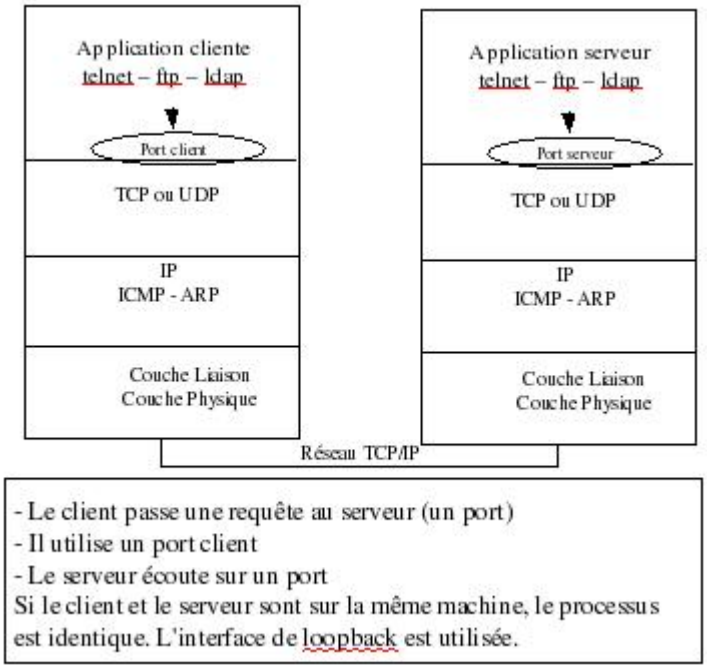
LES APPLICATIONS TCP/IP : COUCHE 4 DU MODELE OSI

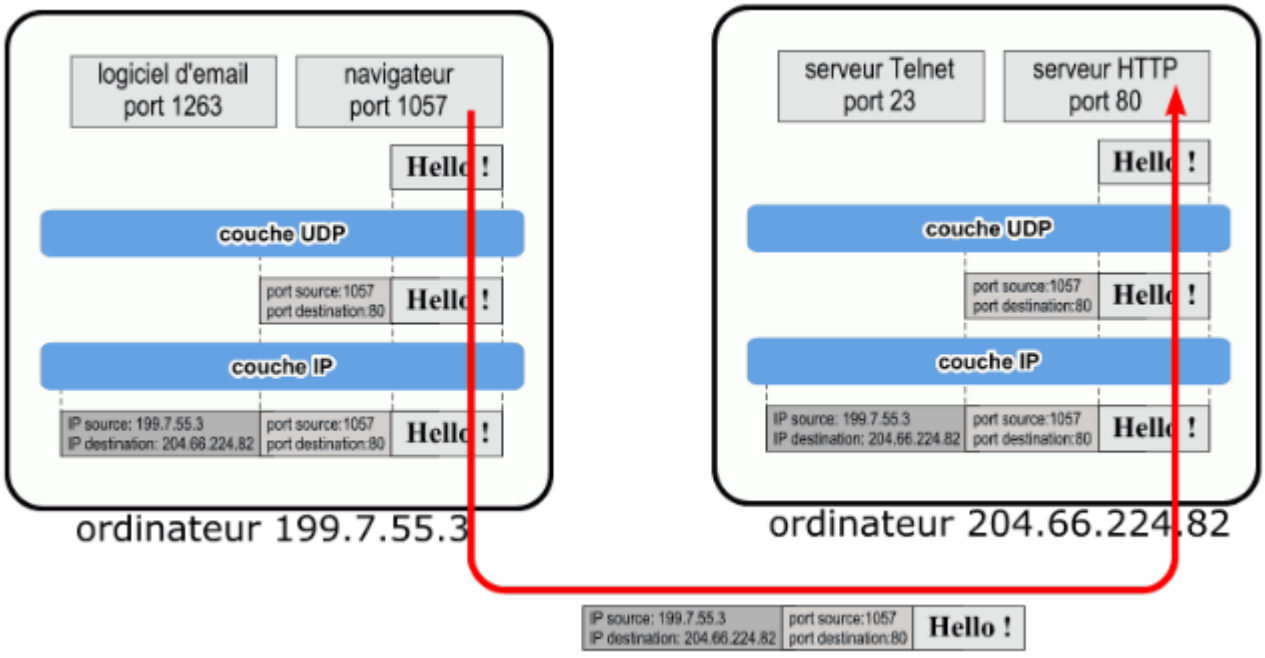
MODÈLE CLIENT/SERVEUR

Les applications réseaux fonctionnent sur le modèle client/serveur. Sur la machine serveur un processus serveur (daemon) traite les requêtes des clients. Client et serveur dialoguent en échangeant des messages qui contiennent des requêtes et des réponses.

