ANNEXE

TRAME ETHERNET

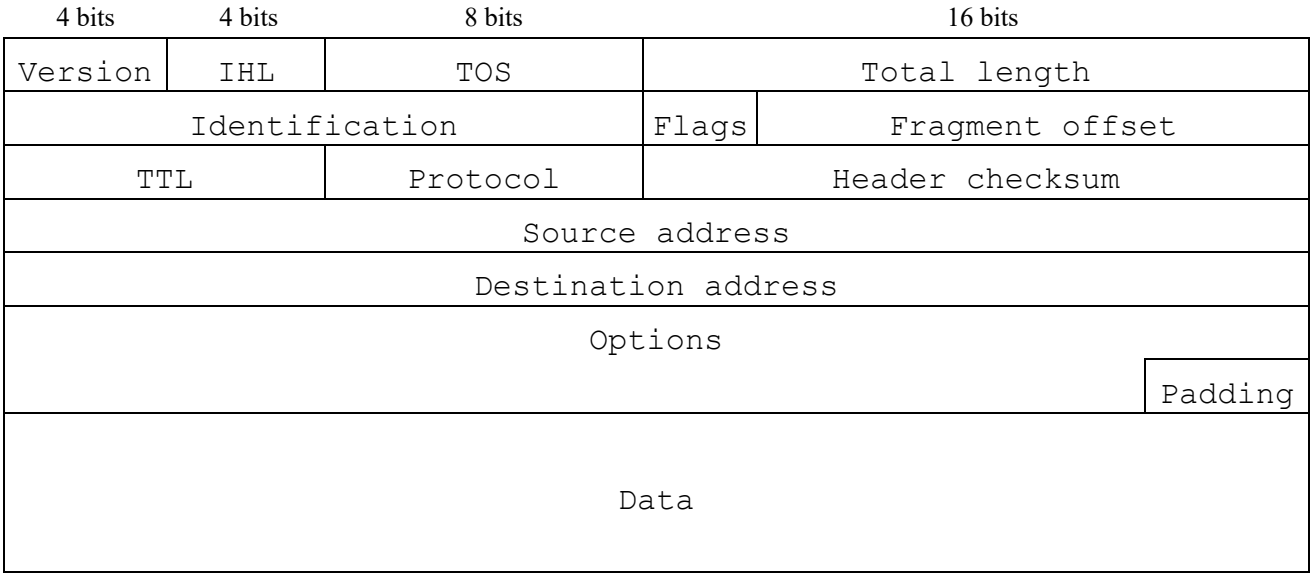
8 octets	6 octets	6 octets	2 octets	46-1500 octets	4 octets
`Preamble`	`Destination address`	`Source address`	`Type`	`Data`	`CRC`

- `Preamble` est un préambule qui détermine le début d'une trame (de la forme 101010…101011) ;
- `Destination address` est l'adresse physique (MAC) du destinataire de la trame ;
- `Source address` est l'adresse physique (MAC) de l'expéditeur de la trame ;
- `Type` indique le protocole de niveau supérieur encapsulé dans le champ `Data` de la trame. Quelques exemples :

Type	Utilisation
0800	DoD Internet (Datagramme IP)
0805	X.25 niveau 3
0806	ARP
8035	RARP
8098	Appletalk
…	…

- `Data` contient les données brutes de la trame à passer au protocole déterminé par le champ Type ;
- `CRC` est le code détecteur d'erreur (total de contrôle) de la trame permettant d'assurer son intégrité.

DATAGRAMME IP



- `Version` indique le format de l'en-tête. Ce champ sert à l'identification de la version courante du protocole. La version décrite ici (et aujourd'hui utilisée) porte le n°4 ;
- `IHL` (*IP Header Length*) est la longueur de l'en-tête IP exprimée en mots de 32 bits ;
- `TOS` (*Type Of Service*) définit le type de service à appliquer au paquet en fonction de certains paramètres comme le délai de transit, la sécurité. Il est peu utilisé et sa valeur est généralement égale à 0 ;
- `Total Length` est la longueur totale du datagramme, exprimée en octets. En pratique, il est rare qu'un datagramme IP fasse plus de 1500 octets ;
- `Identification` sert en cas de fragmentation/réassemblage du datagramme. Ce champ permet alors à l'entité réceptrice de reconnaître les fragments issus d'un même datagramme initial et qui doivent donc faire l'objet d'un réassemblage ;
- `Flags` (3 bits) est utilisé par la fragmentation. Il est composé (de gauche à droite) :
  - d'un bit réservé : mis à 0 ;
  - de l'indicateur DF (*Don't Fragment*) : mis à 1 par l'émetteur pour interdire la fragmentation ;
  - de l'indicateur MF (*More Fragment*) : mis à 1 pour signifier que le fragment courant est suivi d'un autre fragment ;
- `Fragment offset` (13 bits) donne la position relative du fragment dans le datagramme initial, le déplacement étant donné en unités de 64 bits ;
- `TTL` (*Time To Live*) donne une indication de la limite supérieure du temps de vie d'un datagramme ;
- `Protocol` indique le protocole (de niveau supérieur) utilisé pour le champ de données du datagramme. Quelques exemples :

Code (déc)	Abréviation	Nom du protocole	Référence
1	ICMP	Internet Control Message Protocol	[RFC792]
2	IGMP	Internet Group Management Protocol	[RFC1112]
6	TCP	Transmission Control Protocol	[RFC793]
8	EGP	Exterior Gateway Protocol	[RFC888]
9	IGP	any private Interior Gateway Protocol	
17	UDP	User Datagram Protocol	[RFC768]
36	XTP	XTP	
46	RSVP	Reservation Protocol	
…	…	…	

