

# **APPLICATIONS**

## **Client/serveur**

Auteur : Khadija ARFAOUI  
Département Informatique – IUT ANNECY  
Université de SAVOIE  
9, rue de l'Arc-en-ciel  
74016 Annecy-le-Vieux



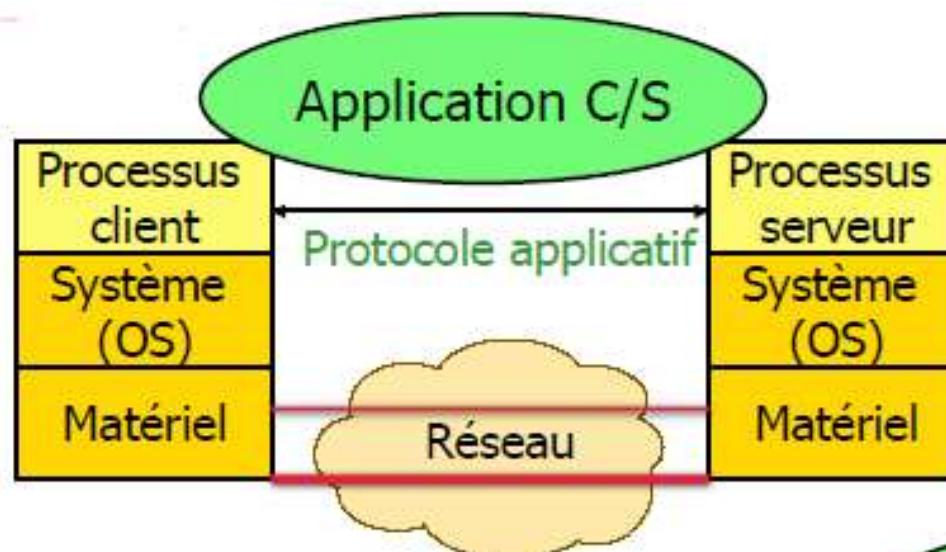
# Sommaire

- **Principes de base**
- **Web serveur**
- **Dhcp**
- **FTP**
- **DNS**
- **NFS, SMB**
- **SMTP**
- **Sockets**



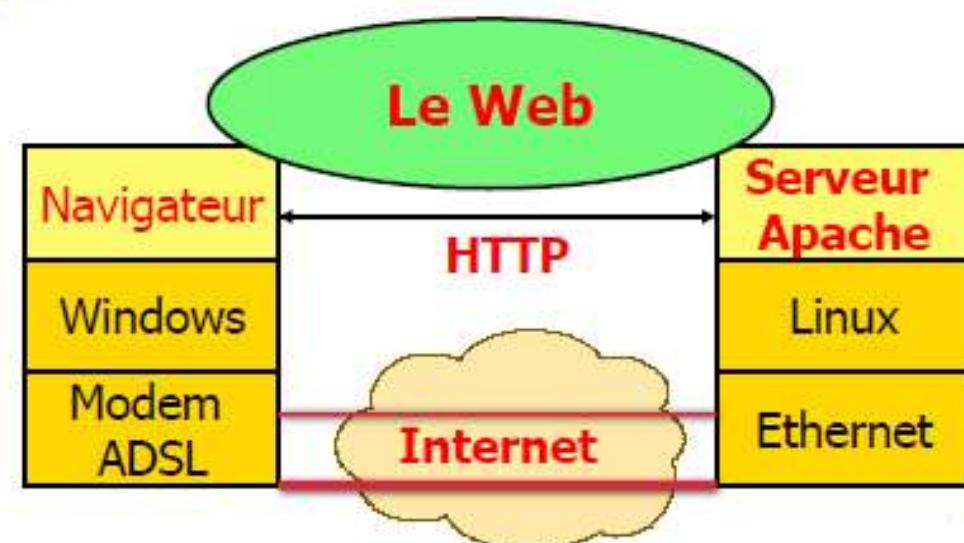
# Principes de base

# Notion d'application Client/Server



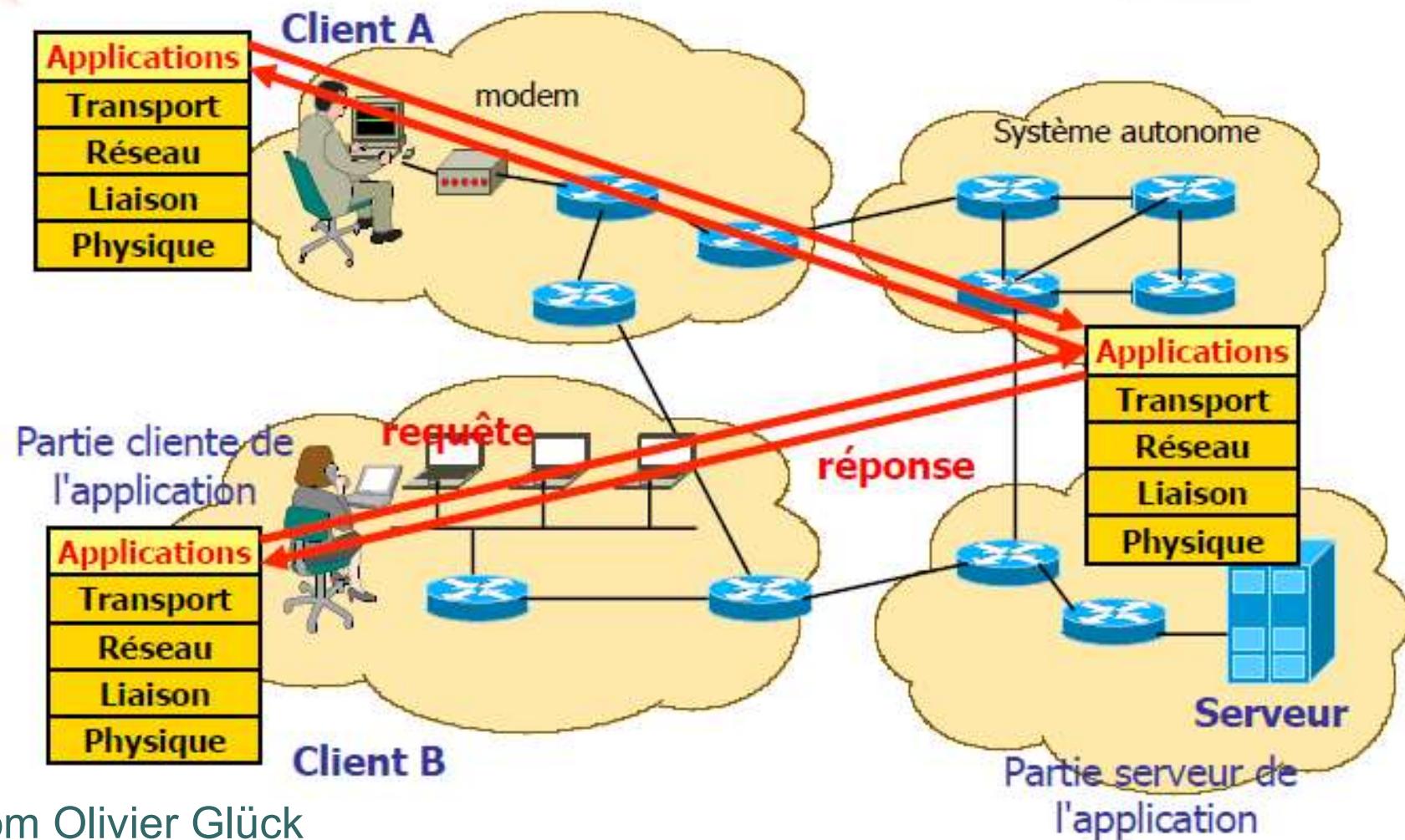
L'application est répartie sur le client et le serveur qui dialoguent selon un protocole applicatif spécifique