Prenons par exemple « telnet » :



FONCTIONNEMENT DU MODELE CLIENT-SERVEUR



L'ADRESSAGE DES APPLICATIFS : LES PORTS

Une fois le datagramme transmis à l'hôte destinataire, il doit parvenir à l'utilisateur (si le système est multi-utilisateur) et à l'application visée (si le système est multi-tâches).

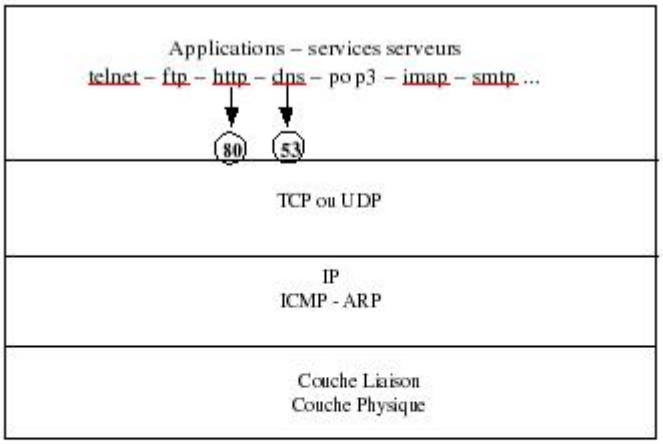
- Sur la machine cliente, l'utilisateur (usager ou programme) effectue une requête vers une machine IP serveur sur le réseau. (par exemple telnet host ou ftp host). Cela se traduit par la réservation d'un port de sortie TCP ou UDP et l'envoi d'un paquet IP à la machine serveur. Ce paquet contient un message TCP ou UDP avec un numéro de port correspondant à l'application demandée sur le serveur.
- Sur le serveur, la requête est réceptionnée par le pilote IP, aiguillée vers TCP ou UDP puis vers le port demandé. Le processus serveur correspondant est à l'écoute des appels sur ce port (par ex: le daemon telnetd traite les requêtes telnet, le daemon ftpd traite les requêtes ftp).
- Les processus client et processus serveur échangent ensuite des messages.

Des numéros de port (entre 0 et 1023) sont réservés pour les applications « standards : les ports « **bien connus** » (**Well Known Ports**), ils ont été assignés par l'IANA. Sur la plupart des systèmes ils peuvent être seulement employés par des processus du système (ou root) ou par des programmes exécutés par les utilisateurs privilégiés (liste complète : http://www.iana.org/assignments/port-numbers ou dans le fichier /etc/services).

D'autres numéros de port sont disponibles pour les applications développées par les utilisateurs (1024 à 65535).

Exemples de ports bien connus :

- HTTP port TCP 80 ;
  - SSH port TCP 22 ;
  - DNS port UDP 53 (TCP 53 pour les transferts de zones et les requêtes supérieures à 512 octets) ;
  - RIP port UDP 520.



- Un service serveur « écoute » sur un port et utilise un protocole de transport
- Ici le service http sur le port 80, le service dns sur le port 53
- Un client utilise un port client et un protocole de transport
- Le client et le serveur utilisent le même protocole de transport sur une session.

On identifie le protocole de communication entre applications par un numéro de protocole et l'application par un numéro de port.

Par exemple, les serveurs HTTP dialoguent de manière traditionnelle par le port 80 :

```
http ://www.sncf.com <=> http :// www.sncf.com:80
```

Les numéros de protocole et de port sont inclus dans le datagramme.

Une fois la connexion établie entre le client et le serveur, ceux-ci peuvent s'échanger des informations selon un protocole défini selon l'applicatif.

Le client soumet des requêtes auxquelles répondra le serveur. Ce mode de communication s'appuie sur la couche “socket”. Cette couche est une interface entre la couche présentation et transport. Elle permet la mise en place du canal de communication entre le client et le serveur. On peut schématiquement dire qu'un socket fournit un ensemble de fonctions. Ces fonctions permettent à une application

client/serveur d'établir un canal de communication entre 2 ou plusieurs machines, qui utilisent un protocole de transport (TCP ou UDP) et un port de communication.

