Protocoles TCP IP

Nous nous intéressons ici uniquement au protocole IPV4.

1.1 Organisations de l'adressage.

Chaque ordinateur du réseau Internet dispose d'une adresse IP codée sur 4 octets (32 bits) qui doit être unique au sein de l'internet. Plus précisément, chaque interface d'un matériel dispose d'une adresse IP particulière. En effet, un même routeur interconnectant plusieurs réseaux différents possède une adresse IP pour chaque interface de réseau. Une adresse IP est toujours représentée dans une *notation décimale pointée* constituée de 4 nombres (1 par octet) compris chacun entre 0 et 255 et séparés par un point. Ainsi `193.49.144.1` est l'adresse IP d'une des principales machines du réseau de l'université d'Angers.

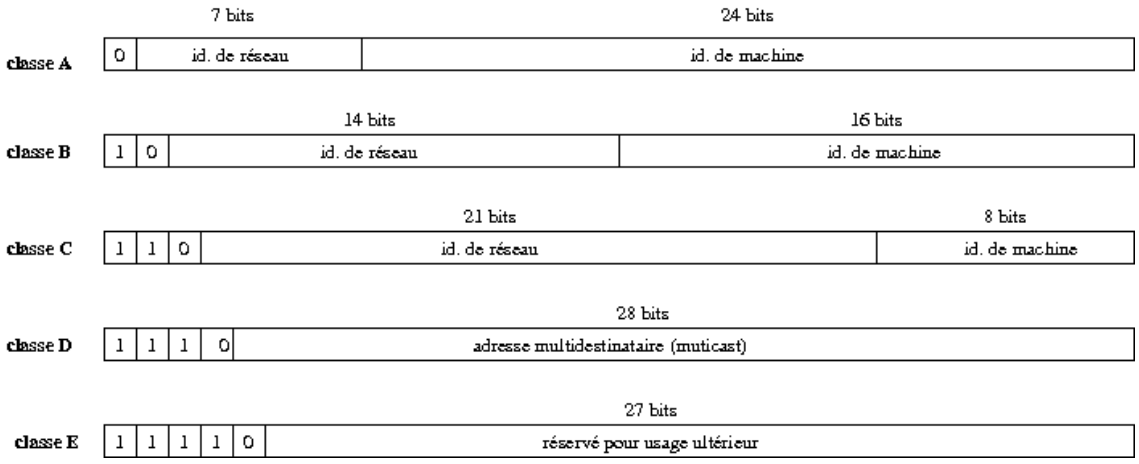


FIG. 1 – Les cinq classes d'adresses IP

Plus précisément, une adresse IP est constituée d'une paire *(identifiant de réseau, identifiant de machine)* (on utilise aussi *identifiant d'interface* au lieu d'identifiant de machine). L'architecture d'adressage d'origine de l'internet définissait cinq classes d'adresse (A, B, C, D

et E) selon la valeur du premier octet de l'adresse comme détaillé dans la figure 1. Le tableau ci-après donne l'espace d'adresses possibles pour chaque classe.

classe	adresses
A	`0.0.0.0` à `127.255.255.255`
B	`128.0.0.0` à `191.255.255.255`
C	`192.0.0.0` à `223.255.255.255`
D	`224.0.0.0` à `239.255.255.255`
E	`240.0.0.0` à `247.255.255.255`

Ainsi, les adresses de classe A étaient utilisées pour les très grands réseaux comportant plus de $2^{16} = 65536$ ordinateurs. Au niveau mondial, il ne pouvait donc exister plus de 127 tels réseaux. Les adresses de classe B étaient utilisées pour les réseaux ayant entre $2^8 = 256$ et $2^{16} = 65536$ ordinateurs, 14 bits définissent l'adresse du réseau et 16 bits celle d'une machine sur le réseau. Seules 256 machines sont possibles sur un réseau de classe C dont le nombre possible dépassaient les 2 millions ($= 2^{21}$).

Au lieu d'utiliser un adressage plat 1, 2, 3, ... la méthode retenue est plus efficace car elle permet une extraction rapide du numéro de réseau à l'intérieur d'une adresse IP ce qui facilitera le routage. Toutes les combinaisons mathématiquement possibles pour identifier un réseau ou une machine ne sont pas permises car certaines adresses ont des significations particulières.

- `0.0.0.0` est utilisée par une machine pour connaître sa propre adresse IP lors d'une processus d'amorçage par exemple
- `<id. de réseau nul>.<id. de machine>` est également utilisée pour désigner une machine sur son réseau lors d'un boot également
- `<id. de réseau>.<id. de machine nul>` n'est jamais affectée à une machine car elle permet de désigner le réseau lui-même
- `<id. de réseau>.<id. de machine avec tous ses bits à 1>` est une *adresse de diffusion* ou de *broadcasting*, c'est-à-dire qu'elle désigne toutes les machines du réseau concerné. Un datagramme adressé à cette adresse sera ainsi envoyé à toutes les machines du réseau.
- `255.255.255.255` est une adresse de diffusion locale car elle désigne toutes les machines du réseau auquel appartient l'ordinateur qui utilise cette adresse. L'avantage par rapport à l'adresse précédente est que l'émetteur n'est pas obligé de connaître l'adresse du réseau auquel il appartient.
- `127.X.Y.Z` est une adresse de rebouclage qui est utilisée pour permettre les communications inter-processus sur un même ordinateur ou réaliser des tests de logiciels car tout logiciel de communication recevant des données pour cette adresse les retourne simplement à l'émetteur.
- Les adresses de classe A de `10.0.0.0` à `10.255.255.255`, de classe B de `172.16.0.0` à `172.31.255.255` et de classe C de `192.168.0.0` à `192.168.255.255` sont réservées à la constitution de réseaux privés autrement appelés *intranet*.

Cependant, ce système d'adressage par classe s'est révélé inefficace devant la demande croissante d'adresses IP. En effet, une entreprise voulant numéroter 2000 machines ne peut se contenter d'un réseau de classe C. Par contre si on lui attribue un réseau de classe B, on perd plus de 60 000 adresses qui resteront inutilisées. On voit qu'aucune des classes n'est satisfaisante.

Devant la pénurie d'adresses de classes B le système CIDR (*Classless Inter Domain Routing* RFC 1518, 1519) est apparu en 1993 (aujourd'hui voir la RFC 4632 `http://www.ietf.org/rfc/rfc4632.txt`) Une telle adresse CIDR est de la forme `A.B.C.D/M` où M est un entier appelé *masque* et compris entre 13 et 27. Ce masque désigne le nombre de bits constituant l'identifiant

de réseaux dans l'adresse `A.B.C.D`. Par exemple, dans `172.20.41.7/16`, les 16 premiers bits de l'adresse désigne le réseau qui est donc `172.20.0.0` et l'identifiant de machine est `0.0.41.7`. Le masque, ici égal à 16, peut également être donné sous la forme décimale `255.255.0.0` correspondant à une adresse IP ayant ses 16 premiers bits à 1.

L'utilisation de tous les masques possibles de 0 à 32 bits permet de dimensionner des réseaux de manière plus précise et selon les possibilités suivantes par exemple.

masque	équivalent en classe C	nombre d'adresses
...	...	...
/27	1/8	32
/26	1/4	64
...	...	...
/14	1 024	262 144
/13	2 048	524 288
...	...	...

Étant donné une adresse IP donnée sous la forme CIDR comme `150.50.215.200/21` on retrouve les identifiants de réseau et de machines en effectuant un `et logique` entre l'adresse complète et le masque tous les deux mis sous forme binaire comme décrit ci-dessous.

adresse IP	`150.50.215.200`	`10010110.00110010.11010111.11001000`
masque (21)	`255.255.248.0`	`11111111.11111111.11111000.00000000`
adresse réseau	`150.50.208.0`	`10010110.00110010.11010` `000.00000000`
adresse machine	`0.0.7.200`	`00000000.00000000.00000` `111.11001000`

De cette manière si une société a besoin de 100 000 adresses on lui fournira une part de réseau de classe A en l'associant à un masque de 15 bits. Ainsi, elle disposera de $2^{(32-15)} = 131072$ (la plus petite puissance de 2 supérieure à 100 000) adresses. Dans l'ancien système, un réseau de classe B n'aurait pas été suffisant et un réseau de classe A, avec ses 16 millions d'adresses, aurait été largement surdimensionné.

Cette technique a permis d'agréger des réseaux par région géographique et fournisseurs d'accès. Ce système de *sur-réseau* permet ainsi de faire apparaître dans les tables de routage plusieurs réseaux sous le même identifiant. Évidemment, les réseaux agrégés doivent avoir des adresses contigües de manière à avoir des préfixes identiques. Par exemple, `193.127.32.0` et `193.127.33.0` peuvent être agrégés sous la notation `193.127.32.0 / 23`. On peut donc voir le réseau `193.127.32.0 / 23` comme un réseau de 512 machines, ou comme 2 réseaux de 256 machines chacun, car le 24^{e}bit permet de coder l'un ou l'autre des 2 réseaux. Et ainsi, 1 seule ligne dans la table de routage est suffisante pour traiter 2 réseaux simultanément.

De manière locale à un réseau d'entreprise par exemple, la système CIDR peut permettre de (re)découper un réseau en *sous-réseaux* grâce à un *masque de sous-réseau* ou *subnet netmask*. La figure 2 illustre le cas d'un réseau `X.Y.Z.0/24` découpé en deux sous-réseaux `X.Y.Z.0/25` et `X.Y.Z.128/25`. Pour tout le reste de l'internet, il n'existe qu'un seul réseau `X.Y.Z.0/24` et tous les routeurs traitent les datagrammes à destination de ce réseau de la même façon. Par contre, le routeur R se sert du 25^{e}bit (égal à 0 ou 1) de l'adresse contenue dans les datagrammes qui lui proviennent pour les diriger vers le sous-réseau auquel ils sont destinés, assurant ainsi un routage hiérarchique.

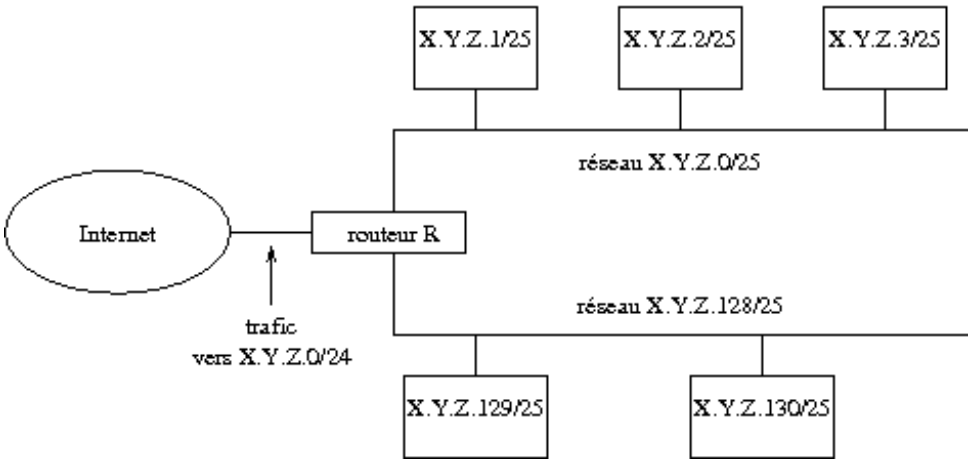


FIG. 2 – Adressage de sous-réseau

1.2 Attribution d'une adresse IP.

De manière administrative, l'obtention d'une plage d'adresses IP pour créer un nouveau réseau est gérée par l'ICANN de manière décentralisée et hiérarchique. Par exemple, pour l'Europe, c'est le **RIPE Network Coordination Centre** (`http://www.ripe.net`) qui assure cette gestion. D'une manière générale, les FAI (Fournisseurs d'Accès à Internet) disposent ainsi de plages d'adresses qui leurs sont attribuées par l'un de ces organismes (le document `ftp://ftp.ripe.net/pub/stats/ripencc/membership/alloclist.txt` donne la liste des plages d'adresses IP attribuées par le RIPE).

De manière technique, une machine peut avoir une adresse IP *statique*, qu'elle conserve de manière permanente, ou *dynamique*, qui change (ou peut changer) à chaque redémarrage ou quand cette adresse n'est plus valide. Par ailleurs, l'attribution de cette adresse IP peut être réalisée via une configuration manuelle, ou via le protocole **DHCP** (Dynamic Host Configuration Protocol). DHCP est un protocole client-serveur où le client est une machine qui demande à s'intégrer au réseau IP « géré »par le serveur DHCP. La principale phase du protocole se découpe en 4 étapes illustrées dans la figure3.

Tous les messages DHCP d'un client vers un serveur sont envoyés dans des datagrammes UDP adressés au port 67 et les messages d'un serveur vers un client sont envoyés dans des datagrammes UDP adressés au port 68.

1. **DHCPDISCOVER** : le client envoie en diffusion (à l'adresse IP 255.255.255.255) une requête en spécifiant 0.0.0.0 comme adresse IP d'origine puisqu'il ne possède pas d'adresse IP pour l'instant. Il indique aussi son adresse matérielle et un numéro de transaction. Ce message est reçu par toutes les machines du réseau, et notamment par les serveurs DHCP qui vont y répondre.
2. **DHCPOFFER** : les serveurs DHCP répondent par un message contenant l'identifiant de transaction, l'adresse IP proposée, le masque de sous-réseau et la durée du *bail* (durée de vie de cette adresse avant expiration).
3. **DHCPREQUEST** : le client accepte l'une des propositions (a priori la première) et répond en envoyant en diffusion un message contenant les divers paramètres.
4. **DHCPACK** : le serveur concerné confirme le bail et mémorise de son côté que cette adresse IP est désormais inutilisable jusqu'à sa libération.

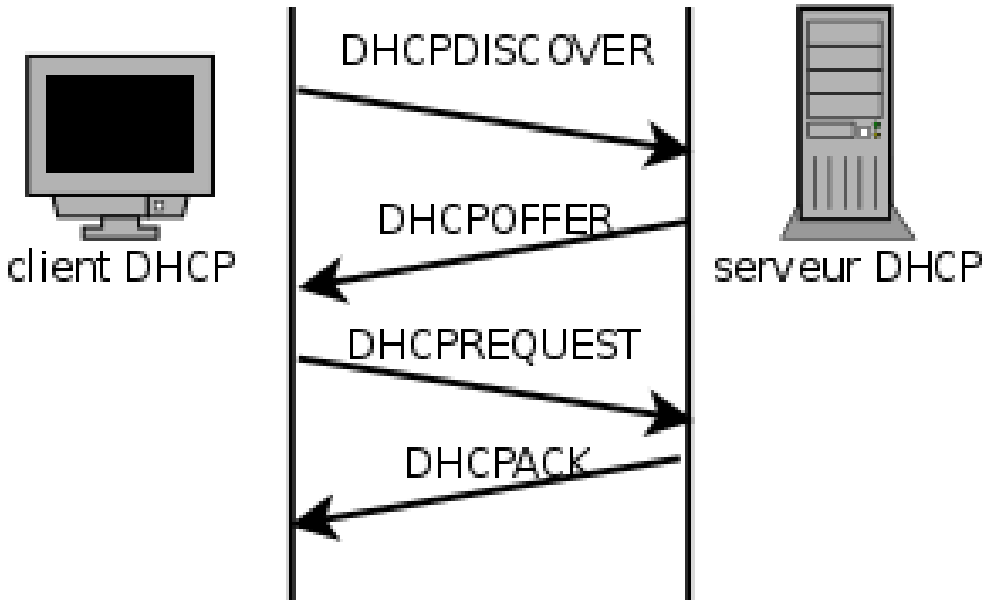


Fig. 3 – DHCP

Les autres points du protocole sont gérés par les messages suivants.
- DHCPNACK : le serveur informe le client que le bail est terminé
- DHCPDECLINE : le client refuse l'adresse IP car elle est déjà utilisée
- DHCPRELEASE : le client libère l'adresse IP et annule le bail
- DHCPINFORM : le client possède une IP et il demande des paramètres de configuration locaux

2 Nommage.

Bien que la numérotation IP à l'aide d'adresses numériques soit suffisante techniquement, il est préférable pour un humain de désigner une machine par un nom explicite. Mais se pose alors le problème de la définition des noms et de leur mise en correspondance avec les numéros IP. Au début des années 80, le réseau ARPANET comportait un peu plus de 200 ordinateurs et chacun possédait un fichier `/etc/hosts` identifiant les noms de ces ordinateurs suivis de leur numéro IP. Lorsqu'une modification intervenait, il suffisait de mettre à jour ce fichier. Pour faire face à l'explosion du nombre d'ordinateurs reliés à Internet, il a été mis en place un système de base de données distribuées : le *système de noms de domaines* **DNS** (Domain Name System) qui fournit la correspondance entre un nom de machine et son numéro IP.

En fait, le DNS est un espace de noms hiérarchisé comme illustré dans la figure 4. Chaque nœud a un nom d'au plus 63 caractères et la racine de l'arbre a un nom nul (les minuscules et majuscules sont indifférenciées). Une *zone* est un sous-arbre de cette hiérarchie. Le *nom de domaine* d'un nœud est la concaténation de son nom avec celui de ses ancêtres dans l'arbre. La responsabilité du nommage est subdivisée par niveau, les niveaux supérieurs délégant leur autorité aux sous-domaines qu'ils créent eux-mêmes. Le système est géré au niveau mondial par l'ICANN. Des organismes ou sociétés assurent ce service par délégation pour les sous-domaines. En France, l'AFNIC (Association Française pour le Nommage Internet en Coopération `http://www.afnic.fr` assure ce service pour le domaine `.fr`.