- `Header Checksum` est une zone de contrôle d'erreur portant uniquement sur l'en-tête du datagramme ;
- `Source Address` est l'adresse IP de la source du datagramme ;
- `Destination Address` est l'adresse IP de destination du datagramme ;
- `Options` est de longueur variable. Il sert à des fonctions de contrôle utiles dans certaines situations. Il est constitué d'une succession d'options élémentaires, également de longueurs variables. Les options sont codées sur le principe TLV (Type, Longueur, Valeur). La longueur indique la taille complète de l'option en octets. Quelques options possibles :

Type (déc.)	Option	Objet
0	*End of Options List* (EOOL)	Utilisée si la fin des options ne coïncide pas avec la fin de l'en-tête.
1	*No Operation* (NOP)	Pour aligner le début de l'option suivante sur 32 bits.
7	*Record Route* (RR)	Permet d'enregistrer la route d'un datagramme (l'adresse IP de chaque passerelle traversée).
68	*Time Stamp* (TS)	Enregistrement de l'heure de chaque passage de passerelle.
131	*Loose Source Route* (LSR)	Permet à la source d'indiquer les adresses IP des passerelles par lesquels le datagramme doit passer.
137	*Strict Source Route* (SSR)	Idem LSR, si ce n'est que le chemin du datagramme ne peut traverser d'autres passerelles que celles indiquées.

- `Padding` est de longueur variable : il permet d'aligner l'en-tête sur 32 bits.

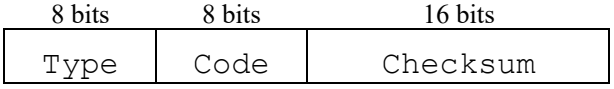
PAQUET ARP / RARP

<table>
<tr><th colspan="2">16 bits</th><th>16 bits</th></tr>
<tr><td colspan="2">Hardware</td><td>Protocol</td></tr>
<tr><td>Hlen</td><td>Plen</td><td>Operation</td></tr>
<tr><td colspan="3">Sender HA (bytes 0–3)</td></tr>
<tr><td colspan="2">Sender HA (bytes 4–5)</td><td>Sender IA (bytes 0–1)</td></tr>
<tr><td colspan="2">Sender IA (bytes 2–3)</td><td>Target HA (bytes 0–1)</td></tr>
<tr><td colspan="3">Target HA (bytes 2–5)</td></tr>
<tr><td colspan="3">Target IA (bytes 0–3)</td></tr>
</table>

- `Hardware` définit le type d'interface pour laquelle l'émetteur cherche une réponse ;
  Exemple : 0x0001 pour une interface Ethernet ;
- `Protocol` définit le type de protocole pour lequel une requête a été émise ;
  Exemple : 0x0800 pour une adresse logique IP ;
- `Hlen` définit la taille de l'adresse physique (Ethernet) en octets ;
- `Plen` définit la taille de l'adresse au niveau protocolaire (IP) ;
- `Operation` décrit le type d'opération à effectuer par le récepteur ;
  Exemple : 0x0001 pour une requête ARP ; (0x0003 pour une requête RARP)
  0x0002 pour une réponse ARP ; (0x0004 pour une réponse RARP)
- `Sender HA` définit l'adresse physique (Ethernet) de l'émetteur ;
- `Sender IA` définit l'adresse de niveau protocolaire (IP) demandé de l'émetteur ;
- `Target HA` définit l'adresse physique (Ethernet) du récepteur ;
- `Target IA` définit l'adresse de niveau protocolaire (IP) demandé du récepteur.

MESSAGE ICMP

Les messages ICMP sont encapsulés dans des datagrammes IP. Ils ont tous en commun le même format pour le premier mot de 32 bits :



Type	Message	Objet
0	Echo Reply	Réponse en écho
3	Destination Unreachable	Destination inaccessible
4	Source Quench	Interruption de la source
5	Redirect	Redirection, changement de route
8	Echo	Demande d'écho
11	Time Exceeded	Temps de vie d'un datagramme dépassé
12	Parameter Problem	Datagramme mal formé
13	Timestamp	Demande de date d'estampillage
14	Timestamp Reply	Réponse à une demande d'estampillage
15	Information Request	Demande d'information
16	Information Reply	Réponse à une demande d'information
17	Address Mask Request	Demande de masque d'adresse
18	Address Mask Reply	Réponse à une demande de masque d'adresse

A titre d'exemple, `Echo` et `Echo Reply` sont utilisés pour vérifier l'état d'activité d'une machine. Une machine source envoie alors un message `Echo` à la machine destinataire dont elle veut vérifier l'activité. Celle-ci doit alors lui répondre par un message `Echo Reply`. L'adresse source dans un Echo (`Type` = 8) sera l'adresse destinataire du `Echo Reply` (`Type` = 0). Pour constituer un `Echo Reply`, les adresses source et destination sont donc simplement inversées. Les données reçues dans un `Echo` doivent être retournées dans le `Echo Reply`. Deux champs du message, `Identifier` et `Sequence Number`, sont utilisés par l'émetteur de l'`Echo` pour mettre en correspondance les réponses avec les requêtes. Par exemple, l'identificateur peut correspondre à un port TCP ou UDP pour identifier une session et le numéro de séquence être incrémenté pour chaque requête d'écho émise. Le répondeur retourne les mêmes valeurs dans sa réponse.