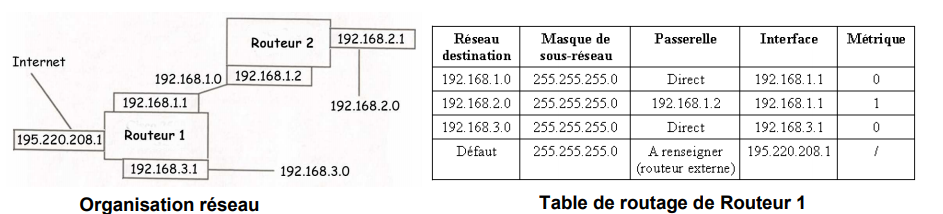
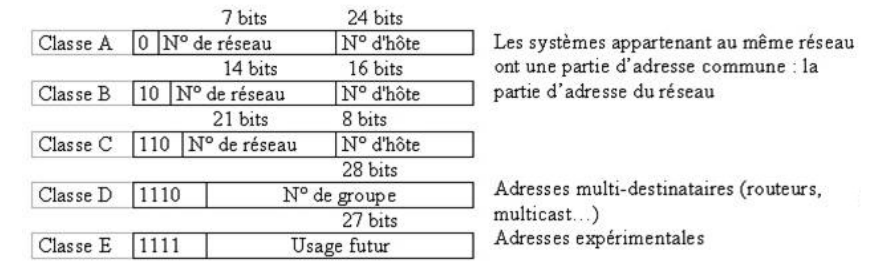
**Réseaux**

Couche réseau

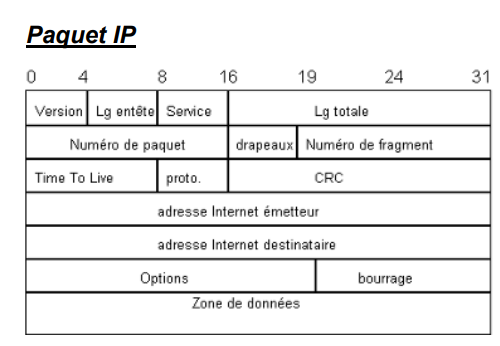




**Classe A :** nombre restreint de très gros réseaux (27 réseaux de 224 -2 machines).

**Classe C :** nombre potentiellement très grand de petits réseaux (221 réseaux de 28 -2 machines).

**Exemple :** Adresse de classe B avec 4 sous-réseaux (identifiés sur 2 bits, soit 00, 01,10 et 11) : 11111111.11111111.11000000.00000000 => Soit en notation décimale : 255.255.192.0



**Version** : numéro de version, habituellement 4

**Lg en-tête** : longueur de l’en-tête, en multiple de 4 octets (pour IPv4, vaut 5 si champs Options vide)

**Type de service** : qualité de service souhaitée pour l’acheminement (priorité …).

**Longueur totale** : en-tête + données

**Numéro de paquet** (identification) : permet de reconstituer les différents fragments.

**Drapeaux** : indicateurs pour la fragmentation

**Numéro** **de** **fragment** (décalage) : 0 pour le premier fragment, 1 pour le second …

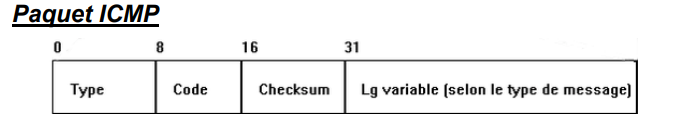
**Durée** **de** **vie** (TTL) : décrémenté de 1 à chaque fois que le paquet traverse un routeur ; à 0, le paquet est détruit.

**Protocole** : ICMP, TCP, UDP …

**Options** : de taille variable (éventuellement plusieurs options).

**Bourrage** : octets à 0 pour s’assurer que la taille de l’en-tête soit un multiple de 4 octet

**ICMP** est un protocole de la couche réseau (3) qui contrôle les erreurs de transmission (pas leur correction).

Ce protocole assure donc plusieurs fonctions : - Eprouver la connectivité du réseau - Optimiser le réseau, à une certaine échelle - Gérer les erreurs de transmission (réseau / machine / port / … inaccessible …)

