Chapter 4

Quelques protocoles réseau

Nous allons décrire quelques protocoles réseau dans ce chapitre, plus précisément Ethernet, le protocole de résolution d'adressele ARP, le protocole de couche réseau IPv4 et les protocoles de couche transport UDP et TCP.

4.1 Protocole Ethernet

4.1.1 Protocole de sous-couche MAC pour Ethernet

4.1.1.1 Adresse physique MAC

Ethernet [IEEE-802.3] et quelques autres standards utilisent une adresse physique sur 48 bits, appelée **adresse MAC**, dont la structure est donnée sur la figure suivante :

1 bit	1 bit	22 bits	24 bits
I/G	U/L	Adresse de sous-réseau	Partie locale

- Le bit `I/G`, le moins significatif de l'adresse, est le bit d'adresse d'individu ou de groupe. Si ce bit vaut 0, l'adresse est celle d'un individu. S'il vaut 1, le reste du champ de l'adresse identifie une adresse de groupe qui nécessite une résolution supplémentaire.

  Lorsqu'une trame est adressée à un groupe, toutes les stations du groupe la reçoivent. Une diffusion de ce type est appelée **diffusion restreinte**, **diffusion multidestinataire** ou **diffusion multidiffusion** (*multicast* en anglais).

- Le deuxième bit, `U/L`, est le bit local ou universel. S'il vaut 0, l'adresse de sous-réseau a été définie par l'organisme d'administration universel. S'il vaut 1, l'adresse de sous-réseau a été définie localement.

  L'organisme d'administration universel fut d'abord Xerox (le créateur d'Ethernet, comme nous l'avons vu) puis ensuite l'IEEE au niveau mondial.

- Dans le cas où le deuxième bit vaut 0, les 24 premiers bits constituent l'**OUI** (pour *Organization Unique Identifier*).

- Les 24 derniers bits identifient l'adresse dans le réseau local et est gérée localement.

Si la totalité de l'adresse est composée de bits à 1, l'adresse a une signification spéciale : elle indique que toutes les stations du réseau sont considérées comme destinataires. On parle de **diffusion générale** (*broadcast* en anglais).

Un sous-réseau est donc identifié par 22 bits. Ce système autorise donc 2^{22} combinaisons – soit 4 millions environ – et il est fort possible qu'au rythme de croissance actuel d'Internet, on se retrouve à court d'OUI. Si une organisation est à court d'adresses physiques, l'IEEE a la possibilité de lui affecter une deuxième adresse de sous-réseau.

4.1.1.2 Les trames Ethernet DIX

Commençons par décrire une trame de l'Ethernet d'origine, appelée **trame DIX** (pour *DEC-Intel-Xerox*, rappelons-le). Une trame Ethernet ne comprend pas seulement un en-tête et des données. Elle est suivie d'un suffixe comme le montre le schéma suivant :

Préambule	Adresse destinataire	Adresse émetteur	Type	Données	CRC
64 bits	48 bits	48 bits	16 bits	Longueur variable	32 bits

- Le **préambule** est un ensemble de bits dont la fonction principale est de synchroniser le processus de communication et de mettre en évidence tout bruit parasite affectant les premiers bits envoyés.

Dans le cas d'une trame de l'Ethernet d'origine, chacun des 8 octets du préambule contient la séquence binaire `10101010`. Le codage Manchester de cette séquence produit un signal de forme rectangulaire de 10 MHz pendant 6,4μs qui permet à l'horloge du récepteur de se caler avec celle de l'émetteur. Les parties doivent ensuite rester synchronisées pour le reste de la trame et utiliser le code Manchester pour identifier les délimitations des bits.

- Les adresses MAC du destinataire et de l'expéditeur suivent. La norme autorise des adresses sur 2 ou 6 octets, mais les paramètres définis pour la norme à 10 Mbit/s en bande de base n'emploient que des adresses sur 6 octets.

- Dans le cas d'une trame DIX, le champ suivant de 16 bits, le **type**, permet au récepteur le démultiplexage au niveau de la couche réseau, autrement dit ce qu'il doit faire avec la trame. En effet plusieurs protocoles de couche réseau peuvent être employés en même temps sur un même ordinateur. Ainsi, lorsqu'une trame Ethernet arrive, le noyau du système doit savoir à quel protocole la remettre. C'est ce champ qui lui permet de le savoir.

  Voici quelques valeurs de ce champ :

Valeur décimale	Description
512	Xerox PUP
2048	Internet Protocol (IP)
2051	ECMA
2053	X.25 Level 3
2054	Address Resolution Protocol (ARP)
32821	Reverse ARP
32823	Appletalk

  dont 2048 et 2054 seront les seules valeurs qui nous intéresseront.

- Les données suivent l'indicateur de type, dont la longueur varie entre 46 et 1 500 octets suivant le type de protocole.

  La valeur maximum de 1 500 a été choisie de façon arbitraire, principalement parce qu'un transcepteur devait disposer de suffisamment de mémoire pour contenir une trame entière et qu'en 1978 la mémoire RAM était coûteuse.

  Outre une longueur maximale, une trame doit respecter une longueur minimale. Bien qu'un champ de données de zéro octet soit parfois utile, cela pose problème. Lorsqu'un transcepteur détecte une collision, il tronque la trame en cours, si bien que des bits ou des fragments de trame égarés apparaissent continuellement sur le câble. Pour faciliter la distinction entre les trames valides et les rebuts parasites, Ethernet exige que les trames valides possèdent au moins 64 octets, du champ d'adresse de destination au champ de somme de contrôle, ces deux champs inclus. Si le champ des données a une taille inférieure à 46 octets, elles sont complétées par des 0 jusqu'à obtenir 46 octets, ce qui constitue le **champ de remplissage**.

- À la fin de la trame se trouvent quatre octets de vérification, (**CRC** pour *Cyclic Redundancy Check*), qui permet de s'assurer plus ou moins que le contenu de la trame n'a pas été modifié lors de la transmission. Chaque passerelle qui figure sur la route de transmission doit calculer une valeur de CRC pour la trame et la comparer à la valeur qui figure à la fin de celle-ci. Si les deux sont égales, la trame est envoyée un peu plus loin dans le réseau. Sinon, c'est que la trame a été altérée en cours de route et elle doit donc être détruite (elle

sera retransmise – après expiration d'un temporisateur (*timer*) – par la machine qui l'a envoyée).

4.1.2 Protocole 802.3 de la sous-couche LLC

La sous-couche LLC comprend un certain nombre de protocoles. Dans le cas de 802.3, le champ données de la trame ne comporte pas d'en-tête de sous-couche 2b ; par contre la signification des champs de l'en-tête Ethernet (*a priori* de la sous-couche MAC) a légèrement changé. On parle alors de **trame Ethernet 802.3**. Il y a deux changements :

- Le premier changement concerne le préambule. Les sept premiers octets restent inchangés ; le huitième octet sert de délimiteur de début de trame (**SFD** pour *Start Frame Delimiter*) afin de rendre le format compatible avec les standards 802.4 et 802.5.

- Le second changement a d'abord transformé le champ de type en champ de **longueur** de trame. Bien sûr, il n'était plus possible pour un récepteur de savoir ce qu'il fallait faire avec une trame entrante, mais ce problème a été résolu par l'ajout d'un petit en-tête dans la portion des données pour apporter cette information.

  Malheureusement, à l'époque où la norme 802.3 fut publiée, il y avait déjà tellement d'équipements et de logiciels en circulation pour l'Ethernet DIX que peu de fabricants et d'utilisateurs ont accueilli avec enthousiasme le changement du champ de type en champ de longueur. En 1997, l'IEEE jeta l'éponge et annonça que les deux formats étaient valides. Heureusement, la plupart des champs de type utilisés avant 1997 étaient d'une valeur supérieure à 1 500 (la longueur maximum des données). Par conséquent, une valeur inférieure ou égale à 1 500 peut être considérée comme étant une longueur et une valeur supérieure à 1 500 comme étant un type.

4.2 Le protocole de résolution d'adresse ARP

La conversion des adresses entre les différentes couches de protocoles représente une tâche essentielle lors de l'identification claire des ressources dans un réseau informatique. Elle est requise au moment de la transition entre couches car chaque couche utilise ses propres types d'adresses (adresses IP, MAC, ATM, etc.). Lors de l'envoi d'un paquet *via* le protocole Internet, l'ordinateur cible est indiqué sous la forme d'une adresse IP. Sur la couche liaison de données on peut implémenter différentes technologies (par exemple Ethernet, Token Ring ou ATM) avec leurs propres formats d'adresses. Il s'agit par exemple des adresses MAC de 48 bits pour Ethernet.