L'exemple du Web

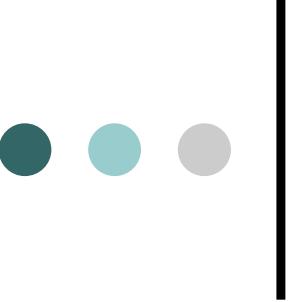


From Olivier Glück

# Notion d'application Client/Server



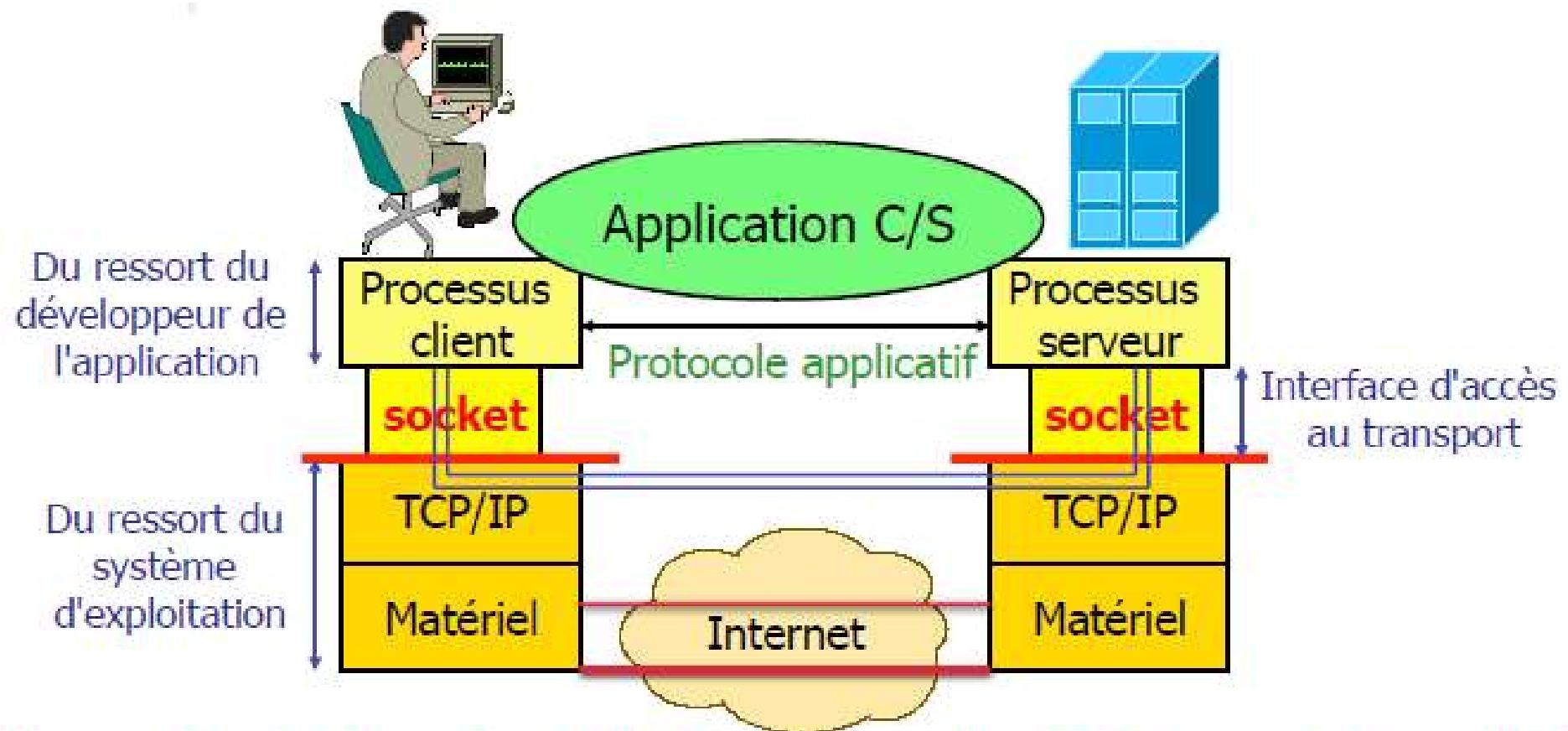
From Olivier Glück



# Notion d'application Client/Server

- Nécessité d'une interface entre l'application réseau et la couche transport
- Couche transport = tuyau (TCP/UDP dans Internet)
- API (*Application Programming Interface*) moyen pour y accéder (interface de programmation)
- Principales APIs de l'Internet : sockets

# Notion d'application Client/Server



**Une socket : interface locale à l'hôte, créée par l'application, contrôlée par l'OS  
Porte de communication entre le processus client et le processus serveur**