* **Type 0, Code 0 :** Réponse d’ECHO (echo-reply) ; réponse au message de type 8.
* **Type 3, Codes 0 à 15 :** Le réseau ou destinataire inaccessible, accès interdit, service indisponible.
  + **Code 0 :** le réseau n'est pas accessible.
  + **Code 1 :** la machine n'est pas accessible.
  + **Code 2 :** le protocole n'est pas accessible.
  + **Code 3 :** le port n'est pas accessible.
* **Type 4, Code 0 :** le routeur ou le destinataire est saturé (vitesse de transfert trop importante / buffer de réception plein) et demande à l’émetteur de ralentir le rythme des envois.
* **Type 5, Codes 0 à 3 :** Le routeur constate que la route empruntée par un ordinateur ou un réseau auquel est connecté ce routeur n’est pas optimale et envoie l’adresse du routeur à rajouter dans la table de routage de l’ordinateur ou des ordinateurs du réseau.
* **Type 8, Code 0 :** demande d’ECHO (echo-request) ; demande de renvoi d’informations, avec la commande PING par exemple.
* **Type 11, code 0 & 1** : durée de vie d’un datagramme ou temps de réassemblage de ses fragments dépassé.
* **Type 12, Code 0** : en-tête erroné.
* **Type 13, Code 0 et Type 14, Code 0 :** demande d’heure système et réponse.
* **Type 15, Code 0 et Type 16, Code 0 :** demande de son adresse IP et réponse.
* **Type 17, Code 0 et Type 18, Code 0 :** demande de son masque de sous-reseaux

**NAT (Network Address Translation)**

NAT est une solution pour faire face à la pénurie d'adresses IP ; elle présente aussi un intérêt en matière de sécurité

Il existe 3 plages d'adresses privées qui ne doivent jamais circuler sur internet

* 10.0.0.0 à 10.255.255.255 (classe A)
* 172.16.0.0 à 172.31.255.255 (classe B)
* 192.168.0.0 à 192.168.255.255 (classe C)

NAT fonctionne en établissant une correspondance entre des adresses IP publiques et les adresses IP privées.

Problème avec NAT : si trop de machines veulent se connecter en même temps vers l'extérieur, il risque de manquer d'adresses IP disponibles

Donc on utilise PAT

**PAT (Port Address Translation)**

**Avec PAT, ce n'est pas seulement l'adresse privée qui est traduite mais aussi le port source de la connexion.** La table de traduction ne contient plus des paires @IPprivée - @IPpublique

mais des quadruplets @IPprivée, PSprivé - @IPpublique, PSpublic.

**Le protocole ARP permet d'obtenir une adresse MAC à partir d'une adresse IP.**

**Avec RARP, c’est l’inverse.**

**Exemple avec A et B situées sur le même réseau local :**

* A partir de son adresse IP et de son masque de sous-réseau, A peut connaitre l'adresse du réseau local. A partir de l'adresse IP de B, elle constate que B appartient au même réseau local.
* Pour connaitre l'adresse physique de B, A envoie alors une requête ARP en broadcast éthernet (FF.FF.FF.FF.FF.FF), c'est à dire à destination de toutes les machines situées sur le réseau éthernet (supportant le réseau logique en question) : "Qui a la machine d'adresse IP 134.59.59.5 ?" (la requête ARP est encapsulée dans une trame éthernet : elle constitue les données de cette trame).
* Toutes les machines du réseau vont donc recevoir la requête mais seule B va y répondre. Réponse ARP (après avoir récupéré l'adresse MAC de A) : "C'est moi, mon adresse MAC est 00:10:5A:AF:AB:E9".
* A peut maintenant envoyer son premier paquet de commande Telnet à destination de B.
* A l'issue de cette communication, A aura mémorisé l'adresse MAC de B dans une table ARP (cache ARP).

**A et B ne sont pas situées sur le même réseau local**

* A constate que B n'est pas sur son réseau local
* A envoie alors une requête ARP afin de récupérer l'adresse physique de la passerelle du réseau.
* A envoie ensuite son premier paquet de connexion Telnet dont l'adresse de destination éthernet (MAC) est celle de sa passerelle et dont l'adresse de destination IP est celle de B
* Ainsi de suite ...

Couche transport

A ce niveau, on décide de l’établissement ou non d’une connexion entre l’émetteur et le destinataire

**Protocole UDP/Mode datagramme (non connecté) :**

* Pas de connexion
* Envoi sans savoir si le destinataire est prêt
* Chaque paquet emprunte un chemin indépendant (la détermination du chemin optimal se fait pour chaque paquet)
* Pas de garantie d’ordre (une remise en ordre des paquets doit être faite à l’arrivée)
* Aucune garantie sur la bonne livraison des paquets (pas d’acquittement)
* Pas contrôle de flux
* Avantage : sa rapidité et le fait qu’il permet de faire de la multidiffusion.