Pour pouvoir envoyer un paquet à une instance IP de l'ordinateur cible ou du routeur le plus proche, l'adresse MAC du prochain saut doit être déterminée dans l'instance de protocole émettrice. Ceci s'effectuait à l'aide de tables statiques dans chaque ordinateur au début de l'ARPANET. Toutefois le développement de l'ARPANET a rendu cette méthode trop rigide et trop onéreuse en mémoire. C'est pourquoi la [RFC 826] a introduit l'ARP (*Address Resolution Protocol*) en vue de la conversion des formats d'adresses.

Bien que la famille de protocoles TCP/IP soit devenue la norme de premier plan pour la presque totalité des réseaux informatiques, l'ARP n'a pas seulement été mis au point pour l'attribution spécifique d'adresses IP et MAC.

4.2.1 Requête et cache ARP

Lorsqu'un ordinateur A veut envoyer un paquet à l'ordinateur (ou routeur) B, situé dans le même réseau local et dont il connaît l'adresse IP mais pas l'adresse MAC, il envoie une demande (**requête ARP**) à tous les ordinateurs du réseau local (diffusion générale d'adresse MAC `FF:FF:FF:FF:FF:FF`). L'ordinateur recherché reconnaît à l'adresse IP placée dans le paquet ARP que la requête lui est destinée et envoie une réponse à l'ordinateur demandeur A, qui lui communique son adresse MAC.

Pour ne pas être contraint de demander à nouveau l'adresse MAC lors des paquets suivants, A enregistre l'adresse MAC de B dans une table locale, appelée **cache ARP**. L'ordinateur B peut également extraire l'adresse MAC de l'ordinateur A à partir de la requête et l'enregistrer à titre conservatoire dans son cache ARP.

4.2.2 Structure des commandes ARP

Les requêtes et les réponses ARP ont une structure identique. Elles sont différenciées par le champ `opération`. Cette structure est la suivante :

<table>
<tr><td>0 7</td><td>8 15</td><td>16 31</td></tr>
<tr><td colspan="2">Type matériel</td><td>Type de protocole</td></tr>
<tr><td>Longueur d'adresse couche 2 (n)</td><td>Longueur d'adresse couche 3 (m)</td><td>Opération</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 3) : m octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 3) : m octets</td></tr>
</table>

- Le **type matériel** spécifie le protocole de la couche 2 utilisé. Il s'agit par exemple de la valeur 1 pour un réseau Ethernet.
- Le **type de protocole** spécifie le protocole de la couche 3 utilisé, par exemple `0x08 00` pour IPv4.
- La **longueur d'adresse de couche 2** spécifie la longueur n, en octets, de l'adresse de la couche 2 pour le protocole utilisé. Dans le cas d'une adresse MAC de 48 bits, on a donc 6.
- La **longueur d'adresse de couche 3** spécifie la longueur m, en octets, de l'adresse de la couche 3 pour le protocole utilisé. Dans le cas d'une adresse IPv4 de 32 bits, on a donc 4.
- Le champ **opération** indique le type de l'unité de données de protocole ARP : 1 pour une requête ARP, 2 pour une réponse ARP.
- Les champs **adresse de la couche 2 de l'expédieur** et **adresse de la couche 2 du destinataire** sont constitués chacun de n octets.
- Les champs **adresse de la couche 3 de l'expédieur** et **adresse de la couche 3 du destinataire** sont constitués chacun de m octets.

Ces commandes sont encapsulées entre un en-tête et un suffixe de couche 2 et constituent ainsi une trame qui est envoyée en diffusion générale.

Exemple. L'ordinateur A, d'adresse MAC `49:72:16:08:64:14` et d'adresse IP `129.25.10.72`, qui veut rechercher l'adresse MAC de l'ordinateur d'adresse IP `129.25.10.11`, envoie l'unité suivante à `FF:FF:FF:FF:FF:FF` :

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 01
49 72 16 08 64 14		
129 25 10 72		
00 00 00 00 00 00		
129 25 10 11		

L'ordinateur B, d'adresse MAC `49:78:21:21:23:90` et d'adresse IP `129.25.10.11`, répond à `49:72:16:08:64:14`, c'est-à-dire à l'ordinateur A :

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 02
49 72 16 08 64 14		
129 25 10 72		
49 78 21 21 23 90		
129 25 10 11		

4.2.3 La commande *arp*

La commande *arp* permet à un utilisateur d'afficher la table ARP (cache ARP) d'un ordinateur, de générer des adresses permanentes ou de supprimer des entrées.

4.2.3.1 Affichage de la table

Lorsque la commande *arp* est appelée avec l'option `-a`, elle affiche la table ARP de l'ordinateur :

```
linux # arp -a
Adresse IP   Type HW         Adresse HW
129.25.10.97 10Mb/s Ethernet 49:72:16:08:80:70
129.25.10.72 10Mb/s Ethernet 49:72:16:08:64:14
```

La première colonne indique l'adresse IP de l'ordinateur cible. La deuxième colonne donne des renseignements sur la catégorie du réseau local (par exemple Ethernet 10 Mb/s) et la dernière contient l'adresse de couche 2 de l'adaptateur réseau.

Si le texte `incomplete` apparaît au lieu de l'adresse matérielle, cela indique une interruption ou une panne du périphérique réseau correspondant.

4.2.3.2 Suppression d'une entrée ARP

Lorsqu'on appelle `arp` avec l'option `-d Ordinateur`, l'entrée est supprimée de la table. De cette façon, une nouvelle requête ARP s'impose lors de la prochaine demande d'adresse de couche 2 pour cet ordinateur. La suppression d'une attribution d'adresse ARP peut être utile lorsqu'un ordinateur a été mal configuré ou que l'adresse de couche 2 a changé, par exemple lors du changement de la carte réseau.

De toute façon, pour éviter ce cas, on déclare automatiquement les entrées ARP non valides au bout d'un certain temps. Cette période s'élève à quelques minutes, de sorte que le changement d'une carte réseau ne puisse pas poser de réel problème.

4.2.3.3 Ajout manuel d'une entrée ARP

Il peut parfois être très utile de saisir manuellement une entrée dans la table ARP. Il existe, pour ce faire, l'option :

```
arp -s Ordinateur AdresseCouche2
```

Par exemple :

```
linux # arp -s tux 49:72:16:08:64:14
```

Ceci peut être utile lorsqu'un ordinateur ne répond pas aux requêtes ARP. Contrairement aux entrées déterminées automatiquement dans le cache ARP, celles qui sont générées à l'aide de cette commande ne sont pas éliminées au bout d'un certain temps mais restent stockées dans le cache jusqu'au redémarrage de l'ordinateur.

4.3 Le protocole de couche réseau IPv4

4.3.1 Étude générale de la couche réseau

4.3.1.1 Ce que fait la couche réseau

La couche réseau gère le déplacement des paquets dans un réseau, c'est-à-dire sur des machines qui ne sont pas nécessairement adjacantes. Elle **fragmente** éventuellement les données de la couche transport, pour obtenir des paquets d'une taille standard, et est responsable du **routage** des paquets, c'est-à-dire de la détermination du chemin à suivre par le paquet en proposant des itinéraires de substitution en cas de problème.

4.3.2 Ce que ne fait pas la couche réseau

Nous venons de voir que la couche réseau s'occupe de la fragmentation et du routage. Elle ne se préoccupe pas de la fiabilité de la livraison des paquets. Cette dernière n'est pas garantie car le paquet peut être retardé, mal routé ou altéré lors du fractionnement et du réassemblage des fragments. Il n'existe pas de fonctionnalité permettant de vérifier qu'un paquet envoyé a été correctement reçu. Le protocole IP de couche réseau dispose d'une somme de contrôle du contenu de l'en-tête d'un paquet mais pas de son contenu. Elle peut essayer de deviner le meilleur itinéraire jusqu'au prochain nœud sur le chemin, mais ne vérifie pas que le chemin choisi est le plus rapide ou le plus efficace.

La couche réseau est **sans connexion**, ce qui signifie qu'elle ne se soucie pas de savoir par quels nœuds passe un paquet, ni même quelles sont ses machines de départ et d'arrivée. Ces informations figurent dans l'en-tête, mais l'analyse et la transmission d'un paquet n'ont rien à voir avec l'analyse faite par la couche réseau de l'adresse d'émission et de réception.

4.3.3 Les tribulations d'un paquet IP

4.3.3.1 Hôte de départ

Lorsqu'une application doit envoyer un paquet sur le réseau, le protocole IPv4 de la couche réseau effectue un certain nombre de tâches simples :

- Elle vérifie tout d'abord si le paquet doit être fragmenté. IP autorise une taille de paquet maximale de 65 535 octets, ce qui est bien supérieur à ce que peuvent traiter la plupart des couches inférieures. Si c'est le cas elle crée les fragments.