Il faut bien avoir à l'esprit que le découpage n'a dans certains cas aucune base géographique ;

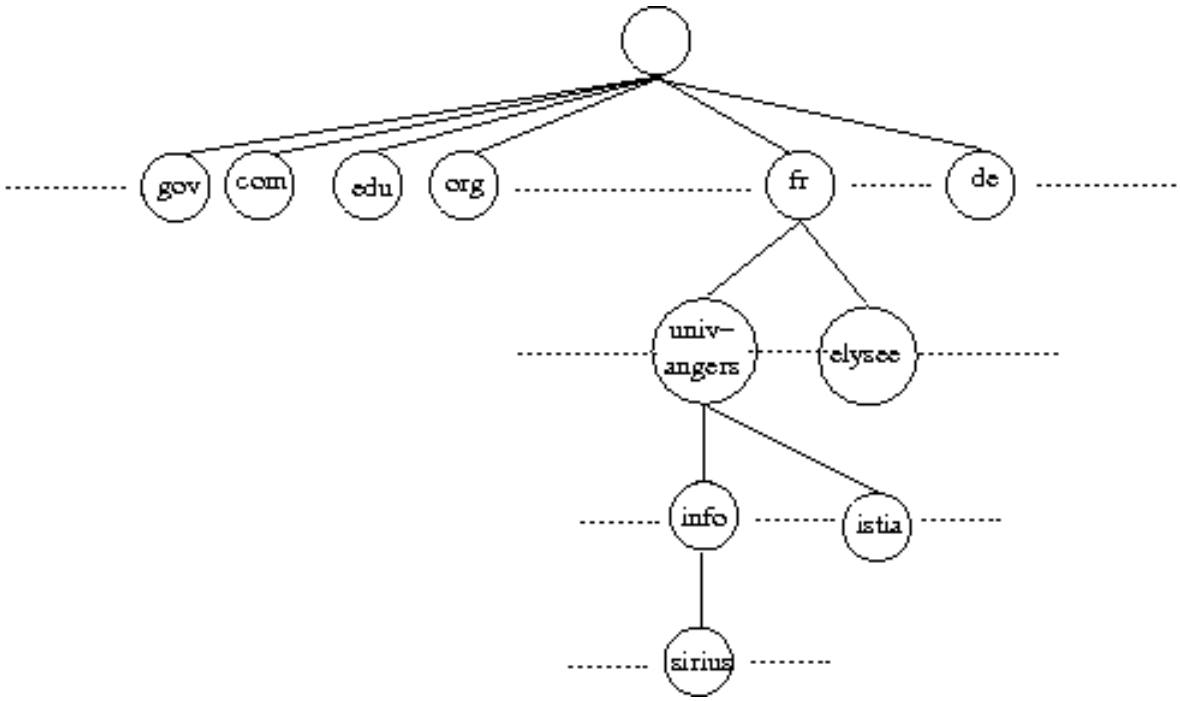


FIG. 4 – Système de noms de domaines.

on trouve des domaines `.com` partout dans le monde.

Le mécanisme qui permet la *résolution* d'un nom en une adresse IP est gérée par des *serveurs de noms* qui représentent une base de données distribuée des noms de domaine. Quand une entité a reçu l'autorité de gérer une zone elle doit maintenir au moins deux serveurs de noms : un *primaire* et un ou plusieurs *secondaires*. Les secondaires sont des serveurs redondants par rapport au primaire de manière à faire face à une défaillance d'un système. Lorsqu'une machine est ajoutée à une zone, l'administrateur de la zone doit ajouter son nom et son numéro IP sur le serveur primaire qui se reconfigure alors en fonction de ces nouvelles données. Quant à eux, les serveurs secondaires interrogent régulièrement (toutes les 3 h) le primaire et font les mises à jour nécessaires en cas d'évolution de la base de données.

Les serveurs de noms peuvent fonctionner en mode récursif ou non, mais ils doivent toujours implanter le mode non récursif. Dans tous les cas, lorsqu'un serveur de noms reçoit une demande, il vérifie si le nom appartient à l'un des sous-domaines qu'il gère. Si c'est le cas il traduit le nom en une adresse IP en fonction de sa base de données et renvoie la réponse au demandeur Sinon,

- en mode non-récursif, le serveur indique au client un autre serveur de noms qui saura lui répondre ou à son tour transmettre la requête à un autre serveur.
- en mode récursif, c'est le serveur qui se charge de l'interrogation successive des serveurs de noms et qui retourne finalement la réponse au client.

Dans tous les cas, lorsqu'un serveur ne sait pas répondre il utilise l'adresse d'un serveur de nom hiérarchiquement supérieur qui connaît le nom et l'adresse IP de chaque serveur de noms pour les domaines de ses sous-niveaux. Ce serveur de nom supérieur renvoie alors l'adresse d'un serveur de noms à contacter. Et ainsi de suite, par interrogations successives de serveurs de noms (soit par le client en mode non-récursif soit par le serveur lui-même en mode récursif) le client initial obtiendra l'adresse demandée. Pour éviter de faire trop souvent de telles requêtes, tout serveur de noms stocke dans une mémoire cache les correspondances (numéro IP, nom de

machine) de manière à pouvoir fournir la réponse immédiatement si une même demande lui parvient ultérieurement.

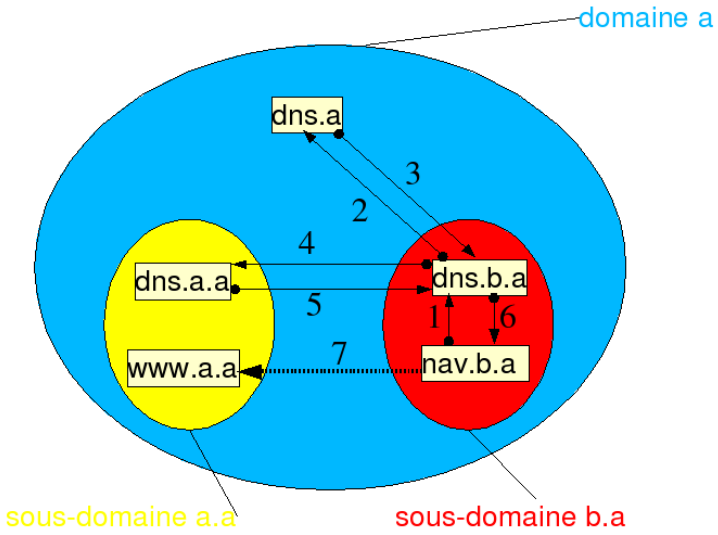


FIG. 5 – Résolution de noms.

La figure 5 décrit la résolution (en mode non récursif) du nom du serveur web `www.a.a` lorsque le navigateur sur la machine `nav.b.a` cherche à joindre ce site.

1. Le navigateur envoie à son DNS `dns.b.a` une requête de résolution pour le nom `www.a.a`
2. `dns.b.a` ne connaissant pas cette adresse, car elle ne dépend pas de sa zone et qu'il ne l'a pas dans son cache, transmet cette adresse à `dns.a` puisque c'est le DNS d'autorité de niveau supérieur qu'il connaît.
3. `dns.a` ne connaissant pas non plus l'adresse demandée, renvoie au demandeur l'adresse d'un ou plusieurs DNS pour le sous-domaine recherché `a.a` et auquel il a lui-même délégué son autorité.
4. Le serveur `dns.b.a` réémet sa requête vers ce nouveau DNS `dns.a.a`.
5. Le serveur `dns.a.a` connaît l'adresse demandée car la machine `www.a.a` appartient à sa zone et peut donc renvoyer l'adresse IP demandée à `dns.b.a`.
6. Le DNS mémorise dans son cache la réponse et la retourne au demandeur initial : le navigateur.
7. Le navigateur peut se connecter au serveur web désiré.

La page `http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/dns/dns.html` propose une illustration de ce mécanisme.

Sous linux, les commandes `host`, `whois`, `dig` sont divers outils pour interroger les DNS. Par exemple :

```
pn@pn:~$ host www.univ-angers.fr
www.univ-angers.fr is an alias for web-serv.univ-angers.fr.
web-serv.univ-angers.fr has address 193.49.144.131
web-serv.univ-angers.fr mail is handled by 0 lagaffe.univ-angers.fr.
```

montre la résolution de nom du serveur web de l'université d'Angers (en 2007).

```
pn@pn:~$ host 193.49.144.1
1.144.49.193.in-addr.arpa domain name pointer ns.univ-angers.fr.
```

montre le résultat de la résolution inverse qui permet à partir d'une adresse IP de trouver le ou les noms de domaines auxquels elle correspond.

Il faut noter aussi que plusieurs adresses IP peuvent être associées à un nom unique, comme l'exemple suivant l'illustre.

```
pn@pn:~$ host www.google.fr
www.google.fr is an alias for www.google.com.
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 209.85.129.99
www.l.google.com has address 209.85.129.104
www.l.google.com has address 209.85.129.147
```

En fait, chaque résolution du nom `www.google.fr` retourne la liste des adresses IP dans un ordre particulier (round robin). Ainsi, les très nombreuses requêtes interrogeant un même service peuvent être réparties entre différentes machines assurant le même service.

```
pn@pn:~$ whois univ-angers.fr
%%
%% This is the AFNIC Whois server [bonnie.nic.fr].
%%
%% Rights restricted by copyright.
%% See http://www.afnic.fr/afnic/web/legal
%%
%% Use '-h' option to obtain more information about this service.
%%
%% [::ffff:193.49.146.121 REQUEST] >> -V Md4.7 univ-angers.fr
%%
domain:      univ-angers.fr
address:     Universite d'Angers
address:     30, rue des Arenes
address:     49035 Angers Cedex 1
address:     FR
phone:       +33 41 73 53 84
fax-no:      +33 41 73 54 31
e-mail:      jacques.allo@univ-angers.fr
admin-c:     JA69-FRNIC
tech-c:      GR1378-FRNIC
zone-c:      NFC1-FRNIC
nserver:     ns.univ-angers.fr 193.49.144.1
nserver:     resone.univ-rennes1.fr 129.20.254.1
mnt-by:      FR-NIC-MNT
mnt-lower:   FR-NIC-MNT
changed:     nic@nic.fr 20050623
source:      FRNIC
...
```

permet d'obtenir les informations relatives à un nom de domaine.

```
pn@pn:~$ dig univ-angers.fr any

; <<>> DiG 9.3.4 <<>> univ-angers.fr any
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6336
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4, ADDITIONAL: 6
;; QUESTION SECTION:
```

```
;univ-angers.fr.                        IN      ANY
;; ANSWER SECTION:
univ-angers.fr.         172702  IN      MX      50 smtp.univ-angers.fr.
univ-angers.fr.         142614  IN      NS      dns1.univ-nantes.fr.
univ-angers.fr.         142614  IN      NS      dns2.univ-nantes.fr.
univ-angers.fr.         142614  IN      NS      resone.univ-rennes1.fr.
univ-angers.fr.         142614  IN      NS      ns.univ-angers.fr.
;; AUTHORITY SECTION:
univ-angers.fr.         142614  IN      NS      resone.univ-rennes1.fr.
univ-angers.fr.         142614  IN      NS      ns.univ-angers.fr.
univ-angers.fr.         142614  IN      NS      dns1.univ-nantes.fr.
univ-angers.fr.         142614  IN      NS      dns2.univ-nantes.fr.
;; ADDITIONAL SECTION:
smtp.univ-angers.fr.    172702  IN      A       193.49.144.2
smtp.univ-angers.fr.    172702  IN      AAAA    2001:660:7201:709::20
dns1.univ-nantes.fr.    6573    IN      A       193.52.108.41
dns2.univ-nantes.fr.    6573    IN      A       193.52.101.20
resone.univ-rennes1.fr. 51620   IN      A       129.20.254.1
resone.univ-rennes1.fr. 51620   IN      AAAA    2001:660:7307:31:129:20:254:1
;; Query time: 2 msec
;; SERVER: 172.20.41.2#53(172.20.41.2)
;; WHEN: Wed Oct  3 16:44:35 2007
;; MSG SIZE  rcvd: 330
```

montre les informations relatives au domaine `univ-angers.fr` (en 2007). On y voit notamment qu'un serveur DNS fournit également, par son champ MX, l'adresse des serveurs de courrier gérant le domaine concerné.

3 La couche de liens d'internet.

Le but de la couche de liens de la pile TCP/IP est d'envoyer et recevoir des datagrammes IP pour la couche IP, d'envoyer des requêtes ARP (respt. RARP) et de recevoir des réponses pour le module ARP (respt. RARP). Elle concentre donc les caractéristiques des deux premières couches du modèle OSI : couche physique et couche de liaison.

3.1 La liaison SLIP.

SLIP (*Serial Line Internet Protocol*, RFC 1055) est un protocole permettant d'envoyer des paquets IP entre deux ordinateurs reliés par une liaison série (par exemple, grâce à deux modems branchés sur les ports RS-232 et une ligne téléphonique). Dans ce cas il n'y a pas besoin de prévoir un adressage de niveau 2, puisque la liaison est point à point (une seule machine à chaque extrémité du lien). Par contre, il s'agit de délimiter le début et la fin des paquets IP.

Le protocole SLIP définit 2 caractères spéciaux : `END` (192 en décimal,c0 en hexadécimal) et `ESC` (219 en décimal, db en hexadécimal). Dans sa version la plus simple, l'encapsulation d'un paquet IP consiste à envoyer tous les octets du paquet IP et de terminer par l'envoi du caractère `END`. Si le caractère `END` se trouve parmi les données à expédier, alors la suite de 2 octets `ESC` et `dc` (220 en décimal) est émise à la place. S'il s'agit d'expédier le caractère `ESC`, alors la suite de 2 octets `ESC` et `dd` (221 en décimal) est émise à la place. Pour éviter des problèmes de bruit, certaines implantations de SLIP font également débuter l'envoi du paquet IP par un caractère `END`. Au final, l'encapsulation SLIP se résume à l'illustration de la figure 6.

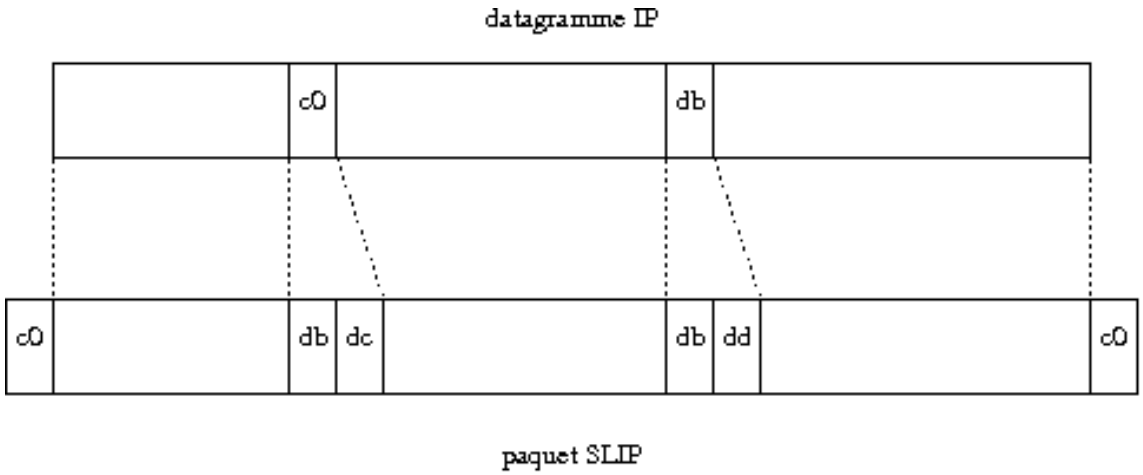


FIG. 6 – Encapsulation SLIP.

Un des défauts de ce protocole est qu'il faut que les deux extrémités aient fixé préalablement leurs adresses IP, car la liaison SLIP ne leur permet pas de se les échanger. Si un site offre via un seul modem l'accès à Internet à plusieurs personnes, cela ne posera pas de problème. En effet, chaque personne aura configuré son ordinateur avec le numéro IP fournit par l'administrateur du réseau et comme une seule connexion est possible à la fois la duplication du même numéro IP n'est pas gênante. Seulement, si le site offre un deuxième modem sur le même numéro téléphonique, les utilisateurs ignoreront à quel modem ils sont connectés. À ce moment là, il faudra que le système indique à chaque utilisateur comment configurer son ordinateur en fonction de l'utilisation ou non de l'autre modem de telle manière que la même adresse IP ne soit pas donnée à deux machines différentes simultanément. Dans ce genre d'utilisation SLIP a

le défaut de ne pas offrir d'accès contrôlé par mot de passe. De plus, il n'y a pas de champ type donc la ligne ne peut pas être utilisée en même temps pour un autre protocole. Et enfin, il n'y a pas de contrôle de la transmission. Si une trame subit des perturbations, c'est aux couches supérieures de de le détecter. Enfin, il existe une version améliorée CSLIP (*Compressed SLIP* RFC 1144) qui assure la (dé)compression des données à transmettre.

3.2 La liaison PPP.

PPP (*Point to Point Protocol*) (RFC 1661) est un protocole qui corrige les déficiences de SLIP en offrant les fonctionnalités suivantes :

- utilisation sur des liaisons point à point autres que série, comme X25 ou RNIS
- transport de différents protocoles de niveau 3 (IP, Decnet, Appletalk, ...)
- compression des en-têtes IP et TCP pour augmenter le débit de la liaison
- gestion d'un contrôle d'accès au réseau par authentification selon le protocole PAP qui nécessite la donnée d'un mot de passe au début de la communication ou le protocole CHAP qui permet l'échange de sceaux cryptés tout au long de la communication
- détection et correction d'erreurs de transmission
- ne pas utiliser des codes qui risquent d'être interprétés par les modems
- configuration automatique de la station client selon ses protocoles de couche réseau (IP, IPX, Appletalk).

Le protocole PPP est celui classiquement utilisé par les fournisseurs d'accès à Internet pour connecter leurs abonnés via le réseau RTC (téléphone) selon le schéma de la figure 7.

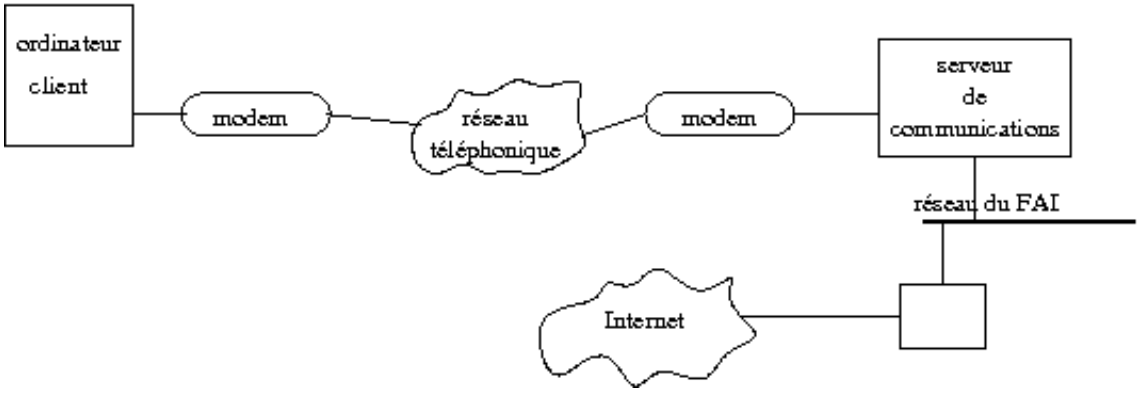


FIG. 7 – Connexion à Internet par modem et PPP.