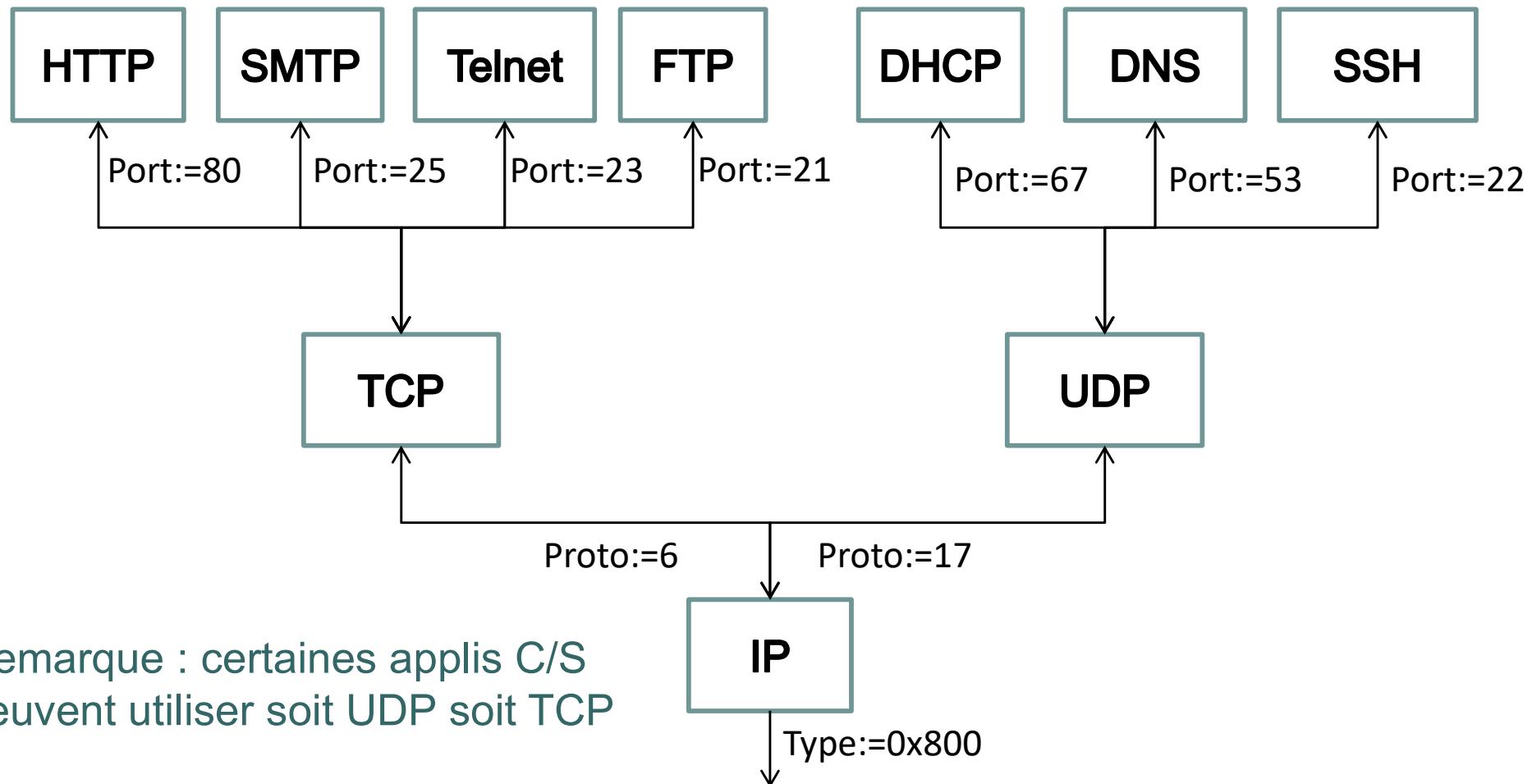


# TCP/IP et applis C/S

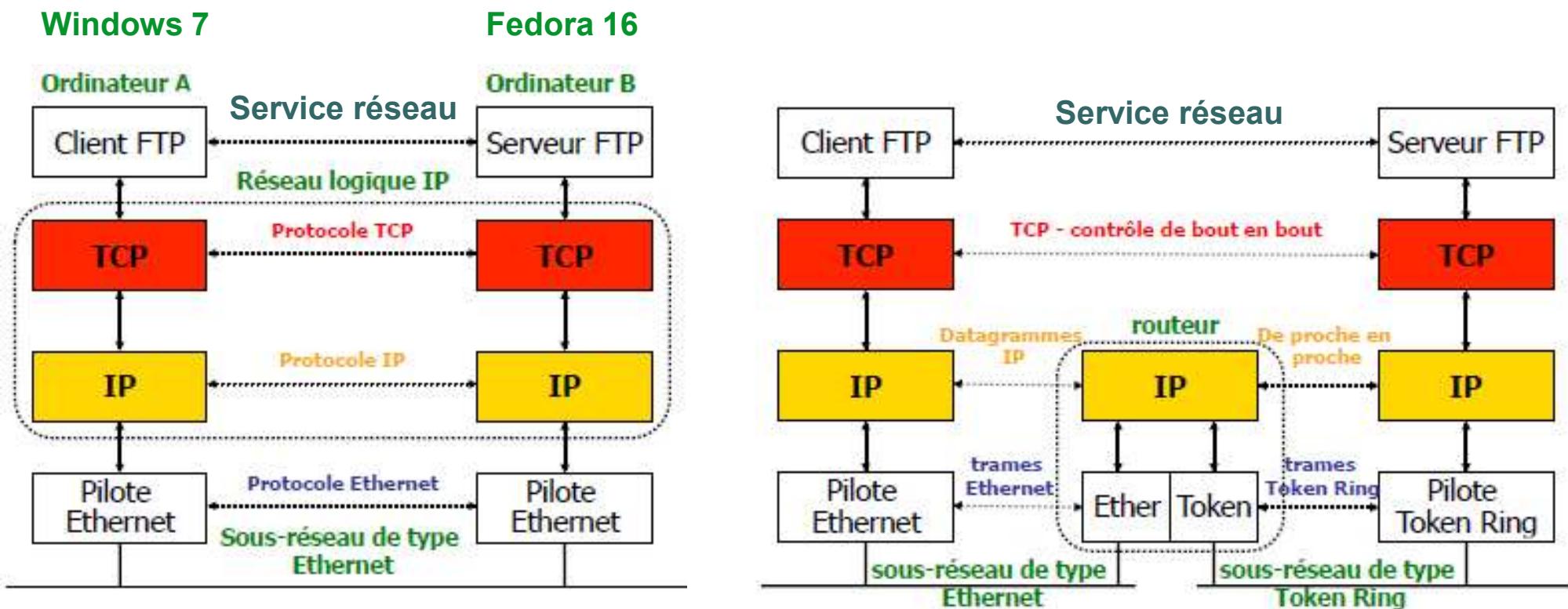
- Conséquences de l'utilisation de TCP/IP : chaque appli C/S de ce type devra spécifier :
  - Le protocole utilisé TCP ou UDP
  - Un numéro de port serveur au niveau TCP/UDP -> fichier **/etc/services**
  - L'adresse IP du serveur
- Les identifiants client (n° port et adresse IP client) sont gérés par l'API



# TCP/IP et applis C/S

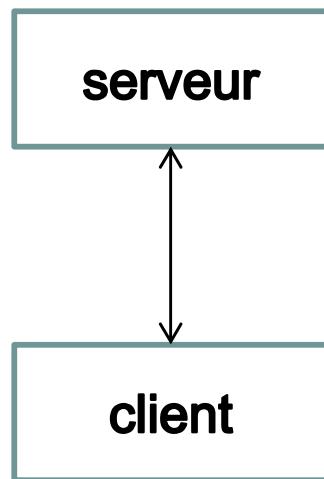


# Gestion de l'hétérogénéité

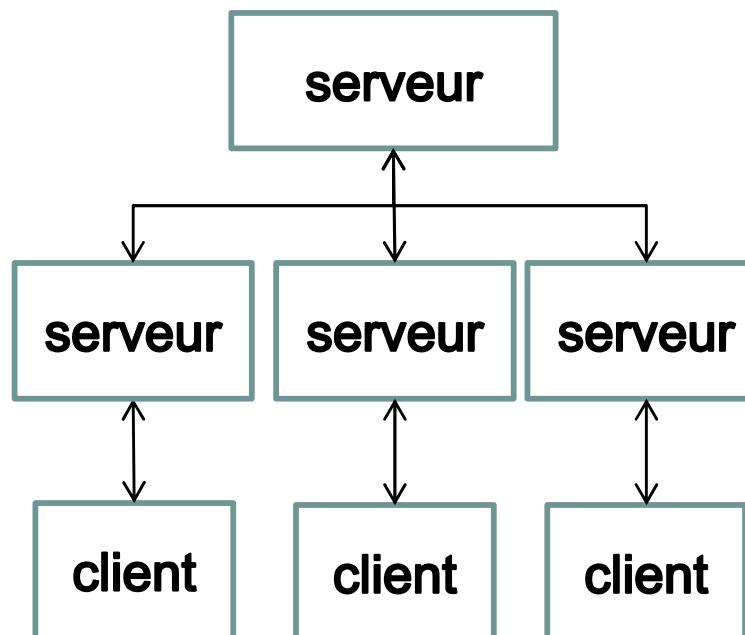




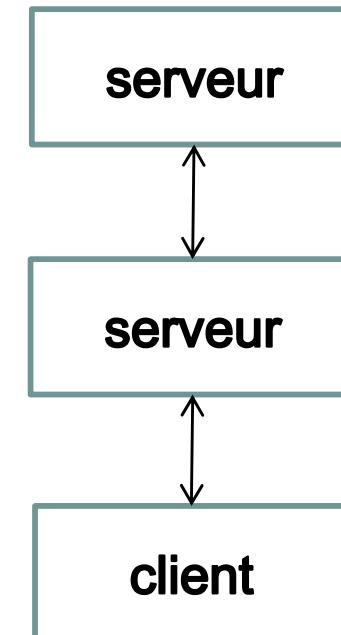
# Types de serveur



Serveur itératif



Serveur parallèle



Multi-serveurs



# Exemples de protocoles C/S

- SMTP : *Simple Mail Transfer Protocol*  
service de messagerie électronique
  - Base pour les services POP, IMAP, ...)
- DNS : *Domain Name System*
  - traduction nom symbolique - adresse IP
  - bases de données réparties sur Internet
- DHCP : *Dynamic Host Configuration Protocol*
  - Allocation dynamique d'adresses IP



# Exemples de protocoles C/S

- HTTP : *HyperText Transport Protocol*
    - LE protocole du web
    - Transactions entre client(s) et serveur web
  - Telnet : *TELetypewriter NETwork protocol*
    - émulation de terminal virtuel
    - ouverture de sessions distantes (sécurité!)
  - FTP : *File Transfer Protocol*
    - Transferts de fichiers distants (down/upload)
    - transfert, suppression, création, ...
- <sub>13</sub>

