TCP/IP

Avant propos

Dans le chapitre précédent, nous avons eu l'occasion de voir comment un réseau est construit et quels protocoles de bas niveau (niveau 1 du modèle OSI) sont employés pour transporter ses données "brutes" sur le réseau.

Le réseau Ethernet est le plus employé (Avec ATM pour les opérateurs Télécoms). Ici, nous allons nous intéresser au protocole situé juste au dessus, du moins au plus utilisé d'entre eux: TCP/IP. Ce protocole est en effet omniprésent sur le Net.

Une bonne compréhension de TCP/IP est nécessaire si l'on souhaite d'une part savoir comment les données transitent sur les réseaux et, d'autre part, ne pas être trop perdu dans les règles "sanitaires" qui permettent de mettre nos machines connectées le plus possible à l'abri des agressions. Il existe plusieurs outils de protection (Firewalls en anglo saxon, Parefeu en français), mais ces outils n'ont qu'un effet psychologique, le plus souvent néfaste d'ailleurs, si l'on n'a aucune compétence pour les paramétrer de façon efficace. Le problème de la sécurité est abordé dans un autre chapitre[1] sur ce site.

Au programme :

- les adresses logiques de l'Internet Protocol (couche 3 du modèle OSI)
- Les modes connecté (TCP) et non connecté (UDP) (couche 4 du modèle OSI)
- Les protocoles applicatifs (HTTP, FTP, SMTP, POP etc.) (couche 7 du modèle OSI)

1 http://christian.caleca.free.fr/securite/index.html

Plan du chapitre

Les protocoles

C'est quoi un protocole ?

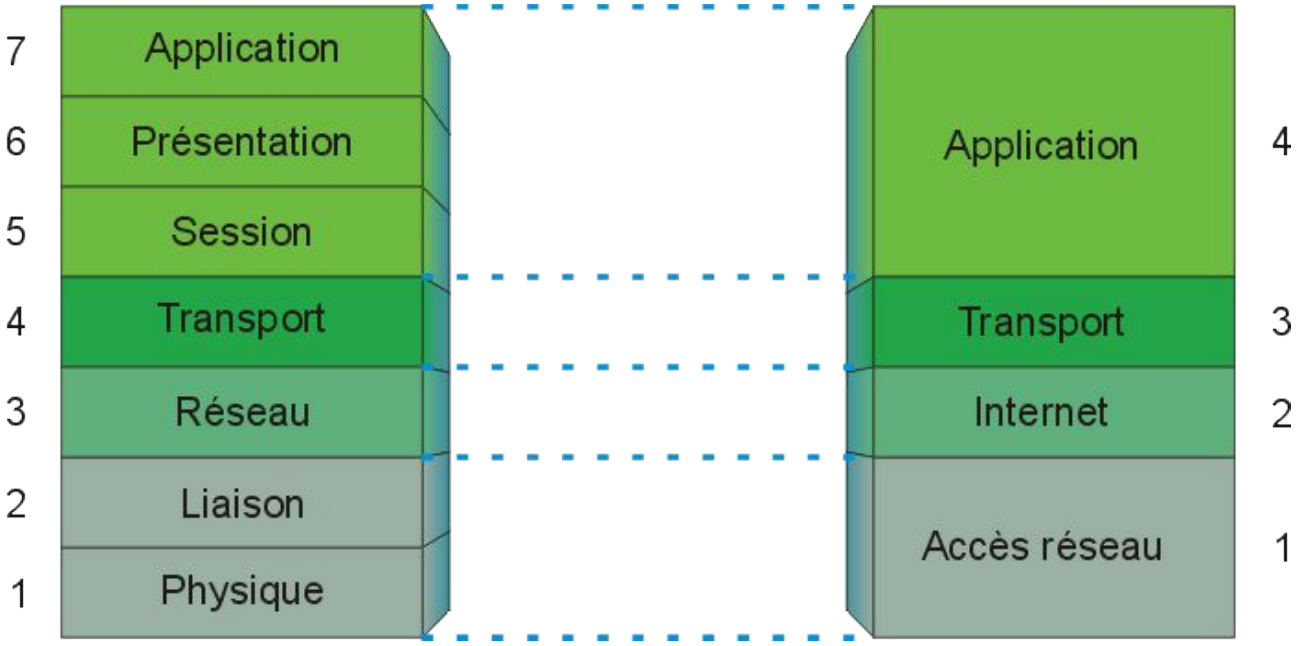
Essayons d'en donner une définition satisfaisante...

C'est un mode opératoire qui doit être commun à tous les éléments qui désirent communiquer entre eux. Il n'y a pas de communication possible sans avoir recours à un protocole. Bien entendu, le protocole doit être adapté au type de communication que l'on souhaite mettre en oeuvre.

Nous passons notre vie à utiliser des protocoles, heureusement sans en être conscients la plupart du temps.

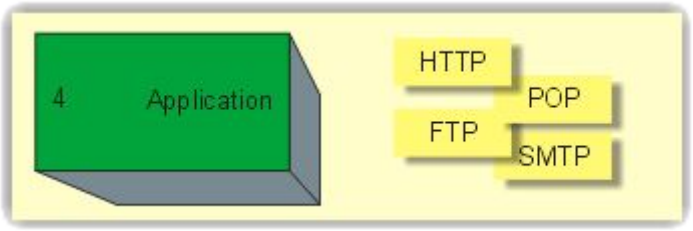
Rappel

Le modèle OSI définit sept couches. TCP/IP est basé sur le modèle DOD, qui ne comporte que quatre couches, mais en cohérence avec le modèle OSI.



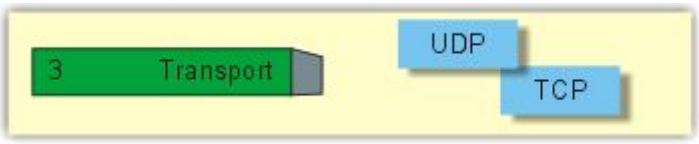
Les principaux protocoles rencontrés sur un réseau TCP/IP

Organisation hiérarchique



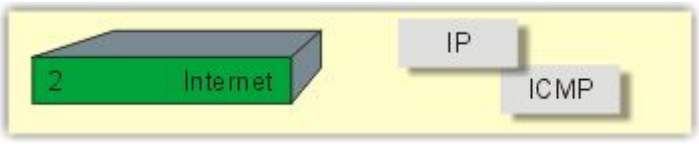
Nous trouvons ici les protocoles applicatifs. Ce sont des protocoles de haut niveau, destinés à permettre le dialogue entre applications serveurs et clientes.

HTTP, FTP, POP et SMTP sont loin d'être les seuls. Ce sont cependant ceux que les internautes utilisent le plus souvent. Parmi l'un des plus "dangereux", il y a TELNET qui permet de piloter une machine à distance.



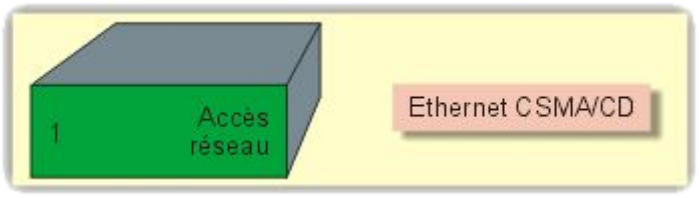
Ici, ce sont les protocoles orientés transport de données. UDP est dit "sans connexion" et TCP "est dit "avec connexion". Nous verrons plus loin ce que ceci veut dire.

Ces protocoles permettent à ceux de la couche 4 de transporter leurs données de façon fiable.



Ce sont ici des protocoles de haut niveau de la couche réseau. IP permet le routage des informations entre réseaux, c'est ici que l'adresse IP est utilisée.

ICMP est un protocole de "contrôle" il met à disposition des outils de dépistage d'erreur et de signalisation. C'est un protocole important qui mérite que l'on s'y arrête. Nous en reparlerons plus en détail.



Protocole de plus bas niveau sur le réseau, il assure la bonne gestion du médium (détection de collisions) et permet l'acheminement des informations entre émetteur et destinataire au niveau des adresses MAC.

IP s'appuie dessus bien évidement.

Ethernet

Le vocable "Ethernet" est souvent employé à contre sens. Peut-être n'est-il pas inutile de préciser un peu, même si, pour l'utilisateur (qui travaille sur la couche supérieure), ce qu'il se passe sur la couche 1 n'a pas beaucoup de répercussions.

Le mot "Ethernet" fait référence au support de propagation des informations utilisé. Historiquement,

de trois types (mais d'autres peuvent être utilisés) :

- Coaxial épais
- Coaxial fin (RG58)
- Paire torsadée.

Pour être tout à fait précis, la norme qui décrit les réseaux de type Ethernet qui sont utilisés sur la majorité des réseaux locaux est la norme IEEE 802.3.

Cette norme décrit dans le détail le fonctionnement du réseau sur les supports cités précédemment. Elle définit entre autre, le protocole d'émission de données utilisé : le CSMA/CD[2] persistant 1 (qui n'est pas le plus performant).

Remarque fine...