  Remarquons que la fragmentation ne concerne que le protocole de transport UDP. Le protocole TCP n'en a pas besoin car la connexion lui permet de savoir le nombre d'octets qu'il peut envoyer.

- Elle construit ensuite, pour chaque fragment, le paquet IP en respectant les longueurs valides, telles qu'elles sont spécifiées par l'implémentation IP locale. Une **somme de contrôle** est calculée pour les données, puis l'en-tête IP est construit.

- Ensuite commence le routage. Il est nécessaire de déterminer le premier "saut" (*hop* en anglais), c'est-à-dire la première machine de la route vers la destination, afin de router le paquet vers la machine de destination : la machine elle-même ou une autre machine du réseau local, la machine de destination ou une **passerelle** si on utilise un interréseau. Si un routage précis est prédéterminé (à titre de contrôle en général), les informations sur celui-ci sont ajoutées à l'en-tête au moyen d'une **option**.

- Enfin le paquet est transmis à la couche inférieure pour être envoyé sur le réseau physique.

4.3.3.2 Passerelles et routeurs

À mesure qu'un paquet traverse l'interréseau, chaque passerelle effectue une série de tests :

- Une fois que la couche inférieure a enlevé l'en-tête de la trame reçue, la couche IP de la passerelle calcule la somme de contrôle et vérifie l'intégrité du paquet. Si les sommes de contrôle ne correspondent pas, le paquet est détruit et un message d'erreur est envoyé au composant émetteur *via* ICMP.

- Ensuite, un champ de **durée de vie** (TTL pour *Time To Live* en anglais) de l'en-tête IP est décrémenté puis examiné. Si la durée de vie du paquet a expirée, le paquet est écarté et un message d'erreur est envoyé au composant émetteur *via* ICMP.

- Après avoir déterminé le prochain saut de la route, en analysant l'adresse de destination ou à partir du routage spécifique indiqué dans le champ option de l'en-tête IP, le paquet est reconstruit avec une nouvelle valeur de TTL et donc une nouvelle somme de contrôle.

- Si une fragmentation est nécessaire, soit parce que la longueur du paquet a augmentée, soit du fait d'une limitation du logiciel d'envoi de la passerelle, le paquet est divisé, et de nouveaux paquets, contenant les informations d'en-tête adéquates, sont créés.

- Si un routage ou une estampille temporelle est nécessaire, il est ajouté.

- Enfin, le paquet est renvoyé à la couche inférieure.

4.3.3.3 Hôte de destination

Lorsque le paquet est finalement reçu par le composant récepteur :

- Le système effectue un calcul de la somme de contrôle et vérifie que celle-ci concorde avec le champ adéquat de l'en-tête. Si ce n'est pas le cas, un message ICMP est envoyé à l'émetteur.

- Il regarde ensuite s'il y a des fragments supplémentaires. L'opération inverse de la fragmentation s'appelle le **réassemblage**. Si d'autres paquets sont nécessaires pour réassembler le paquet originel, le système attend, après avoir lancé un **minuteur de réassemblage**, pour ne pas être bloqué si les paquets suivants mettent trop de temps à arriver.

- Si le composant ne peut réassembler toutes les parties du paquet originel avant que le minuteur n'atteigne 0, la partie du paquet arrivée est détruite et un message d'erreur est envoyé à l'émetteur *via* ICMP.

  Une des conséquences de cette étape est qu'un paquet fragmenté a moins de chances d'arriver à bon port qu'un paquet non fragmenté, ce qui explique pourquoi la plupart des applications essaient d'éviter autant que possible la fragmentation.

- Enfin l'en-tête IP est enlevé, le paquet originel est reconstruit (s'il a été fragmenté) et il passe à travers les couches pour arriver à la couche d'application.

4.3.4 Fragmentation

Pour pouvoir envoyer des paquets IP sur tous les types de réseau physique, le protocole IP doit être en mesure d'adapter la dimension de ceux-ci au type de réseau en place. Nous avons déjà vu que tout réseau physique comporte une taille de trame maximale, qualifiée de *MTU* (pour l'anglais *Maximum Transfert Unit*, unité de transfert maximale). Seuls des trames de cette taille peuvent être envoyées sur le réseau.

Si la MTU d'un support de transfert est inférieure à la taille d'un paquet à expédier, ce dernier doit être divisé en paquets IP plus petits, appelés **fragments**.

Il ne suffit pas que les protocoles de la couche de transport n'expédient que des paquets convenant à la MTU, tout au moins dans le cas non connecté. Un paquet peut traverser plusieurs réseaux différents avec des MTU différentes sur le trajet de l'hôte source à l'hôte cible. C'est pourquoi il faut appliquer une procédure plus flexible capable également de générer des paquets plus petits au sein d'un système de transmission (routeur) sur la couche IP. Cette procédure est appelée **fragmentation**.

La fragmentation sous-entend que le protocole IP de tout ordinateur IP (que ce soit un routeur ou un système d'extrémité) doit être en mesure de reconstituer ces fragments pour obtenir le paquet d'origine (*réassemblage*).

Chaque fragment d'un paquet IP découpé est un paquet IP complet qui contient un en-tête IP. Le paquet originel auquel appartient un fragment se reconnait grâce au champ identificateur de l'en-tête IP. Ce champ seul ne suffit pas à déterminer le fragment. On utilise également les champs adresse de l'émetteur, adresse cible et protocole.

Les différents fragments d'un datagramme peuvent emprunter des itinéraires différents pour parvenir à l'ordinateur cible et être également fragmentés plusieurs fois sur leurs trajets. La position des données d'un fragment au sein du datagramme IP d'origine est identifiée grâce au champ `décalage de fragment` de leur en-tête IP. Les sous-paquets sont repérés par un décalage et non par un numéro parce qu'un sous-paquet peut être fragmenté à tout moment le long de sa route. On ne peut pas revenir en arrière pour renuméroter les fragments.

Tous les fragments, jusqu'à l'avant-dernier, comportent un bit `MF` (pour l'anglais *More Fragments*) pour indiquer que d'autres fragments vont suivre.

La figure 4.1 ([WPRMB–02], p. 280) montre l'exemple d'un paquet IP qui doit être fragmenté plusieurs fois.

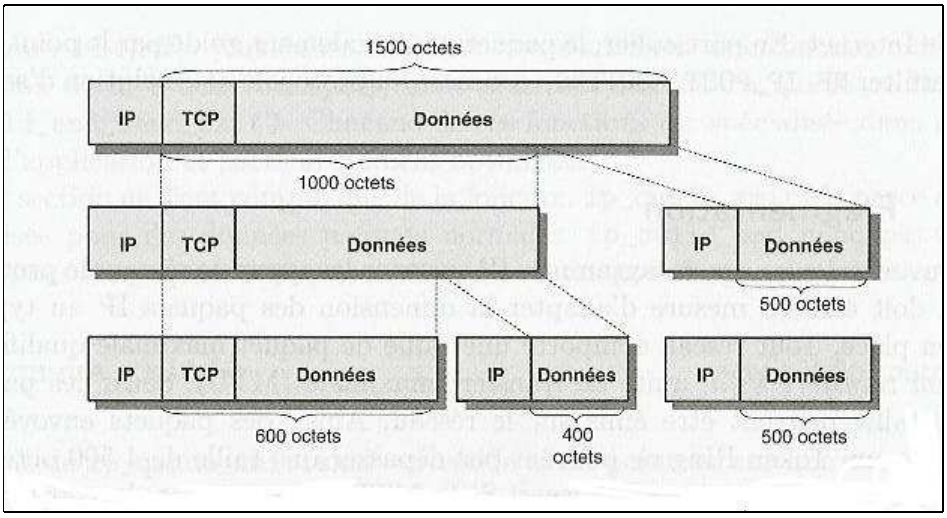


Figure 4.1: Fragmentation d'un paquet IP

4.3.5 Le routage

4.3.5.1 Notion

Le **routage** (*routing* en anglais) est l'une des tâches de la couche réseau. Il s'agit de déterminer la route à suivre pour aller d'un point A à un point B. Ceci est l'objet des ordinateurs spécialisés appelés **routeurs** (*router* en anglais) et non des hôtes d'extrémité. Le routage ne devrait donc pas nous intéresser dans ce livre mais en fait il a également des répercussions sur les hôtes d'extrémité.

Le routage comprend deux modules :

- La **redirection** (*forwarding* en anglais) recherche, pour chaque paquet, à partir de la destination de celui-ci dans une table l'interface sur laquelle transmettre ce paquet. Cette table est appelée **table de redirection** (*forwarding table* en anglais).
- La tâche de routage proprement dit consiste à construire et à maintenir la table de redirection.