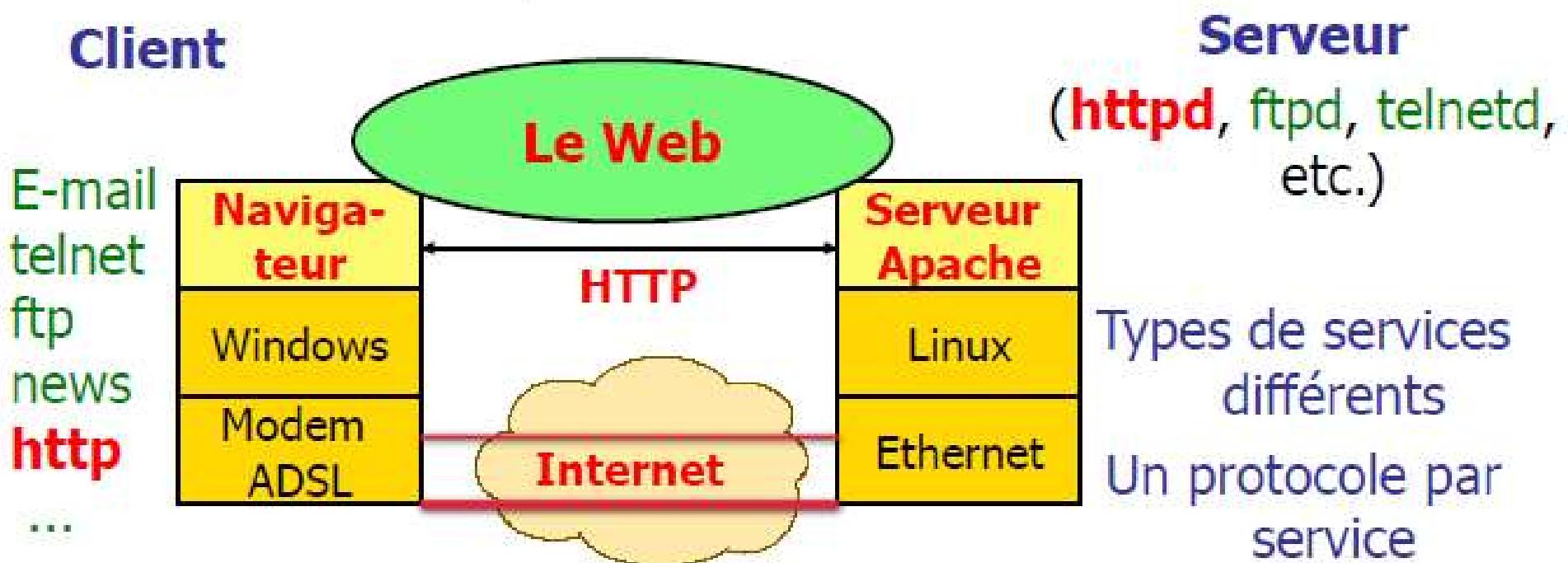


# HTTP



# Serveur Web

- Utilise TCP, port 80 sur un serveur fixé (@ IP)



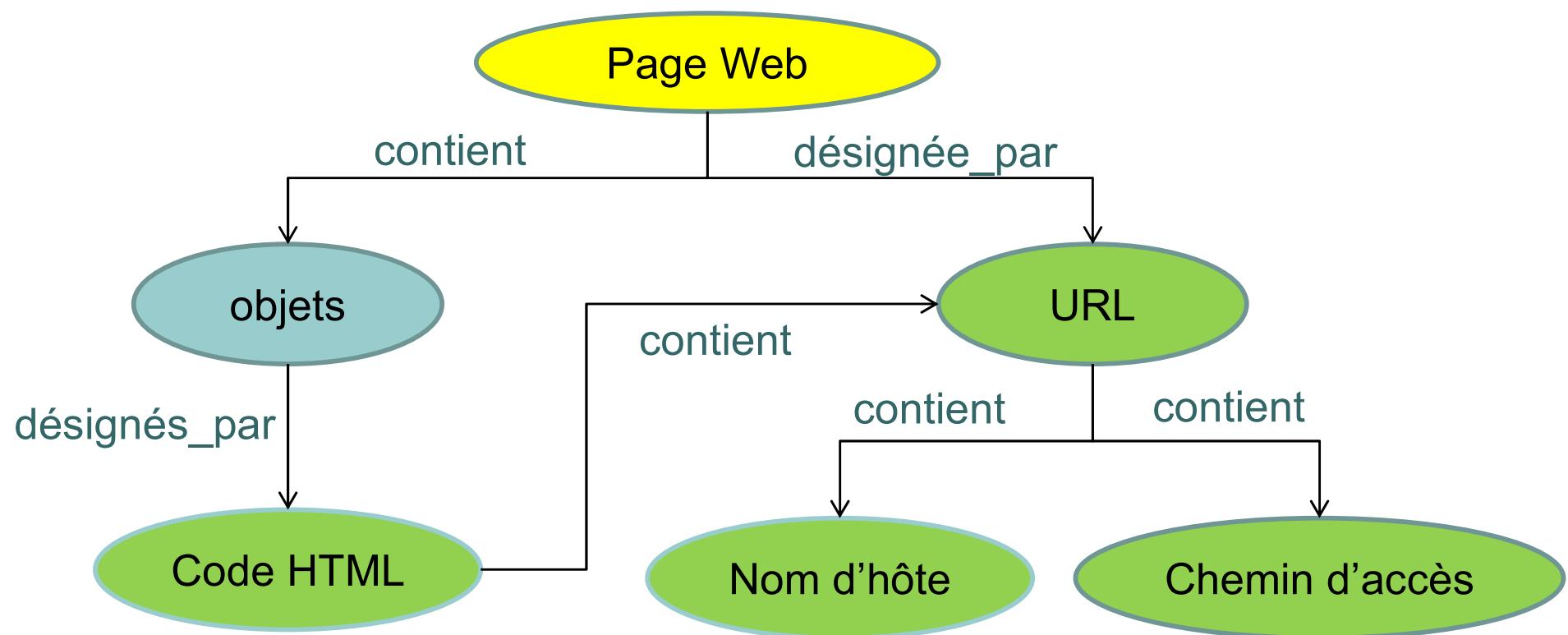


# Principe

- Principe : le client accède à des documents reliés entre eux et situés sur des machines interconnectées par Internet
- Localisation : notion d'**URL**
- protocole utilisé : **HTTP**
- langage utilisé : **HTML5**

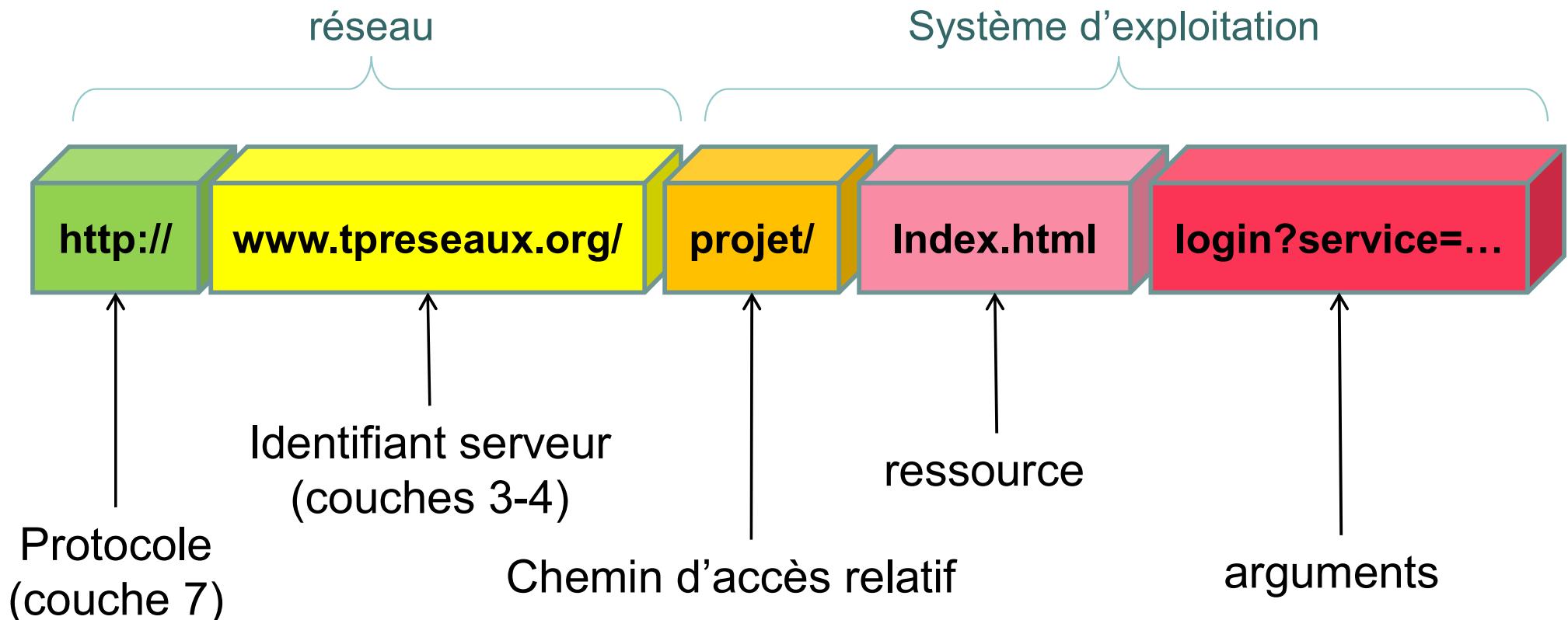


# Principe





# URL



le browser reconstruit l'URL absolue pour faire la requête



# Protocole Http

- Hyper Text Transfer Protocol
- Fonctionne en mode Client/Serveur
- Rôle : gérer le dialogue entre clients et serveur
- Contenu d'une transaction :
  - type de la requête ou de la réponse
  - un en-tête
  - une ligne vide
  - un contenu (parfois vide)