Les réseaux France Télécom ne sont pas des réseaux IEEE 802.3, mais des réseaux ATM (Asynchronous Transfer Mode). Le réseau ATM a été développé dans l'optique d'un transport de données de natures diverses (voix, vidéo, informatique...). ATM est capable de gérer finement le partage des ressources d'une dorsale.

Bien que cette technologie soit pas mal controversée, c'est tout de même elle qui est utilisée par notre opérateur "historique" (et d'autres également). Cependant, les trames IEEE 802.3 peuvent être encapsulées sur de l'ATM, TCP/IP peut s'appuyer sur ATM, si bien que nous autres, utilisateurs, "voyons" tout de même un réseau classique de l'Internet. En fait, le Com21 est connecté sur un réseau ATM via le câble.

IP

Internet Protocol.

C'est le protocole dont on parle le plus, il est en effet directement impliqué dans la configuration réseau de l'hôte. C'est lui qui, en fonction de l'adresse IP du destinataire acheminera l'information sur la bonne route.

- Les considérations relatives à la topologie d'une adresse IP sont vues un peu plus loin dans ce chapitre.
- Les concepts du routage sont vus dans le chapitre suivant sur ce site[3].

ICMP

Internet Control Message Protocol.

En termes de sécurité, ce protocole fait peur à beaucoup de monde (parfois à juste titre d'ailleurs), il est cependant fondamental pour le bon fonctionnement de l'Internet. C'est grâce à ce protocole que les anomalies de fonctionnement peuvent être signalées à l'émetteur, afin qu'il puisse essayer d'y remédier.

ICMP génère des messages de types différents, selon la nature du problème à traiter :

2 http://christian.caleca.free.fr/reseaux/software.htm#csmacd
3 http://christian.caleca.free.fr/routage/index.html

Valeur	Nom	Description
0	Réponse d'écho	Rien de plus que la réponse à un PING.
3	Destination inaccessible	C'est un message intéressant, parce qu'il permet à celui qui le reçoit d'être informé que l'hôte avec lequel il veut communiquer n'est pas accessible. Ça peut souvent éviter à une application de rester "plantée" à attendre une réponse qui ne viendra pas.
4	Étranglement de la source	Principalement utilisés par les routeurs, ce signal permet d'expliquer à un hôte qui parle un peu trop qu'il faut qu'il se taise, parce qu'il inonde la file d'attente.
5	Redirection nécessaire	Information utile pour la mise à jour des tables de routage.
8	Demande d'écho	C'est la question posée à un hôte par la commande PING.
11	TTL Expiré	Un paquet est toujours émis avec une durée de vie. Cette durée de vie est décrémentée à chaque nœud qui traite le paquet (d'une durée minimum d'une seconde, ou du temps qu'a mis la paquet à traverser le nœud). Si le paquet arrive en fin de vie, il est jeté et un message ICPM de type 11 est envoyé à l'émetteur. Cette propriété est utilisée dans la commande "tracert" ("traceroute" sur Linux) pour calculer les temps d'accès sur les diverses passerelles du chemin parcouru.
12	Problème de paramètre	Ce message indique qu'il y a une erreur dans le champ d'en-tête du paquet. Ce problème ne peut normalement arriver que dans le cas d'un bug du logiciel.
13	Requête d'horodatage	Assez similaire à la requête d'écho, avec en plus le marquage de l'heure. Ce type d'écho permet de connaître l'heure d'arrivée de la requête et l'heure de départ de la réponse sur l'hôte cible.
14	Réponse de d'horodatage	
17	Requête de masque d'adresse	Ces messages sont utilisés pour effectuer des tests au sein d'un réseau ou d'un sous-réseau.
18	Réponse de masque d'adresse	

Les protocoles de la couche transport permettent, comme l'indique le nom de la couche, de mettre à disposition des niveaux supérieurs des outils de transport de données fiables.

Les deux modes de transfert

Il existe deux modes de transfert :

Le mode connecté (TCP)

Dans ce mode, il se met en place un processus de "handshake" (poignée de main) entre le client et le serveur. Ce processus permet d'établir un dialogue à propos du transfert de données. Il y a des accusés réception, des demandes d'émission etc. qui permettent aux applications de savoir exactement où en est le processus de transfert de données.

Ce protocole est très robuste et permet un transfert de données dans de bonnes conditions. Voir les détails plus loin dans ce chapitre[4].

Le "handshake" est un concept fondamental dans un protocole de dialogue robuste. En gros, ça veut dire : "Chaque fois que tu envoies un message à son destinataire, assures-toi qu'il l'a reçu et compris"

La lettre recommandée avec accusé de réception est un bon exemple de mode connecté. Si l'émetteur reçoit l'accusé réception, alors il est certain que sa lettre est arrivée à destination.

Le mode non connecté (UDP)

C'est un mode simple, de type "on envoie les données et on espère qu'elles arriveront". Il n'y a pas de "connexion", au sens où on l'a vu pour le mode connecté. En revanche, il est possible de mettre en place un processus d'acquittement.

Ce mode est utilisé, par exemple, pour les requêtes DNS. Il offre l'avantage d'être moins gourmand en ressources, mais ne peut être efficace pour un transfert de fichiers et en général, pour les transferts de données volumineuses.

Là aussi, vous aurez plus de détails un peu plus loin sur ce site[5].

Dans ce mode, il n'y a pas de "handshake". Une lettre simple et ici un bon exemple. L'émetteur ne reçoit à priori aucune confirmation de réception.

Les protocoles d'application utilisant TCP ou UDP

Les protocoles d'application sont des protocoles de haut niveau, adaptés aux besoins d'applications spécifiques. Ils s'appuient sur UDP et TCP pour permettre le transfert d'informations entre une application serveur et ses applications clientes.

Il en existe un grand nombre, nous allons effectuer un rapide tour d'horizon de ceux qui sont le plus souvent utilisés.

4 Aller au mode connecté
5 Aller au mode non connecté

HTTP	***Hyper Text Transfert Protocol*** Ce protocole est utilisé pour la navigation web entre un serveur HTTP et un butineur. Le protocole assure (normalement) qu'un client comme Internet Explorer ou Netscape Communicator peut envoyer des requêtes et recevoir les réponses de serveurs HTTP comme APACHE ou Internet Information Server sans problèmes particuliers. Les ennuis viennent du fait que les clients supportent bien souvent des extensions "propriétaires" du protocole. Ces extensions sont d'ailleurs la plupart du temps entérinées dans les versions successives du protocole, c'est comme ça que tout évolue.
FTP	***File Transfert Protocol*** Protocole qui permet d'assurer le transfert de fichiers de façon indépendante des spécificités des NOS (Network Operating System, pour mémoire). Ainsi, un client FTP sous Windows peut télécharger un fichier depuis un serveur UNIX
SMTP	***Simple Mail Transfert Protocol*** Le protocole qui permet d'acheminer le courrier depuis le serveur SMTP de l'émetteur, jusqu'au serveur SMTP du destinataire, qui le classe dans les Boîtes aux lettres de ses clients. (Décrit en détail par ailleurs dans ce site[6]).
POP3	***Post Office Protocol version 3*** Le protocole qui permet au client de relever à distance le courrier classé dans sa boîte aux lettres. Également détaillé par ailleurs sur ce site[7].
IMAP4	***Interactive Mail Access Protocol version 4*** Normalement, ce protocole devrait prendre la place de POP3. Certains fournisseurs sérieux, comme FREE l'implémentent déjà. Contrairement à POP3 qui ne permet une gestion des messages qu'une fois qu'ils sont rapatriés localement, IMAP propose des fonctionnalités plus fines.
NNTP	***Network News Transfert Protocol*** Très proche de SMTP, ce protocole est employé par les forums usenet. Bien que l'usage des forums NNTP n'entre que tardivement dans les mœurs des internautes "débutants", ce moyen de communication offre des avantages incomparables par rapport aux listes de diffusion par exemple.

6 http://christian.caleca.free.fr/smtp/index.html
7 http://christian.caleca.free.fr/pop3/index.html

TELNET	C'est le "couteau suisse" du travail à distance. En fait, un client TELNET est une console en mode texte, capable de se connecter sur la plupart des serveurs, comme POP3 ou SMTP. Il devient alors possible d'envoyer et de lire des messages, si l'on connaît les commandes inhérentes aux protocoles SMTP et POP3. Un serveur TELNET permet cependant des choses bien plus puissantes et "dangereuses" puisqu'il devient possible de prendre à distance le contrôle d'un hôte. C'est un outil qui permet l'administration distante d'une machine, du moment que l'on est capable d'ouvrir une session et d'acquérir les droits de "super utilisateur".

Il en existe bien entendu beaucoup d'autres, il n'est pas, encore une fois, question ici de référencer tous les protocoles applicatifs de la création.