La redirection est une procédure du noyau alors que la construction de la table est une procédure utilisateur. Le routage proprement dit ne nous intéressera ici qu'à travers la façon d'ajouter ou de retrancher un élément à cette table.

En gros le rôle de la table de redirection est le suivant : on lui fournit une adresse IP et elle renvoie le prochain saut (*next hop* en anglais, abrégé en `nh`), sous la forme de l'interface réseau sur laquelle envoyer le paquet (ce qui détermine le réseau local) et une adresse IP (ce qui détermine l'élément de ce réseau local). La détermination de l'interface réseau est importante car un routeur possède au minimum deux interfaces réseau et en moyenne quatre à cinq.

4.3.5.2 Routage et tables de hachage

Les adresses IPv4 ont une longueur de 32 bits. Une table à deux colonnes, une colonne pour placer une adresse IP par ligne et une colonne pour le prochain saut, est peu réaliste. Rien que la première colonne occuperait 4×2^{32} octets, soit 16 Go de mémoire.

L'idée est donc d'utiliser une table de hachage (figure 4-2, [C-P–02], p. 548). Connaissant la distribution approximative des adresses IP actuellement assignées, une fonction de hachage peut envoyer une adresse IP, parmi les 2^{32} adresses possibles, dans un ensemble plus petit.

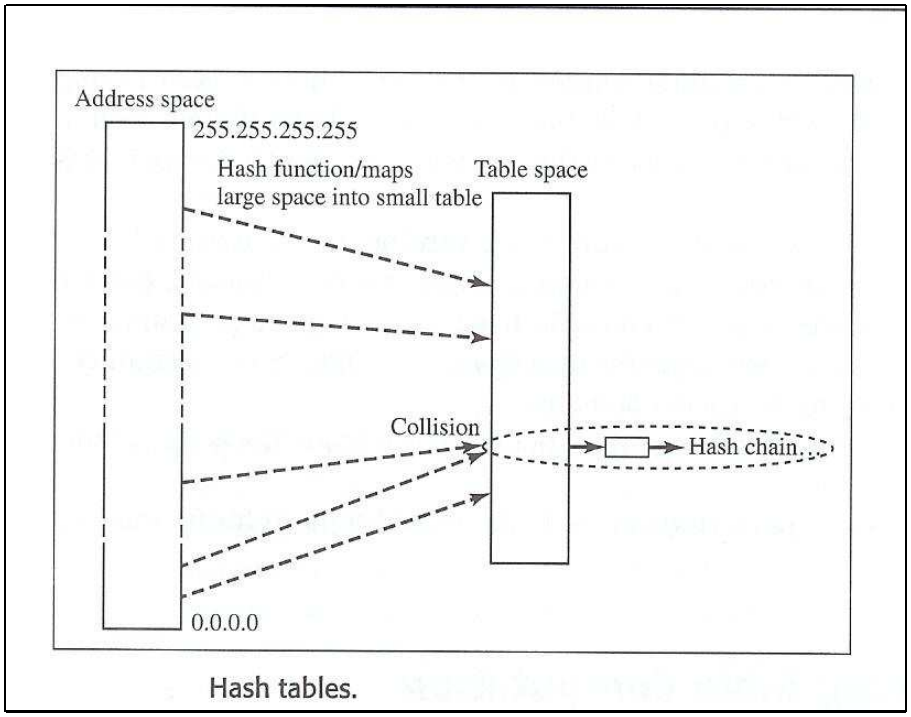


Figure 4.2: Table de hachage

Bien entendu il y aura inévitablement des collisions, aussi prevoit-on un chaînage sous la forme d'une liste chaînée pour les adresses qui donnent lieu au même index de hachage.

4.3.6 Routage et adresse IP

La distribution des adresses IP simplifie la conception de la fonction de hachage. En effet les adresses IPv4 ne sont pas de simple identifiants de nœuds réseau. Elles sont distribuées de façon à faciliter le routage. Les adresses IP sont structurées de manière hiérarchique et sont formées de deux parties distinctes : une **adresse de sous-réseau** (*network part* en anglais) et un **identifiant de système terminal** (*host part* en anglais). L'adresse de sous-réseau est identique pour tous les éléments présents dans un sous-réseau donné. Quant à l'identifiant de système terminal, il est destiné à distinguer les différents éléments au sein d'un sous-réseau donné.

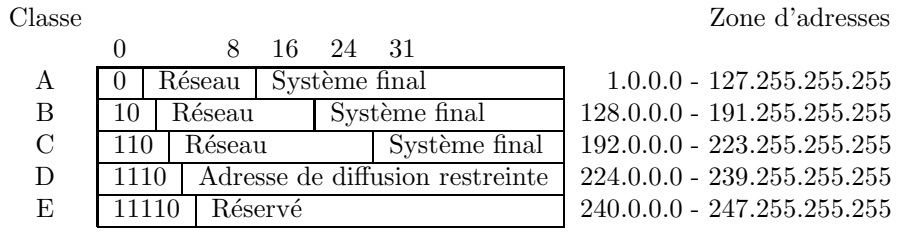
Dans le cas du routage, l'identifiant du système terminal peut être ignoré jusqu'à ce que le paquet soit parvenu dans le bon sous-réseau. De ce fait, les routeurs n'ont pas besoin de connaître les identifiants des systèmes terminaux, ce qui permet ainsi de réduire de manière notable la quantité d'informations à enregistrer, grâce au découpage de l'adresse IP en deux parties.

Il existe deux façons de savoir quelle est la taille de la partie de l'adresse IP correspondant au sous-réseau et celle correspondant à l'identifiant de système terminal, comme nous allons le

voir. Dans les deux cas la partie sous-réseau d'une adresse IP se trouve toujours au début, d'où le synonyme **préfixe réseau** que l'on rencontre quelquefois.

4.3.6.1 Classes d'adresses

Le premier système ([RFC 791]), datant de 1981, consiste à distinguer trois **classes d'adresses**, comme le montre la figure ci-dessous :



Les classes d'adresses A, B et C comprennent une partie sous-réseau respectivement de 7, 14 et 21 bits et un identifiant de système terminal de 24, 16 et 8 bits. L'appartenance à l'une de ces classes est déterminée par les premiers bits (forts) de l'adresse.

Ce schéma d'adressage a été conçu de manière à permettre à chaque réseau physique d'obtenir un identifiant sous-réseau appartenant à l'une des trois classes A, B ou C en fonction de la taille du réseau physique. L'idée était qu'il pouvait y avoir 256 sous-réseaux de classe A, tels que ARPANET et SATNET, 65 536 sous-réseaux de classe B, tels que Berkeley, le MIT, UCLA et UCL, et 16 777 216 sous-réseaux de classe C.

Cependant, il est très rapidement apparu que cette approche épuiserait très rapidement les préfixes réseau disponibles. Par ailleurs, les classes présentes se sont fréquemment montrées peu adaptées : un réseau de classe A pourrait compter près de 2^{24}, c'est-à-dire 16 777 216 systèmes terminaux, une valeur qui dépasse les besoins des plus grandes entreprises – indépendamment du fait qu'aucune technique réseau n'est en mesure de gérer un aussi grand nombre de systèmes terminaux. Les sous-réseaux de classe C – qui sont très nombreux – ne peuvent en revanche gérer que 254 systèmes terminaux, ce qui est généralement insuffisant.

4.3.6.2 Adressage sans classe

C'est la raison pour laquelle des techniques visant à une meilleure utilisation de l'espace d'adressage disponible ont été développées. L'idée fondamentale est de permettre que la limite entre le préfixe réseau et l'identifiant du système terminal soit placé à n'importe quelle position de bit, au lieu de ne l'autoriser qu'aux trois positions prévues par les classes d'adresses A, B et C. On utilise actuellement sur Internet le système **CIDR** (*Classless Inter-Domain Routing*) défini dans [RFC 1518] et [RFC 1519].

Les préfixes réseau peuvent avoir une longueur quelconque. L'information concernant la longueur de l'identifiant du sous-réseau ne peut plus être déterminée à partir des premiers bits de l'adresse. Cette information doit être transmise par ailleurs. On a le choix pour cela entre deux notations utilisateur :

- Dans la première notation, le nombre de bits correspondant au préfixe du sous-réseau est indiqué sous forme d'un nombre décimal, séparé de l'adresse par une barre oblique. Ainsi, l'adresse 192.168.152.0/21 désigne un sous-réseau dont le préfixe est représenté par les 21 premiers bits de l'adresse IP.

- La seconde notation consiste à ajouter à l'adresse IP un masque de bits de même longueur,

le **masque réseau**, dans lequel tous les bits appartenant au préfixe du sous-réseau dans l'adresse IP ont la valeur 1. Par exemple le réseau évoqué ci-dessus est noté :

192.168.152.0/255.255.248.0.