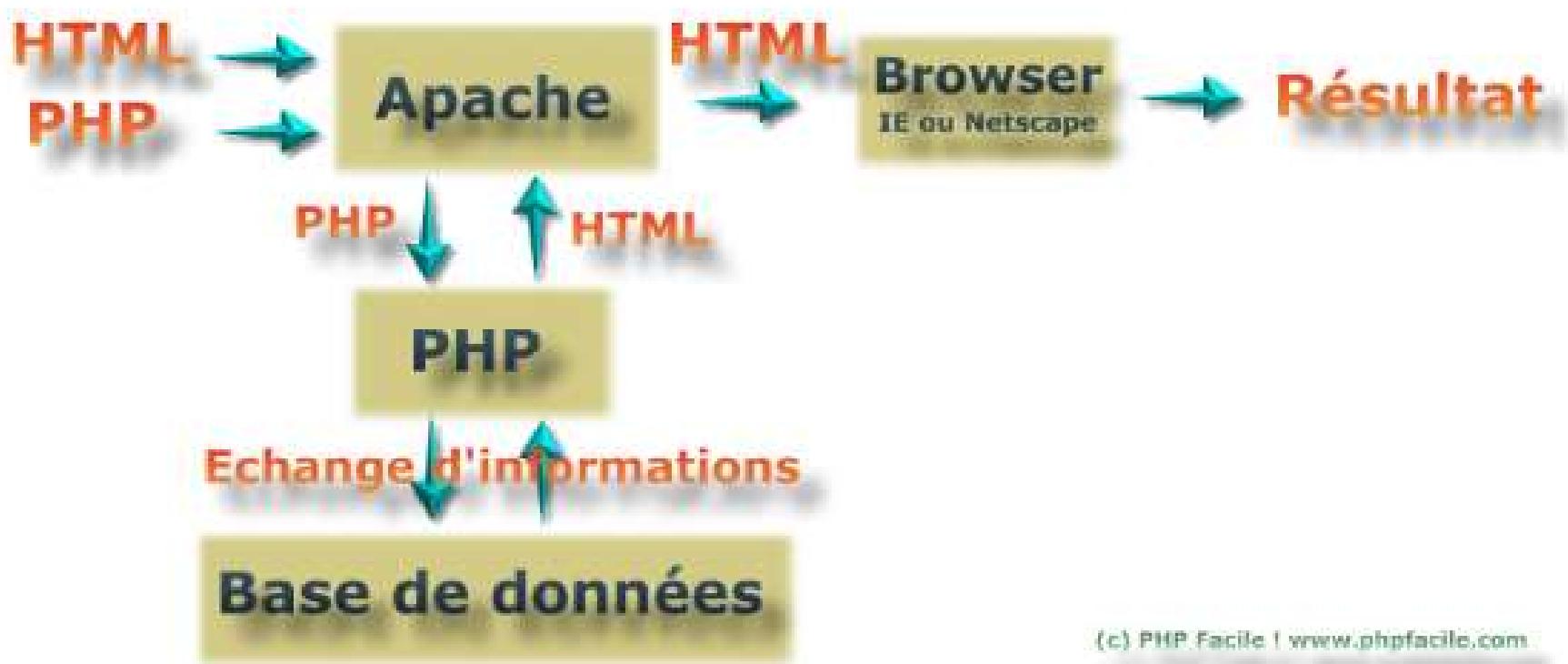


# Protocole Http

- Version : HTTP 1.1 (RFC 2068)
  - par défaut, la connexion maintenue tant que le serveur ou le client ne décide pas de la fermer
- HTTP est un protocole **sans état**
  - aucune info conservée entre 2 connexions
  - permet au serveur HTTP de servir plus de clients en un temps donné
  - pour conserver des infos entre deux transactions, utilisation de cookies, champs cachés de formulaire



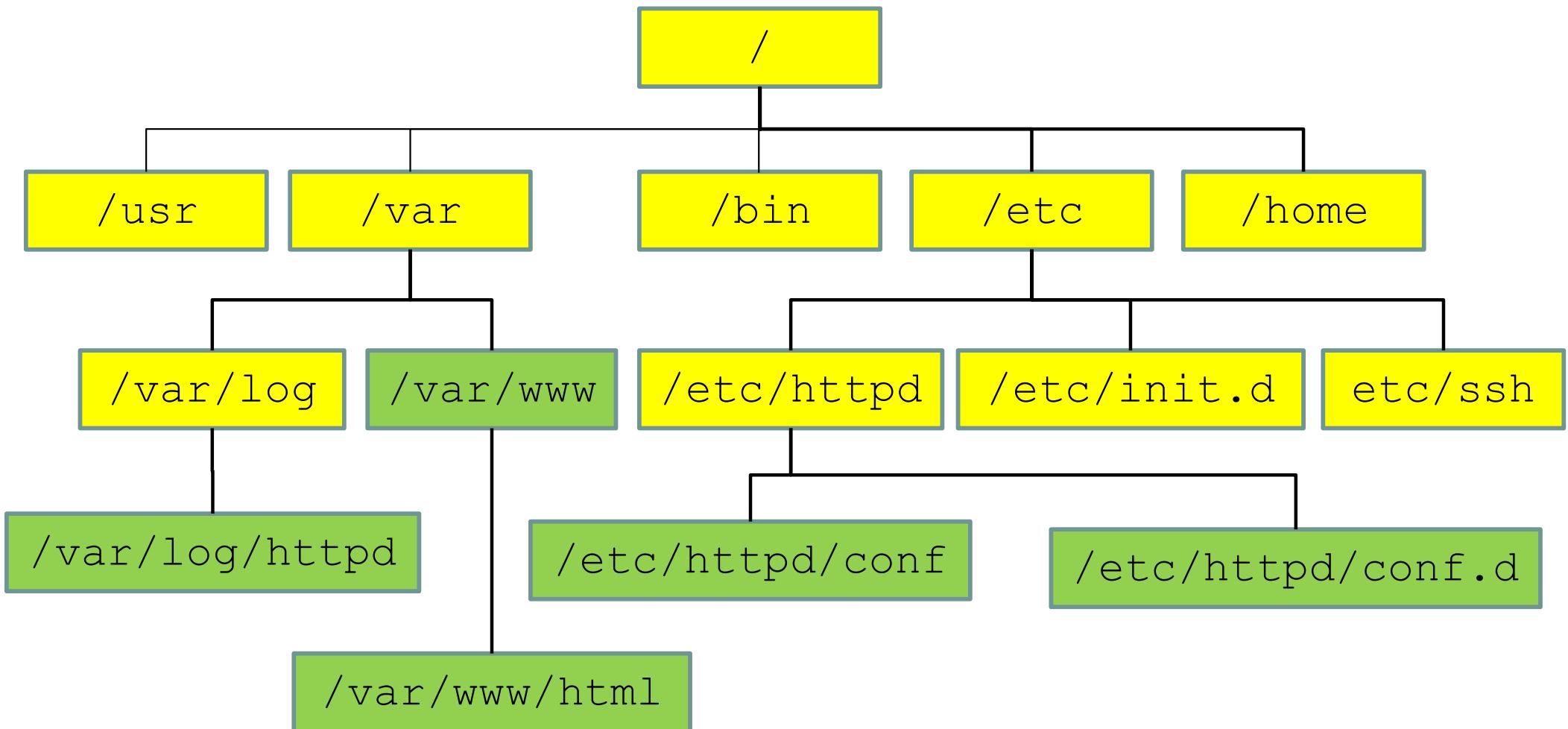
# Apache Server



(c) PHP Facile | [www.phpfacile.com](http://www.phpfacile.com)



# Apache Server





# Apache Server

- Les fichiers **.htaccess** :

- Modifient la config du serveur au niveau d'un **répertoire**
- Ses directives s'appliquent au répertoire et **TOUS** ses **sous-réertoires**
- Syntaxe identique à celle de httpd.conf
- Si AllowOverride None -> fichiers .htaccess ignorés
- Si AllowOverride All -> toute directive valable dans le répertoire .htaccess sera autorisée dans ce fichier