L'adresse IP

Avant de commencer

Il est bon de savoir qu'il existe une adresse "MAC" (Media Access Control), écrite normalement en "dur" dans la ROM de l'interface réseau et donc théoriquement ineffaçable et infalsifiable (mais ce n'est que la théorie, tous les pirates vous le diront). Cette adresse est réputée unique et décidée par le constructeur de la carte. Elle est la seule adresse exploitée au niveau 2 pour l'identification des hôtes qui dialoguent. Cette méthode ne permettant pas l'interconnexion de réseaux, il va être nécessaire d'ajouter dans la couche supérieure (niveau 3), une adresse logique qui sera attribuée par l'administrateur du réseau, en coordination avec les organismes chargés de gérer l'attribution de ces adresses. Dans le cas qui nous intéresse ici, il s'agit de la fameuse adresse IP.

Définition d'une adresse IP

Internet Protocol

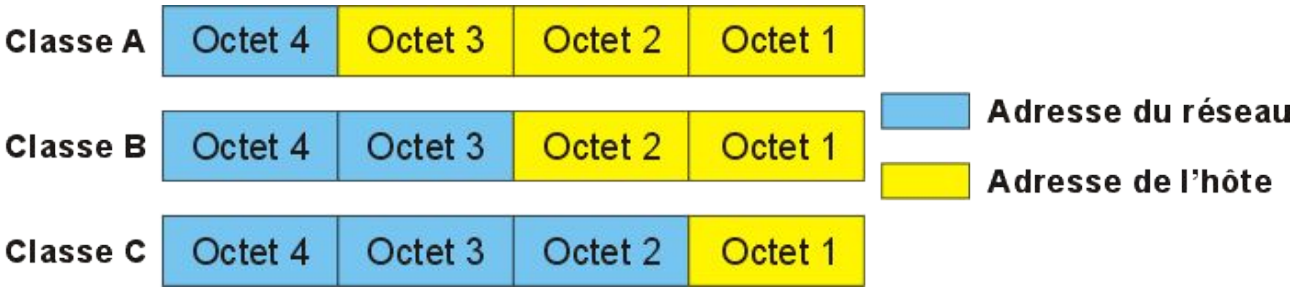
Il existe déjà sur le Net une multitude de pages qui traitent du sujet, ça ne fait rien, on va en mettre une de plus...

Dans sa version 4, IP définit une adresse sur 4 octets. Une partie définit l'adresse du réseau (NetID ou SubnetID suivant le cas), l'autre partie définit l'adresse de l'hôte dans le réseau (HostID). La taille relative de chaque partie varie suivant la classe choisie.

Les classes d'adresses

Topologie

Hormis la classe D multicast, destinée à faire de la diffusion d'information pour plusieurs hôtes simultanément, il existe trois classes d'adresses IP :



Comme vous le voyez, la classe A permet de créer peu de réseaux, mais avec beaucoup d'hôtes dans chaque réseau, La classe C faisant l'inverse.

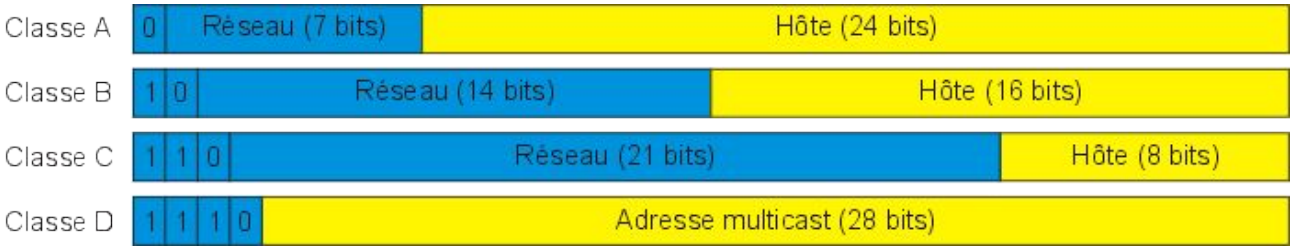
Étendue de chaque classe

Comment fait-on pour savoir à quelle classe appartient une adresse ? Il y a deux méthodes pour le savoir :

- La triviale, qui consiste à apprendre par cœur le tableau.
- La subtile, qui consiste à retenir la règle, qui est logique.

Voici donc la règle :

- La classe est définie par les bits les plus lourds (les plus à gauche)
- Le bit le moins signifiant pour la classe est toujours un 0
- Les autres sont tous à 1
- La classe A est signalée par un seul bit, donc obligatoirement un 0
- La classe B par deux bits, donc 1 0
- La classe C par trois bits, donc 1 1 0
- La classe D (multicast) par 4 bits donc 1 1 1 0



Il existe même une classe E, dont les bits les plus lourds sont 11110, qui est "réservée à un usage ultérieur".

Si l'on arrive à retenir la définition ou son image, ça devient facile de retrouver l'étendue de chaque classe :

Classe	*Première adresse*	*Dernière adresse*
A	0.0.0.1	127.255.255.254
B	128.0.0.1	191.255.255.254
C	192.0.0.1	223.255.255.254
D	224.0.0.1	239.255.255.254

A ce stade, nous pourrions penser qu'il peut y avoir, par exemple, 128 réseaux de classe A, avec la possibilité d'avoir 16 777 216 hôtes dans chaque réseau. C'est bien entendu, un peu plus compliqué que çà.

Il y a déjà quelques adresses que l'on ne peut pas attribuer à un hôte :

- L'adresse d'hôte =0 (exemple: 192.168.1.0 dans une classe C)
  Par convention, l'adresse IP dont la partie hôte est nulle est réservée à l'identification du réseau.
- L'adresse d'hôte avec tous ses bits à 1 (exemple: 192.168.1.255)
  Par convention, cette adresse signifie que tous les hôtes du réseau 192.168.1.0 sont concernés (Adresse de broadcast).

Les réseaux privés

Et ce n'est pas tout. Nous savons qu'une adresse Internet doit être unique dans un inter réseau. Cette considération, qui ne posait pas trop de problèmes pour des réseaux d'entreprise coupés du reste du monde, devient très restrictive à l'échelle de l'Internet où chaque adresse IP doit être unique à l'échelle planétaire. Ceci représente une contrainte énorme, et qui fait que la pénurie d'adresses IP est une catastrophe annoncée bien plus certaine que celle du bug de l'an 2000. (Rassurez-vous, le prochain protocole IP v6 prévoit de la marge, il faudra juste tout ré apprendre).

Pour permettre aux entreprises de construire leur réseau privé, il a donc été réservé dans chaque classe A, B et C des adresses de réseaux qui ne sont jamais attribuées sur l'Internet (RFC 1918)[8]. Tout paquet de données contenant une adresse appartenant à ces réseaux doit être éliminé par le premier routeur établissant une connexion avec l'Internet.

Ces réseaux privés sont:

Classe	*Réseaux privés*	*Identification*
A	10.0.0.0	Pour les réseaux privés
	127.0.0.0	Pour l'interface de boucle locale (*)
B	172.16.0.0 à 172.31.0.0	Pour les réseaux privés
C	192.168.0.0 à 192.168.255.0	Pour les réseaux privés
(*) L'adresse qui correspond à "localhost". Cette adresse locale est nécessaire au fonctionnement de la pile IP.		

Le masque de sous réseau

Le masque de sous-réseau a une importance que peu d'utilisateurs connaissent, elle est pourtant fondamentale. C'est un ensemble de 4 octets destiné à isoler :

- Soit l'adresse de réseau (NetID ou SubnetID) en effectuant un ET logique bit à bit entre l'adresse IP et le masque.
- Soit l'adresse de l'hôte (HostID) en effectuant un ET logique bit à bit entre l'adresse IP et le complément du masque (!masque).

Les masques de sous-réseau par défaut sont, suivant les classes d'adresses:

Classe	*Masque par défaut*	*Nbe d'octets pour l'hôte*
A	255.0.0.0	3
B	255.255.0.0	2
C	255.255.255.0	1

Par défaut, un masque de sous réseau englobe donc la totalité de la classe.

8 http://abcdrfc.free.fr/rfc-vf/rfc1918.html

Mais pourquoi "sous réseau"?

Le principe en est simple: Imaginons que nous disposions d'une classe B. Nous disposons donc de deux octets pour les adresses d'hôtes, soit 65 534 hôtes possibles (les adresses x.x.0.0 et x.x.255.255 sont réservées). Ca ferait tout de même beaucoup de machines sur le même réseau. En pareil cas, il est bien préférable d'organiser son réseau logique en plusieurs sous réseaux, connectés entre eux par des routeurs.

Si par exemple, bien qu'étant en classe B, on choisit comme masque de sous réseau 255.255.255.0, nous obtiendrons 256 sous réseaux de 254 hôtes chacun dans le même réseau. Mais il est possible de définir des masques plus subtils.

Deux hôtes, bien qu'appartenant au même réseau logique, s'ils sont placés dans des sous réseaux logiques différents, ne pourront communiquer entre eux que par l'intermédiaire d'un routeur. Cette solution est très commode pour des réseaux d'entreprise constitués de réseaux locaux distants et même pour des réseaux locaux comportant plusieurs centaines d'hôtes.

Les sur-réseaux

IPv4 est au bout du rouleau... Les adresses sont rares, les classes A ne sont plus disponibles, en classe B, pas grand chose de libre et les classes C sont exiguës. Que faire alors ? Par exemple créer un seul réseau logique avec plusieurs classes C contiguës. Dans ce cas, le masque de "sous réseau" sera un masque de "sur réseau" et définira un réseau avec plus d'hôtes qu'une classe C ne le permet.