Dans ce cas, le processus de connexion d'un client est le suivant.

- Le modem du client appelle le numéro de téléphone du fournisseur et la connexion téléphonique s'établit si l'un au moins de ses modems est libre.
- L'identification du client se fait par envoi d'un nom d'utilisateur et d'un mot de passe soit directement par l'utilisateur, soit selon l'un des protocoles PAP ou CHAP. Pour PAP (*Protocol Authentification Protocol*) le serveur de communication envoie à l'ordinateur un paquet pour demander le nom d'utilisateur et le mot de passe et l'ordinateur renvoie ces informations directement. CHAP (*Challenge Handshake Authentification Protocol*) fonctionne de la même manière sauf que le serveur de communication envoie d'abord une clef qui va permettre de crypter l'envoi du nom d'utilisateur et du mot de passe.
- Une fois l'identification du client controlée, le serveur de communication envoie une adresse IP, dite *dynamique* car elle varie selon les connexions, à l'ordinateur du client qui à partir

de là se retrouve intégré au réseau Internet avec une adresse IP pour tout le temps que durera sa connexion.

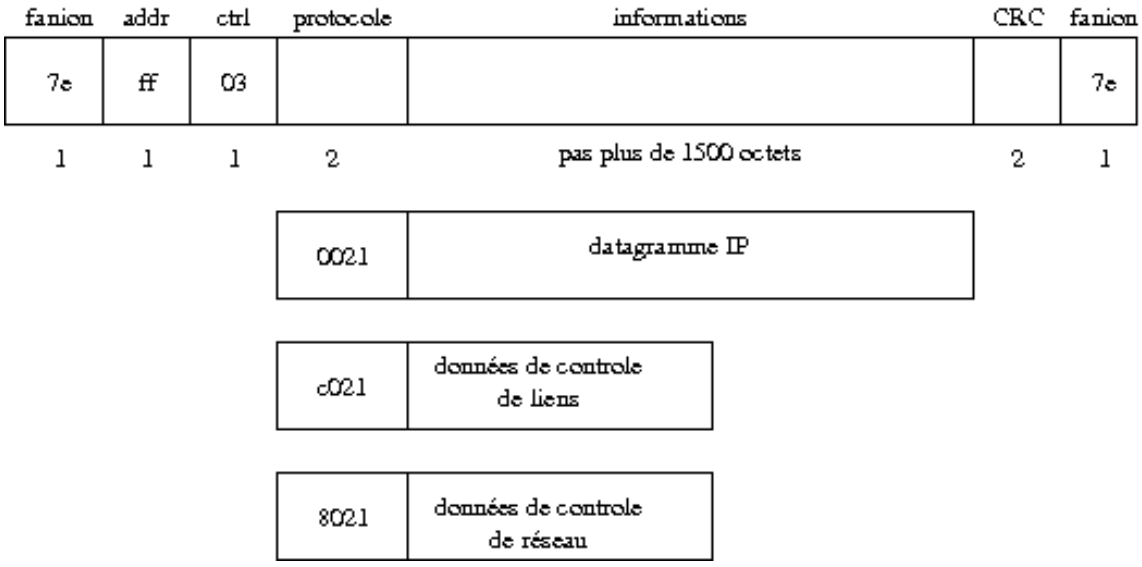


FIG. 8 – Encapsulation PPP.

De manière plus technique l'encapsulation PPP illustrée dans la figure 8 est proche du standard HDLC de l'ISO et est telle que chaque trame commence et finit par un fanion de valeur `7e` soit en binaire `01111110`. La valeur du champ adresse est toujours fixée à `ff` puisqu'elle est inutile ici dans le cas d'une liaison point à point. Le champ contrôle est fixé à `03`. Le champ protocole précise la nature des données transportées : IP, appletalk ...Le CRC assure la détection des erreurs de transmission. Il est calculé à l'aide du polynome générateur $P(x) = x^{16} + x^{12} + x^5 + 1$ sur l'ensemble des champs adresse, contrôle, protocole et informations.

Le problème de l'apparition du fanion `01111110` au milieu des données à transmettre est réglé des deux manières suivantes.

- Dans le cas d'une liaison synchrone, à l'émission un bit à `0` est systématiquement ajouté après 5 `1` et il est retiré à la réception.
- Dans le cas d'une liaison asynchrone, le fanion `7e` est remplacé par la suite `7d 5e`, et le code `7d` est lui-même remplacé par la suite `7d 5d`. De plus tout octet `O` de valeur inférieure à `20` (32 en décimal), correspondant donc à un code de contrôle ASCII, sera remplacé par la séquence `7d O'` où `O'`=`O`$\oplus$20. Ainsi, on est sûr que ces caractères ne seront pas interprétés par les modems comme des caractères de commandes. Par défaut, les 32 valeurs sont traitées ainsi mais il est possible d'utiliser le protocole de contrôle de liens pour spécifier pour quels caractères uniquement on fait cette transformation.

3.3 Le réseau Ethernet.

Ethernet est le nom donné à une des technologies les plus utilisées pour les réseaux locaux en bus. Elle a été inventée par Xerox au début des années 70 et normalisée par l'IEEE (*Institute for Electrical and Electronics Engineers)* vers 1980 sous la norme IEEE 802 et la déclinaison la plus usitée est la 802.3.

Tout d'abord, il existe plusieurs technologies physiques pour établir un réseau local de type Ethernet comme illustré dans la figure9.

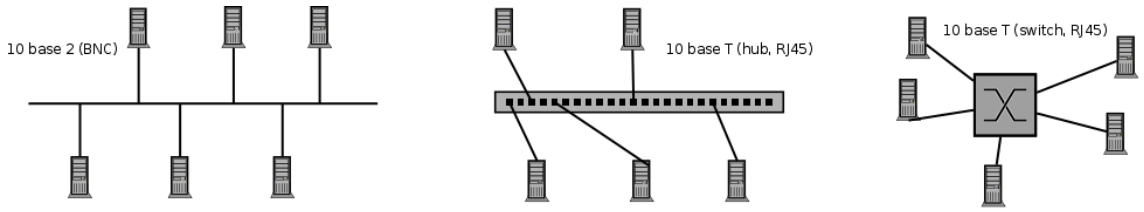


FIG. 9 – Topologie de réseaux locaux Ethernet.

- *10 base 5 ou thick Ethernet* est un réseau à base de câble coaxial de 1,27 cm de diamètre, d'une longueur de 500 m maximum et terminé à chaque extrémité par une résistance. Chaque ordinateur est relié, par un cordon AUI *Attachment Unit Interface*), à un boîtier appelé *transceiver* lui-même connecté au câble par l'intermédiaire d'une prise « vampire ». Le transceiver est capable de détecter si des signaux numériques transitent sur le câble et de les traduire en signaux numériques à destination de l'ordinateur, et inversement.
- *10 base 2 ou thin Ethernet* est un réseau à base d'un câble coaxial plus fin et plus souple, moins résistant aux perturbations électromagnétiques que le 10 base 5, mais d'un coût inférieur. Le transceiver et le câble AUI ne sont plus utiles car l'ordinateur est relié directement au câble par l'intermédiaire d'une prise BNC en T intégrée à la carte Ethernet de l'ordinateur.
- *10 base T ou twisted pair Ethernet* est un réseau dans lequel chaque ordinateur est relié, par un câble de type paire torsadée, à un point central qui est soit un *répartiteur (hub)* ou un *commutateur (switch)* simulant l'effet d'un transceiver et de son câble AUI. La connexion des câbles se fait par l'intermédiaire d'une prise RJ45 et les concentrateurs doivent être alimentés électriquement. Ils simulent ainsi le fonctionnement d'un bus alors que la topologie physique du réseau est une étoile. Le débit d'un tel réseau est très généralement de 10 ou 100 Mbit/sec maximal mais l'on trouve aussi les technologies 1000 baseT (ou Gigabit Ethernet sur fil de cuivre ou fibre optique) et 10 GigaBit portant le débit à 1 ou 10 Gbit/sec.

Dans tous les cas les informations sont transmises sur le bus sans garantie de remise. Chaque transceiver capte toutes les trames qui sont émises sur le câble et les redirige vers le contrôleur de l'ordinateur qui rejettera les trames qui ne lui sont pas destinées et enverra au processeur celles qui le concernent, c'est-à-dire celles dont l'adresse de destination est égale à celle de la carte réseau.

Comme il n'y a pas d'autorité centrale qui gère l'accès au câble, il est possible que plusieurs stations veuillent émettre simultanément sur le câble. C'est pourquoi chaque transceiver écoute le câble pendant qu'il émet des données afin de détecter des éventuelles perturbations. Si une collision est détectée par le transceiver, celui-ci prévient le coupleur qui arrête d'émettre et attend un laps de temps aléatoire compris entre 0 et une certaine durée δ avant de réémettre ses données. S'il y a encore un problème de collision, alors un nouveau temps d'attente est tiré au sort entre 0 et $2*\delta$, puis entre 0 et $4*\delta$, etc... jusqu'à ce que la trame soit émise. Ce principe est justifié par le fait que si une première collision se produit, il y a de fortes chances que les délais d'attente tirés au sort par chacune des 2 stations soient très proches, donc il ne sera pas surprenant d'avoir une nouvelle collision. En doublant à chaque fois l'intervalle des délais d'attente possibles on augmente les chances de voir les retransmissions s'étaler sur des durées relativement longues et donc de diminuer les risques de collision. Cette technologie s'appelle

CSMA/CD (*Carrier Sense Multiple Access with Collision Detect)*. Elle est efficace en général mais a le défaut de ne pas garantir un délai de transmission maximal après lequel on est sûr que la trame a été émise, donc cela ne permet pas de l'envisager pour des applications temps réel.

Les adresses physiques Ethernet ou MAC adresses (Medium Access control) sont codées sur 6 octets et données sous la forme de 6 nombres hexadécimaux. Ils sont censées être uniques car les constructeurs et l'IEEE gèrent cet adressage de manière à ce que deux coupleurs ne portent pas la même adresse[1]. Elles sont de trois types

- *unicast* dans le cas d'une adresse monodestinataire désignant un seul coupleur
- *broadcast* dans le cas d'une adresse de diffusion générale (tous les bits à 1 donc égale à `FF :FF :FF :FF :FF :FF`) qui permet d'envoyer une trame à toutes les stations du réseau
- *multicast* dans le cas d'une adresse multidestinataire qui permet d'adresser une même trame à un ensemble de stations qui ont convenu de faire partie du groupe que représente cette adresse multipoint.

On voit donc qu'un coupleur doit être capable de reconnaitre sa propre adresse physique, l'adresse de multicast, et toute adresse de groupe dont il fait partie.

Au niveau des trames, plusieurs formats ont été définis, mais le plus courant aujourd'hui est celui dit de type II illustré dans la figure 10.

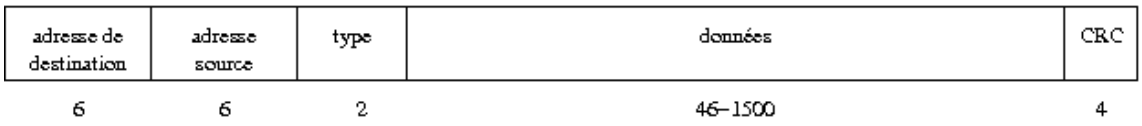


FIG. 10 – Trame Ethernet.

Les adresses matérielles source et destination sont codées sur 6 octets (adresse Ethernet). Le troisième champ contient le *type* de données transmises selon que c'est un datagramme IP, une requête ou réponse ARP ou RARP. Dans la norme 802.3, ce champ *type* est remplacé par un champ *longueur* indiquant la taille des données transmises (la partie données est également subdivisée en plusieurs zones correspondant à la sous couche Logical Link Control). Les deux types de trames (II et 802.3) cohabitent sur un même réseau, sachant que la version avec champ type correspond au transport de données utilisateurs, quand la version avec champ longueur correspond au dialogue entre éléments du réseau (ex : Spanning Tree Protocol). Puis, viennent les données transmises qui peuvent avoir une taille allant de 46 à 1500 octets. Dans le cas de données trop petites, comme pour les requêtes et réponse ARP et RARP (voir la sous-section 3.5) on complète avec des *bits de bourrage* ou *padding*. Enfin, un CRC de 4 octets termine la trame, ce CRC est calculé à partir du polynome générateur : $P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

De nombreux équipements matériels interviennent dans la constitution physique d'un réseau Ethernet, ce paragraphe décrit quelques uns de ceux qui interviennent aux niveaux 1 et 2 du modèle OSI.

Les équipements matériels suivants peuvent intervenir dans la constitution physique d'un réseau Ethernet.

- Un *répéteur* opére de manière physique uniquement, donc au niveau de la couche 1 du modèle OSI. Il se contente de retransmettre et d'amplifier tous les signaux qu'il reçoit,

[1]En pratique, certains constructeurs ne respectent pas cette règle et plusieurs cartes peuvent avoir la même adresse, voire une adresse nulle. On peut aussi programmer l'adresse matérielle.

sans aucun autre traitement. Un *hub* est un répéteur 10 base T multiport qui renvoie donc le signal qu'il reçoit par l'un de ses ports vers tous ses autres ports.

– Un *pont* est un équipement qui intervient dans l'architecture d'un réseau en reliant deux segments disjoints de ce réseau. Le pont appartient à la couche 2 du modèle OSI car il va filtrer les trames du réseau en fonction de leur origine et destination, mais il ne se préoccuppe pas du logiciel réseau de niveau supérieur (TCP/IP, DECNet, IPX, ...).

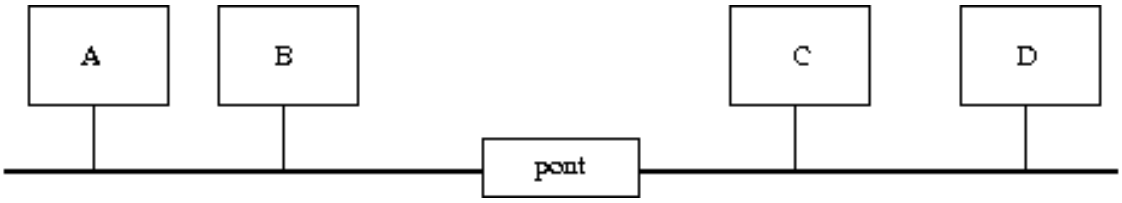


FIG. 11 – Fonctionnement d'un pont..

Dans la configuration de la figure 11 le pont sera capable de déterminer que les ordinateurs A et B sont sur le segment 1 et les ordinateurs C et D sur le segment 2. Il peut obtenir ces informations car il « voit passer »toutes les trames provenant des ordinateurs appartenant aux deux segments qu'il relie et grâce aux adresses d'origine contenues dans les trames, il peut se construire une table d'adresses mémorisant la cartographie du réseau. Ainsi, si une trame est envoyée de A vers B, ou de C vers D, elle ne franchira pas le pont car celui-ci aura détecté que c'est inutile. Mais si la trame provenant de A est destinée à C ou D, elle le traversera sans aucun autre traitement.
L'utilisation d'un pont peut ainsi améliorer le débit d'un réseau car toutes les trames ne sont pas transmises sur tout le réseau. D'autre part, cela peut permettre d'augmenter la confidentialité du réseau en isolant certains ordinateurs des autres de manière à ce que certaines trames soient impossibles à capturer par des ordinateurs « espions »collectionnant toutes les trames qui circulent sur le réseau, même celles qui ne lui sont pas destinées.

– Un *commutateur* ou *switch* est en fait un pont multiport qui va aiguiller chacune des trames qu'il reçoit vers le segment sur lequel se trouve l'ordinateur de destination de la trame. Cependant, chacun de ses ports est habituellement relié à un segment contenant un nombre restreint d'ordinateurs, voire à un seul s'il s'agit par exemple d'un serveur très sollicité. Les performances globales du réseau seront donc bien meilleures que lorsque les postes sont reliés via un hub puisque les changes entre 2 machines du réseau n'inondent pas inutilement les autres machines du réseau. Les meilleures débits et la meilleure confidentialité possibles sur un réseau local Ethernet sont atteints si l'on peut mettre en place un réseau « tout commuté ». Ce terme signifie que chaque poste du réseau est relié par un lien direct à un port du commutateur et ne reçoit ainsi que son trafic personnel.
La page `http://www.info.univ-angers.fr/pub/pn/CommutEthernet` présente une applet illustrant les principes de diffusion et de commutation au sein d'un réseau Ethernet

– La technologie du CPL (Courant Porteur en ligne `http://www.cpl-france.org`) permet d'étendre un réseau local Ethernet au moyen du réseau électrique interne d'un batiment (*accès indoor*). Cette technique peut aussi s'utiliser pour relier à internet une habitation (*accès outdoor*). Dans les deux cas, Chaque équipement informatique est connecté, via une prise RFJ45, à un adaptateur lui-meme branché sur une prise de courant électrique. L'adaptateur multiplexe le signal informatique d'une fréquence comprise entre 2 et 30 Mhz en technique OFDM (Orthogonal Frequency Division Multiplexing) avec le signal électrique de 50Hz du réseau électrique. Le réseau électrique se comporte alors comme

un bus Ethernet au débit de 14Mbit/s. Comme illustré dans la figure 12, cela permet la constitution d'un réseau local à domicile pour partager un accès à internet (voir section 6).