# Dhcp

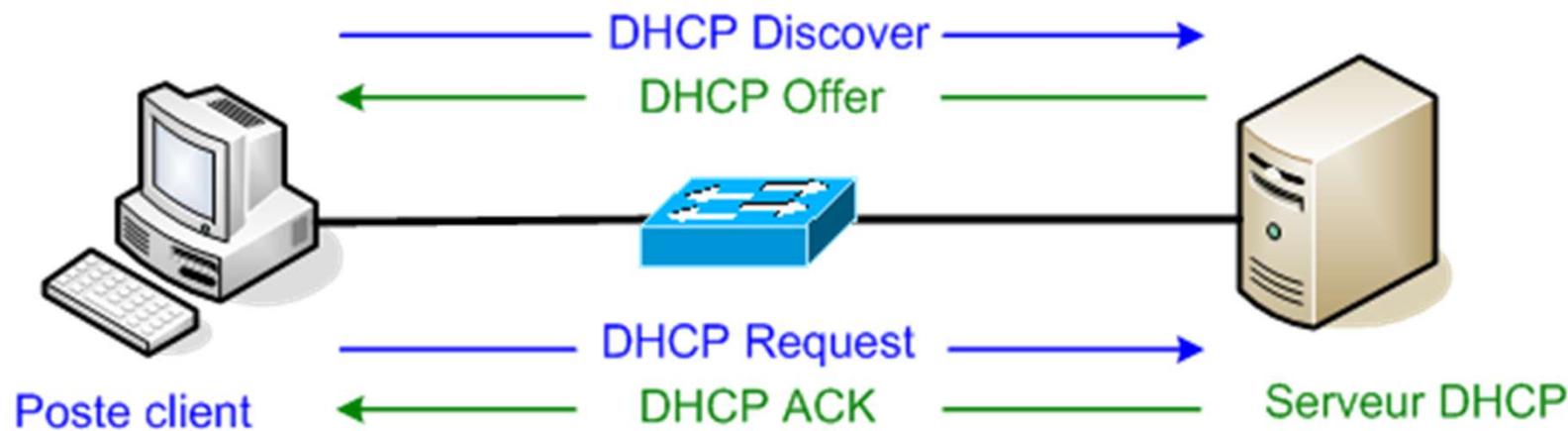


# DHCP

- Dynamic Host Configuration Protocol.
- Mode client-serveur
- Défini dans les RFC 1533, 1534, 1541 et 1542
- objectif : centraliser et automatiser la configuration TCP/IP des machines d'un réseau. Pour cela le serveur utilise des plages d'adresses IP ou **étendues** (scope) ainsi qu'une durée de validité (**bail**).



# DHCP



acquisition du bail d'adresse IP



# DHCP – mécanisme 1

- Au démarrage de la machine client, diffusion d'une première trame DHCP DISCOVER pour rechercher un serveur DHCP.
  - **MAC dest** : FF:FF:FF:FF:FF:FF
  - **IP dest** : 255.255.255.255
  - **UDP dest** : 67 (**serveur Bootp**)
  - **MAC src** : **adresse MAC client**
  - **IP src** : 0.0.0.0
  - **UDP src** : 68 (**client Bootp**)
- Tous les serveurs DHCP potentiels proposent au client une offre s'ils ont des adresses disponibles pour le réseau considéré.



## DHCP – mécanisme 2

- Une trame de diffusion DHCP OFFER est transmise au client par le serveur le plus rapide avec les informations:
  - adresse MAC client
  - adresse IP proposée
  - masque associé
  - durée de bail
  - adresse IP serveur DHCP.
  - Options diverses.
- Si la tentative de connexion n'aboutit pas, itération toutes les 5 mn.



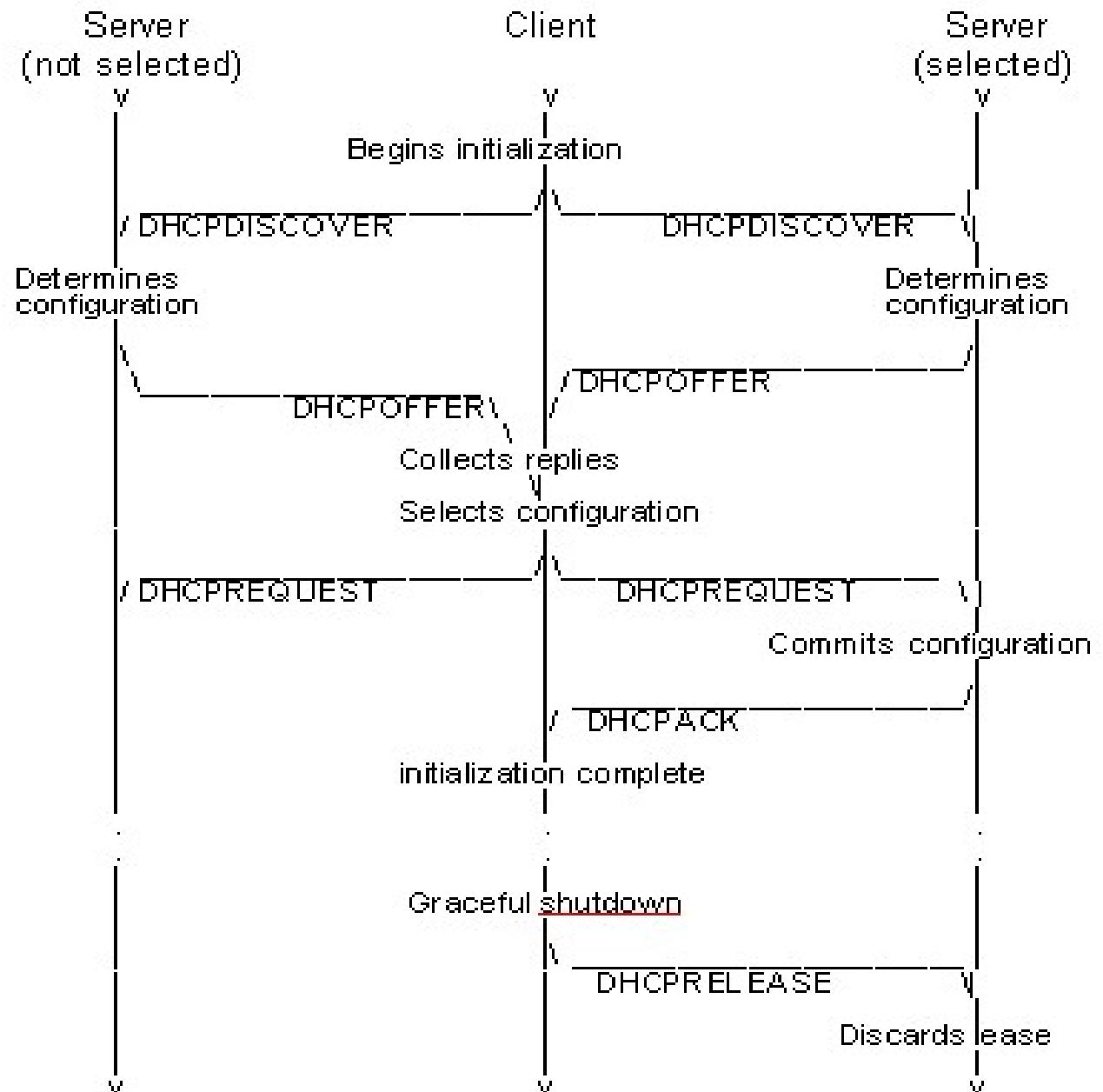
## DHCP – mécanisme 3-4

3. Le client répond au serveur choisi via une trame de diffusion DHCP REQUEST transmise à tous les serveurs par le client pour que tous les serveurs aient connaissance du choix du client. Tous les serveurs non concernés se retirent.
4. A réception du choix client, le serveur confirme le bail au client par une trame DHCP ACK.



# Exemple

- Les différentes étapes de l'allocation d'adresse IP.





# FTP



# Serveur FTP

- Le rôle du serveur FTP consiste à:
  - accepter les requêtes des clients;
  - rechercher la ressource;
  - renvoyer le tout au client.
- Une ressource peut se composer d'un ensemble de fichiers.
- Condition : il est nécessaire que la machine hébergeant le serveur ait une adresse IP fixe



# Exemple : serveur vsftp

Options	Description
<b>listen</b>	Permet de définir si le démon est en standalone (YES) ou dirigé par (x)inetd (NO)
<b>anonymous_enable</b>	Permet d'accepter les connexions anonymes
<b>local_enable</b>	Oblige les personnes à s'identifier avec un compte utilisateur
<b>write_enable</b>	Permission d'écriture
<b>chroot_local_user</b>	Permet de restreindre la connexion utilisateur
<b>xferlog_file</b> 33	Fichiers de log des accès au serveur FTP

1. All users are jailed by default:

```
chroot_local_user=YES
```

```
chroot_list_enable=NO
```

2. Just some users are jailed:

```
chroot_local_user=NO
```

```
chroot_list_enable=YES
```

```
# Create the file /etc/vsftpd.chroot_list with a list of the  
jailed users.
```

3. Just some users are "free":

```
chroot_local_user=YES
```

```
chroot_list_enable=YES
```

```
# Create the file /etc/vsftpd.chroot_list with a list of the  
"free" users.
```



# Exemple de config

## vsftp.conf

```
listen=YES  
anonymous_enable=YES  
local_enable=YES  
write_enable=NO  
xferlog_file=YES  
ftpd_banner=/etc/ma_banniere  
chroot_local_user=NO  
chroot_list_enable=YES  
chroot_list_file=/etc/vsftpd.chroot_list
```



# DNS



# Le problème

Niveau utilisateur

**www.google.fr**

Niveau réseau

En-tête  
IP



- Problème à résoudre : Comment relier les adresses IP utilisées pour acheminer les paquets aux noms utilisés par les utilisateurs ou les applications ?