Sur un réseau privé par exemple, nous pourrions prendre les deux classes C 192.168.0.0 et 192.168.1.0. En utilisant un masque de type 255.255.254.0, ceci nous permettra de réunir les deux classes C au sein d'un même réseau logique.

Bidouillage ? Probablement, mais ça fonctionne... Avec quelques restrictions cependant. Certaines piles IP n'accepteront pas les adresses 192.168.0.255 et 192.168.1.0 comme adresses d'hôtes valides (elles devraient être réservées dans un réseau "normal", nous l'avons vu, mais dans le cas d'un "sur réseau" constitué comme celui de l'exemple, il est logiquement possible de les utiliser).

Un exemple de configuration chez FT :

Cet exemple appartient désormais au passé, du temps où le Câble Wanadoo utilisait encore DHCP. Il reste cependant intéressant, comme cas d'école.

Un client Wanadoo Câble à Marseille se connecte et récupère l'adresse 62.161.99.115. C'est une adresse de classe A. Nous allons essayer de voir toutes les informations que l'on peut en tirer, au niveau du réseau. La base RIPE nous dit :

```
whois -h whois.geektools.com 62.161.99.115 ...
Query: 62.161.99.115
Registry: whois.ripe.net
Results:

% Rights restricted by copyright. See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 62.161.96.0 - 62.161.120.255
netname: FR-FTCI-3
descr: FTCI
descr: 40, rue Gabriel Crie
descr: 92240 Malakoff
country: FR
...
mnt-by: OLEANE-NOC
```

```
...
```

Cette adresse appartient donc au bloc 62.161.96.0 - 62.161.120.255, qui est une portion du réseau de classe A 62.0.0.0.

Ce bloc est géré par Oleane et est utilisé par FTCI.

Voyons maintenant les informations données par le DHCP. (sous Linux avec PUMP, mais possible aussi sous Windows avec winipcfg. Cependant, vous aurez moins d'informations) :

```
Device eth0
        IP: 62.161.99.115
        Netmask: 255.255.248.0
        Broadcast: 62.161.103.255
        Network: 62.161.96.0
        Boot server 62.161.120.11
        Next server 62.161.120.11
        Gateway: 62.161.96.1
        Domain: wanadoo.fr
        Nameservers: 62.161.120.11
        Renewal time: Thu Feb  1 10:17:57 2001
        Expiration time: Thu Feb  1 10:25:27 2001
```

Le masque de sous réseau est inhabituel, mais techniquement tout à fait acceptable.

En binaire il s'écrit: `1111 1111 . 1111 1111 . 1111 1000 . 0000 0000`

Son complément vaut: `0000 0000 . 0000 0000 . 0000 0111 . 1111 1111`

Le nombre d'adresses du sous réseau est égal au complément du masque, soit 2 047 moins les adresses de sous réseau et de broadcast du sous réseau.

Exercice:

- *A quel sous réseau appartient l'adresse 62.161.99.115 (SubnetID)?*

  Adresse IP `0011 1110 . 1010 0001 . 0110 0011 . 0111 0011`

  Masque de sous réseau : `1111 1111 . 1111 1111 . 1111 1000 . 0000 0000`

  Adresse du sous-réseau: (ET logique) `0011 1110 . 1010 0001 . 0110 0000 . 0000 0000`

  donc en décimal : `62.161.96.0`

  - L'opération consiste simplement en un ET logique bit à bit entre l'adresse et le masque. Mais on avait déjà la réponse en consultant les informations du client DHCP

- *Quelle est la partie de l'adresse qui concerne l'hôte (HostID)?*

  Adresse IP `0011 1110 . 1010 0001 . 0110 0011 . 0111 0011`

  Masque de sous réseau: (complément logique) `0000 0000 . 0000 0000 . 0000 0111 . 1111 1111`

  HostID: (ET logique) `0000 0000 . 0000 0000 . 0000 0011 . 0111 0011`

  donc en décimal : `0.0.3.115`

  - L'opération consiste ici en un ET logique entre l'adresse et le complément du masque

Bien entendu, HostID + SubnetID doit reconstituer l'adresse IP, ce qui est bien le cas :
(62.161.96.0) + (0.0.3.115) = 62.161.99.115

- *Quelle est la plus petite adresse possible dans ce sous réseau?*
  - SubnetID+1=62.161.96.1 .
    Qui est d'ailleurs l'adresse de la passerelle (c'est un choix de FTCI, pas une obligation. Toute adresse dans le même sous réseau aurait aussi bien fait l'affaire).
- *Quelle est la plus grande adresse possible dans ce sous réseau?*
  - C'est SubnetID+!SubnetMask-1
    Pourquoi? !SubnetMask-1 correspond à la plus grande HostID possible dans ce sous réseau, !SubnetMask correspondant à l'adresse de "l'hôte de broadcast"

SubnetID :
```
0011 1110 . 1010 0001 . 0110 0000 . 0000 0000
```

!Masque de sous réseau-1 :
```
0000 0000 . 0000 0000 . 0000 0111 . 1111 1110
```

Plus grande adresse possible: (+)
```
0011 1110 . 1010 0001 . 0110 0111 . 1111 1110
```

donc en décimal :
```
62.161.103.254
```

L'opération est une somme binaire. Le résultat était prévisible, une fois encore, en regardant les informations du client DHCP. En effet; l'adresse de broadcast pour le sous réseau étudié est 62.161.103.255 (HostID avec tous les bits à 1).

C'est bien, n'est-ce pas, de pouvoir donner une explication rationnelle à tous ces paramètres IP plus ou moins obscurs à première vue...

Les sockets

Une oreille dans chaque port

Adresse, port et socket

Imaginons la situation suivante (fréquente sur des petits réseaux) :

Un seul "serveur" (entendez par là une machine) héberge plusieurs services bien connus des internautes :

- Un serveur web (HTTP)
- Un serveur de fichiers (FTP)
- Un serveur de messagerie (SMTP et POP3)

Tous ces services cohabitent donc sur un hôte disposant d'une seule adresse IP, disons 62.161.120.45 (pour fixer les idées) et fonctionnent sans problèmes. Vous êtes-vous posé la question de savoir par quel prodige tout ne se mélange pas? Comment se fait-il que le navigateur du client qui invoque l'URL http://62.161.120.45/default.html voit bien arriver la page demandée, alors que le client qui se connecte sur le serveur POP 62.161.120.45 va pouvoir y récupérer son courrier?

Plus fort encore, pendant qu'un client consulte la page http://62.161.120.45/default.html, un autre consulte http://62.161.120.45/sommaire.html. Et chaque client reçoit bien la page qu'il demande...

Grâce aux ports ! Les ports sont des numéros d'identification qui permettent de spécifier le service concerné. Ce numéro de port est écrit sur 2 octets, ce qui donne 65535 ports possibles (parce que le port 0 n'est, à ma connaissance, pas utilisé).

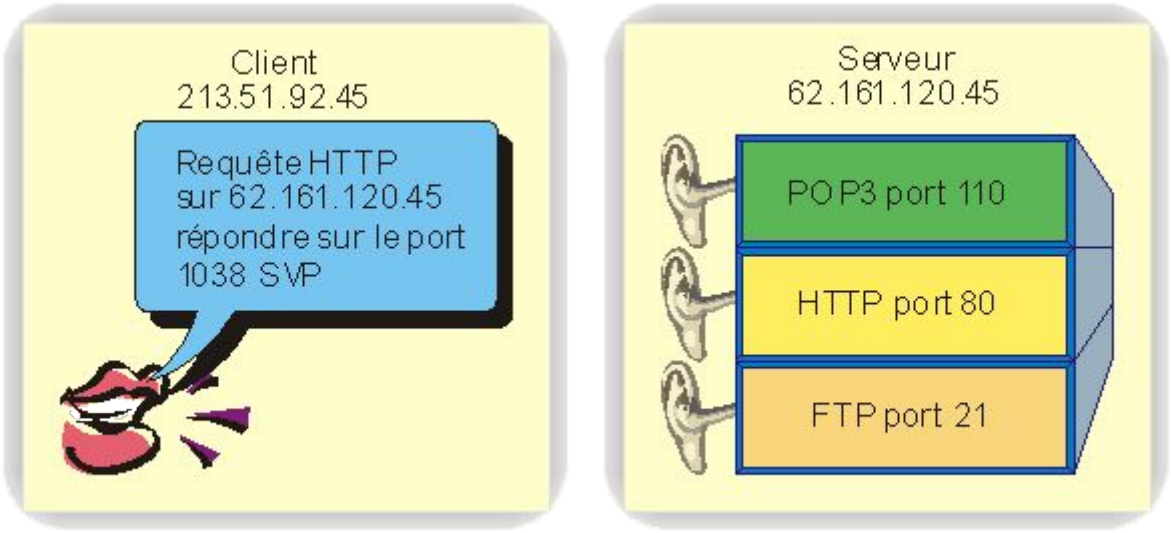
La combinaison "adresse IP:numéro de port " constitue ce que l'on appelle une "socket" (qui veut dire à peu près "connecteur" en anglais).