4.3.6.3 Remarque

Ce système a fonctionné parfaitement durant de nombreuses années, tant qu'un sous-réseau ne possèdait qu'un seul préfixe réseau (*single homed* en anglais). Ce n'est plus le cas de nos jours puisqu'il est apparu plus simple d'attribuer plusieurs préfixes réseau à un même sous-réseau (*multihoming* en anglais).

4.3.7 En-tête IPv4

Rappelons que l'unité de transfert utilisée par IP est appelée **paquet**, ou plus précisément **paquet Internet** ou **paquet IP**. Un paquet est constitué d'un en-tête IP suivi des données. La figure suivante montre l'agencement de l'en-tête IP, telle que définie par [RFC 791] ("*Internet Protocol*" écrite par Postel en 1981) :

<table>
<tr><td>Version
(4 bits)</td><td>Longueur
(4 bits)</td><td>Type
(8 bits)</td><td colspan="4">Longueur du paquet
(16 bits)</td></tr>
<tr><td colspan="3">Identificateur</td><td></td><td>DF</td><td>MF</td><td>Décalage</td></tr>
<tr><td colspan="2">TTL</td><td>Transport</td><td colspan="4">Somme de contrôle</td></tr>
<tr><td colspan="7">Adresse émetteur</td></tr>
<tr><td colspan="7">Adresse destinataire</td></tr>
<tr><td>Options</td><td colspan="6">Remplissage</td></tr>
</table>

Commentons les champs de cet en-tête un à un :

- Le **numéro de version** est un champ de 4 bits contenant le numéro de version IP que le paquet utilise.

  La version la plus utilisée est la version 4 mais quelques systèmes effectuent des tests avec la version 6. Nous décrivons la version 4 dans la suite.

  Ce champ est *a priori* nécessaire pour que le logiciel IP récepteur sache décoder le reste de l'en-tête, qui varie à chaque nouvelle version de protocole IP. En fait nous avons vu que la distinction se fait par le champ *protocole* de l'en-tête Ethernet au niveau de la couche MAC (`0x800` pour IPv4, `0x86DD` pour IPv6).

- La **longueur de l'en-tête** (`IHL` pour *Internet Header Length*) est un champ de 4 bits qui indique la longueur totale de l'en-tête IP construit par la machine émettrice. Cette longueur permet de déterminer où commencent les données puisqu'il n'existe pas de marqueur de début des données.

  Les spécifications d'IP (ainsi que de la plupart des autres protocoles de la suite TCP/IP) définissent comme unité de longueur le **mot**, un mot représentant 32 bits, soit 4 octets.

  L'en-tête IP a une longueur maximale de quinze mots, soit 60 octets. Le plus court en-tête autorisé par IP (sans aucune option) utilise cinq mots, soit 20 octets.

- Le champ suivant, de 8 bits (un octet), spécifiait à l'origine le **type de service** (`TOS` pour *Type Of Service*), qui indiquait aux routeurs comment traiter correctement le paquet.

Les 8 bits de ce champ étaient répartis de la façon suivante :

Priorité	Délai	Débit	Fiabilité	non utilisé

- Les trois premiers bits indiquaient l'**ordre de priorité** (ou précédence) du paquet, avec des valeurs comprises entre 0 (normal) et 7 (contrôle réseau). Plus cette valeur est grande, plus le datagramme est important, ce qui, en théorie du moins, le fera arriver plus vite à destination.
- Les trois bits suivants sont des drapeaux d'un bit qui contrôlent le délai, le débit et la fiabilité du paquet. Un bit positionné à 1 indique un délai bas, un grand débit et une haute fiabilité respectivement.
- Les deux derniers bits ne sont pas utilisés.

En pratique, la plupart des implémentations des routeurs TCP/IP ignoraient ce champ et traitaient tous les paquets avec les mêms valeurs de priorité, de délai, de débit et de fiabilité. Il prenait donc la plupart du temps la valeur 0.

La signification de ce champ a été changée dans [RFC 2474] pour devenir *Differentiation Services Codepoint*. Il indique maintenant le comportement de transmission utilisé, ce qui n'intéresse que les routeurs, aussi ne nous nous en occuperons pas dans ce livre.

- La **longueur du paquet** est un champ de 16 bits, soit deux octets, qui indique la longueur totale du paquet, en-tête compris, en octets.

  On a donc des paquets d'au plus 65 535 octets.

- L'**identificateur de paquet originel** (*Fragment-ID*) est un champ de 16 bits, soit deux octets, qui contient un identificateur unique créé par le nœud émetteur pour chaque paquet, avant que celui-ci ne soit éventuellement fragmenté. Ce numéro est nécessaire lors du réassemblage des fragments, afin de garantir que les fragments d'un paquet ne soient pas mêlés à ceux d'un autre.

  Comme nous l'avons déjà dit, un paquet est identifié par cet identificateur mais également par l'adresse de l'émetteur, l'adresse cible et le protocole de transport.

- Le champ **drapeaux** est un champ de trois bits qui contrôle la gestion des paquets lorsqu'une fragmentation est requise. Le premier bit n'est pas utilisé, les deux bits restants permettent de spécifier des drapeaux appelés `DF` (pour *Don't Fragment*, ne pas fragmenter) et `MF` (pour *More Fragments*, encore des fragments) :

  - Si le drapeau `DF` vaut 1, le paquet ne peut en aucun cas être fragmenté. Si la couche de logiciel IP ne peut pas envoyer le paquet à une autre machine sans le fragmenter et que ce bit vaut 1, le paquet est détruit et un message d'erreur doit être envoyé au composant émetteur *via* ICMP.
  - Si le drapeau `MF` vaut 1, c'est que le paquet en cours est suivi d'autres paquets (parfois appelés **sous-paquets**) qui devront être réassemblés pour former le paquet originel. Le dernier fragment d'un paquet envoyé doit avoir son drapeau `MF` égal à 0, de manière à ce que le composant récepteur sache qu'il doit cesser d'attendre de nouveaux sous-paquets. Comme l'ordre d'arrivée des fragments ne correspond pas forcément à leur ordre d'envoi, le drapeau `MF` est utilisé en conjonction avec le champ suivant pour indiquer à la machine réceptrice à quoi ressemble le paquet originel.

- Si le drapeau `MF` est égal à 1, ce qui indique un paquet fragmenté, le champ `décalage de fragment` de 13 bits contient la position, l'unité étant deux mots ou huit octets, au sein du paquet complet, du début des données contenues dans le paquet en cours. Ceci permet à IP de réassembler les paquets fragmentés dans leur ordre correct.

  Puisque ce champ comporte 13 bits et que les décalages sont calculés en unités de 8 octets, on retrouve bien la longueur de paquet maximale de 65 535 octets. Puisque ce champ a une taille de 13 bits, seulement 8 192 fragments au maximum peuvent appartenir à un paquet IP originel. Tous les fragments, sauf le dernier, doivent avoir une longueur multiple de 8 octets, l'unité élémentaire de fragmentation.

- Le champ `TTL` (pour *Time To Live*, durée de vie), de taille un octet, indiquait à l'origine la période, en secondes, durant laquelle un paquet pouvait rester sur le réseau avant d'être détruit par le nœud en cours avec, dans ce cas, un message ICMP envoyé à la machine émettrice. Cette durée de vie permet d'éviter que des paquets IP circulent sans fin sur le réseau et ne saturent celui-ci.

  Cette période est fixée par le nœud émetteur lors de l'assemblage du paquet. Elle a généralement pour valeur 15 ou 30 secondes.

  Les standards TCP/IP stipulent que le champ `TTL` doit être diminué d'au moins une seconde à chaque nœud qui traite le paquet, même si le temps de traitement est inférieur à une seconde. Par ailleurs, lorsqu'un paquet est reçu par une passerelle, le temps d'arrivée est noté, pour que dans l'hypothèse où le paquet doive attendre avant d'être traité, ce temps d'attente soit comptabilisé dans le `TTL`. En conséquence, si une passerelle est surchargée et ne peut pas traiter rapidement tous les paquets, le minuteur du `TTL` peut expirer alors que le paquet attend d'être traité, et ce dernier est détruit.

  Ce compteur indique aujourd'hui le nombre maximal de systèmes intermédiaires (routeurs ou plutôt sauts).

- Le **protocole de transport** est un champ d'un octet qui contient le numéro d'identification du protocole de transport duquel provient ou sera confié le paquet. Les numéros sont définis par le IANA. Il existe actuellement environ 50 protocoles affectés d'un numéro de transport, codifiés dans [RFC 1700]. Les quatre protocoles les plus importants sont ICMP, qui porte le numéro 1, TCP le numéro 6, UDP le numéro 17 et IGMP le numéro 2.