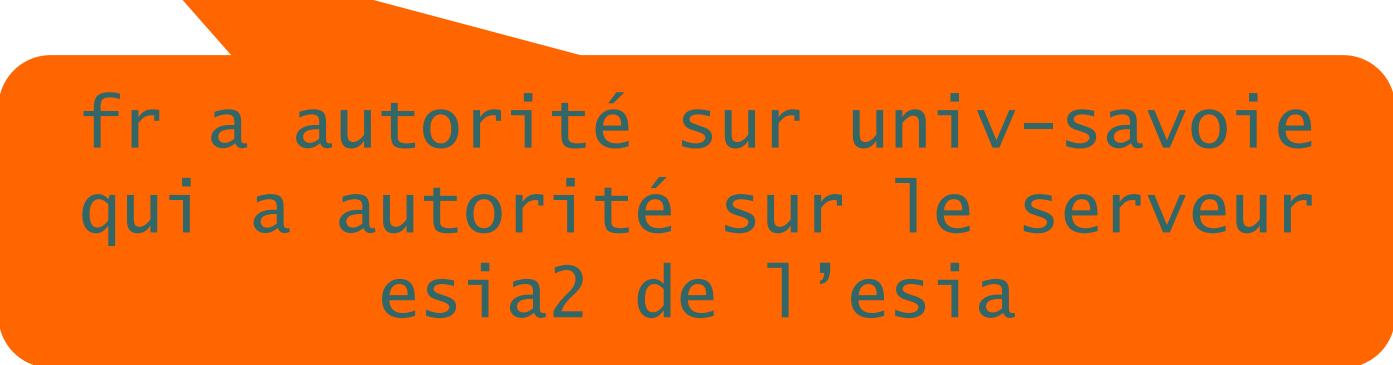
# DNS

- Chaque domaine détient la responsabilité de ses sous-domaines
- Exemples :

**cs.yale.edu**

**Info.itu.ch**

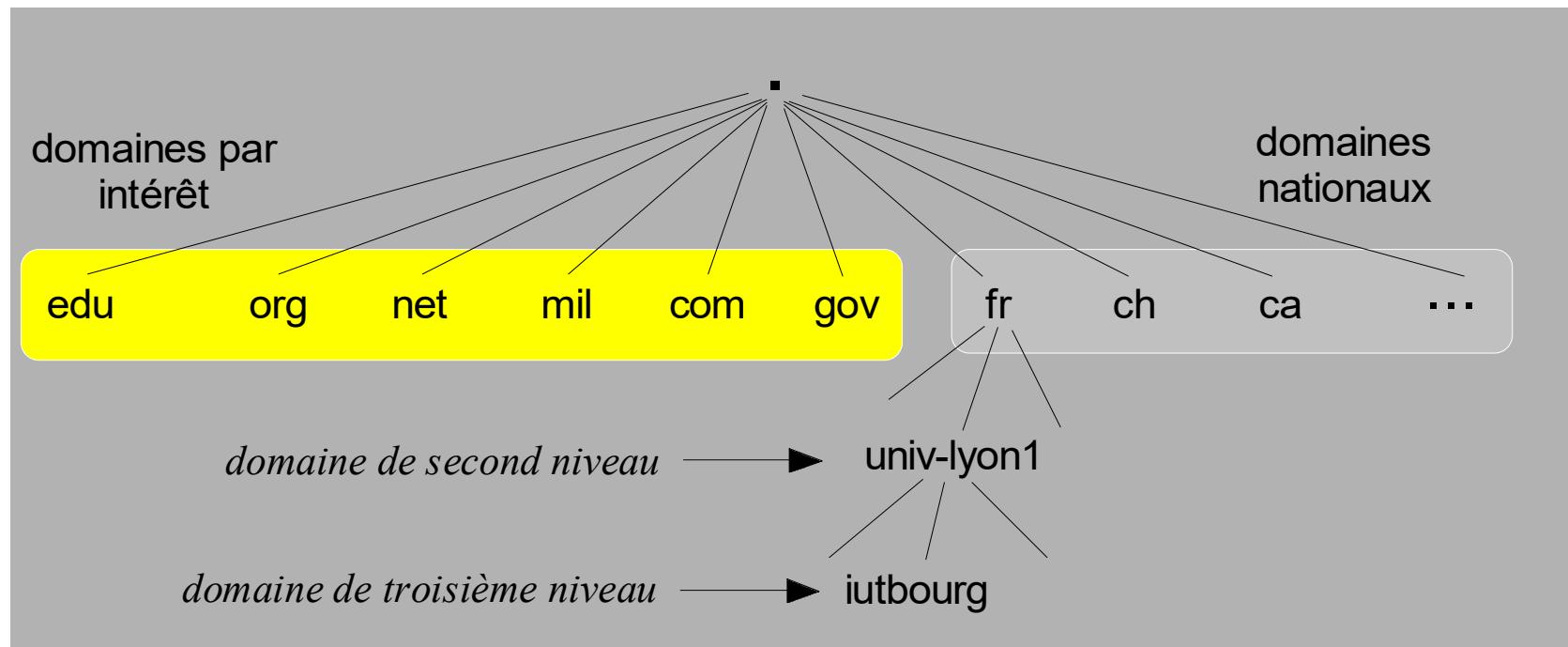
**www.polytech.univ-savoie.fr**



fr a autorité sur univ-savoie  
qui a autorité sur le serveur  
esia2 de l'esia



# DNS



**Modèle hiérarchique de l'Internet**



# DNS root servers

A.ROOT-SERVERS.NET.	6D	IN	A	198.41.0.4
B.ROOT-SERVERS.NET.	6D	IN	A	128.9.0.107
C.ROOT-SERVERS.NET.	6D	IN	A	192.33.4.12
D.ROOT-SERVERS.NET.	6D	IN	A	128.8.10.90
E.ROOT-SERVERS.NET.	6D	IN	A	192.203.230.10
F.ROOT-SERVERS.NET.	6D	IN	A	192.5.5.241
G.ROOT-SERVERS.NET.	6D	IN	A	192.112.36.4
H.ROOT-SERVERS.NET.	6D	IN	A	128.63.2.53
I.ROOT-SERVERS.NET.	6D	IN	A	192.36.148.17
J.ROOT-SERVERS.NET.	6D	IN	A	198.41.0.10
K.ROOT-SERVERS.NET.	6D	IN	A	193.0.14.129
L.ROOT-SERVERS.NET.	6D	IN	A	198.32.64.12
M.ROOT-SERVERS.NET.	6D	IN	A	202.12.27.33