CONNAITRE LES PORTS OUVERTS SUR UNE MACHINE

Utilisation de la commande "netstat"

```
hannibal@box:~$ sudo -s
[sudo] password for hannibal:
root@box:~# netstat -a
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 *:17500                 *:*                     LISTEN
tcp        0      0 box.local:41716         77.67.4.17:http         TIME_WAIT
tcp        0      0 box.local:54051         91-237-218-231-una:http TIME_WAIT
tcp        0      0 box.local:58571         91.226.182.240:http     TIME_WAIT
tcp        0      0 box.local:54361         77.67.28.75:http        ESTABLISHED
tcp        0      0 box.local:38180         77.67.4.10:http         TIME_WAIT
tcp        0      0 box.local:33337         star-01-04-lhr2.fa:http TIME_WAIT
tcp        0      0 box.local:42219         lhr08s03-in-f1.1e:https TIME_WAIT
tcp        0      0 box.local:59577         clpubs.chat-cool.o:http TIME_WAIT
tcp        0      0 box.local:36281         lhr08s03-in-f13.1e:http ESTABLISHED
tcp        0      0 box.local:54362         77.67.28.75:http        ESTABLISHED
tcp        0      0 box.local:40194         lhr08s03-in-f28.1e:http ESTABLISHED
tcp        0      0 box.local:50544         77.67.4.43:http         TIME_WAIT
tcp        0      0 box.local:47347         sjc-not11.sjc.drop:http ESTABLISHED
```

La commande « **netstat** » permet de lister les connexions internet actives, de connaître les numéros de ports et l'état des connexions.

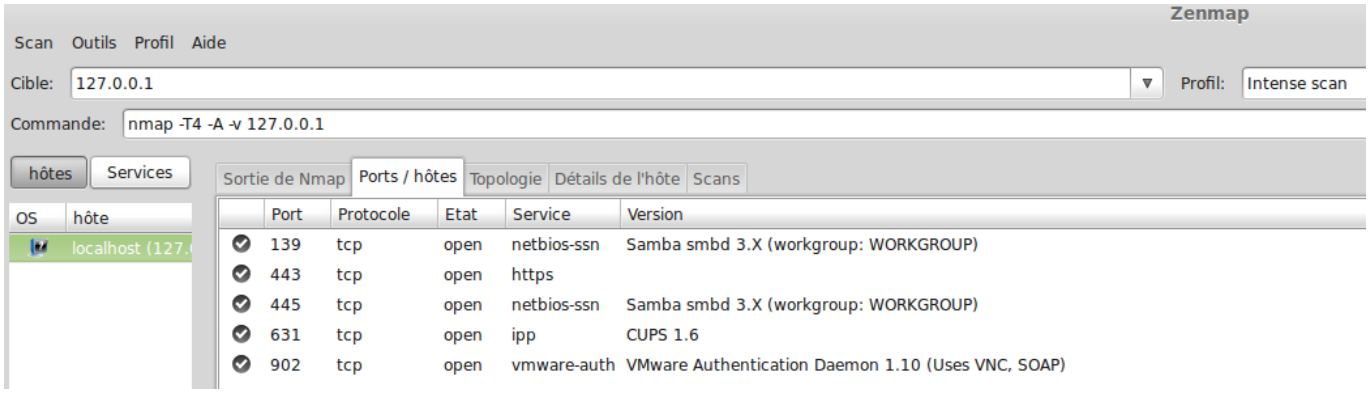
- En état LISTEN, l'application est en écoute et en attente de requêtes de la part de clients sur Internet.
- En état ESTABLISHED, l'application a établi la communication suite à une demande de requête et on considère donc la connexion comme établie.

Quand quelqu'un se connecte à un port (une application web sur un serveur par exemple), l'application va passer de l'état LISTEN à l'état ESTABLISHED.

utilisation de "nmap"

nmap est un utilitaire extrêmement puissant.

Attention, ne jamais utiliser nmap sur un réseau dont vous n'êtes pas l'administrateur légitime (ou sans l'autorisation de celui-ci).



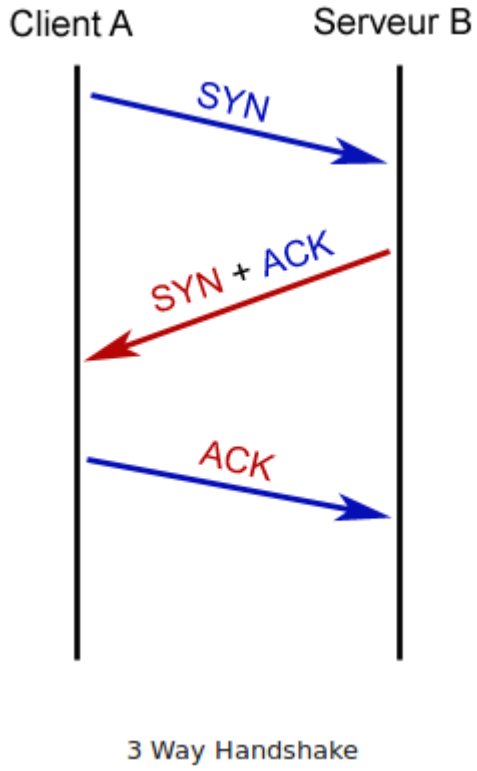
J'ai utilisé ici “zenmap”, version graphique de nmap sur ma propre machine. J'aurai pu me satisfaire de nmap en mode commande :

```
nmap -T4 -A -v 127.0.0.1
```