- La **somme de contrôle** ne porte que sur l'en-tête de protocole (et non sur les données) pour accélérer les traitements.

  L'algorithme de contrôle prend le complément à un de la somme sur 16 bits de tous les mots de 16 bits.

  Comme le champ `TTL` est décrémenté lors du passage de chaque routeur, la somme de contrôle doit être recalculée par chaque routeur par lequel passe le paquet.

- Les champs **adresse de l'émetteur** et **adresse de destination** contiennent les adresses IP à 32 bits des composants émetteur et récepteur.

- Le champ **Options** est optionnel et ne nous intéressera pas vraiment dans cet ouvrage. Il est composé de plusieurs codes de longueurs variables. Si plus d'une option est utilisée dans un paquet, les options apparaissent les unes à la suite des autres dans l'en-tête IP. Toutes les options sont contrôlées par un octet, généralement divisé en trois champs : un *drapeau de copie* sur un bit, une *classe d'options* sur deux bits et un *numéro d'option* sur cinq bits.

– Le **drapeau de copie** est utilisé pour indiquer la manière dont l'option est traitée lorsqu'une fragmentation est nécessaire dans une passerelle. Lorsque ce bit vaut 0, l'option doit être copiée sur le premier fragment mais pas sur les suivants. Si le bit vaut 1, elle doit être copiée sur tous les fragments.

– Les deux bits de la **classe d'options** permet quatre classes. Pour le moment il n'existe que deux classes. La classe 0 indique que l'option s'applique au contrôle du paquet ou du réseau. La classe 2 indique que l'option a trait au débogage ou à l'administration.

– Les valeurs actuellement prises en charge pour les classes et numéros d'options figurent dans le tableau suivant :

Classe	Numéro	Description
0	0	Marque la fin de la liste d'options
0	1	Aucune option (utilisée pour le remplissage de caractères)
0	2	Options de sécurité (utilisation militaire seulement)
0	3	Routage tolérant
0	7	Active l'enregistrement du routage (ajoute des champs)
0	9	Routage strict
2	4	Marquage de la date activé (ajoute des champs)

Les options les plus intéressantes sont celles qui activent l'enregistrement du routage et de l'heure. Elles permettent de tracer le passage d'un datagramme au travers de l'interréseau, ce qui peut être utile pour émettre des diagnostics.

Le **routage tolérant** (*loose routing* en anglais) fournit une série d'adresses IP par lesquelles le paquet doit passer, mais il permet de prendre n'importe quelle route pour parvenir à chacune de ces adresses (correspondant à des passerelles en général).

Le **routage strict** n'autorise aucune déviation de la route spécifiée. Si cette route ne peut pas être suivie, le paquet est détruit et un message d'erreur est envoyé *via* ICMP. Le routage strict est fréquemment utilisé pour tester les routes mais rarement pour transmettre des paquets utilisateur.

– Le contenu de la **zone de remplissage** dépend des options sélectionnées. On utilise le remplissage pour s'assurer que la longueur de l'en-tête est bien un multiple de mots.

4.4 Le protocole de couche de transport UDP

Le protocole UDP (*User Datagram Protocol*), décrit dans [RFC 768], constitue un protocole de transport minimal. Il est basé sur le protocole de réseau Internet (IP) et offre pour l'essentiel la même fonctionnalité qu'IP lui-même : un service de datagrammes sans connexion et non fiable. Le protocole UDP possède cependant une fonctionnalité essentielle, celle de multiplexage/démultiplexage : il transmet les datagrammes aux différentes applications d'un système terminal donné, grâce à la spécification des numéros de port.

Le protocole UDP est utilisé d'une part pour les applications orientées transaction telles que DNS, dans lesquelles seules une requête et une réponse associée doivent être transmise, de telle sorte qu'il est inutile d'établir une connexion pour cela. D'autre part, le protocole UDP est également utilisé dans les contextes où la rapidité de la transmission des données prédomine sur la fiabilité. C'est ainsi que dans le cas des flux audio, qui sont transmis sous forme de petits paquets, il n'est pas excessivement gênant que des paquets isolés se perdent ; au contraire, un

contrôle automatique de flux et des erreurs (assorti de la retransmission des paquets perdus) s'avérerait souvent très pénalisant pour la régularité du flux.

4.4.1 L'en-tête UDP

L'en-tête UDP est défini dans [RFC 768]. Sa structure, très simple, n'occupe que huit octets, comme le montre la figure suivante :

Port source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données	Remplissage

- Le **port source** est un champ facultatif contenant le numéro de port de l'expéditeur, compris entre 1 et 65 535. Si aucun numéro de port n'est spécifié, le champ est mis à 0. Ce champ est nécessaire au destinataire s'il doit renvoyer des données.

  Dans le cas de l'implémentation Linux, le port source sera cependant nécessairement indiqué.

- Le **port de destination** est le numéro de port sur la machine de destination.

- La **longueur** est la taille du datagramme, exprimée en nombre d'octets, comprenant en-tête et données. Sa valeur minimum est 8 (taille de l'en-tête UDP) et le datagramme UDP le plus long peut transporter 65 535 - 8 = 65 527 octets de données utiles.

- la **somme de contrôle** est un champ facultatif sur lequel nous allons revenir ci-dessous. De nombreuses applications n'y ont pas recours. S'il n'est pas utilisé, sa valeur doit être zéro.

- Les **données** sont de longueur variable.

- Le **remplissage** assure que le datagramme a une longueur multiple de 16 bits.

4.4.2 Sécurisation optionnelle par somme de contrôle

UDP permet une connexion non fiable dans le sens où il n'existe aucun mécanisme permettant de reconnaître et de traiter les datagrammes perdus ou dupliqués. Les datagrammes peuvent cependant être protégés, de manière optionnelle, des erreurs à l'aide d'une somme de contrôle. Contrairement au protocole IP, cette somme de contrôle ne se rapporte pas uniquement à l'en-tête du datagramme mais également aux données utiles de celui-ci.

Si le datagramme expédié comporte une somme de contrôle, celle-ci est le complément à 1 de 16 bits du complément à 1 de la somme du datagramme (en-tête et données utiles) ainsi que d'un **pseudo en-tête**, dont la structure est montrée à la figure 4-3 ([TJ–97], p.117).

Le pseudo en-tête contient les adresses IP de l'expéditeur et du destinataire, la taille du datagramme UDP et le code de protocole IP pour UDP (à savoir 17). Toutes ces informations proviennent de la couche réseau.

Lorsque le calcul de la somme de contrôle donne la valeur nulle, on indique `0xFFFF`, soit -1, à la place, qui équivaut également à 0 sans toutefois coïncider avec la valeur constante 0 signalant l'absence de somme de contrôle.

La méthode de calcul de la somme de contrôle s'implémente très efficacement pour tous les types de processeurs. Elle est décrite dans les recommandations [RFC 1071], [RFC 1141] et [RFC 1624].

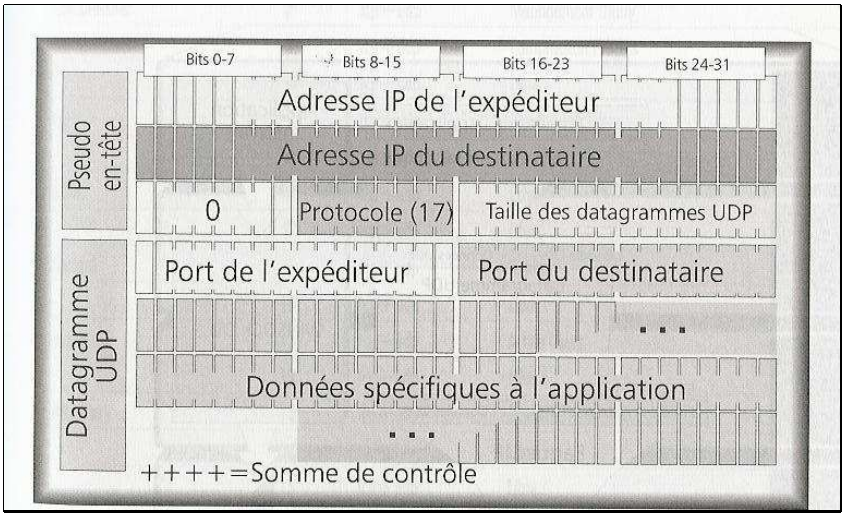


Figure 4.3: Pseudo en-tête UDP

4.5 Le protocole de couche de transport TCP

4.5.1 Fonctionnalités

Le protocole de transport TCP permet, comme UDP, le multiplexage/démultiplexage mais essaie d'aller au-delà : fiabilité (avec transfert des données à nouveau si celles-ci ont été corrompues en cours de route), contrôle du flux (réduction de celui-ci en cas de congestion, augmentation progressive sinon) et connexion.