**Protocoles couches hautes basés sur UDP :** DNS, DHCP, SNMP, NTP, RTP (protocole de transport de flux audio/vidéo en temps réel : voix, visioconf, streaming)

**Protocole TCP/Mode circuit virtuel (connecté) :**

* Principe du handshake pour établir (flags SYN, SYN/ACK, ACK) et terminer (flags FIN, ACK, FIN, ACK) la connexion
* Tous les paquets empruntent le même chemin réseau (un paquet de routage sans données est émis vers le destinataire (il détermine un circuit virtuel qu’emprunteront tous les paquets de la connexion)
* Contrôle des pertes par numéros de séquence et d’acquittement
* Contrôle de flux (taille de fenêtre)
* Liaison en point à point uniquement.

**Protocoles couches hautes basés sur TCP :** la plupart, dont HTTP, FTP, Telnet, SMTP, POP, IMAP.

Couche session

La couche session fournit les moyens d’organiser les dialogues et l’échange de données. Par exemple, qui parle (dialogue simplex, half/full-duplex ?), comment se passe la gestion des accusés de réception.

La couche session est aussi associée à la notion de synchronisation : en cas d’erreur, pouvoir par exemple revenir à un état antérieur stable et connu de tous les communicants (point de reprise)

Couche présentation

Un réseau est un environnement hétérogène où les données peuvent être codées de différentes manières. Exemple : le codage des entiers Big endian/Little endian dépend du processeur, le codage des caractères dépend du système d'exploitation, la représentation de tableaux ... pourra dépendre du langage.

**ASN englobe les notions de syntaxe abstraite et de syntaxe de transfert**

* **Syntaxe réelle** : Représentation des données applicatives sur un hôte donné : dépend de "critères réels" : langage, système d'exploitation, processeur. Ce sont les données telles qu'elles sont codées sur une machine donnée (niveau physique/machine).
* **Syntaxe abstraite :** Représentation qui décrit le contenu sémantique mais qui est indépendante de tout critère réel. Données que l'on cherche à échanger (niveau conceptuel).
* **Syntaxe de transfert** : Représentation la plus approprié lors d'un échange entre deux parties applicatives. Souvent négociée au début d’une connexion (à une syntaxe abstraite, peut correspondre plusieurs syntaxes de transfert). Données telles qu'elles circulent sur le réseau (niveau physique/réseau).

**Syntaxe abstraite ASN :**

* Composants de type simple (primitif) : BOOLEAN, INTEGER, REAL, UTCTIME, GENERALIZEDTIME, BIT STRING, OCTET STRING, etc.
* Sous-types : par exemple, INTEGER (3|4|6|10), INTEGER (4 .. 100), INTEGER (4 < .. < 100), etc.
* Types construits : CHOICE, SEQUENCE, SEQUENCE OF, SET, SET OF
* Si le composant est optionnel : OPTIONAL
* Avec une valeur par défaut : DEFAULT
* Etiquetage : chaque type a une étiquette par défaut (par exemple INTEGER a l'étiquette 2), étiquette que l'on peut parfois être obligé de redéfinir pour lever les ambiguïtés.
  + Etiquetage implicite (mot-clé IMPLICIT) si la nouvelle étiquette attribuée remplace l'étiquette courante
  + Explicite (EXPLICIT) si elle s'y ajoute (EXPLICIT par défaut).
* Exemple :
  + Quantity ::= CHOICE {
  + units [0] INTEGER,
  + millimeters [1] INTEGER,
  + milligrams [2] INTEGER }
* Possibilité de définir des Macros, des Modules (sortes de meta-structures)

**Syntaxe de** **transfert**

Exemple BER (Basic Encoding Rules) :Triplet de 3 champs T L V ou chaque champs est un multiple de 8 bits.