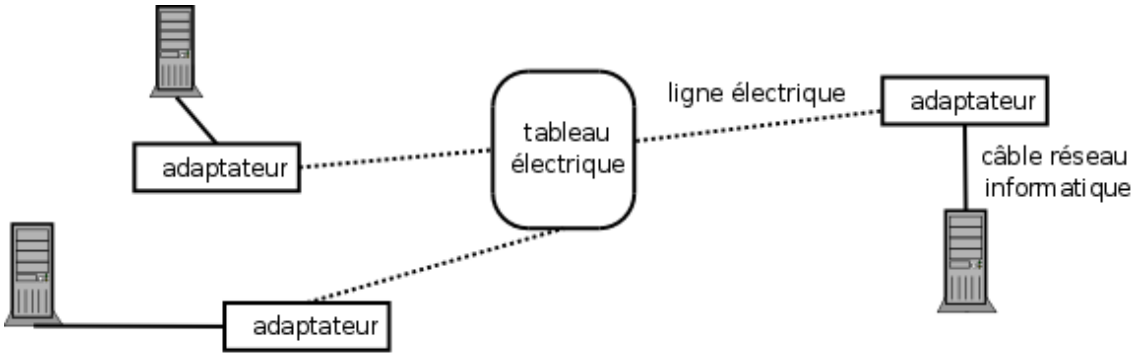


Fig. 12 – Réseau local CPL.

3.4 Le réseau WIFI.

La norme IEEE 802.11 définit les caractéristiques d'un réseau local sans fil (WLAN) plus connu sous le nom WIFI (WIreless FIdelity). Au niveau physique on distingue 3 modes de communication : IR (infrarouge), DSSS (Direct Sequence Spread Spectrum) FHSS (Frequency Hopping Spread Spectrum). Les deux derniers utilisent des ondes radio dans la bande de fréquence de 2.4 Ghz à 2.4835 Ghz et découpent de manière particulière la bande de fréquence en différents canaux. La déclinaison 802.11b (celle du département d'informatique de l'Université d'Angers en 2005) autorise un débit maximal de 11Mbit/s, la 802.11g permet un débit maximal de 54 Mbit/s.

La constitution d'un réseau local sans fil se fait selon le schéma de la figure13 en un mode dit d'*infrastructure*. Le point d'accès est un équipement (antenne émettrice/réceptrice) connecté au réseau filaire Ethernet. Chaque ordinateur, équipé lui-même d'une carte WIFI, communique avec ce point d'accès pour pouvoir dialoguer avec les autres machines du réseau.

Pour sa part, le mode *ad-hoc* consiste à créer un réseau d'ordinateurs équipés de carte WIFI, s'interconnectant entre eux, sans point d'accès particulier ni réseau filaire auquel se raccorder. Cela permet de constituer un réseaux de machines sans aucune structure matérielle préexistante.

Les émissions de données sont régies par la technique CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) qui suit le protocole suivant. Lorsqu'une station veut émettre, elle explore le spectre de fréquences. Si une activité est détectée, elle attend pendant quelques instants avant de réessayer. Si aucune activité n'est détectée, et si cette inactivité dure un temps donné DIFS (Distributed Inter Frame Space), alors la station expédie ses données et attend un message d'aquittement (ACK). Si celui-ci n'arrive pas, soit parce que le récepteur a détecté des incohérences, soit parce qu'il a été lui-meme perturbé, alors la station émettrice réemet les données. Sinon, lorsque le ACK est reçu, l'émission est terminée.

Cependant, la phase d'écoute préalable peut être inefficace car l'absence d'activité ne signifie pas forcément aucune activité, notamment à proximité du point d'accès. En effet, deux stations peuvent ne pas « s'entendre »si elles sont trop éloignées l'une de l'autre, tout en étant chacune à portée du point d'accès (voir la figure 13). Pour pallier cette difficulté, la station émettrice, après avoir constaté une période de silence, va envoyer un message RTS (Ready To Send).

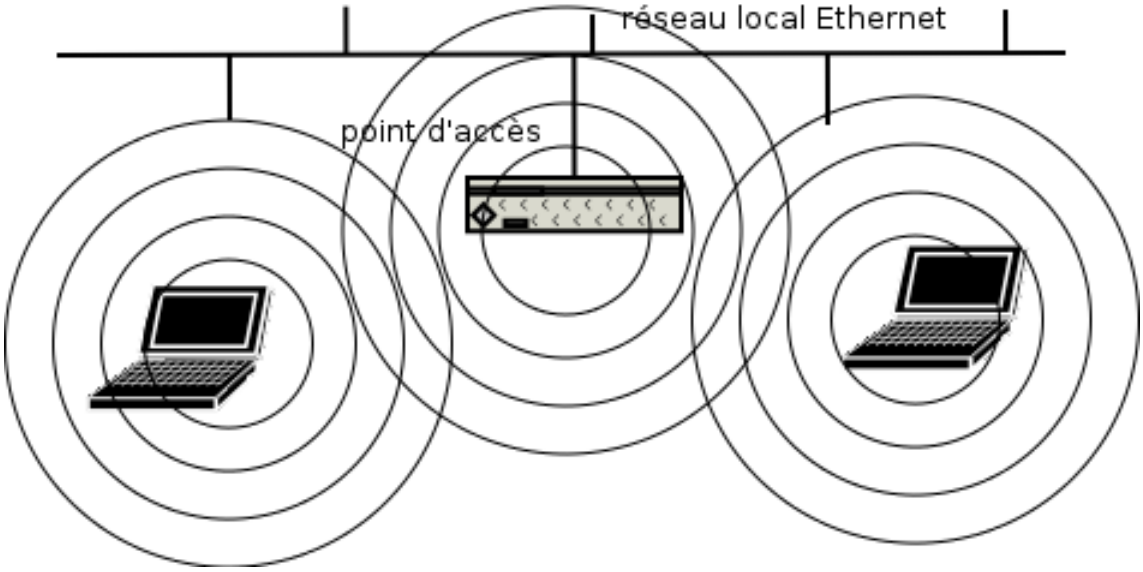


FIG. 13 – Réseau local sans fil.

Ce message contient des informations sur le volume des données à émettre et la vitesse de transmission. Le récepteur répond alors par un message CTS (Clear To Send), puis la station expédie les données. Lorsque toutes les données sont reçues, le récepteur envoie un accusé de réception (ACK). Ainsi, grâce au message CTS qui peut être reçu par toutes les stations (car elles sont toutes à portée du point d'accés), ces dernières vont rester silencieuses pendant le temps nécessaire à la transmission des données à la vitesse annoncée. Ce mécanisme RTS/CTS, qui évite de créer des collisions, n'est utilisé que pour les « gros »paquets de données. Pour les échanges de petite taille, les données sont expédiées immédiatement (sans message RTS/CTS) et l'émetteur attend simplement le ACK.

Les protocoles 802.11 assurent aussi la fragmentation des paquets de manière à expédier peu de données à la fois ce qui améliore les taux de tranfert et chaque paquet est protégé par un CRC de 32 bits.

3.5 Les protocoles ARP et RARP.

Étant donné que le protocole IP, et ses adresses, peuvent être utilisés sur des architectures matérielles différentes (réseau Ethernet, Token-Ring, ...) possédant leur propres adresses physiques, il y a nécessité d'établir les correspondances biunivoques entre adresses IP et adresses matérielles des ordinateurs d'un réseau. Ceci est l'objet des protocoles ARP (*Address Resolution Protocol*) et RARP (*reverse Address Resolution Protocol*). ARP fournit une correspondance dynamique entre une adresse IP et l'adresse matérielle lui correspondant et RARP fait l'inverse.

Nous nous plaçons dans le cas d'une correspondance à établir entre IP et Ethernet. La nécessité de la résolution d'adresse fournie par ARP apparaît dans l'exemple ci-dessous décrivant le début d'une connexion FTP entre un PC et le serveur `sirius` tous deux à l'intérieur du même réseau Ethernet.

1. Le client FTP convertit l'adresse du serveur FTP (ex : `sirius.info-ua`) en une adresse IP (`172.20.41.7`) à l'aide du fichiers `/etc/hosts` ou d'un serveur de noms (DNS).
2. Le client FTP demande à la couche TCP d'établir une connexion avec cette adresse.
3. TCP envoie une requête de connexion à ce serveur en émettant un datagramme IP contenant l'adresse IP

4. En supposant que les machines client et serveur sont sur le même réseau local Ethernet, la machine émettrice doit convertir l'adresse IP sur 4 octets en une adresse Ethernet sur 6 octets avant d'émettre la trame Ethernet contenant le paquet IP. C'est ce que va faire ARP.
5. Le module ARP envoie une requête ARP dans une trame Ethernet (donnée dans la figure 14) avec une adresse de destination multicast. Ainsi, toutes les machines du réseau local reçoivent cette requête contenant l'adresse IP à résoudre.
6. La couche ARP de la machine visée (ici `sirius.info-ua`) reconnaît que cette requête lui est destinée et répond par une réponse ARP contenant son adresse matérielle `00 :20 :AF :AB :42` Les autres machines du réseau ignorent la requête.
7. La réponse ARP est reçue par l'émetteur de la requête. Pour ce retour, il n'y a pas de problème de résolution puisque l'adresse physique de l'émetteur étant envoyée dans la requête elle est connue de la machine qui répond.
8. La réponse ARP est reçue par la couche ARP du client FTP, et le driver Ethernet peut alors émettre le paquet IP avec la bonne adresse Ethernet de destination.

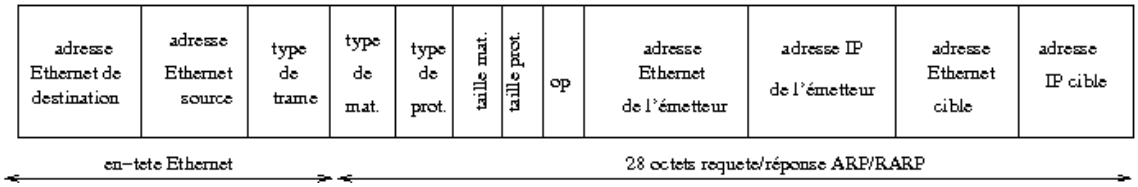


FIG. 14 – Requête ou réponse ARP sur un réseau Ethernet.

Les deux premiers champs d'une trame Ethernet (voir figure 14) émise par ARP sont conformes à l'en-tête d'une trame Ethernet habituelle et l'adresse de destination sera l'adresse multicast `FF :FF :FF :FF :FF :FF` désignant toutes les machines du réseau à la fois. La valeur du champ type de trame est `0806` indiquant le protocole ARP. Le champ type de matériel est égal à 1 pour un réseau Ethernet et celui type de protocole est égal est `0800` pour IP. Les tailles en octets spécifiées ensuite sont 6 (6 octets pour une adresse Ethernet) et 4 (4 octets pour une adresse IP). Le champ op vaut `1` pour une requête ARP et `2` pour une réponse ARP. Les quatre champs suivants contiennent des adresses et sont redondants dans le cas de l'adresse Ethernet metteur d'une requête ARP, et non renseignés dans le cas de l'adresse Ethernet cible d'une requête ARP.

La machine qui reconnaît son numéro IP à l'intérieur d'une requête ARP qu'elle reçoit la renvoie en y intervertissant les adresses IP cible et émetteur, ainsi que les adresses Ethernet cible (après l'avoir substituée à l'adresse de diffusion dans l'en-tête et renseignée dans le corps de la trame) et Ethernet émetteur. Pour éviter la multiplication des requêtes ARP, chaque machine gère un cache dans lequel elle mémorise les correspondances adresses IP/adresses Ethernet déjà résolues préalablement. Ainsi, le module ARP ne lancera une requête que lorsqu'il ne trouvera pas cette correspondance dans le cache, sinon il se contentera d'émettre les données qu'il reçoit d'IP en ayant fixé correctement l'adresse physique de destination. Cependant, les correspondances ne sont pas conservées indéfiniment car cela pourrait provoquer des erreurs lorsque l'on change un ordinateur (ou une carte réseau) du réseau en conservant un même numéro IP pour cet ordinateur mais évidemment pas la même adresse physique.

Comme illustré ci-dessous, la commande `arp -a` (sous Linux et Windows) affiche le contenu du cache mémoire ARP courant.

```
$ arp -a
titan.info-ua (172.20.41.1) à 00:13:46:66:A4:5A [ether] sur eth0
h108-0.info-ua (172.20.41.28) à 00:0F:1F:EA:92:DC [ether] sur eth0
pn.info-ua (172.20.41.43) à 00:12:3F:DD:E2:E0 [ether] sur eth0
neo.info-ua (172.20.41.25) à 00:0E:0C:5B:DF:22 [ether] sur eth0
pollux.info-ua (172.20.41.9) à 08:00:20:88:0F:4E [ether] sur eth0
tines.info-ua (172.20.41.2) à 00:13:46:66:A4:5A [ether] sur eth0
sirius.info-ua (172.20.41.7) à 00:00:1A:18:FF:4C [ether] sur eth0
cerbere.info-ua (172.20.40.9) à 00:10:DC:54:7B:03 [ether] sur eth0
joebar.info-ua (172.20.41.71) à 00:14:85:03:BE:7A [ether] sur eth0
helios.info-ua (172.20.41.5) à 00:03:BA:08:DB:22 [ether] sur eth0
```

Quant à lui, le protocole RARP joue le rôle inverse de ARP en permettant de déterminer l'adresse IP d'un équipement dont on connaît l'adresse physique. Ceci est notamment utile pour amorcer une station sans disques, ou un TX, qui n'a pas en mémoire son adresse IP mais seulement son adresse matérielle. Le format d'une trame RARP est celui de la figure 14) où le champ type de trame vaut `0835` et le champ op vaut `3` pour une requête RARP et `4` pour une réponse. Une requête RARP est diffusée sous forme de broadcast, donc toutes les machines du réseau la reçoivent et la traitent. Mais la plupart des machines ignorent simplement cette demande, seuls, le ou les serveurs RARP du réseau vont traiter la requête grâce à un ou plusieurs fichiers et vont retourner une réponse contenant l'adresse IP demandée.

4 Le protocole IP.

Comme on a peut le voir dans la pile TCP-IP le protocole IP (*Internet Protocol*, RFC 791) est au cœur du fonctionnement d'internet. Il assure *sans connexion* un service *non fiable* de délivrance de datagrammes IP. Le service est non fiable car il n'existe aucune garantie pour que les datagrammes IP arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. On parle de remise au mieux (*best effort delivery*) et ni l'émetteur ni le récepteur ne sont informés directement par IP des problèmes rencontrés. Le mode de transmission est non connecté car IP traite chaque datagramme indépendamment de ceux qui le précèdent et le suivent. Ainsi en théorie, au moins, deux datagrammes IP issus de la même machine et ayant la même destination peuvent ne pas suivre obligatoirement le même chemin. Le rôle du protocole IP est centré autour des trois fonctionnalités suivantes chacune étant décrite dans une des sous-sections à venir.

- définir le format du datagramme IP qui est l'unité de base des données circulant sur Internet
- définir le routage dans internet
- définir la gestion de la remise non fiable des datagrammes

4.1 Le datagramme IP.

Un datagramme IP est constitué d'une en-tête suivie d'un champ de données. Sa structure

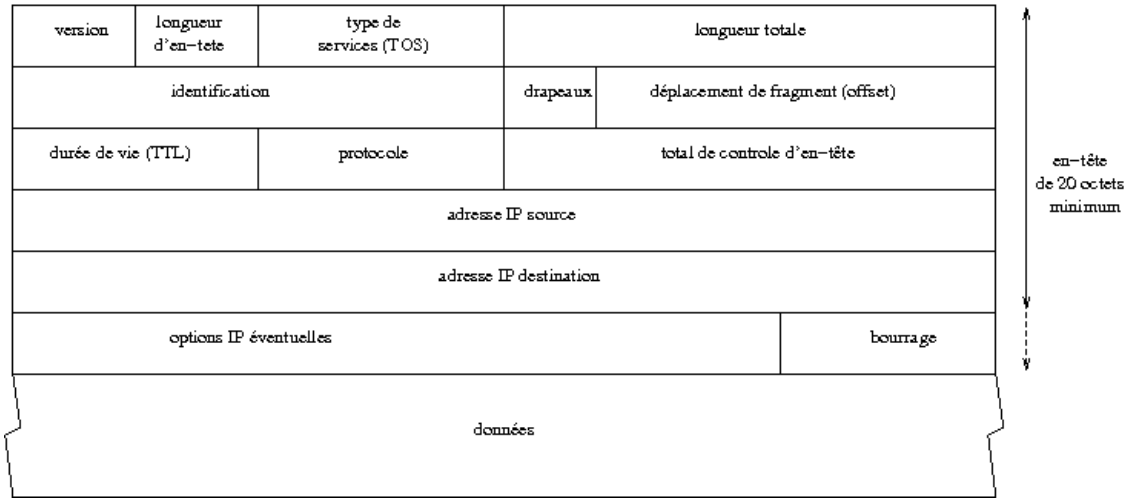


FIG. 15 – Structure d'un datagramme IP.

précise est détaillée dans la figure 15 et comporte les champs suivants.

- La *version* code sur 4 bits le numéro de version du protocole IP utilisé (la version courante est la 4, d'où son nom d'IPv4). Tout logiciel IP doit d'abord vérifier que le numéro de version du datagramme qu'il reçoit est en accord avec lui-même. si ce n'est pas le cas le datagramme est tout simplement rejeté. Ceci permet de tester des nouveaux protocoles sans interférer avec la bonne marche du réseau.
- La *longueur d'en-tête* représente sur 4 bits la longueur, en nombre de mots de 32 bits, de l'en-tête du datagramme. Ce champ est nécessaire car une en-tête peut avoir une taille supérieure à 20 octets (taille de l'en-tête classique) à cause des options que l'on peut y ajouter.

– Le *type de services (TOS)* est codé sur 8 bits, indique la manière dont doit être géré le datagramme et se décompose en six sous-champs comme suit.