- **Transfert continu d'octets**. TCP transfère un flux continu d'octets dans chaque direction. Ce sont les processus d'application qui doivent grouper les octets dans un *segment de message* devant être envoyés/reçus. Les segments de message peuvent être de taille arbitraire. Afin d'être plus efficace dans la gestion des messages, les connexions TCP négocient généralement une taille maximum de segment.

  TCP numérote chaque octet qu'il envoie. Les octets sont remis aux processus d'application à l'autre extrémité dans leur ordre d'envoi. On appelle cela le **séquencement des octets**. Bien enetendu, TCP n'envoie pas les données octet par octet, mais par groupe d'octets ; cependant ce sont bien les octets qui sont numérotés.

  Comme TCP envoie les données sous forme de flux d'octets, on ne trouve pas de marqueur de fin de message dans le flux. Pour s'assurer que toutes les données soumises au module TCP ont bien été transmises, une fonction `push()` est nécessaire : elle fait envoyer par TCP toute donnée reçue jusqu'ici en provenance d'une application. Sinon TCP attend que la taille maximum négociée soit atteinte avant d'envoyer un datagramme.

- **Fiabilité**. La fiabilité totale est impossible, cependant TCP possède un mécanisme qui essaie de remettre de façon fiable le flux d'octets. Il essaie de restaurer les données endommagées, perdues, dupliquées ou remises en mauvais état.

  Il utilise le mécanisme **PAR** (*Positive Acknowledgment Retransmission*, retransmission d'acquittement positif). TCP implémente ce mécanisme en affectant un numéro de séquence à chaque octet transmis et en requérant un acquittement positif (**ACK**) du module

TCP récepteur. Si cet acquittement ne parvient pas avant expiration d'un certain délai, TCP effectue une retransmission.

Au niveau du module TCP récepteur les numéros de séquence servent à ordonner les segments et à éliminer les doublons.

Pour détecter les données erronées, on se sert d'un champ somme de contrôle. Les segments de données dont la somme de contrôle ne correspond pas sont écartés.

- **Contrôle de flux**. Les machines qui émettent et reçoivent les segments de données TCP ne le font pas toutes au même rythme en raison de la diversité des unités centrales et de la bande passante. Il peut donc arriver que l'émetteur envoie ses données beaucoup plus rapidement que le récepteur ne peut les gérer. C'est pourquoi TCP implémente un mécanisme de contrôle des flux de données.

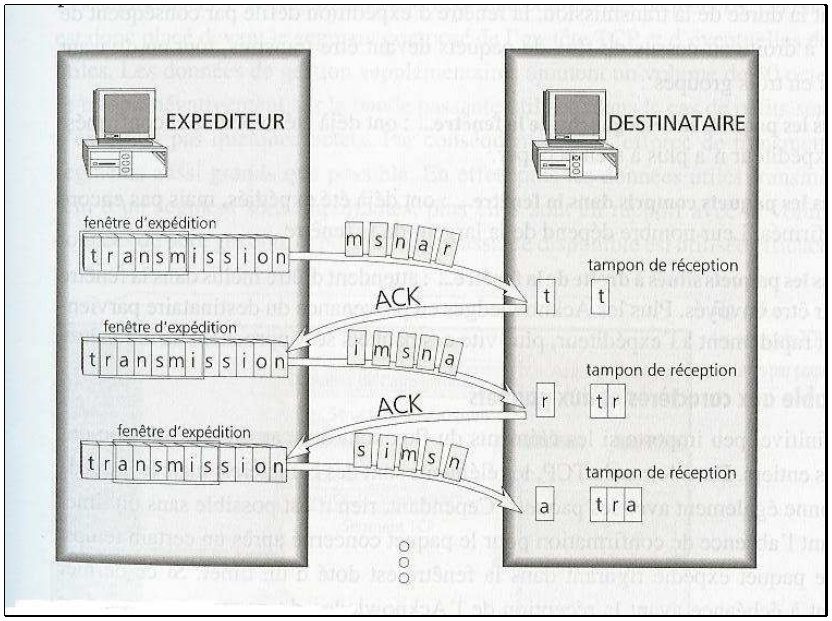


Figure 4.4: Fenêtre glissante

TCP se sert de la technique de la **fenêtre glissante** (voir figure 4.4). Comme nous l'avons déjà dit, le flux de données est numéroté octet par octet. Lors de l'ouverture de la connexion, le module TCP du récepteur envoie à l'émetteur un acquittement qui fixe un intervalle pour les numéros de séquence à partir du dernier segment correctement reçu. Cet intervalle est appelé **fenêtre** : il indique à l'émetteur le nombre d'octets qu'il peut transmettre sans avoir besoin d'une autorisation. Voici les caractéristiques du mécanisme de contrôle de flux :

- Les octets qui précèdent la limite inférieure de la fenêtre ont été envoyés et acquittés.
- Les octets se trouvant dans l'intervalle défini par la fenêtre peuvent être envoyés. S'ils ont déjà été envoyés, leur acquittement n'est pas encore parvenu.
- Les octets qui dépassent la limite supérieure de la fenêtre n'ont pas encore été envoyés et ne pourront l'être que lorsqu'ils passeront dans l'intervalle de la fenêtre.

La limite inférieure de la fenêtre est l'octet non acquitté dont le numéro est le plus petit. La fenêtre progresse, on dit de gauche à droite, dès qu'elle reçoit un acquittement pour les données expédiées. Le paquet qui contient l'acquittement contient aussi des informations concernant l'amplitude de la fenêtre que l'émetteur peut se permettre.

L'amplitude de la fenêtre dépend de la quantité d'espace libre encore disponible dans le tampon de données du récepteur. Lorsque le récepteur est trop sollicité il renvoie une taille de fenêtre réduite. Dans les cas extrêmes cette taille peut se réduire à 1. Lorsque le module TCP du récepteur renvoie une taille de fenêtre égale à zéro, cela indique à l'expéditeur que ses tampons sont pleins et qu'il ne faut plus lui envoyer aucune donnée.

- **Connexion**. Avant de pouvoir envoyer des données à l'aide de TCP, les processus doivent d'abord établir une connexion. UNe connexion s'établit entre le numéro de port de l'émetteur et celui du récepteur. Une **extrémité** (*en point* en anglais) est définie par un couple : l'adresse IP et le numéro de port. On établit une **connexion** entre deux extrémités, autrement dit avace un quadruplet :

  (adresse IP 1, numéro de port 1, adresse IP 2, numéro de port 2)

4.5.2 L'en-tête TCP

La figure suivante montre l'agencement de l'en-tête TCP, telle que définie par la [RFC 793] ("*Transmission Control Protocol*" écrite par Postel en 1981) :

<table>
<tr><td colspan="8">Port source (16 bits)</td><td>Destination (16 bits)</td></tr>
<tr><td colspan="9">Numéro de séquence (32 bits)</td></tr>
<tr><td colspan="9">Numéro d'acquittement (32 bits)</td></tr>
<tr><td>Offset
données
(4 bits)</td><td>Réservé
(6 bits)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td>Fenêtre (16 bits)</td></tr>
<tr><td colspan="8">Somme de contrôle (16 bits)</td><td>Pointeur Urgent (16 bits)</td></tr>
<tr><td colspan="7">Options</td><td colspan="2">Octets de remplissage</td></tr>
</table>

Analysons les champs un à un :

- `Port source` (*Source Port*) : champ de 16 bits qui identifie l'utilisateur TCP local (il s'agit généralement d'une application de la couche supérieure).

- `Port de destination` (*Destination Port*) : champ de 16 bits qui identifie l'utilisateur TCP de la machine distante.

  Les deux premiers champs de l'en-tête TCP servent à identifier l'expéditeur et le destinataire. Étant donné que leurs adresses IP figurent déjà dans l'en-tête du paquet IP, seules les informations de port sont retenues. Les ports et les adresses IP correspondantes permettent d'obtenir les deux extrémités de la connexion, qui fournissent ensemble un critère d'identification univoque permettant de transmettre les données reçues à l'application requise.

- `Numéro de séquence` (*Sequence Number*) : nombre indiquant la position du bloc en cours dans l'ensemble du message. Conserve ainsi le classement du segment, dans l'éventualité où les segments n'arriveraient pas dans l'ordre (attendu) de leur transmission.

`Numéro de séquence` indique au destinataire le début des données contenues dans le paquet TCP. Pour le premier segment TCP, on indique 0 mais pas toujours. Si l'on transmet 300 octets dans ce segment, le segment suivant débutera à l'octet 301 du flux de caractères et son champ `numéro de séquence` indiquera 301.