Une socket identifie pleinement le service qui est concerné sur une machine donnée.

Le serveur et le client

Les serveurs ont une fonction particulière : Ils doivent envoyer des informations pertinentes aux clients qui en réclament. Comme un serveur ne convient pas d'un rendez-vous avec le client, il doit rester attentif en permanence pour ne pas risquer de rater une question. Pour ce faire, on y installe des "daemons", petits programmes qui tournent en tâche de fond et qui écoutent continuellement sur un numéro de port donné. Il y a des conventions pour attribuer ces ports sur des services connus, par exemple le port 80 pour HTTP, le port 110 pour POP3, le port 21 pour FTP. Il faut qu'il y ait des conventions de ce genre pour que les clients puissent atteindre ces services.

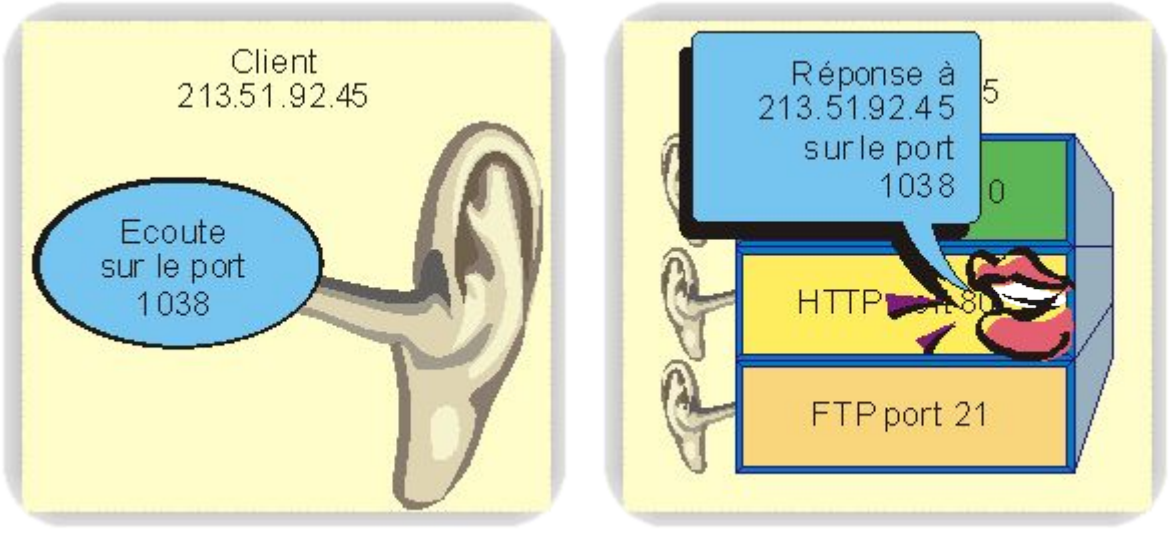
Lorsque l'on écrit http://62.161.120.45, on ne spécifie pas de port; sous-entendu, il s'agit du port 80 parce que l'on invoque un service HTTP. Il serait possible d'écrire: http://62.161.120.45:80 Ici, on spécifie le port. Certaines protections triviales consistent justement à forcer un service à ne pas employer le port standard. Un administrateur pourrait décider de mettre son serveur HTTP à l'écoute du port 88. Dans ce cas, si l'utilisateur n'est pas au courant de cette particularité, il ne pourra pas accéder à ce serveur (sauf s'il dispose d'un scanner de ports et qu'il découvre la supercherie).



En revanche, le client qui émet la requête ne dispose pas de port d'écoute attitré. Ce n'est pas un serveur, c'est un client; il n'a donc rien à écouter d'autre que les réponses à ses questions. Il faut donc, lorsqu'il envoie sa requête, qu'il spécifie sur quel port il va écouter la réponse, de manière à ce que le serveur puisse construire un socket efficace pour ladite réponse.

Vous êtes-vous demandé par quel miracle, si vous ouvrez deux fois votre navigateur pour afficher deux pages différentes sur le même serveur, les informations ne se mélangent pas ?

C'est parce que les deux sessions du navigateur indiquent des ports de réponse différents ! C'est le NOS du client qui choisit les ports de réponse en fonction de ceux qui sont disponibles sur la machine.



Un port d'écoute est une porte ouverte

Lorsqu'un port est ouvert à l'écoute sur un service serveur, c'est une porte ouverte par laquelle un intrus peut entrer.

Ce détail nous mène directement aux problèmes de sécurité et d'intrusions. Mais ne mélangeons pas

tout, cette affaire est traitée ailleurs sur ce site[9].

Quelques infos supplémentaires

NAT, PAT et autres mascarades

Nous y reviendrons plus loin dans le chapitre consacré au routage[10], mais tant qu'on est dans les ports, autant dire quelques mots de ces techniques.

- NAT (Network Address Translation) est une faculté dont dispose un routeur, de modifier les adresses IP des émetteurs lors du passage des datagrammes entre deux réseaux. Ça ne nous intéresse pas directement ici.
- PAT (Port Access Translation) est une fonction qui permet de changer au passage le numéro de port dans le datagramme. Ca peut paraître tordu, mais il existe une foule d'applications possibles pour cette propriété.
- MASQUERADE, qui est un mélange des deux (NAT, PAT) est une fonction très intéressante pour connecter tout un réseau local construit sur une classe IP privée à l'Internet. La passerelle utilisera son IP publique (côté Internet) pour faire du NAT sur les adresses privées du réseau local et fera également du PAT pour savoir à qui il faudra transmettre les réponses.
    - Le principe de fonctionnement et la façon de construire une telle passerelle sont décrits dans la chapitre MASQUERADE[11], ailleurs sur ce site.

La liste des ports réservés

Le mieux est de consulter la RFC 1700 qui définit les ports d'écoute standards :

(Pour mémoire, un site de référence pour ce genre d'informations : http://www.iana.org/)

Ceux qui désirent consulter la liste exhaustive des "ports bien connus", peuvent le faire ici[12].

9 http://christian.caleca.free.fr/securite/index.html
10 http://christian.caleca.free.fr/routage/index.html
11 http://christian.caleca.free.fr/masquerade/index.html
12 http://christian.caleca.free.fr/tcpip/well_known_ports.htm

Mode connecté (TCP)

Le mode connecté de TCP n'est pas d'une grande simplicité. Il est conçu pour être robuste et tient compte des possibilités et des risques des grands réseaux maillés, à savoir :

- Les paquets peuvent circuler de la source vers la cible par des chemins différents (dans ce cas, ils arrivent sur la cible dans le désordre),
- Il peut s'en perdre en route,
- Certains paquets peuvent arriver corrompus
- etc..

TCP en revanche ne prend hélas pas en compte, ou très peu, les problèmes de piratage.

L'exemple est pris sur mon réseau local, mais le principe reste rigoureusement le même sur l'Internet. La manipulation sur le réseau local m'évite d'avoir à faire un filtrage plus ou moins pénible.

La séquence en gros

(Désolé si les lignes sont longues et nécessitent un "scrolling latéral" pour une résolution inférieure à 1024x768, ça reste tout de même plus lisible comme ça).

```
No. Time     Source               Destination          Proto Info
 1 0.000000 00:20:18:b9:49:37    ff:ff:ff:ff:ff:ff    ARP   Who has 192.168.0.250? Tell 192.168.0.10
 2 0.000277 00:20:18:61:90:e3    00:20:18:b9:49:37    ARP   192.168.0.250 is at 00:20:18:61:90:e3
 3 0.000474 chris.maison.mrs     gateway1.maison.mrs  TCP   1927 > pop3 [SYN]
 4 0.000885 gateway1.maison.mrs  chris.maison.mrs     TCP   pop3 > 1927 [SYN, ACK]
 5 0.001111 chris.maison.mrs     gateway1.maison.mrs  TCP   1927 > pop3 [ACK]
 6 0.049836 gateway1.maison.mrs  chris.maison.mrs     POP   Response: +OK
 7 0.050586 chris.maison.mrs     gateway1.maison.mrs  POP   Request: USER chris
 8 0.050998 gateway1.maison.mrs  chris.maison.mrs     TCP   pop3 > 1927 [ACK]
 9 0.051511 gateway1.maison.mrs  chris.maison.mrs     POP   Response: +OK
10 0.051979 chris.maison.mrs     gateway1.maison.mrs  POP   Request: PASS babaorum
11 0.060769 gateway1.maison.mrs  chris.maison.mrs     TCP   pop3 > 1927 [ACK]
12 0.159888 gateway1.maison.mrs  chris.maison.mrs     POP   Response: +OK Mailbox open, 0 messages
13 0.160799 chris.maison.mrs     gateway1.maison.mrs  POP   Request: STAT
14 0.161552 gateway1.maison.mrs  chris.maison.mrs     POP   Response: +OK 0 0
15 0.162801 chris.maison.mrs     gateway1.maison.mrs  POP   Request: QUIT
16 0.167987 gateway1.maison.mrs  chris.maison.mrs     POP   Response: +OK Sayonara
17 0.168562 chris.maison.mrs     gateway1.maison.mrs  TCP   1927 > pop3 [FIN, ACK]
18 0.168957 gateway1.maison.mrs  chris.maison.mrs     TCP   pop3 > 1927 [ACK]
19 0.169465 gateway1.maison.mrs  chris.maison.mrs     TCP   pop3 > 1927 [FIN, ACK]
20 0.169698 chris.maison.mrs     gateway1.maison.mrs  TCP   1927 > pop3 [ACK]
```

Pas moins de 20 trames, pour constater qu'il n'y a pas de nouveau courrier !