0	1	2	3	4	5	6	7
priorité			D	T	R	C	inutilisé

Le champ *priorité* varie de 0 (priorité normale, valeur par défaut) à 7 (priorité maximale pour la supervision du réseau) et permet d'indiquer l'importance de chaque datagramme. Même si ce champ n'est pas pris en compte par tous les routeurs, il permettrait d'envisager des méthodes de contrôle de congestion du réseau qui ne soient pas affectées par le problème qu'elles cherchent à résoudre. Les 4 bits D, T, R et C permettent de spécifier ce que l'on veut privilégier pour la transmission de ce datagramme (nouvel RFC 1455). D est mis à 1 pour essayer de minimiser le délai d'acheminement (par exemple choisir un câble sous-marin plutôt qu'une liaison satellite), T est mis à 1 pour maximiser le débit de transmission, R est mis à 1 pour assurer une plus grande fiabilité et C est mis à 1 pour minimiser les coûts de transmission. Si les quatre bits sont à 1, alors c'est la sécurité de la transmission qui doit être maximisée. Les valeurs recommandées pour ces 4 bits sont données dans la table 1. Ces 4 bits servent à améliorer la qualité du routage et ne sont pas

application	minimise le délai	maximise le débit	maximise la fiabilité	minimise le coût
telnet/rlogin	1	0	0	0
FTP				
contrôle	1	0	0	0
transfert	0	1	0	0
SMTP				
commandes	1	0	0	0
données	0	1	0	0
NNTP	0	0	0	1
SNMP	0	0	1	0

Tab. 1 – Type de service pour les applications standard.

des exigences incontournables. Simplement, si un routeur connaît plusieurs voies de sortie pour une même destination il pourra choisir celle qui correspond le mieux à la demande.

– La *longueur totale* contient la taille totale en octets du datagramme, et comme ce champ est de 2 octets on en déduit que la taille complète d'un datagramme ne peut dépasser 65535 octets. Utilisée avec la longueur de l'en-tête elle permet de déterminer où commencent exactement les données transportées.

– Les champs *identification*, *drapeaux* et *déplacement de fragment* interviennent dans le processus de fragmentation des datagrammes IP et sont décrits dans la sous-section 4.2.

– La *durée de vie* (TTL) indique le nombre maximal de routeurs que peut traverser le datagramme. Elle est initialisée N (souvent 32 ou 64) par la station émettrice et décrémenté de 1 par chaque routeur qui le reçoit et le réexpédie. Lorsqu'un routeur reçoit un datagramme dont la durée de vie est nulle, il le détruit et envoie à l'expéditeur un message ICMP. Ainsi, il est impossible qu'un datagramme « tourne » indéfiniment dans un internet. Ce champ sert également dans la réalisation du programme `tracepath` (sous Linux) ou `tracert` (sous windows). qui sert à déterminer quel est l'itinéraire (l'ensemble des routeurs traversés) entre deux machines, comme dans l'exemple ci-dessous.

```
pn@forge:~$ tracepath www.w3c.org
 1:  forge.info-ua (172.20.41.8)                          0.204ms pmtu 1500
 1:  titan.info-ua (172.20.41.1)                          0.294ms
 1:  titan.info-ua (172.20.41.1)                          0.259ms
 2:  193.49.146.254 (193.49.146.254)                      1.013ms
 3:  ua-jun.net.univ-angers.fr (194.57.169.5)             0.707ms asymm  4
 4:  nantes-rtr-021-g8-4-66.cssi.renater.fr (193.51.182.150)   1.886ms
```

```
 5:  193.51.189.57 (193.51.189.57)                          9.637ms asymm 11
 6:  caen-rtr-021-te4-1.cssi.renater.fr (193.51.189.54)     9.638ms asymm 10
 7:  rouen-rtr-021-te4-1.cssi.renater.fr (193.51.189.46)    9.586ms asymm 10
 8:  193.51.189.49 (193.51.189.49)                          9.716ms
 9:  renater.rt1.par.fr.geant2.net (62.40.124.69)          13.491ms asymm 10
10:  so-3-0-0.rt1.lon.uk.geant2.net (62.40.112.106)        34.455ms asymm 11
11:  so-2-0-0.rt1.ams.nl.geant2.net (62.40.112.137)        25.260ms asymm 12
12:  216.24.184.85 (216.24.184.85)                        109.635ms
13:  216.24.184.102 (216.24.184.102)                      109.036ms
14:  B24-RTR-2-BACKBONE.MIT.EDU (18.168.0.23)             113.754ms asymm 15
15:  MITNET.TRANTOR.CSAIL.MIT.EDU (18.4.7.65)             113.859ms asymm 16
16:  trantor.haven.csail.mit.edu (128.30.0.254)           114.791ms asymm 17
17:  dolph.w3.org (128.30.52.45)                          114.781ms reached
     Resume: pmtu 1500 hops 17 back 47
```

- Le *protocole* permet de coder quel protocole de plus haut niveau a servi a créé ce datagramme. Les valeurs codées sur 8 bits sont 1 pour ICMP, 2 pour IGMP, 6 pour TCP et 17 pour UDP. Ainsi, la station destinataire qui reçoit un datagramme IP pourra diriger les données qu'il contient vers le protocole adéquat.
- Le *total de contrôle d'en-tête* (header checksum) est calculé à partir de l'en-tête du datagramme pour en assurer l'intégrité. L'intégrité des données transportées est elle assurée directement par les protocoles ICMP, IGMP, TCP et UDP qui les émettent. Pour calculer cette somme de contrôle, on commence par la mettre à zéro. Puis, en considérant la totalité de l'en-tête comme une suite d'entiers de 16 bits, on fait la somme de ces entiers en complément à 1. On complémente à 1 cette somme et cela donne le total de contrôle que l'on insère dans le champ prévu. A la réception du datagramme, il suffit d'additionner tous les nombres de l'en-tête et si l'on obtient un nombre avec tous ses bits à 1, c'est que la transmission s'est passée sans problème.
- Les *adresses IP source* et *destination* contiennent sur 32 bits les adresses de la machine émettrice et destinataire finale du datagramme.
- Le champ *options* est une liste de longueur variable, mais toujours complétée par des bits de *bourrage* pour atteindre une taille multiple de 32 bits pour être en conformité avec la convention qui définit le champ longueur de l'en-tête. Ces options sont très peu utilisées car peu de machines sont aptes à les gérer. Parmi elles, on trouve des options de sécurité et de gestion (domaine militaire), d'enregistrement de la route, d'estampille horaire, routage strict, etc...

Les champs non encore précisés le sont dans la section suivante car ils concernent la fragmentation des datagrammes.

4.2 La fragmentation des datagrammes IP.

En fait, il existe d'autres limites à la taille d'un datagramme que celle fixée par la valeur maximale de 65535 octets. Notamment, pour optimiser le débit il est préférable qu'un datagramme IP soit encapsulé dans une seule trame de niveau 2 (Ethernet par exemple). Mais, comme un datagramme IP peut transiter à travers Internet sur un ensemble de réseaux aux technologies différentes il est impossible de définir, a priori (lors de la définition du RFC), une taille maximale (1500 octets pour Ethernet et 4470 pour FDDI par exemple) des datagrammes IP qui permette de les encapsuler dans une seule trame quel que soit le réseau traversé. On appelle la taille maximale d'une trame d'un réseau le *MTU* (*Maximum Transfert Unit*) et elle va servir à fragmenter les datagrammes trop grands pour le réseau qu'ils traversent. Mais, si le MTU d'un réseau traversé est suffisamment grand pour accepter un datagramme, évidemment il sera encapsulé tel quel dans la trame du réseau concerné. Comme on peut le voir dans la figure 16 la fragmentation se situe au niveau d'un routeur qui reçoit des datagrammes issus

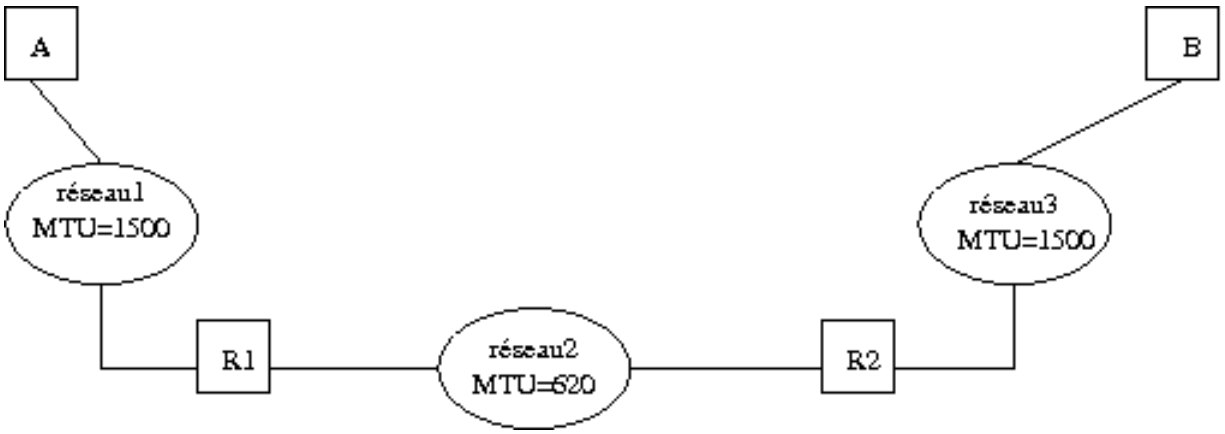


FIG. 16 – Fragmentation d'un datagramme IP.

d'un réseau à grand MTU et qui doit les réexpédier vers un réseau à plus petit MTU. Dans cet exemple, si la station A, reliée à un réseau Ethernet, envoie un datagramme de 1300 octets à destination de la station B, reliée également à un réseau Ethernet, le routeur R1 va devoir fragmenter ce datagramme de la manière décrite ci-dessous.

La taille d'un fragment est choisie la plus grande possible tout en étant un multiple de 8 octets. Un datagramme fragmenté n'est réassemblé que lorsqu'il arrive à destination finale. Même s'ils arrivent sur des réseaux avec un plus grand MTU les routeurs ne réassemblent pas les petits fragments. De plus chaque fragment est routé de manière totalement indépendante des autres fragments du datagramme d'où il provient. Le destinataire final qui reçoit un premier fragment d'un datagramme arme un temporisateur de réassemblage, c'est-à-dire un délai maximal d'attente de tous les fragments. Si, passé ce délai, tous les fragments ne sont pas arrivés il détruit les fragments reçus et ne traite pas le datagramme. Plus précisément, l'ordinateur destinataire décrémente, à intervalles réguliers, de une unité le champ TTL de chaque fragment en attente de réassemblage. Cette technique permet également de ne pas faire coexister au même instant deux datagrammes avec le même identifiant.

datagramme initial	en-tête du datagramme : num=N	données1 600 octets	données2 600 octets	données3 100 octets

fragment1	en-tête du fragment1 : num=N, déplacement=0, à suivre=1	données1 600 octets
fragment2	en-tête du fragment2 : num=N, déplacement=75, à suivre = 1	données2 600 octets
fragment3	en-tête du fragment3 : num=N, déplacement=150, à suivre=0	données3 100 octets

TAB. 2 – Exemple de fragmentation IP.

Le processus de fragmentation-réassemblage illustré dans la table 2 est rendu possible grâce aux différents champs suivants. Le champ *déplacement de fragment* (ou *offset*) précise la localisation du début du fragment dans le datagramme initial. Cet offset est égal au décalage réel

divisé par 8 (voir l'exemple de la table2). À part cela, les fragments sont des datagrammes dont l'en-tête est quasiment identique à celle du datagramme original. Le champ *identification* est un entier qui identifie de manière unique chaque datagramme émis et qui est recopié dans le champ identification de chacun des fragments si ce datagramme est fragmenté. Par contre, le champ longueur total est recalculé pour chaque fragment. Le champ *drapeaux* comprend trois bits dont deux qui contrôlent la fragmentation. S'il est positionné à 1 le premier bit indique que l'on ne doit pas fragmenter le datagramme et si un routeur doit fragmenter un tel datagramme alors il le rejette et envoie un message d'erreur à l'expéditeur. Un autre bit appelé *fragments à suivre* est mis systématiquement à 1 pour tous les fragments qui composent un datagramme sauf le dernier. Ainsi, quand le destinataire reçoit le fragment dont le bit fragment à suivre est à 0 il est apte à déterminer s'il a reçu tous les fragments du datagramme initial grâce notamment aux champs offset et longueur totale de ce dernier fragment. Si un fragment doit être à nouveau fragmenté lorsqu'il arrive sur un réseau avec un encore plus petit MTU, ceci est fait comme décrit précédemment sauf que le calcul du champ déplacement de fragment est fait en tenant compte du déplacement inscrit dans le fragment à traiter.

4.3 Le routage IP.

Le routage est l'une des fonctionnalités principales de la couche IP et consiste à choisir la manière de transmettre un datagramme IP à travers les divers réseaux d'un internet. On appellera *ordinateur* un équipement relié à un seul réseau et *routeur* un équipement relié à au moins deux réseaux (cet équipement pouvant être un ordinateur, au sens classique du terme, qui assure les fonctionnalités de routage). Ainsi un routeur réémettra des datagrammes venus d'une de ses interfaces vers une autre, alors qu'un ordinateur sera soit l'expéditeur initial, soit le destinataire final d'un datagramme.

D'une manière générale on distingue la *remise directe*, qui correspond au transfert d'un datagramme entre deux ordinateurs du même réseau, et la *remise indirecte* qui est mise en œuvre dans tous les autres cas, c'est-à-dire quand au moins un routeur sépare l'expéditeur initial et le destinataire final. Par exemple, dans le cas d'un réseau Ethernet, la remise directe consiste à encapsuler le datagramme dans une trame Ethernet après avoir utilisé le protocole ARP pour faire la correspondance adresse IP adresse physique (voir les sections 3.3 et 3.5) et à émettre cette trame sur le réseau. L'expéditeur peut savoir que le destinataire final partage le même réseau que lui-même en utilisant simplement l'adresse IP de destination du datagramme. Il en extrait l'identificateur de réseau et si c'est le même que celui de sa propre adresse IP[2] alors la remise directe est suffisante. En fait, ce mécanisme de remise directe se retrouve toujours lors de la remise d'un datagramme entre le dernier routeur et le destinataire final. Pour sa part, la remise indirecte nécessite de déterminer vers quel routeur envoyer un datagramme IP en fonction de sa destination finale. Ceci est rendu possible par l'utilisation d'une *table de routage* spécifique à chaque routeur qui permet de déterminer vers quelle voie de sortie envoyer un datagramme destiné à un réseau quelconque. À cause de la structure localement arborescente d'internet la plupart des tables de routage ne sont pas très grandes. Par contre, les tables des routeurs interconnectant les grands réseaux peuvent atteindre des tailles très grandes ralentissant d'autant le trafic sur ces réseaux.

L'essentiel du contenu d'une table de routage est constitué de quadruplets *(destination, passerelle, masque, interface)* où :

– destination est l'adresse IP d'une machine ou d'un réseau de destination

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

– passerelle (gateway) est l'adresse IP du prochain routeur vers lequel envoyer le datagramme pour atteindre cette destination
– masque est le masque associé au réseau de destination
– interface désigne l'interface physique par laquelle le datagramme doit réellement être expédié.

Une table de routage contient notamment une *route par défaut* qui spécifie un routeur vers lequel sont envoyés tous les datagrammes pour lesquels il n'existe pas de route spécifique dans la table.

Tous les routeurs mentionnés dans une table de routage doivent bien sûr être directement accessibles à partir du routeur considéré. Cette technique, dans laquelle un routeur ne connaît pas le chemin complet menant à une destination, mais simplement la première étape de ce chemin, est appelée *routage par sauts successifs* (*next-hop routing*).

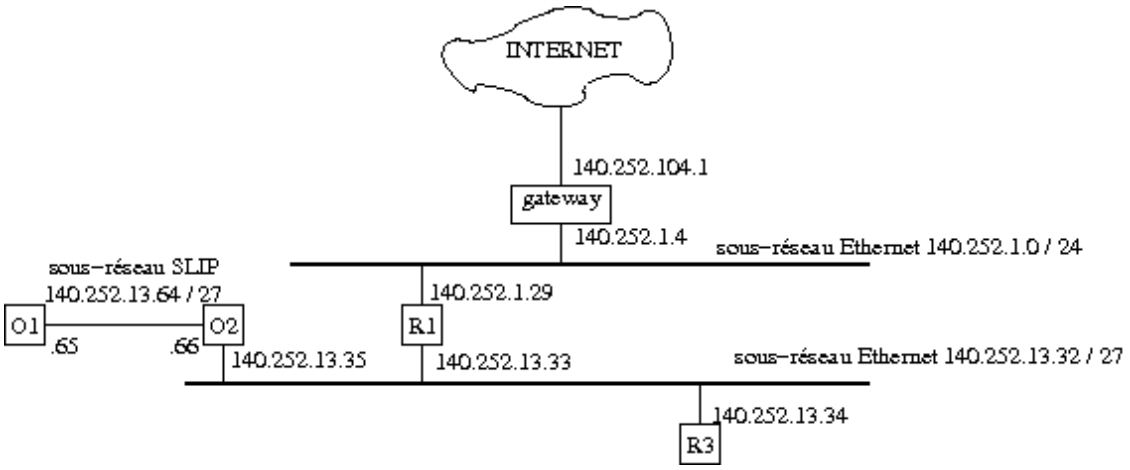


FIG. 17 – Exemple d'interconnexion de réseau.

La figure 17 donne un exemple d'interconnexion d'un réseau de classe B `140.252.0.0` subdivisé en 3 sous-réseaux d'adresses respectives `140.252.1.0/24` pour l'Ethernet du haut, `14.252.13.32/27` pour l'Ethernet du bas et `140.252.13.64/27` pour le sous-réseau SLIP. Les masques de sous-réseau sont associés par défaut à chaque interface d'une machine ou sont éventuellement spécifiés sur chaque ligne des tables de routage. Dans l'exemple donné ici, la table de routage simplifiée de l'ordinateur *R3* (sous système unix Sun) est la suivante[3].

destination	passerelle (gateway)	masque (genmask)	indicateurs (flags)	interface
140.252.13.65	140.252.13.35	255.255.255.255	UGH	emd0
127.0.0.1	127.0.0.1	255.255.255.255	UH	lo0
140.252.13.32	140.252.13.34	255.255.255.224	U	emdO
default	140.252.13.33	0.0.0.0	UG	emd0

Les *indicateurs (flags)* ont la signification suivante.

U La route est en service.

G La route est un routeur (*gateway*). Si ce flag n'est pas positionné la destination est directement connectée au routeur, c'est donc un cas de remise directe vers l'adresse IP de destination.

[3]La commande `netstat -rn` affiche la table de routage sous Linux ou Windows.