ETABLISSEMENT D'UNE CONNEXION TCP

Même s'il est possible pour 2 systèmes d'établir une connexion entre eux simultanément, dans le cas général, un système ouvre une 'socket' (point d'accès à une connexion TCP) et se met en attente passive de demandes de connexion d'un autre système. Ce fonctionnement est communément appelé ouverture passive, et est utilisé par le côté serveur de la connexion. Le côté client de la connexion effectue une ouverture active en 3 temps (poignée de mains -handshaking- en trois temps) :

- Le client envoie un segment SYN au serveur,
    1. Le serveur lui répond par un segment SYN/ACK,
    2. Le client confirme par un segment ACK.



3 Way Handshake

On visualise :

```
CONSOLE 1 :
hannibal@box:~$ wget www.free.fr
--2012-12-13 17:19:16--  http://www.free.fr/
Résolution de www.free.fr (www.free.fr)... 212.27.48.10, 2a01:e0c:1:1599::1
Connexion vers www.free.fr (www.free.fr)|212.27.48.10|:80... connecté.
requête HTTP transmise, en attente de la réponse... 301 Moved Permanently
Emplacement: http://portail.free.fr/ [suivant]
--2012-12-13 17:19:16--  http://portail.free.fr/
Résolution de portail.free.fr (portail.free.fr)... 212.27.48.10, 2a01:e0c:1:1599::1
Réutilisation de la connexion existante vers www.free.fr:80.
requête HTTP transmise, en attente de la réponse... 200 OK
Longueur: 76794 (75K) [text/html]
Sauvegarde en : «index.html.4»

100%[======================================>] 76 794       481K/s   ds 0,2s

2012-12-13 17:19:16 (481 KB/s) - «index.html.4» sauvegardé [76794/76794]
```

```
CONSOLE 2 :
root@box:~# tcpdump -n -i eth0 port http
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:19:16.220143 IP 192.168.0.108.36989 > 212.27.48.10.80: Flags [S], seq 417009659, win 14600, options
[mss 1460,sackOK,TS val 8379546 ecr 0,nop,wscale 6], length 0
17:19:16.255106 IP 212.27.48.10.80 > 192.168.0.108.36989: Flags [S.], seq 2963902162, ack 417009660, win
5840, options [mss 1460,nop,nop,sackOK], length 0
```

```
17:19:16.255146 IP 192.168.0.108.36989 > 212.27.48.10.80: Flags [.], ack 1, win 14600, length 0
```

On visualise ici les 3 échanges poignées de main.

TCP ou UDP ?

La couche 4 du modèle OSI utilise 2 protocoles (UDP et TCP) pour satisfaire 2 types de communication :

- **Communications générées par des applications qui nécessitent un transport fiable des données**, mais qui n'ont pas de besoin particulier en ce qui concerne la vitesse de transmission.
- **Communications générées par des applications qui nécessitent un transport immédiat des informations**, mais qui pouvent se permettre de perdre quelques informations.

La première catégorie regroupe une très grande majorité des applications d'Internet, car bon nombre d'entre elles ont besoin que chaque paquet émis soit reçu coûte que coûte ! Ce sont notamment les applications comme le web, la messagerie, le ssh, etc. Si un paquet est perdu, une page web ne pourra pas s'afficher correctement, ce sera pareil pour un mail, etc.

La seconde catégorie concerne les applications de streaming, comme la radio ou la télé sur Internet. Pour une radio en ligne, il est essentiel que les informations soient envoyées en temps réel, le plus rapidement possible. Par contre, si un ou plusieurs paquets sont perdus, on ne va pas arrêter la radio pour autant. L'utilisateur aura des coupures de connexion, mais la radio continuera d'émettre.

On identifie donc ainsi deux besoins bien distincts l'un de l'autre :

- Un protocole TCP fiable mais sans nécessité de rapidité.
    - Un protocole UDP rapide sans nécessité de fiabilité.

  C'est pour cela que nous aurons deux protocoles pour la couche 4. Le protocole TCP et le protocole UDP.

- TCP est de la première catégorie, c'est un protocole extrêmement fiable.
    - Chaque paquet envoyé doit être acquitté par le receveur, qui en ré-émettra un autre s'il ne reçoit pas d'accusé de réception. On dit alors que c'est un protocole connecté.
    - UDP, lui, est un protocole rapide, mais peu fiable. Les paquets sont envoyés dès que possible, mais on ne s'intéresse pas de savoir s'ils ont été reçus ou pas. On dit qu'UDP est un protocole non-connecté.