* **T =Tag :** valeur d'étiquette du type (bits 1-5) + indication type primitif/construit (bit 6 : resp 0/1) + classe Universal/Application/Context-specific/Private (bit 7-8 : resp 00/10/01/11).
* **L=Length** (longueur de la valeur en octets). Si elle tient sur un octet, bit le plus à gauche à 0 et les 7 bits suivants indiquent cette longueur, sinon bit de gauche 1 et les 7 bits suivants indiquent le nombre d'octets pour coder cette longueur et si cette dernière n'est pas définie, champ L à 10000000 et on utilise un délimiteur de fin (2 octets à 0) après le champ Value.
* **V= Value (valeur).** Si la taille de codage n'est pas un multiple de 8 (exemple : BIT STRING ‘111100001’), on utilise des bits de bourrage.

Couche application

Elle comporte des protocoles de communication de haut niveau. Cela correspond à des services réseaux directement utilisables par les applications.

Il ne faut pas confondre protocole et application :

* Une application Web (écrite en PHP …) utilise les services du protocole HTTP pour permettre la communication entre un client et un serveur Web.
* Une même application peut supporter plusieurs protocoles (ex : POP, IMAP et SMTP pour les applications de messagerie).

**Quelques exemples de protocoles de la couche application**

* **HTTP** (Hypertext Transport Protocol) : protocole du Web ; communément utilisé pour faire communiquer un navigateur et un serveur Web.
* **FTP** : (File Transfert Protocol) : protocole de manipulation de fichiers distants : création, modification, suppression … Alternatives : NFS (réseaux locaux UNIX/Linux), SMB (réseaux locaux, interopérabilité Windows/Unix) …
* **DHCP** (Dynamic Host Configuration Protocol) : protocole prévu pour distribuer automatiquement des adresses IP aux hôtes qui en font la demande.
* **Telnet** (TELetypewriter Network Protocol) : système permettant à un terminal virtuel d’ouvrir une session et d’exécuter des commandes à distance.
* **SMTP** (Simple Mail Transfert Protocol) : service d’envoi de courrier électronique ; pour la réception : POP, IMAP …
* **POP3** (Post Office Protocol Version 3) : protocole qui permet au client de relever à distance le courrier stocké dans sa boite aux lettres
* **IMAP4** (Interactive Mail Access Protocol Version 4) : protocole de réception également. Il devrait prendre la place de POP3 ; il propose des fonctionnalités plus fines que POP3 qui ne permet de traiter les messages qu’une fois rapatriés localement ; il est déjà implémenté par Free
* **DNS** (Domain Name Server) : Correspondance entre adresses IP et nom(s) de domaine dans des bases de données réparties dans le monde.
* **SNMP** (Simple Network Management Protocol) : protocole d’administration de réseau (interrogation, configuration des équipements).
* **SSH (Secure Shell)** : protocole sécurisé permettant d’ouvrir une connexion sécurisée sur un shell distant : il impose un échange de clés de chiffrement au début de la connexion, tous les segments TCP envoyés sont ensuite chiffrés.
* **NNTP** (Network News Transfert Protocol) : il est utilisé pour les forums de discussion Usenet.
* **NTP** (Network Time Protocol) : protocole horaire en réseau qui permet de synchroniser des horloges sur celle d’un serveur de temps.

**Les méthodes de HTTP :**

* **GET** : demander une ressource au serveur
* **HEAD** : obtenir des informations sur une ressource du serveur
* **POST** : envoyer à une ressource du serveur des données pour traitement ou une ressource subalterne pour création
* **PUT** : ajouter ou remplacer une ressource sur le serveur
* **DELETE** : supprimer une ressource du serveur
* **OPTIONS** : obtenir des informations de communication …
* **TRACE** : demander au serveur de renvoyer dans le corps de sa réponse les en-têtes qu’il a reçu dans sa requête
* **CONNECT** : demander à un serveur proxy de se connecter au serveur désiré (tunneling). Utile pour les connexions chiffrées de type HTTPS (HTTP +SSL/TLS)

Exemples :

**PUT /mon\_repertoire/mon\_fichier.txt HTTP/1.1**

Host : mon\_serveur.com

Content-Type : text/plain

Content-Length: 44 …

Bonjour, voici le contenu d'un fichier texte

**GET /mon\_repertoire/mon\_fichier.xml HTTP/1.1**

Host: mon\_serveur.com

Content-Type: application/xml

**DNS (Domain Name System)**

DNS permet de faire correspondre des adresses IP à des noms de domaines.

A une adresse IP peut correspondre plusieurs noms de domaine, un nom principal et plusieurs alias.