H La route est un ordinateur (*host*), la destination est une adresse d'ordinateur. Dans ce cas, la correspondance entre l'adresse de destination du paquet à « router »et l'entrée destination de la table de routage doit être totale. Si ce flag n'est pas positionné, la route désigne un autre réseau et la destination est une adresse de réseau ou de sous-réseau. Ici, la correspondance des identificateurs de réseaux est suffisante.

D La route a été créée par une redirection.

M La route a été modifiée par une redirection.

L'adresse 127.0.0.1 est celle de *lo0*, l'interface de *loopback*, qui sert à pouvoir faire communiquer une machine avec elle-même sans passer physiquement par la carte réseau.

La destination *default* sert à indiquer la destination de tous les datagrammes qui ne peuvent être « routés »par l'une des autres routes. Elle est placée en dernière position, puisque la décision de routage se fait selon l'ordre des lignes de la table. La première entrée (ligne) dont l'adresse de destination est égale à l'adresse de destination du paquet à router « modulo le masque »indique l'interface de sortie par laquelle expédier ce paquet.

4.4 La gestion des erreurs.

Le protocole ICMP (*Internet Control Message Protocol*) organise un échange d'information permettant aux routeurs d'envoyer des messages d'erreurs à d'autres ordinateurs ou routeurs. Bien qu'ICMP « tourne »au-dessus de IP il est requis dans tous les routeurs c'est pourquoi on le place dans la couche IP. Le but d'ICMP n'est pas de fiabiliser le protocole IP, mais de fournir à une autre couche IP, ou à une couche supérieure de protocole (TCP ou UDP), le compte-rendu d'une erreur détectée dans un routeur. Un message ICMP étant acheminé à l'intérieur d'un datagramme IP[4] illustré dans la figure 18, il est susceptible, lui aussi, de souffrir d'erreurs de transmission. Mais la règle est qu'aucun message ICMP ne doit être délivré pour signaler une

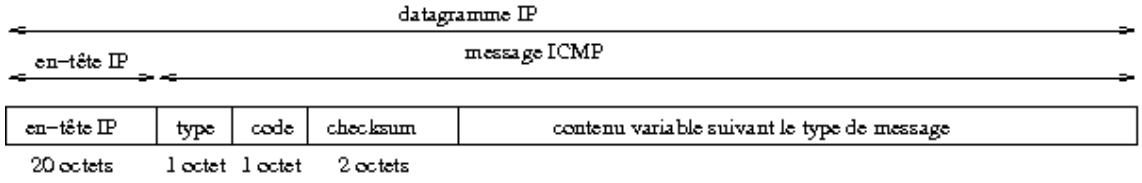


Fig. 18 – Encapsulation d'un message ICMP.

erreur relative à un message ICMP. On évite ainsi une avalanche de messages d'erreurs quand le fonctionnement d'un réseau se détériore.

Le champ *type* peut prendre 15 valeurs différentes spécifiant de quelle nature est le message envoyé. Pour certains types, le champ *code* sert à préciser encore plus le contexte d'émission du message. Le *checksum* est une somme de contrôle de tout le message ICMP calculée comme dans le cas de l'en-tête d'un datagramme IP (voir la section 4.1). Le détail des différentes catégories de messages est donné dans la liste ci-dessous où chaque alinéa commence par le couple $(type, code)$ de la catégorie décrite.

(0,0) ou (8,0) Demande (type 8) ou réponse (type 0) d'écho dans le cadre de la commande `ping`.

(3,0-13) Compte-rendu de destination inaccessible délivré quand un routeur ne peut délivrer un datagramme. Le routeur génère et envoie ce message ICMP à l'expéditeur de ce datagramme. Il obtient l'adresse de cet expéditeur en l'extrayant de l'en-tête du datagramme,

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

il insère dans les données du message ICMP toute l'en-tête ainsi que les 8 premiers octets du datagramme en cause. Une liste non exhaustive des différents codes d'erreurs possibles est :

0 Le réseau est inaccessible.
1 La machine est inaccessible.
2 Le protocole est inaccessible.
3 Le port est inaccessible.
4 Fragmentation nécessaire mais bit de non fragmentation positionné à 1.
5 Échec de routage de source.
6 Réseau de destination inconnu.
7 Machine destinataire inconnue.
8 Machine source isolée (obsolète)
9 Communication avec le réseau de destination administrativement interdite.
10 Communication avec la machine de destination administrativement interdite.
11 Réseau inaccessible pour ce type de service.
12 Machine inaccessible pour ce type de service.
13 Communication administrativement interdite par filtrage.

(4,0) Demande de limitation de production pour éviter la congestion du routeur qui envoie ce message.

(5,0-3) Demande de modification de route expédiée lorsqu'un routeur détecte qu'un ordinateur utilise une route non optimale, ce qui peut arriver lorsqu'un ordinateur est ajouté au réseau avec une table de routage minimale. Le message ICMP généré contient l'adresse IP du routeur à rajouter dans la table de routage de l'ordinateur. Les différents codes possibles ci-après expliquent le type de redirection à opérer par l'ordinateur.

0 Redirection pour un réseau.
1 Redirection pour une machine.
2 Redirection pour un type de service et réseau.
3 Redirection pour un type de service et machine.

(9,0) Avertissement de routeur expédié par un routeur.

(10,0) Sollicitation de routeur diffusé par une machine pour initialiser sa table de routage.

(11,0) TTL détecté à 0 pendant le transit du datagramme IP, lorsqu'il y a une route circulaire ou lors de l'utilisation de la commande `traceroute`.

(11,1) TTL détecté à 0 pendant le réassemblage d'un datagramme.

(12,0) Mauvaise en-tête IP.

(12,1) Option requise manquante.

(13-14,0) Requête (13) ou réponse (14) *timestamp*, d'estampillage horaire.

(15,0) et (16,0) devenues obsolètes.

(17-18,0) Requête (17) ou réponse (18) de masque de sous-réseau.

5 Les protocoles TCP et UDP.

On présente ici les deux principaux protocoles de la couche transport d'internet que sont les protocoles *TCP* (*Transmission Control Protocol*) et *UDP* (*User Datagram Protocol*). Tous les deux utilisent IP comme couche réseau, mais TCP procure une couche de transport fiable (alors même que IP ne l'est pas), tandis que UDP ne fait que transporter de manière non fiable des datagrammes.

5.1 Le protocole UDP.

Le protocole UDP (rfc 768) utilise IP pour acheminer, d'un ordinateur à un autre, en mode non fiable des datagrammes qui lui sont transmis par une application. UDP n'utilise pas d'accusé de réception et ne peut donc pas garantir que les données ont bien été reçues. Il ne réordonne pas les messages si ceux-ci n'arrivent pas dans l'ordre dans lequel ils ont été émis et il n'assure pas non plus de contrôle de flux. Il se peut donc que le récepteur ne soit pas apte à faire face au flux de datagrammes qui lui arrivent. C'est donc à l'application qui utilise UDP de gérer les problèmes de perte de messages, duplications, retards, déséquencement, ...

Cependant, UDP fournit un service supplémentaire par rapport à IP, il permet de distinguer plusieurs applications destinataires sur la même machine par l'intermédiaire des *ports*. Un port est une destination abstraite sur une machine identifié par un numéro qui sert d'interface à l'application pour recevoir et émettre des données. Par exemple,

```
...
discard         9/udp           sink null
systat          11/udp          users
daytime         13/udp
...
```

est un court extrait du fichier `/etc/services` de la machine `deneb.info-ua` dans lequel sont enregistrés les numéros de port utilisés par chaque application. On y voit que l'application daytime utilise le port 13[5]. Chaque datagramme émis par UDP est encapsulé dans un datagramme IP en y fixant à 17 la valeur du protocole (voir la section 4.1). Le format détaillé d'un datagramme UDP est donné dans la figure 19. Les *numéros de port* (chacun sur 16 bits) identifient

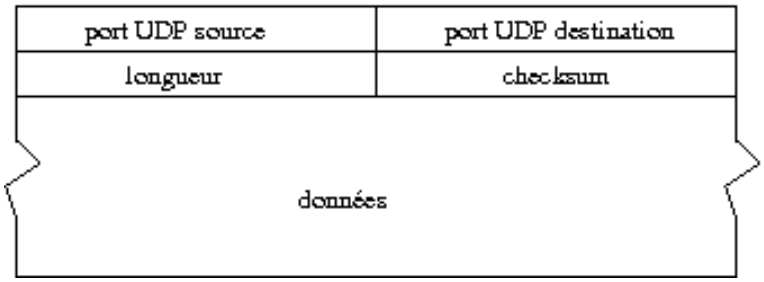


FIG. 19 – Structure d'un datagramme UDP.

les processus émetteur et récepteur. Le champ *longueur* contient sur 2 octets la taille de l'en-tête

[5]Certains numéros de port (comme 13) sont statiques c'est-à-dire que tous les systèmes les associent de la même manière aux applications et tous les logiciels applicatifs se conforment à cette règle. D'autres sont dynamiques et un numéro de port est attribué par les logiciels de communication à l'application au moment où celle-ci le demande.

et des données transmises. Puisqu'un datagramme UDP peut ne transmettre aucune donnée la valeur minimale de la longueur est 8. Le *checksum* est un total de contrôle qui est optionnel car il n'est pas indispensable lorsque UDP est utilisé sur un réseau très fiable. S'il est fixé à 0 c'est qu'en fait il n'a pas été calculé. De manière précise, UDP utilise l'en-tête et les données mais également une *pseudo-entête* pour aboutir à l'ensemble décrit figure 20. Cette pseudo en-tête

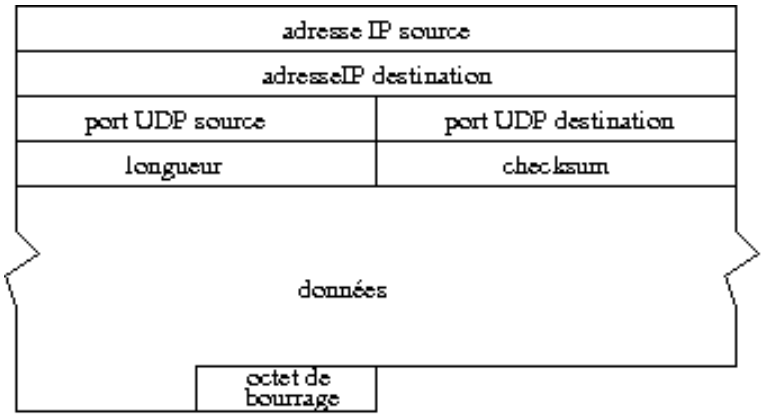


FIG. 20 – Champs utilisés pour le calcul du checksum UDP.

comprend les adresses IP source[6] et destination du datagramme ainsi qu'un éventuel octet de bourrage pour aboutir à un nombre d'octets total pair. À partir de cet ensemble, le total de contrôle est calculé de la même manière que dans le cas du datagramme IP (voir section 4.1). Si le résultat donne un checksum nul, son complément à 1, c'est-à-dire 65535 (16 bits 1), est en fait placé dans la zone de contrôle. Ce détail permet d'éviter la confusion avec le checksum nul qui indique qu'il n'a pas été calculé. Précisons enfin que la pseudo en-tête et l'octet de bourrage ne sont pas transmis et qu'ils n'interviennent pas dans le calcul du champ longueur. À la réception UDP utilise l'adresse IP de destination et l'adresse IP émettrice inscrite dans l'en-tête du datagramme IP pour calculer, de la même manière qu'à l'émission, une somme de contrôle qui permettra d'assurer que le datagramme est délivré sans erreur et à la bonne machine. Si une erreur de transmission est détectée, le datagramme UDP est détruit « en silence ». Sinon, UDP oriente les données du datagramme vers la file d'attente associée au numéro de port destination pour que l'application associée à celui-ci puisse les y lire.

5.2 Le protocole TCP.

Contrairement à UDP, TCP est un protocole qui procure un service de flux d'octets orienté connexion et fiable. Les données transmises par TCP sont encapsulées dans des datagrammes IP en y fixant la valeur du protocole à 6.

Le terme *orienté connexion* signifie que les applications dialoguant à travers TCP sont considérées l'une comme un *serveur*, l'autre comme un *client*, et qu'elles doivent établir une connexion avant de pouvoir dialoguer (comme dans le cas de l'utilisation du téléphone). Les ordinateurs vérifient donc préalablement que le transfert est autorisé, que les deux machines sont

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

prêtes en s'échangeant des messages spécifiques. Une fois que tous les détails ont été précisés, les applications sont informées qu'une connexion a été établie et qu'elles peuvent commencer leurs échanges d'informations. Il y a donc exactement deux extrémités communiquant l'une avec l'autre sur une connexion TCP[7]. Cette connexion est bidirectionnelle simultanée (*full duplex*) et composée de deux flots de données indépendants et de sens contraire. Il est cependant possible d'inclure dans l'en-tête de segments TCP d'une communication de A vers B des informations relatives à la communication de B vers A. Cette technique de superposition (*piggybacking*) permet de réduire le trafic sur le réseau.

Tout au long de la connexion, TCP échange un *flux d'octets* sans qu'il soit possible de séparer par une marque quelconque certaines données. Le contenu des octets n'est pas du tout interprété par TCP, c'est donc aux applications d'extrémité de savoir gérer la structure du flot de données.

Si elles sont trop volumineuses, les données à transmettre pour une application sont fractionnées en fragments dont la taille est jugée optimale par TCP. A l'inverse, TCP peut regrouper des données d'une application pour ne former qu'un seul datagramme de taille convenable de manière à ne pas charger inutilement le réseau. Cette unité d'information émise est appelée *segment*. Certaines applications demandent que les données soient émises immédiatement, même si le tampon n'est pas plein. Pour cela, elles utilisent le principe du *push* pour forcer le transfert. Les données sont alors émises avec un bit marquant cela pour que la couche TCP réceptrice du segment remette immédiatement les données à l'application concernée.

La *fiabilité* fournie par TCP consiste à remettre des datagrammes, sans perte, ni duplication, alors même qu'il utilise IP qui lui est un protocole de remise non fiable. Ceci est réalisé à l'aide de la technique générale de l'accusé de réception (ACK) présentée de manière simplifiée dans la figure 21 Chaque segment est émis avec un numéro qui va servir au récepteur pour envoyer

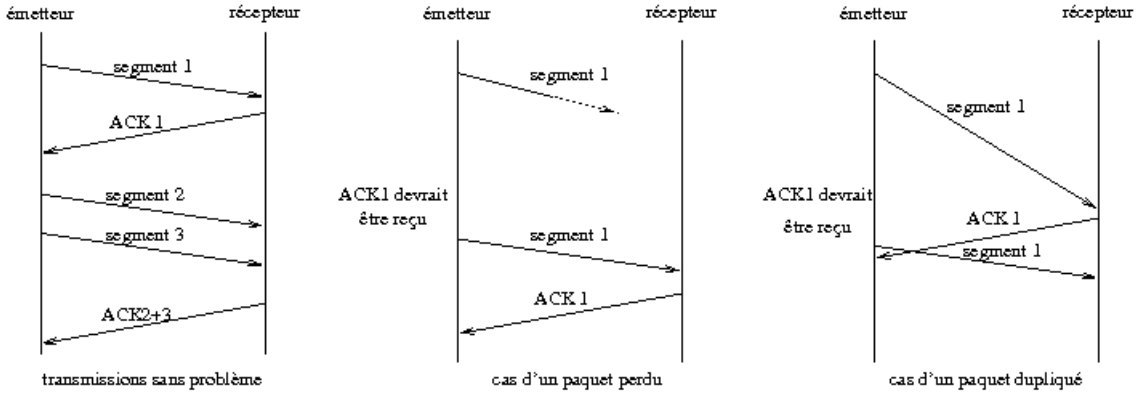


FIG. 21 – Échanges de segments TCP.

un accusé de réception. Ainsi l'émetteur sait si l'information qu'il voulait transmettre est bien parvenue à destination. De plus, à chaque envoi de segment, l'émetteur arme une temporisation qui lui sert de délai d'attente de l'accusé de réception correspondant à ce segment. Lorsque la temporisation expire sans qu'il n'ait reçu de ACK, l'émetteur considère que le segment s'est perdu[8] et il le réexpédie. Mais il se peut que la temporisation expire alors que le segment a été transmis sans problème, par exemple suite à un engorgement de réseau ou à une perte de

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

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

l'accusé de réception correspondant. Dans ce cas, l'émetteur réémet un segment alors que c'est inutile. Mais le récepteur garde trace des numéros de segments reçus, donc il est apte à faire la distinction et peut éliminer les doublons.

La figure 22 donne le format d'un segment TCP qui sert aux trois fonctionnalités de TCP : établir une connexion, transférer des données et libérer une connexion. L'en-tête, sans option,

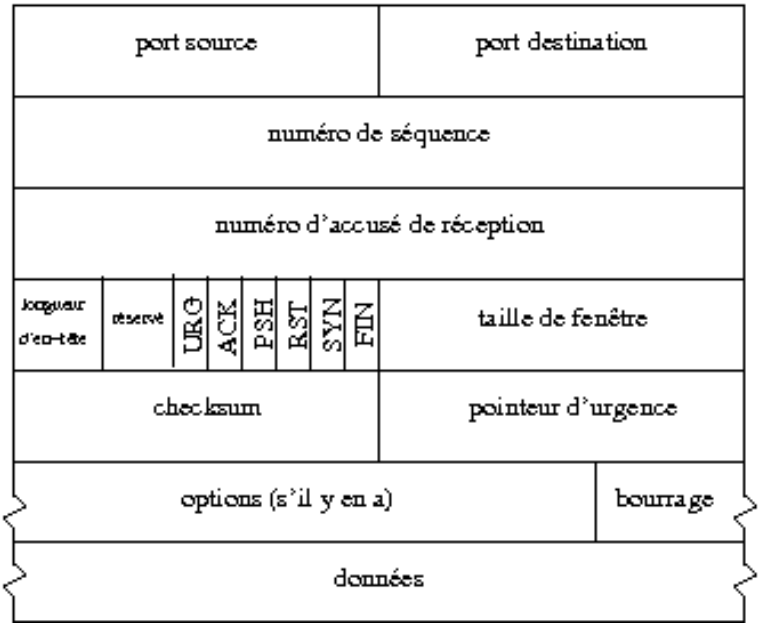


FIG. 22 – Format du segment TCP.

d'un segment TCP a une taille totale de 20 octets et se compose des champs suivants.