TCP en détail

Un petit coup d'ARP...

Les séquences 1 et 2 ne sont pas inintéressantes, bien que ne faisant pas directement partie du protocole TCP. C'est de l'ARP, ça vient de la couche basse d'Ethernet,

- Trame 1 :
  Mon poste n'a pas en mémoire la correspondance MAC Address / IP pour le serveur. Il pose

donc la question sur un broadcast ARP :
A qui appartient l'adresse IP 192.168.0.250 (le serveur)? répondez à 192.168.0.10 (mon poste).

- Trame 2 :
  Le serveur répond :
  192.168.0.250 à la MAC Address:00:20:18:61:90:e3

Si, sans trop attendre, je lance la commande ARP sur mon poste, voici le résultat :

```
E:\>arp -a

Interface : 192.168.0.10 on Interface 0x4000003
Adresse Internet Adresse physique Type
192.168.0.250 00-20-18-61-90-e3 dynamique
```

Au bout d'un "certain temps" sans servir, cette ligne sera effacée de la mémoire. Rappelons qu'à l'intérieur d'un réseau, la couche d'accès physique (la plus basse du modèle DOD) utilise exclusivement les adresses MAC.

Et la connexion TCP

Établissement de la connexion

Accrochez-vous, c'est un peu compliqué :)

Il faut d'abord savoir que les connexions TCP mettent en oeuvre deux pointeurs de 32 bits, respectivement appelés :

- Sequence number
- Acknowledgment number

Ces deux pointeurs permettent le suivi des paquets :

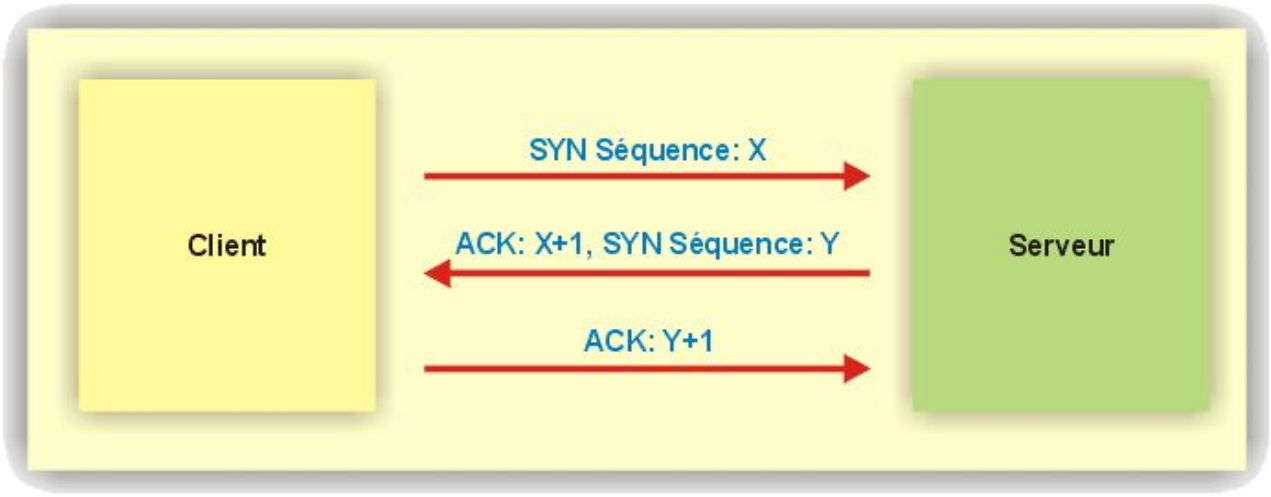
- L'accusé réception pour la source pour chaque paquet émis.
- La remise en ordre des paquets reçus sur la cible.

D'autres pointeurs permettent également de fiabiliser la connexion, comme le checksum. Nous n'allons pas rentrer dans tous les détails, d'autres sites sur l'Internet le font déjà très bien. Le point particulier des numéros de séquence et d'acquittement va en revanche être regardé de près, parce qu'il est utilisé pour une attaque particulièrement redoutable: le "spoofing".

Par ailleurs, un ensemble de "flags" (drapeaux, bits significatifs d'un état particulier) permet de donner des informations sur la nature du paquet.

Voyons ce que les livres disent :

L'établissement d'une connexion se fait en trois temps :



1	2	3
Le client envoie une séquence de synchronisation, avec un numéro de séquence. Le Flag "SYN" est positionné	Le serveur répond par une acceptation dans laquelle il renvoie : • un numéro d'acquittement égal au numéro de séquence qu'il a reçu+1 • un numéro de séquence les flags SYN et ACK sont positionnés.	Le client acquitte la réponse en envoyant : • un numéro d'acquittement égal au numéro de séquence envoyé par le serveur +1 • un numéro de séquence égal au numéro d'acquittement envoyé par le serveur

Mais voyons cela sur l'exemple :

```
Frame 3 (62 on wire, 62 captured)
    Arrival Time: Oct 12, 2000 11:19:15.3756
    Time delta from previous packet: 0.000197 seconds
    Frame Number: 3
    Packet Length: 62 bytes
    Capture Length: 62 bytes
Ethernet II
    Destination: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Source: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    *** Au niveau Ethernet, nous trouvons les deux adresses MAC
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 48
    Identification: 0x7624
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    *** C'est bien un protocole TCP
    Header checksum: 0x024f (correct)
    Source: chris.maison.mrs (192.168.0.10)
```

```
    Destination: gateway1.maison.mrs (192.168.0.250)
Transmission Control Protocol, Src Port: 1927 (1927), Dst Port: pop3 (110)
    Source port: 1927 (1927)
    *** Le port du client (l'émetteur de cette trame)
    Destination port: pop3 (110)
    *** Le port de destination (110 pour POP3)
    Sequence number: 3662573346
    *** Et un numéro de séquence (à mémoriser pour la suite).
    Header length: 28 bytes
    Flags: 0x0002 (SYN)
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        *** C'est bien une requête "SYN"
        .... ...0 = Fin: Not set
    Window size: 16384
    Checksum: 0x6f64
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted
```

La seconde doit être la réponse du serveur POP. Normalement, c'est un ACK (Acknowledgment, Acceptation de la synchronisation du client, suivi d'une demande de synchronisation du numéro de séquence du serveur), c'est ce que disent les livres. Voyons ça :

```
Frame 4 (62 on wire, 62 captured)
    Arrival Time: Oct 12, 2000 11:19:15.3760
    Time delta from previous packet: 0.000411 seconds
    Frame Number: 4
    Packet Length: 62 bytes
    Capture Length: 62 bytes
Ethernet II
    Destination: 00:20:18:b9:49:37 (00:20:18:b9:49:37) *** Le client
    Source: 00:20:18:61:90:e3 (00:20:18:61:90:e3)       *** Le serveur
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 48
    Identification: 0x088a
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xafe9 (correct)
    Source: gateway1.maison.mrs (192.168.0.250)
    Destination: chris.maison.mrs (192.168.0.10)
Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 1927 (1927)
    Source port: pop3 (110)
    *** Le serveur continue sur le port 110
    Destination port: 1927 (1927)
    *** Et répond bien sur le port ouvert par le client
    Sequence number: 4089248825
    *** Le numéro de séquence proposé par le serveur
    Acknowledgement number: 3662573347
    *** Rappelez-vous, le n° de séquence du client était 3662573346
    *** Le numéro d'acquittement est 3662573346 + 1
    Header length: 28 bytes
    Flags: 0x0012 (SYN, ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
```

```
        *** Nous avons bien un acquittement de la part du serveur
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        *** Et une demande de synchronisation (sur le numéro de séquence 4089248825)
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0x41e4
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted
```

Ici, nous devrions trouver un acquittement du client sur le numéro de séquence 4089248825

```
Frame 5 (60 on wire, 60 captured)
    Arrival Time: Oct 12, 2000 11:19:15.3762
    Time delta from previous packet: 0.000226 seconds
    Frame Number: 5
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Source: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    *** C'est bien le client qui répond au serveur
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 40
    Identification: 0x7625
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0x0256 (correct)
    Source: chris.maison.mrs (192.168.0.10)
    Destination: gateway1.maison.mrs (192.168.0.250)
Transmission Control Protocol, Src Port: 1927 (1927), Dst Port: pop3 (110)
    Source port: 1927 (1927)
    Destination port: pop3 (110)
    *** Les ports ne sont toujours pas changés
    Sequence number: 3662573347
    *** Le numéro de séquence est ici égal à l'acquittement de la trame précédente
    *** souvenez-vous: "Acknowledgement number: 3662573347"
    Acknowledgement number: 4089248826
    *** comme tout à l'heure, 4089248825 + 1
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        *** Il n'y a bien qu'un acquittement.
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 17520
    Checksum: 0xa7b0
```

C'est bien comme dans les livres (ouf!)