Si le drapeau SYN (voir ci-dessous) vaut 1, ce champ définit le numéro de séquence initial (**ISN**) correspondant à la session en cours.

- `Numéro d'acquittement` (*Acknowledgement Number*) : l'expéditeur utilise cette information pour préciser à son correspondant le numéro de séquence qu'il attend dans le segment TCP suivant. Ceci donne une idée des confirmations que l'expéditeur a déjà recues.

  Le destinataire doit tenir compte de ce champ lorsqu'il reçoit un en-tête TCP contenant le drapeau `ACK` (voir ci-dessous). Celui-ci indique que le segment contient, outre les données éventuellement présentes, une confirmation provenant du correspondant de la réception de données. Le champ `numéro d'acquittement` contient le numéro du prochain octet attendu en provenance de l'interlocuteur. Pendant qu'il sert au transport des données du point A vers le point B, le segment signale à B les données déjà reçues par A. Bien que circulant de A à B au sein du segment, cette partie du paquet concerne donc la communication entre B et A. C'est assez astucieux car une confirmation peut ainsi être transmise sans occasionner pour autant l'envoi d'un segment TCP comprenant uniquement un en-tête TCP, comme le montre la figure 4.5 ([TJ–97], p.146). En anglais ce procédé est appelé *piggybacking*.

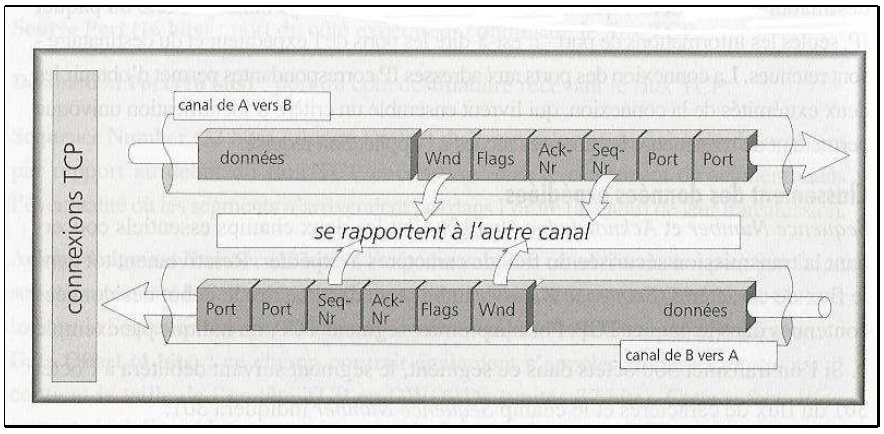


Figure 4.5: Piggybacking

Un acquittement peut servir à acquitter plusieurs segments de message TCP. L'acquittement signifie que le module TCP a reçu les données, mais il ne garantit pas que les données sont parvenues à l'application.

- `Décalage de données` (*Data Offset*) : il pourrait s'appeler longueur de l'en-tête (*Header Length*) car il contient le nombre de mots de 32 bits qui se trouvent dans l'en-tête TCP. Ce champ est utilisé pour identifier le début du champ des données. Cette information est nécessaire parce que le champ options a une longueur variable. Lorsque le champ options est vide, la valeur de décalage des données est cinq (soit 20 octets).

- `Réservé` : champ de 6 bits réservé pour des utilisations ultérieures. Les 6 bits doivent être mis à 0.

- Drapeau `Urg` : s'il est activé (valeur égale à 1), il indique que le segment contient des données urgentes et donc que le champ de pointeur urgent est significatif.

- Drapeau `Ack` : s'il est activé, il indique que le segment contient une confirmation et donc que le champ `Acquittement` est significatif.

- Drapeau `Psh` : s'il est activé, il indique qu'il faut faire suivre immédiatement les données reçues (par exemple appel à la fonction `push()` sous UNIX).

- Drapeau `Rst` (pour *ReSeT*) : s'il est activé, il indique que la connexion doit être réinitialisée pour cause d'erreurs irrécupérables. À la réception du drapeau `Rst`, le récepteur doit immédiatement terminer la connexion.

- Drapeau `Syn` : s'il est activé, il indique que les numéros de séquence doivent être synchronisés. Ce drapeau est utilisé lors de l'établissement d'une connexion.

- Drapeau `Fin` : s'il est activé, il indique que l'émetteur n'a plus de données à envoyer. C'est l'équivalent d'un marqueur de fin de transmission.

- `Longueur de la fenêtre` (*window*) : nombre de blocs de données que la machine réceptrice est prête à recevoir, compte tenu de l'octet affiché dans le champ `numéro d'acquittement`.

- `Somme de contrôle` (*checksum*) : calculée en prenant le complément à un en 16 bits du complément à un de la somme des mots de 16 bits dans l'en-tête et le texte réunis. Ce champ est le résultat du calcul de la somme de contrôle basée sur la taille du segment entier, comprenant un pseudo en-tête de 96 bits préfixé à l'en-tête TCP lors du calcul. Le **pseudo en-tête** de 96 bits contient l'adresse source, l'adresse de destination, l'identificateur de protocole et la longueur du segment. Ce sont les paramètres qui sont passés à IP lorsqu'une instruction d'envoi est passée, et ceux qui sont lus par IP lorsqu'une livraison est tentée.

- `Pointeur Urgent` (*Urgent Pointer*) : ce champ n'a de sens que si le drapeau `Urg` est activé. Il indique la portion des données du message qui est urgente, en en spécifiant le décalage à partir du numéro de séquence dans l'en-tête.

  Aucune action spécifique n'est entreprise par TCP en ce qui concerne les données urgentes. C'est l'application qui s'en charge éventuellement. Peut servir pour mettre en place les fonctions `send()` et `recv()`.

- `Options` : chaque option consiste en un type d'option (1 octet), suivi éventuellement du nombre d'octets dans l'option (1 octet) puis des données (le nombre d'octets précédent moins deux). Dans la RFC 793, seules trois types sont définis pour TCP :

  - 0 : fin de la liste d'option (*End-of-Options*), option constituée d'un seul octet. Cette option n'est pas nécessaire lorsque la fin des options coïncide avec la fin de l'en-tête TCP.
  - 1 : pas d'opération (*No-Operation*), option constituée d'un seul octet. Ce type sert à aligner le commencement de l'option suivante sur le début d'un mot, mais les émetteurs TCP ne prennent pas toujours cette précaution.

  - 2 : taille de segment maximale (*MMS* pour *Maximum Segment Size*) de longueur 4. Ce champ sert à spécifier la taille maximale de tampon qu'une implémentation TCP réceptrice peut accepter. Comme TCP utilise des zones de données à longueur variable, il peut arriver qu'une machine émettrice crée un segment plus long que ce que peut gérer le logiciel récepteur.

- `Remplissage` : rempli pour être sûr que la longueur de l'en-tête est un multiple de 32 bits.

4.5.3 Procédure de négociation d'ouverture d'une session TCP en trois étapes

On ouvre une session TCP à l'aide d'une procédure de négociation en trois temps (*Three-way handshake* en anglais) :

- Le client envoie au serveur un segment TCP d'ouverture de session avec les valeurs suivantes pour les drapeaux : SYN = 1 et ACK = 0. Ces valeurs caratérisent cette première étape. Le numéro de séquence d'ouverture est SEQ = X, appelé **ISS** pour *Initial Sender Sequence*. On a éventuellement Z octets de données mais en général Z = 0. Le numéro d'acquittement est nul : #ACK = 0.

- Le serveur acquitte ce segment d'ouverture par un segment TCP dont les valeurs des drapeaux sont les suivantes : SYN = 1 et ACK = 1. Ces valeurs caractérisent cette seconde étape. Le numéro de séquence d'ouverture de la part du serveur est SEQ = Y, appelé **IRS** pour *Initial Receiver Sequence*. Le numéro d'acquittement doit être #ACK = X + 1 + Z.

- Dans l'étape trois, le client acquitte le numéro de séquence que lui a envoyé le serveur avec les valeurs suivantes des drapeaux : SYN = 0 et ACK = 1. On doit avoir SEQ = X + 1 + Z et #ACK = Y + 1.

4.5.4 Transmission des données et fin de session

Après l'établissement de la connexion TCP, le drapeau ACK des segments doit être positionné pour indiquer que le champ numéro d'acquittement est valide. La transmission des données pourra se faire dans les deux sens, jusqu'à la fermeture de la session.

Au moment de la fermeture de la session, une extrémité envoie le drapeau FIN. Mais la fermeture de la connexion TCP n'est réalisée que lorsque l'autre extrémité envoie également le drapeau FIN pour accepter la fermeture. Cette façon de faire permet d'éviter les pertes de données que pourrait occasionner une fermeture unilatérale.