- Le *port source* et le *port destination* identifient les applications émettrice et réceptrice. En les associant avec les numéros IP source et destination du datagramme IP qui transporte un segment TCP on identifie de manière unique chaque connexion[9].
- Le *numéro de séquence*[10] donne la position du segment dans le flux de données envoyées par l'émetteur ; c'est-à-dire la place dans ce flux du premier octet de données transmis dans ce segment.
- Le *numéro d'accusé de réception* contient en fait le numéro de séquence suivant que le récepteur s'attend à recevoir ; c'est-à-dire le numéro de séquence du dernier octet reçu avec succès plus 1. De manière précise, TCP n'acquitte pas un à un chaque segment qu'il reçoit, mais acquitte l'ensemble du flot de données jusqu'à l'octet $k-1$ en envoyant un acquittement de valeur k. Par exemple, dans une transmission de 3 segments de A vers B, si les octets de 1 à 1024 sont reçus correctement, alors B envoie un ACK avec la valeur 1025. Puis, si le segment suivant contenant les octets de 1025 à 2048 se perd et que B reçoit d'abord correctement le segment des octets de 2049 à 3072, B n'enverra pas d'accusé de réception positif pour ce troisième segment. Ce n'est que lorsque B recevra le deuxième segment, qu'il pourra envoyer un ACK avec la valeur 3073, que A interprétera comme l'acquittement des deux derniers segments qu'il a envoyés. On appelle cela un acquittement cumulatif.

[9] Cette association (numéro IP, port) est appelée *socket*.

[10] C'est un entier non signé codé sur 32 bits qui retourne à 0 après avoir atteint la valeur $2^{31}-1$.

- La *longueur d'en-tête* contient sur 4 bits la taille de l'en-tête, y compris les options présentes, codée en multiple de 4 octets. Ainsi une en-tête peut avoir une taille variant de 20 octets (aucune option) à 60 octets (maximum d'options).
- Le champ *réservé* comporte 6 bits réservés à un usage ultérieur.
- Les 6 champs *bits de code* qui suivent permettent de spécifier le rôle et le contenu du segment TCP pour pouvoir interpréter correctement certains champs de l'en-tête. La signification de chaque bit, quand il est fixé à 1 est la suivante.
  - *URG*, le *pointeur de données urgentes* est valide.
  - *ACK*, le champ *d'accusé de réception* est valide.
  - *PSH*, ce segment requiert un *push*.
  - *RST*, réinitialiser la connexion.
  - *SYN*, synchroniser les numéros de séquence pour initialiser une connexion.
  - *FIN*, l'émetteur a atteint la fin de son flot de données.
- La *taille de fenêtre* est un champ de 16 bits qui sert au contrôle de flux selon la méthode de la fenêtre glissante. Il indique le nombre d'octets (moins de 65535) que le récepteur est prêt à accepter. Ainsi l'émetteur augmente ou diminue son flux de données en fonction de la valeur de cette fenêtre qu'il reçoit.
- Le *checksum* est un total de contrôle sur 16 bits utilisé pour vérifier la validité de l'en-tête et des données transmises. Il est obligatoirement calculé par l'émetteur et vérifié par le récepteur. Le calcul utilise une pseudo-entête analogue à celle d'UDP (voir la section 5.1).
- Le *pointeur d'urgence* est un offset positif qui, ajouté au numéro de séquence du segment, indique le numéro du dernier octet de donnée urgente. Il faut également que le bit *URG* soit positionné à 1 pour indiquer des données urgentes que le récepteur TCP doit passer le plus rapidement possible à l'application associée à la connexion.
- L'*option* la plus couramment utilisée est celle de la taille maximale du segment TCP qu'une extrémité de la connexion souhaite recevoir. Ainsi, lors de l'établissement de la connexion il est possible d'optimiser le transfert de deux manières. Sur un réseau à haut débit, il s'agit de remplir au mieux les paquets, par exemple en fixant une taille qui soit telle que le datagramme IP ait la taille du MTU du réseau. Sinon, sur un réseau à petit MTU, il faut éviter d'envoyer des grands datagrammes IP qui seront fragmentés, car la fragmentation augmente la probabilité de pertes de messages.

L'*établissement* et la *terminaison* d'une connexion suit le diagramme d'échanges de la figure 23. L'extrémité demandant l'ouverture de la connexion, est le *client*. Il émet un segment SYN (où le bit SYN est fixé à 1) spécifiant le numéro de port du *serveur* avec lequel il veut se connecter. Il expédie également un numéro de séquence initial N. Cette phase est appelée ouverture active et « consomme »un numéro de séquence. Le serveur répond en envoyant un segment dont les bits ACK et SYN sont fixés à 1. Ainsi, dans un même segment il acquitte le premier segment reçu avec une valeur de ACK=$N+1$ et il indique un numéro de séquence initial. Cette phase est appelée ouverture passive. Le client TCP doit évidemment acquitter ce deuxième segment en renvoyant un segment avec ACK=$P+1$[11].

La terminaison d'une connexion peut être demandée par n'importe quelle extrémité et se compose de deux *demi-fermetures* puisque des flots de données peuvent s'écouler simultanément dans les deux sens. L'extrémité qui demande la fermeture (le client dans l'exemple de la figure 23 émet un segment où le bit FIN est fixé à 1 et où le numéro de séquence vaut N'. Le récepteur du segment l'acquitte en retournant un ACK=$N'+1$ et informe l'application de la demi-fermeture

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

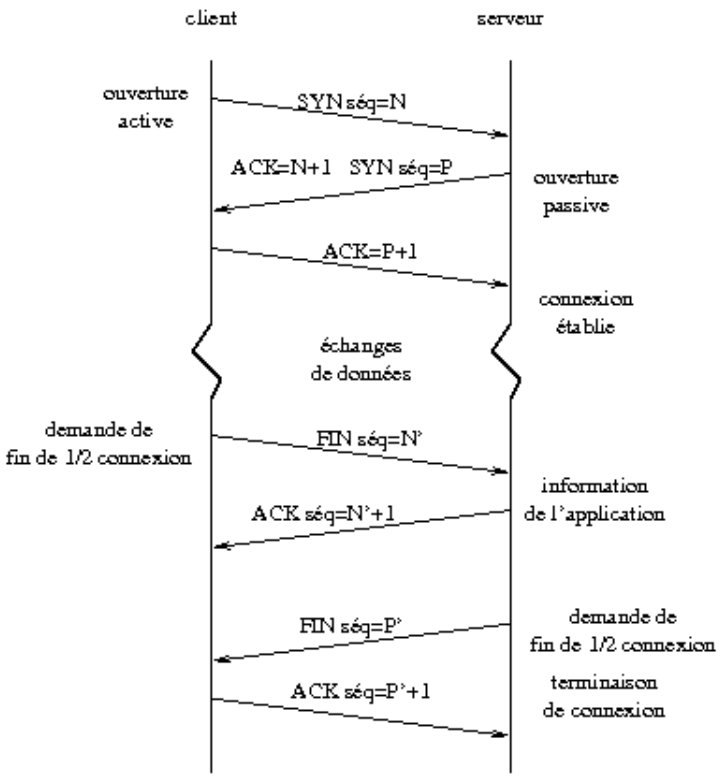


FIG. 23 – Établissement et terminaison d'une connexion TCP.

de la connexion. À partir de là, les données ne peuvent plus transiter que dans un sens (de l'extrémité ayant accepté la fermeture vers l'extrémité l'ayant demandée), et dans l'autre seuls des accusés de réception sont transmis. Quand l'autre extrémité veut fermer sa demi-connexion, elle agit de même que précédemment ce qui entraîne la terminaison complète de la connexion.

Le *transfert de données* de TCP est de deux types : le transfert *interactif* dans lequel chaque segment transporte très peu d'octets, voire un seul, et le transfert *en masse* où chaque segment transporte un maximum d'octets. Un exemple de transfert interactif est celui généré par la commande `rlogin` lancée depuis un client vers un serveur. Dans ce cas de figure, tous les caractères tapés par l'utilisateur sur le client sont envoyés vers le serveur en utilisant un caractère par segment, et ils sont ensuite renvoyés en sens inverse par le serveur pour un écho sur l'écran du client. Tous les segments échangés dans ce cas là ont leur bit *PSH* fixé à 1. Or, tout segment doit être acquitté dans un sens comme dans l'autre ; ce qui devrait amener au diagramme d'échanges illustré dans le a) de la figure 24. En fait, TCP gère ce type d'échange avec la procédure d'*acquittement retardé* qui consiste à envoyer l'acquittement en l'incluant dans un segment qui transporte également des données comme illustré dans le b) de la figure 24. Pour cela l'acquittement est généralement retardé de 200ms dans le but d'attendre d'éventuelles données à transmettre. Cependant, dans ce contexte la frappe de la commande `date` sur le client provoquerait quand même l'échange de 15 datagrammes IP de 41 octets[12], pour transporter à chaque fois seulement 1 octet utile. Ce type de situation est peu gênant sur un LAN mais devient très vite préjudiciable au bon fonctionnement d'un WAN. Une solution à ce problème est l'*algorithme de Nagle* (RFC 896) qui spécifie qu'une connexion TCP ne peut avoir seulement un petit segment non encore acquitté. Ainsi, si un réseau a un fort débit et n'est pas du tout

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

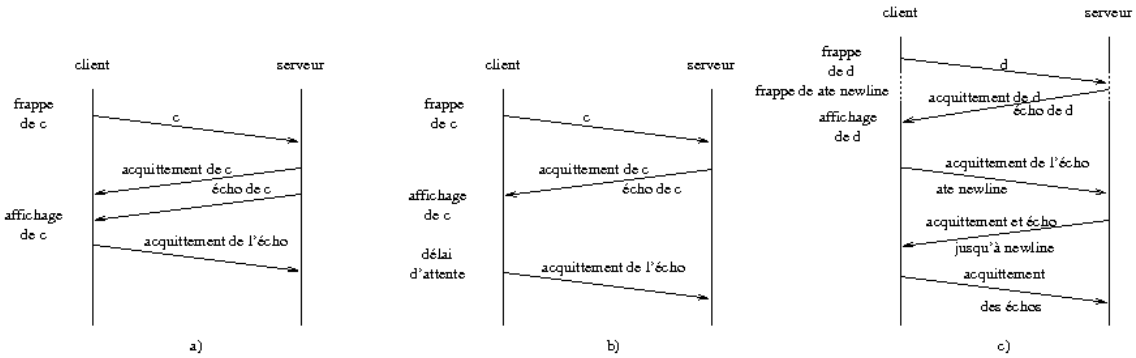


FIG. 24 – Échange de données interactif.

chargé la procédure sera celle du b) de la figure 24. Par contre, dès que le temps de transfert est non négligeable, les acquittements vont arriver plus lentement que la frappe des caractères par l'utilisateur du poste client. Dans ce cas, la couche TCP du client va accumuler de petits volumes de données (quelques caractères) et les envoyer dans le même segment TCP diminuant ainsi considérablement le nombre d'échanges comme illustré dans le c) de la figure 24. Dans les cas où de petits messages doivent être remis sans délai (mouvement de souris dans le cas d'un serveur X) l'algorithme de Nagle doit être invalidé pour donner une impression de temps réel.

Dans le cas d'un transfert de données en masse, TCP utilise la technique de la *fenêtre glissante* pour contrôler le flux des échanges. Ceci est primordial quand un micro ordinateur communique avec un gros ordinateur, sinon le tampon d'entrée du micro sera très vite saturé. Ceci consiste en un contrôle de flux de *bout en bout*. Mais il s'agit aussi de réguler le trafic en fonction de la charge des routeurs et du débit des réseaux traversés. On rappelle que l'ensemble d'un flux de données unidirectionnel d'une machine A vers une machine B est constitué d'une séquence d'octets tous numérotés individuellement. La fenêtre glissante va consister à fixer quels sont les octets appartenant à ce flux que A peut émettre comme c'est illustré dans la figure 25. Dans cet exemple la fenêtre couvre les octets de 4 à 9, car la taille de la fenêtre courante est 6

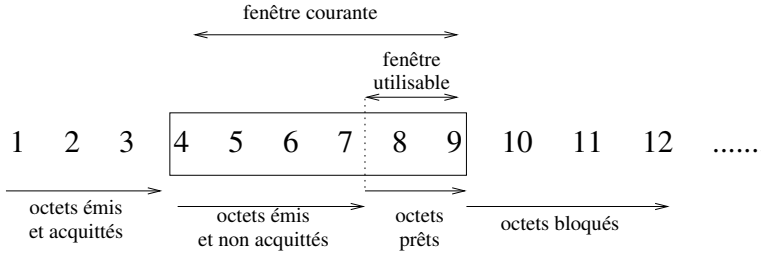


FIG. 25 – Fenêtre glissante de TCP.

et que tous les octets jusqu'au troisième inclus ont été émis et acquittés. A tout instant TCP calcule sa fenêtre utilisable qui est constituée des octets présents dans la fenêtre et non encore envoyés. Ces octets sont généralement immédiatement transmis. Pour le flot de A vers B, la taille de la fenêtre est contrôlée par B qui envoie dans chacun de ses accusés de réception la taille de la fenêtre qu'il désire voir utiliser. Si la demande exprime une augmentation, A déplace le bord droit de sa fenêtre courante et émet immédiatement les octets qui viennent d'y entrer. Si la demande exprime une diminution, il est déconseillé de déplacer réellement le bord droit de la fenêtre vers la gauche. Ce rétrécissement est opéré lors des glissements de la fenêtre vers

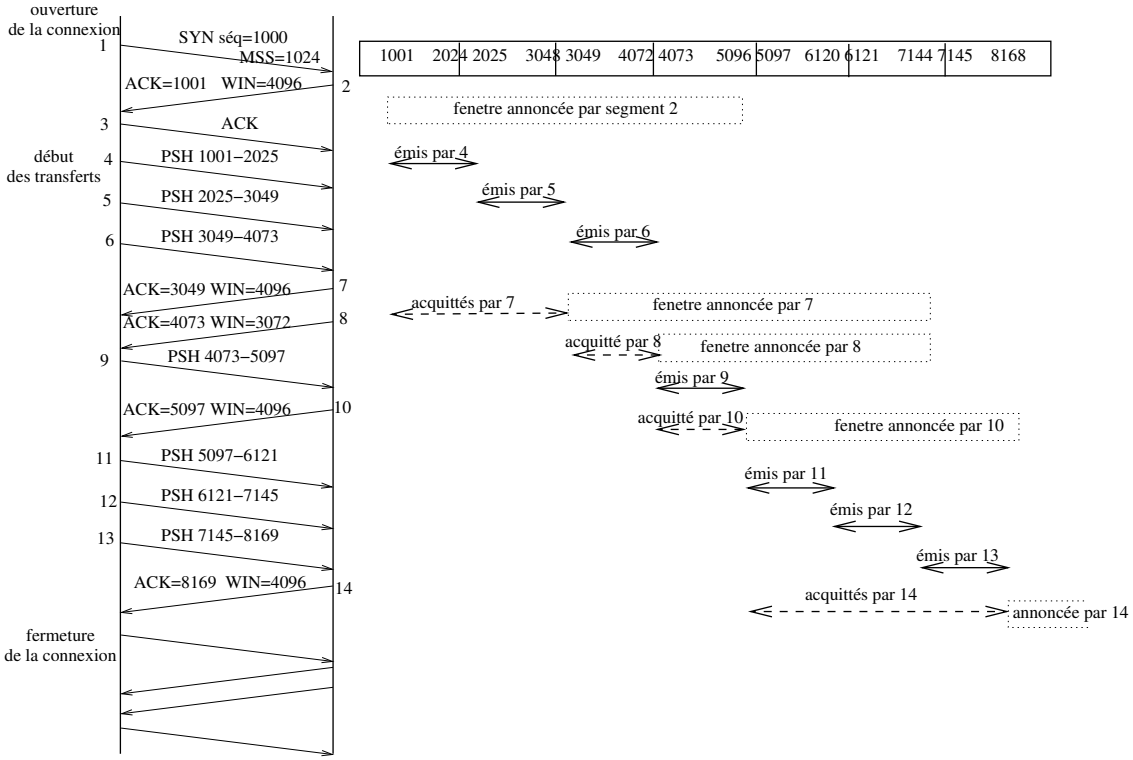


FIG. 26 – Exemple d'évolution de fenêtre glissante.

la droite avec l'arrivée des accusés de réception.

La figure 26 illustre[13] les échanges de segments entre un émetteur A et un récepteur B et l'évolution de la fenêtre glissante de A suivant les indications de B.

6 Réseau privé et accès à l'internet

Le réseau du département d'informatique de l'université d'Angers schématisé dans la figure 27 est un exemple de réseau IP privé. C'est-à-dire que l'ensemble des machines qui le composent ont reçu une adresse IP dite privée (« non routable ») comme défini à la section 1 et appartenant à l'un des deux réseaux `172.20.0.0/16`, ou `192.168.99.0/24`. Seule la machine `tines` est une passerelle dont l'une des interfaces est publique `193.49.146.121`. Étant donné cet adressage privé, aucune machine du réseau privé n'est directement joignable de l'extérieur. Par conséquent, sans configuration supplémentaire, elle ne pourrait pas non plus établir une connexion vers l'extérieur, puisque l'adresse d'émission des paquets IP qu'elle émettrait vers l'extérieur ne pourrait en aucun cas servir « d'adresse de retour » pour lui renvoyer des informations.

La suite décrit les différents moyens mis en œuvre pour rendre possibles certaines communications entre les postes du réseau privé et d'autres postes sur internet, tout en assurant un maximum de sécurité (éviter les intrusions, contrôler les accès en entrée et en sortie, ...). Ainsi, avec seulement 1 adresse IP publiques il devient possible de connecter à l'internet un grand nombre de machines.