La transmission des données

Que disent les livres? Ils disent que maintenant, les échanges de données vont se faire.

- Le flag PUSH sert à signaler à TCP qu'il doit transmettre les données reçues aux couches supérieures.
- Chaque paquet aura :
  - Pour numéro d'acquittement le numéro de séquence du dernier paquet reçu, augmenté du nombre d'octets de données qu'il contenait.
  - Pour numéro de séquence le numéro d'acquittement du dernier paquet reçu.

Sur le protocole POP3, c'est le serveur qui va envoyer un message de bienvenue. La trame qui suit doit donc provenir du serveur, elle doit contenir :

- Un acquittement du numéro de séquence de la trame précédente:3662573347, puisque la trame 5 ne contenant pas de données.
- Un numéro de séquence égal au numéro d'acquittement de la trame précédente:4089248826.

```
Frame 6 (103 on wire, 103 captured)
    Arrival Time: Oct 12, 2000 11:19:15.4249
    Time delta from previous packet: 0.048725 seconds
    Frame Number: 6
    Packet Length: 103 bytes
    Capture Length: 103 bytes
Ethernet II
    Destination: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Source: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 89
    Identification: 0x088b
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xafbf (correct)
    Source: gateway1.maison.mrs (192.168.0.250)
    Destination: chris.maison.mrs (192.168.0.10)
    *** C'est bien le serveur qui répond.
Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 1927 (1927)
    Source port: pop3 (110)
    Destination port: 1927 (1927)
    Sequence number: 4089248826
    Acknowledgement number: 3662573347
    * Nous partons bien sur les numéros de séquence et d'acquittement définis
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        *** L'acquittement au paquet précédent est donné
        .... 1... = Push: Set
        *** Il va y avoir des données...
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0xa343
```

```
Post Office Protocol
    Response: +OK
    Response Arg: POP3 gateway1.maison.mrs v7.64 server ready
    *** Et voilà les données.
```

Pour le moment, tout se passe encore conformément aux écritures. Voyons la suite.

Le paquet précédent contient les données :

```
+OK POP3 gateway1.maison.mrs v7.64 server ready
```

Ceci nous fait 47 octets (Il ne faut pas oublier de compter les espaces, ce sont des caractères comme les autres). Il ne faut pas oublier non plus qu'une ligne de texte se termine par les caractères CR (retour à la ligne) et LF (saut de ligne), ce qui nous fait deux caractères de plus, soit au total 49 octets.

Normalement :

- Le numéro d'acquittement du prochain paquet devrait donc être 4089248826+49=4089248875 (séquence + 49)
- Le numéro de séquence devrait être 3662573347

Le client doit maintenant envoyer son login. Nous devrions donc trouver un PUSH et un ACK.

```
Frame 7 (66 on wire, 66 captured)
    Arrival Time: Oct 12, 2000 11:19:15.4257
    Time delta from previous packet: 0.000750 seconds
    Frame Number: 7
    Packet Length: 66 bytes
    Capture Length: 66 bytes
Ethernet II
    Destination: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Source: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 52
    Identification: 0x7626
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0x0249 (correct)
    Source: chris.maison.mrs (192.168.0.10)
    Destination: gateway1.maison.mrs (192.168.0.250)
    *** C'est bien le client qui s'adresse au serveur
Transmission Control Protocol, Src Port: 1927 (1927), Dst Port: pop3 (110)
    Source port: 1927 (1927)
    Destination port: pop3 (110)
    Sequence number: 3662573347
    *** Oui, c'est l'Acknowledgment number du paquet précédent
    Acknowledgement number: 4089248875
    *** OUI!!! C'est le Sequence number du paquet précédent augmenté de 49
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        *** voici l'ACK...
        .... 1... = Push: Set
        *** et voilà le PUSH, donc il y aura des données
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
```

```
    Window size: 17471
    Checksum: 0x0da4
Post Office Protocol
    Request: USER
    Request Arg: chris
    *** Ce sont les données.
```

Tout s'est passé comme prévu.

Bien. Comme nous connaissons par cœur le protocole POP[13], nous savons que le serveur va envoyer la réponse "+OK" et une invite à communiquer le mot de passe.

Nous devrions trouver :

- Un ACK et un PUSH,
- Un Sequence number égal à l'Acknowledgment number du paquet précédent.
- Un Acknowledgment number égal au Sequence number du paquet précédent +12 (comptez les octets de données du paquet précédent).

```
Frame 8 (60 on wire, 60 captured)
    Arrival Time: Oct 12, 2000 11:19:15.4261
    Time delta from previous packet: 0.000412 seconds
    Frame Number: 8
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Source: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 40
    Identification: 0x088c
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xafef (correct)
    Source: gateway1.maison.mrs (192.168.0.250)
    Destination: chris.maison.mrs (192.168.0.10)
    *** C'est bien le serveur qui répond.
Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 1927 (1927)
    Source port: pop3 (110)
    Destination port: 1927 (1927)
    Sequence number: 4089248875
    Acknowledgement number: 3662573359 =3662573347+12
    Toujours normal, 12 octets de données dans le paquet précédent.
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        *** Voici ACK...
        .... 0... = Push: Not set
        *** Mais il n'y a pas de PUSH, donc pas de données?
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0x6e6b
```

13 http://christian.caleca.free.fr/pop3/index.html

Ben non, il n'y a pas eu de données. Comme on est parfaitement certain que le serveur doit en envoyer (parce que l'on connaît le protocole POP3 par cœur), c'est que c'est encore le serveur qui va parler...

```
Frame 9 (95 on wire, 95 captured)
    Arrival Time: Oct 12, 2000 11:19:15.4266
    Time delta from previous packet: 0.000513 seconds
    Frame Number: 9
    Packet Length: 95 bytes
    Capture Length: 95 bytes
Ethernet II
    Destination: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Source: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 81
    Identification: 0x088d
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xafc5 (correct)
    Source: gateway1.maison.mrs (192.168.0.250)
    Destination: chris.maison.mrs (192.168.0.10)
    *** Oui, c'est encore le serveur
Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 1927 (1927)
    Source port: pop3 (110)
    Destination port: 1927 (1927)
    Sequence number: 4089248875
    Acknowledgement number: 3662573359
    *** Et les numéros sont identiques au paquet précédent
        (normal, pas de données dans le paquet précédent)
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        *** Ce coup-ci, il y a des données.
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0x6428
Post Office Protocol
    Response: +OK
    Response Arg: User name accepted, password please
    *** Les voilà!
```

Et voilà. Le client va envoyer son mot de passe, le dialogue continue.

```
Frame 10 (68 on wire, 68 captured)
    Arrival Time: Oct 12, 2000 11:19:15.4271
    Time delta from previous packet: 0.000468 seconds
    Frame Number: 10
    Packet Length: 68 bytes
    Capture Length: 68 bytes
Ethernet II
    Destination: 00:20:18:61:90:e3 (00:20:18:61:90:e3)
    Source: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
```

```
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 54
    Identification: 0x7627
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0x0246 (correct)
    Source: chris.maison.mrs (192.168.0.10)
    Destination: gateway1.maison.mrs (192.168.0.250)
    *** C'est bien le client
Transmission Control Protocol, Src Port: 1927 (1927), Dst Port: pop3 (110)
    Source port: 1927 (1927)
    Destination port: pop3 (110)
    Sequence number: 3662573359
    *** = Acknowledgement number précédent
    Acknowledgement number: 4089248916
    *** = Sequence number précédent augmenté de 41, nous avons compris le principe
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        *** Il va y avoir des données.
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 17430
    Checksum: 0x8f50
Post Office Protocol
    Request: PASS
    Request Arg: babaorum
    *** M**** alors! le mot de passe circule en clair !
        rassurez-vous, ce n'est pas le bon que vous lisez :-)
        Mais le mot de passe circule vraiment en clair. Stupéfiant non?
```

Bien. Il ne reste plus que 10 trames à regarder (on a fait la moitié), mais maintenant, ça va devenir monotone. Je vous laisse analyser le reste[14] si ça vous amuse. Je vous conseille tout de même de jeter un œil au moins sur les trames 17, 18 19 et 20[15] qui servent à fermer proprement la connexion. C'est une procédure importante qui permet au serveur "d'oublier" cette connexion.

Ce qu'il est intéressant d'étudier, c'est le mécanisme des numéros de séquence et d'acquittement. Parce que celui qui va être capable de prédire la séquence de ces numéros, s'il dispose d'outils qui savent bricoler les trames, pourra se faire passer pour un autre dans un dialogue TCP existant. C'est ce qu'on appelle le "spoofing", méthode de piratage délicate mais dangereuse.

Vous avez également la RFC 793[16] dont une traduction en français est disponible ici:

http://abcdrfc.free.fr/rfc-vf/rfc793.html

14 http://christian.caleca.free.fr/tcpip/exemple_tcp.htm
15 http://christian.caleca.free.fr/tcpip/exemple_tcp.htm#fin
16 http://www.freesoft.org/CIE/RFC/793/index.htm

Mode non connecté (UDP)

User Datagram Protocol. Ici, la discussion se fait sans trop de précautions. Le principe est le suivant :

- Celui qui doit parler s'adresse à son interlocuteur, la plupart du temps en posant une question, directement, sans vérifier que l'interlocuteur est présent et peut répondre.
- Si la réponse ne vient pas, l'initiateur décidera de la stratégie à appliquer. En général, il n'y a pas trop de solutions :
  - Répéter la question au même interlocuteur
  - Répéter la question à un autre interlocuteur (C'est le cas par exemple des résolutions de noms)
  - Abandonner et arrêter le dialogue.

Parmi les usages les plus connus du mode sans connexion (UDP), notons :

- La résolution des noms ou la résolution inverse des adresses (DNS)
- La recherche d'une adresse IP dynamique (DHCP)
- La plupart des jeux en réseau.

En général, partout où le paquet de données à transmettre peut tenir dans un seul datagramme.

A titre d'exemple, nous allons regarder ça sur quelque chose de nouveau: le protocole NTP (Network Time Protocol). C'est un protocole applicatif qui permet à un hôte de synchroniser son horloge sur un serveur de temps.

L'exemple est pris sous linux par la commande :

```
#ntpdate ntp0.oleane.net
```

Comme d'habitude, le sniffer ne perd rien de la conversation...

```
No. Source           Destination      Protocol Info
16  193.248.36.34    ntp0.oleane.net  NTP      NTP
17  ntp0.oleane.net  193.248.36.34    NTP      NTP
18  193.248.36.34    ntp0.oleane.net  NTP      NTP
19  ntp0.oleane.net  193.248.36.34    NTP      NTP
20  193.248.36.34    ntp0.oleane.net  NTP      NTP
21  ntp0.oleane.net  193.248.36.34    NTP      NTP
22  193.248.36.34    ntp0.oleane.net  NTP      NTP
23  ntp0.oleane.net  193.248.36.34    NTP      NTP
```

A première vue, nous constatons un dialogue entre le client (193.248.36.34) et le serveur ntp0.oleane.net.

L'objectif est ici, non pas de décortiquer le protocole NTP, encore que ce ne soit pas sans intérêt, mais d'observer un dialogue UDP. Voyons donc le détail :

Le surlignage jaune représente le protocole, la source et la destination, le sur lignage "bleu" représente les données transmises, dans l'organisation décrite par le protocole NTP :

```
Frame 16 (86 on wire, 86 captured)
    ...
    Protocol: UDP (0x11)
```

```
    Header checksum: 0x40a6 (correct)
    Source: Mix-Marseille-107-1-34.abo.wanadoo.fr (193.248.36.34)
    Destination: ntp0.oleane.net (194.2.0.28)
User Datagram Protocol
    Source port: ntp (123)
    Destination port: ntp (123)
    Length: 56
    Checksum: 0x5b48
Network Time Protocol
    Flags: DB
        11.. .... = Leap Indicator: alarm condition (clock not synchronized)
        ..01 1... = Version number: NTP Version 3
        .... .011 = Mode: client
    Peer Clock Stratum: unspecified or unavailable (0)
    Peer Pooling Interval: 4 (16 sec)
    Peer Clock Precision: 0,015625 sec
    Root Delay:    1,0000 sec
    Clock Dispersion:    1,0000 sec
    Reference Clock ID: Unindentified reference source ''
    Reference Clock Update Time: NULL
    Originate Time Stamp: NULL
    Receive Time Stamp: NULL
    Transmit Time Stamp: 2001-06-13 12:36:30,1227 UTC

Frame 17 (76 on wire, 76 captured)
    ...
    Protocol: UDP (0x11)
    Header checksum: 0xa111 (correct)
    Source: ntp0.oleane.net (194.2.0.28)
    Destination: Mix-Marseille-107-1-34.abo.wanadoo.fr (193.248.36.34)
User Datagram Protocol
    Source port: ntp (123)
    Destination port: ntp (123)
    Length: 56
    Checksum: 0xa367
Network Time Protocol
    Flags: 1C
        00.. .... = Leap Indicator: no warning
        ..01 1... = Version number: NTP Version 3
        .... .100 = Mode: server
    Peer Clock Stratum: secondary reference (2)
    Peer Pooling Interval: 4 (16 sec)
    Peer Clock Precision: 0,000004 sec
    Root Delay:    0,0130 sec
    Clock Dispersion:    0,1491 sec
    Reference Clock ID: 49.110.238.145
    Reference Clock Update Time: 2001-06-13 12:33:43,8840 UTC
    Originate Time Stamp: 2001-06-13 12:36:30,1227 UTC
    Receive Time Stamp: 2001-06-13 12:35:18,4244 UTC
    Transmit Time Stamp: 2001-06-13 12:35:18,4246 UTC

Frame 18 (86 on wire, 86 captured)
    ...
    Protocol: UDP (0x11)
    Header checksum: 0x40a5 (correct)
    Source: Mix-Marseille-107-1-34.abo.wanadoo.fr (193.248.36.34)
    Destination: ntp0.oleane.net (194.2.0.28)
User Datagram Protocol
    Source port: ntp (123)
    Destination port: ntp (123)
    Length: 56
    Checksum: 0x5a96
Network Time Protocol
    Flags: DB
        11.. .... = Leap Indicator: alarm condition (clock not synchronized)
        ..01 1... = Version number: NTP Version 3
        .... .011 = Mode: client
    Peer Clock Stratum: unspecified or unavailable (0)
    Peer Pooling Interval: 4 (16 sec)
    Peer Clock Precision: 0,015625 sec
    Root Delay:    1,0000 sec
```

```
    Clock Dispersion:    1,0000 sec
    Reference Clock ID: Unindentified reference source ''
    Reference Clock Update Time: NULL
    Originate Time Stamp: NULL
    Receive Time Stamp: NULL
    Transmit Time Stamp: 2001-06-13 12:36:30,1879 UTC

Frame 19 (76 on wire, 76 captured)
    ...
    Protocol: UDP (0x11)
    Header checksum: 0xa10d (correct)
    Source: ntp0.oleane.net (194.2.0.28)
    Destination: Mix-Marseille-107-1-34.abo.wanadoo.fr (193.248.36.34)
User Datagram Protocol
    Source port: ntp (123)
    Destination port: ntp (123)
    Length: 56
    Checksum: 0x66a1
Network Time Protocol
    Flags: 1C
        00.. .... = Leap Indicator: no warning
        ..01 1... = Version number: NTP Version 3
        .... .100 = Mode: server
    Peer Clock Stratum: secondary reference (2)
    Peer Pooling Interval: 4 (16 sec)
    Peer Clock Precision: 0,000004 sec
    Root Delay:    0,0130 sec
    Clock Dispersion:    0,1491 sec
    Reference Clock ID: 49.110.238.145
    Reference Clock Update Time: 2001-06-13 12:33:43,8840 UTC
    Originate Time Stamp: 2001-06-13 12:36:30,1879 UTC
    Receive Time Stamp: 2001-06-13 12:35:18,5105 UTC
    Transmit Time Stamp: 2001-06-13 12:35:18,5106 UTC
```

Etc...

Ce n'est pas nécessaire de voir la suite pour montrer ce qui est important ici. Contrairement à ce qui a été vu en mode connecté avec TCP :

- Toute la partie "synchronisation" entre l'hôte et le client n'existe pas ici.
- Le client pose tout de suite sa "question", en fait, ce n'est pas vraiment une question, le client se contente de dire :
  alarm condition (clock not synchronized)
  Suivi de quelques indicateurs nuls et de la date UTC dont il dispose.
- Le serveur répond simplement en envoyant son heure UTC et quelques autres paramètres destinés à informer sur la précision de la date qu'il donne.
- Ce dialogue va s'arrêter lorsque le client estimera qu'il dispose de toutes les informations nécessaires pour synchroniser son horloge dans de bonnes conditions. (Je vous laisse étudier le protocole NTP[17] en détail, si ça vous intéresse).
- Notez que le dialogue s'arrête sans signalisation particulière, ce qui n'est pas le cas en mode connecté où le signal FIN doit être envoyé et confirmé.
- Notez également que les informations à transmettre sont entièrement contenues dans un seul datagramme. Dans un tel cas, le protocole UDP est tout à fait acceptable et plus léger que le mode connecté.

Vous pouvez trouver d'autres exemples de dialogue UDP dans les paragraphes DNS et DHCP sur ce

17 http://www.ntp.org/

site[18].

Liens associés

- Suite de l'exemple de connexion : http://christian.caleca.free.fr/tcpip/exemple_tcp.htm
- Liste des ports reconnus : http://christian.caleca.free.fr/tcpip/well_known_ports.htm

18 http://christian.caleca.free.fr/