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

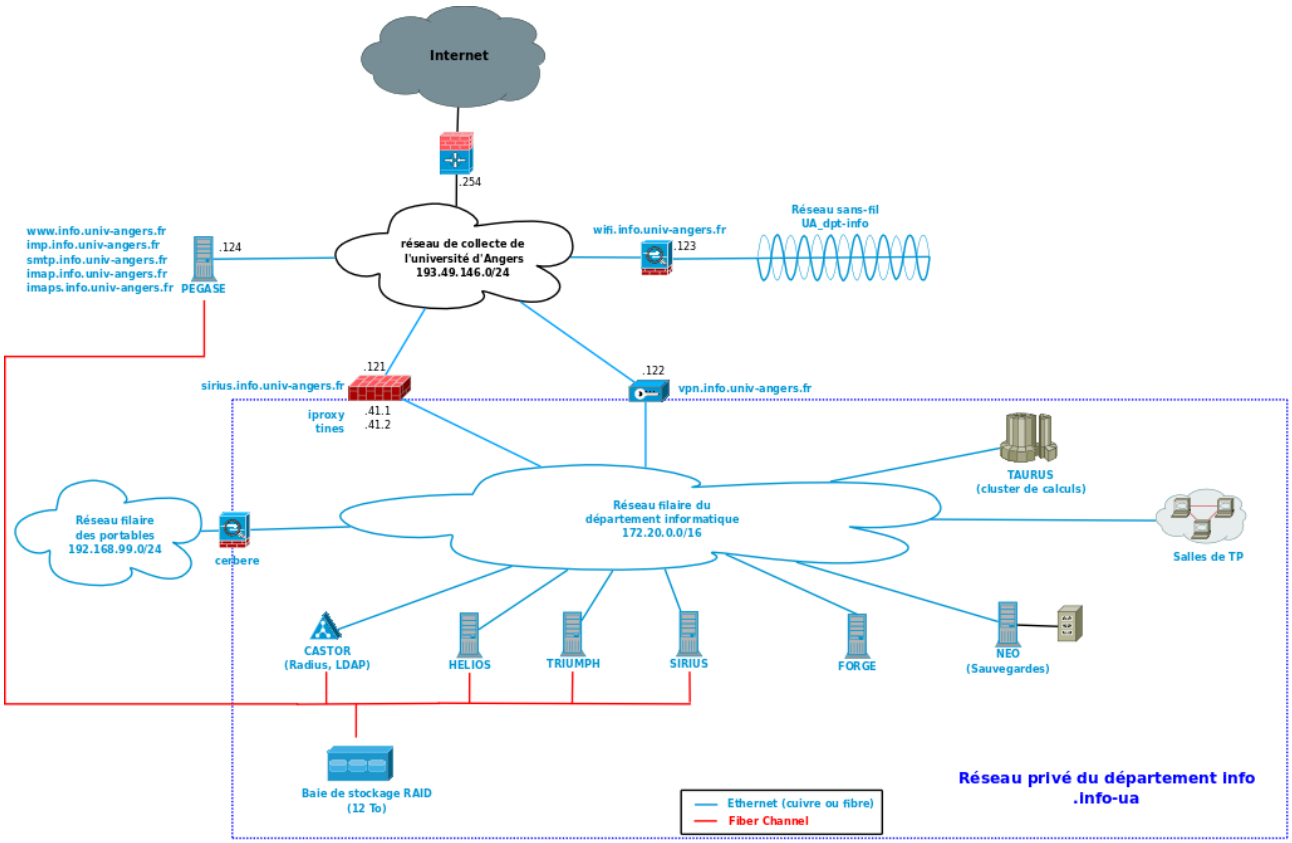


FIG. 27 – Schéma du réseau du département d'informatique de l'Université d'Angers

En premier lieu il existe un système de proxy applicatif hébergé par la machine tines. Le principe du mandataire ou « proxy »web permet à tout navigateur (moyennant sa configuration indiquant que le proxy est le service `iproxy.info-ua`) du réseau privé d'accéder aux sites webs externes. Cela consistera à faire relayer la requête par le proxy comme illustré dans la figure 28. Ici le relai est dit applicatif ou de niveau 7 (en référence au modèle OSI) car c'est au niveau de l'application elle-même (ici HTTP) que s'effectue le travail. Les mêmes principes peuvent s'appliquer pour la mise en place d'un proxy FTP ou pour toute autre application basée sur TCP ou UDP.

Cependant, si l'on veut gérer globalement l'accès à l'internet des postes internes on peut travailler à un plus bas niveau (TCP et UDP)en utilisant des techniques de translation d'adresses de réseau dite NAT (Network Address Translation). Nous ne décrirons pas le principe de NAT statique (voir `http://www.usenet-fr.net/fur/comp/reseaux/nat.html`) car il ne permet pas de faire d'économies d'adresses IP publiques puisqu'il associe chaque adresse privée à une adresse publique.

De son côté la NAT dynamique, dite IP masquerading, permet d'associer plusieurs adresses privées à une même adresse publique en agissant sur les ports TCP ou UDP des connexions qu'ils relaient comme ceci est décrit dans la figure 29. Évidemment, pour les protocoles ou applications qui ne s'appuient pas sur TCP ou UDP, comme ICMP par exemple, il faut mettre en place un relayage particulier.

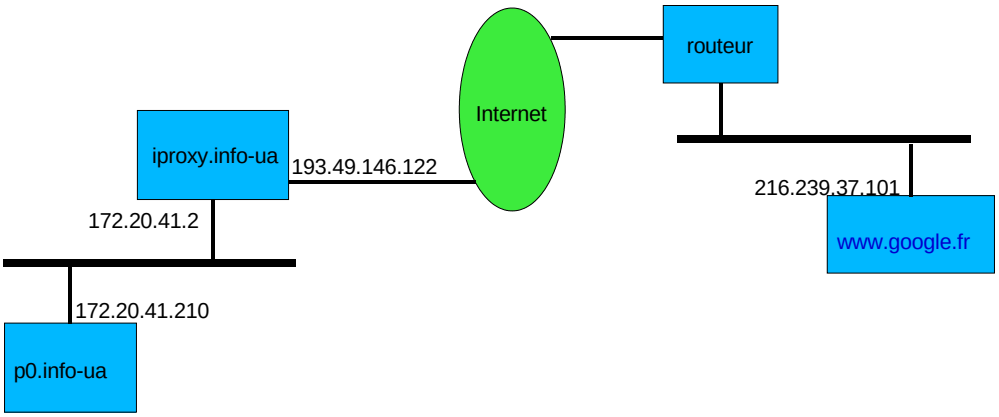
Mais, la principale limitation de l'IP masquerading est qu'il ne permet la communication entre une machine du réseau privé et une sur l'internet que dans le cas où la connexion est initiée de l'intérieur. D'un côté, cela augmente la sécurité du réseau interne. D'un autre, cela empêche toute offre de service à partir d'une machine interne. Cette limitation d'accès depuis l'extérieur

peut être levée si l'on utilise la technique de portforwarding décrite dans la figure 30. Dans ce cas, la machine passerelle va relayer vers une machine interne assurant le service demandé toutes les demandes qui lui parviennent sur le port correspondant à l'application demandée. Dans le cas du réseau de la figure 27 ce système est implanté sur la machine `tines`. Ainsi, si l'on cherche à établir une connexion ssh depuis l'internet vers la machine `sirius.info.univ-angers.fr` (nom symbolique pour l'adresse IP publique `193.49.146.121`) cela aboutira à établir cette connexion ssh vers la machine du réseau privé `sirius.info-ua` d'adresse IP privée `172.20.41.7`.

Proxy (mandataire) web / 1

Comment accède-t-on à un site ?

Exemple de l'accès à google depuis un poste étudiant du département d'info de l'Université d'Angers via le proxy web



Proxy (mandataire) web / 2

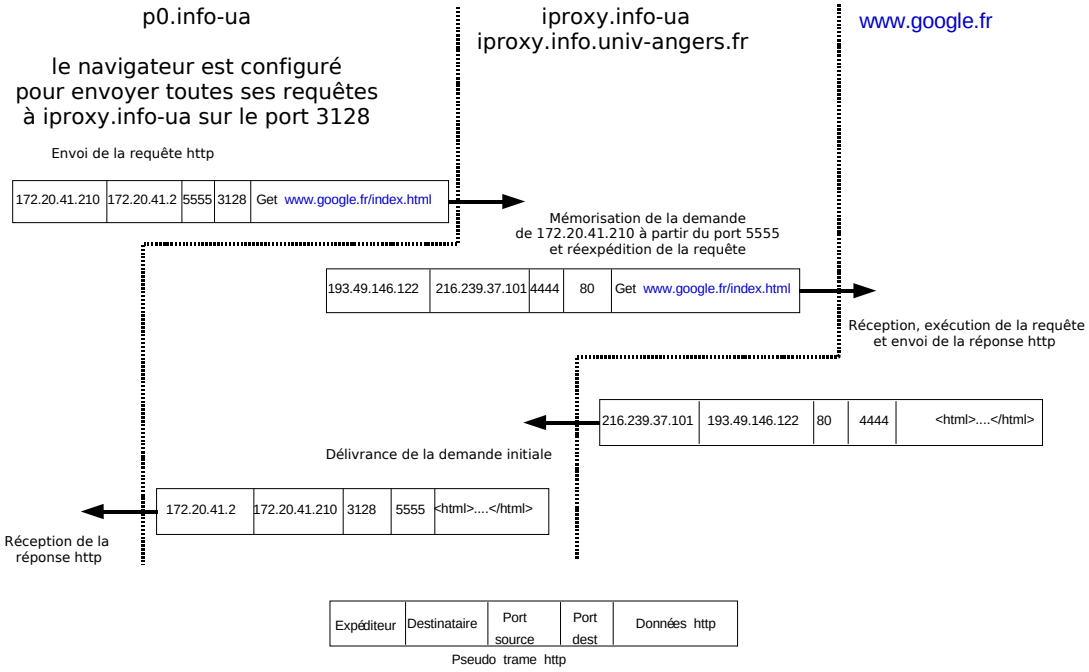
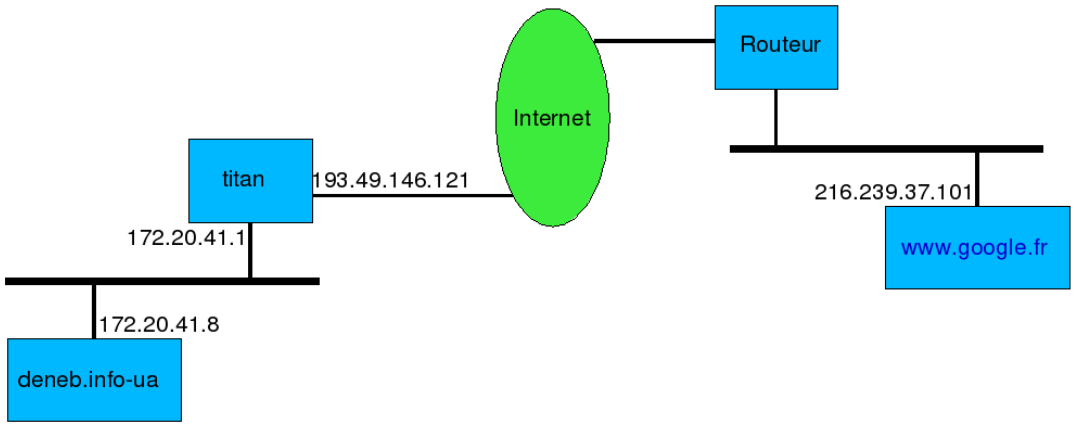


FIG. 28 – Proxy web

NAT dynamique (IP masquerading) / 1

Comment sort-on ?

Exemple de l'accès à google (sans utilisation de proxy web) depuis l'un des serveurs du département d'info de l'Université d'Angers



NAT dynamique (IP masquerading) / 2

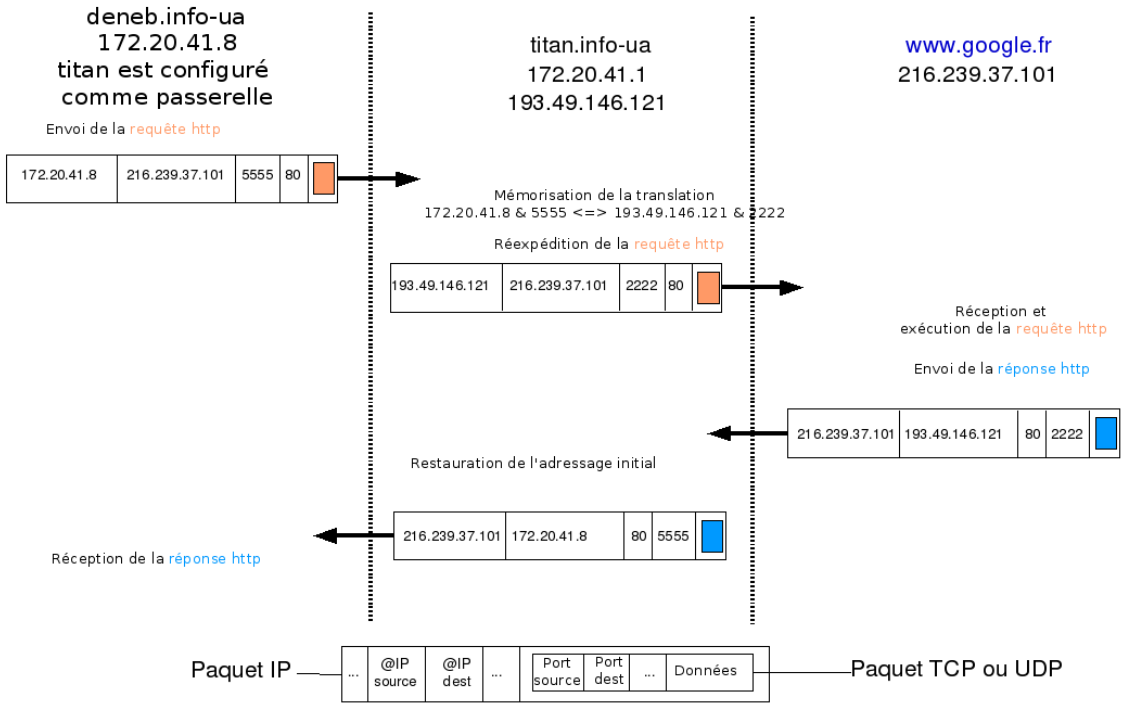
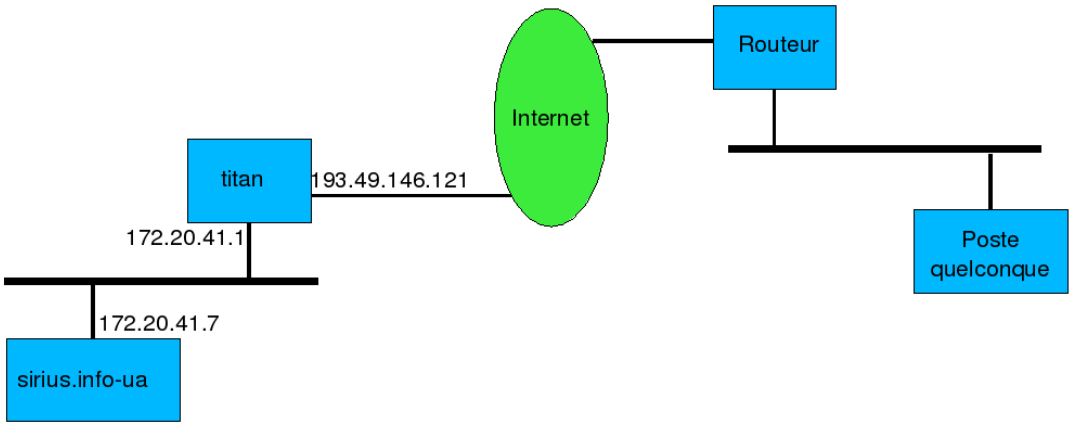


FIG. 29 – IP masquerading

NAT dynamique (port forwarding) / 1

Comment entre-t-on ?

Exemple de l'accès ssh au serveur sirius depuis un poste situé sur internet.



NAT dynamique (port forwarding) / 2

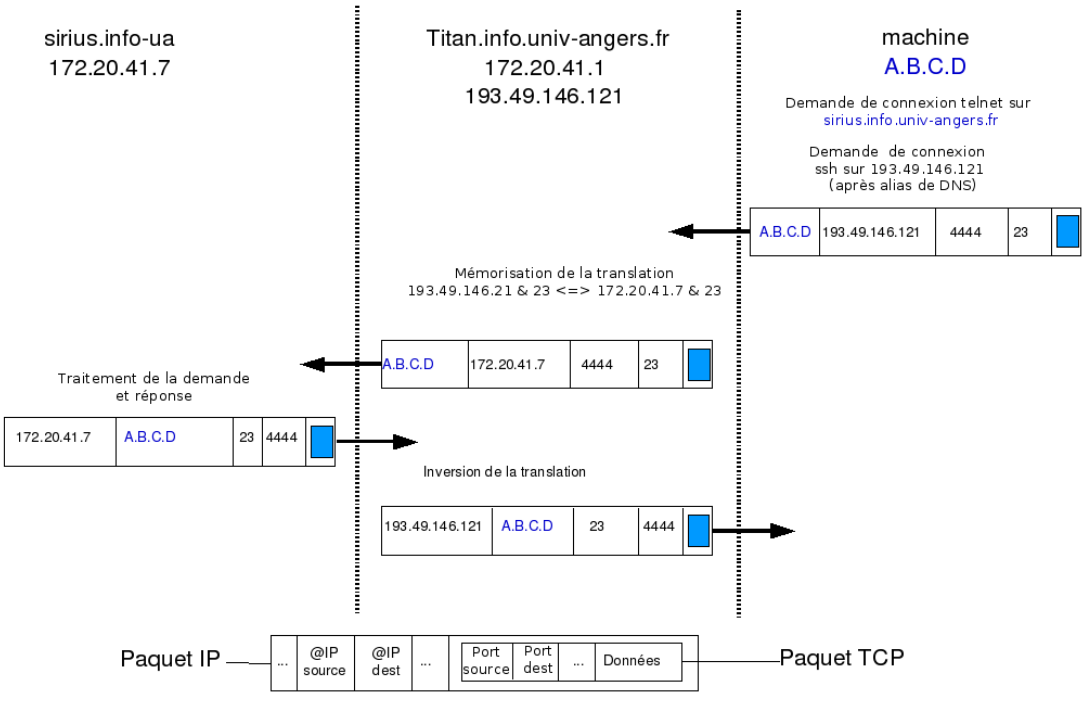


FIG. 30 – Port forwarding