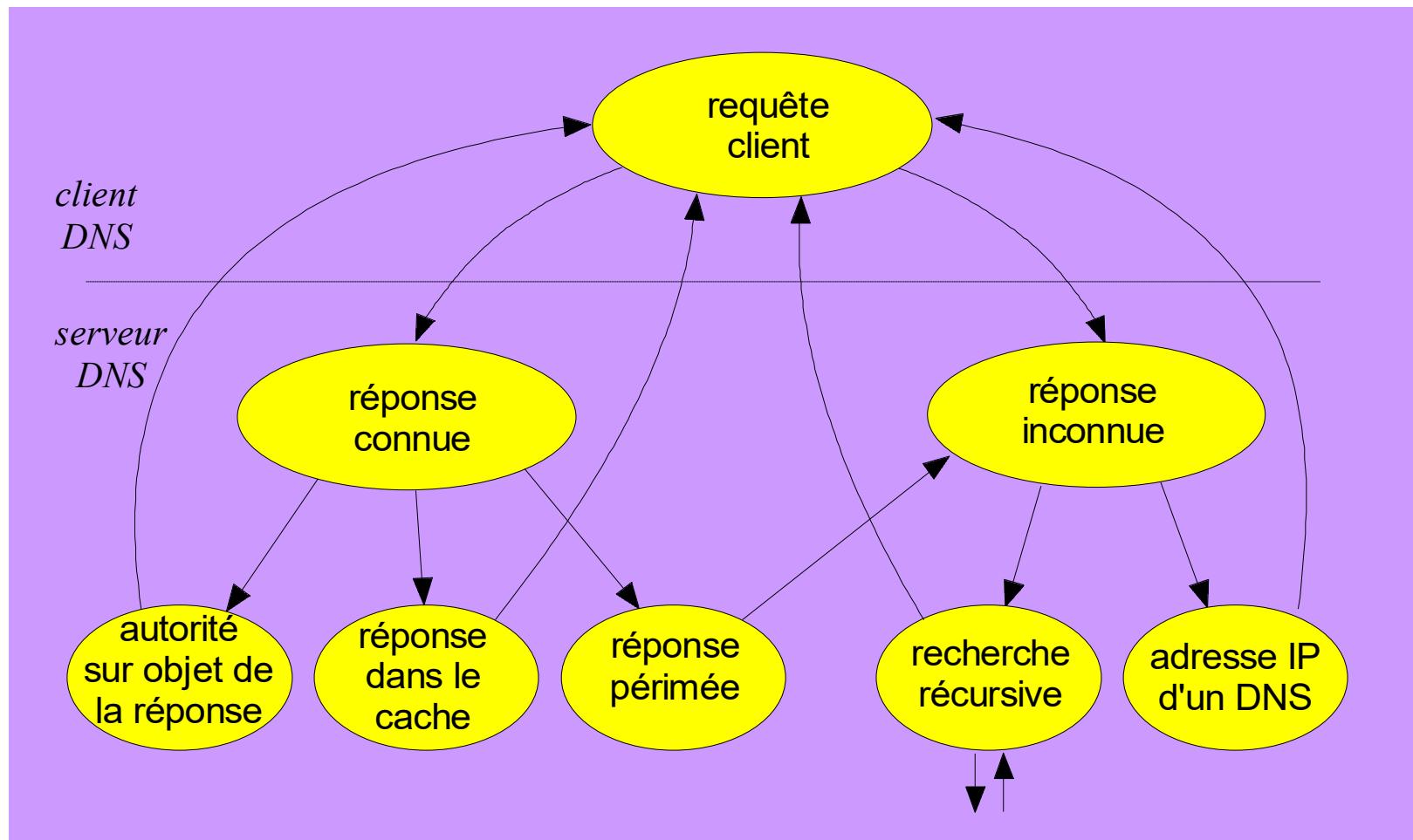


# DNS - principe

- Équivalence entre noms et adresses IP gérée par le DNS (Domain Name System)
- Avant de consulter le serveur : consultation d'un fichier de cache
- **UNIX** : /etc/hosts
- **Windows** :  
\\Windows\system32\drivers\etc\hosts
- Résolution de noms d'hôtes



# DNS : client/serveur





# Recherche récursive

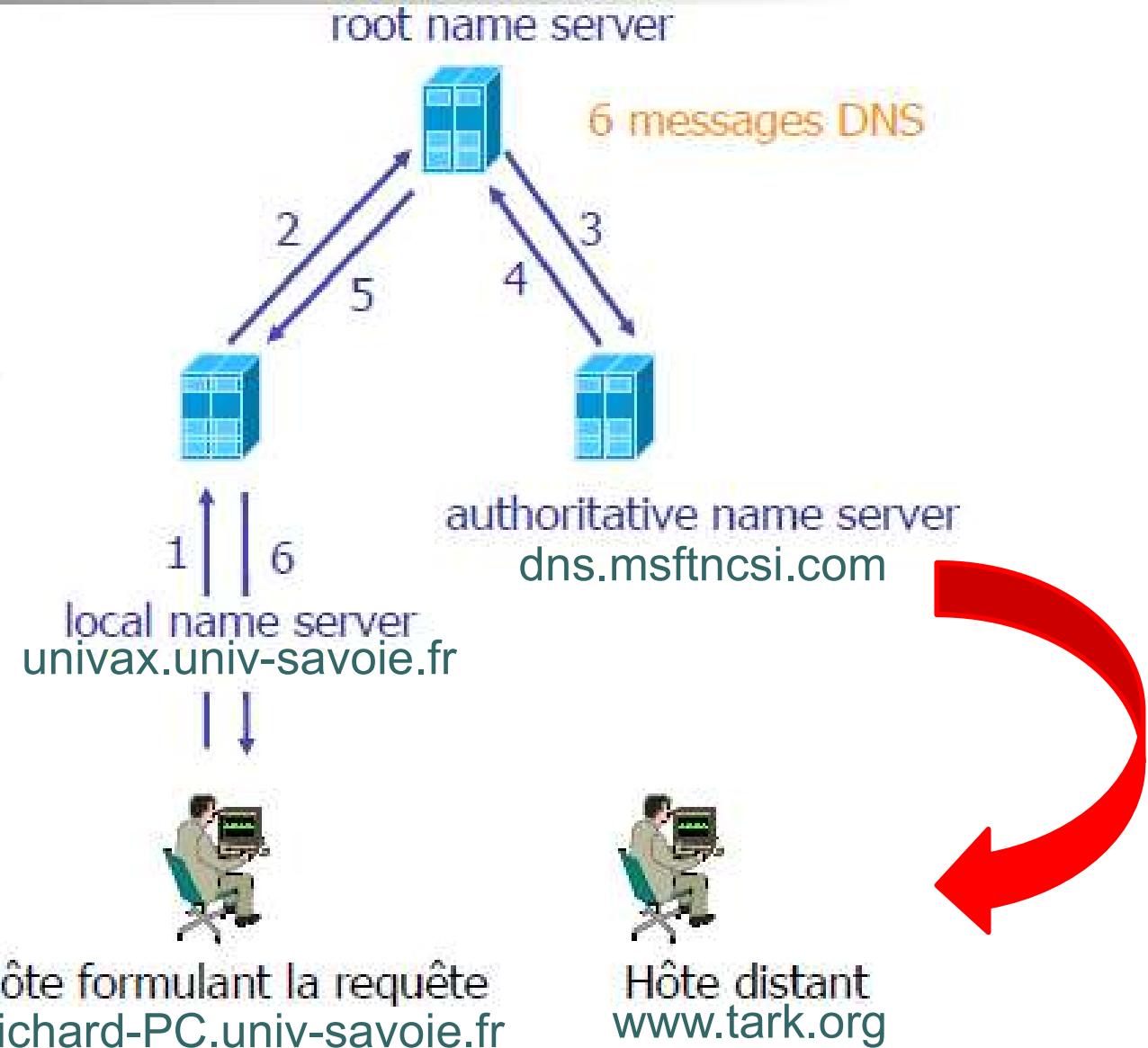
le serveur de noms racine  
contacte le serveur de nom  
"authoritative" (si nécessaire)



Le DNS local contacte le  
serveur de noms racine (si  
nécessaire)



L'hôte contacte son  
serveur DNS local





# Le cache DNS

- Principe : réduire le temps de réponse d'une résolution de nom, en diminuant le nombre de messages DNS en transit nécessaires
  - le serveur de noms stocke dans son cache les informations récentes (en particulier les enregistrements de type NameServer)
  - Expiration des données après un certain temps TTL (environ 2 jours)
  - "no authoritative" le serveur n'a pas autorité sur l'enregistrement



# Messages DNS

- RFC 1034, 1035
- Format unique pour les Requêtes/Réponses
- 12 octets d'en-tête DNS
- identifiant qui permet au client d'associer la réception d'une réponse à une requête formulée
- les fanions : 2 octets





# Messages DNS

512 octets max. avec UDP

Identification	Fanions	12 octets d'en-tête	
Nombre de questions	Nombre de réponses (RR)		
Nombre de "authority servers"	Nombre de RR supplémentaires		
Questions (nombre variable de questions)			
Réponses (RRs répondant à la demande)			
Noms des serveurs de source autorisée (RRs des serveurs primaire/secondaire)			
Adresses des serveurs de source autorisée (nb. variable de RRs)			

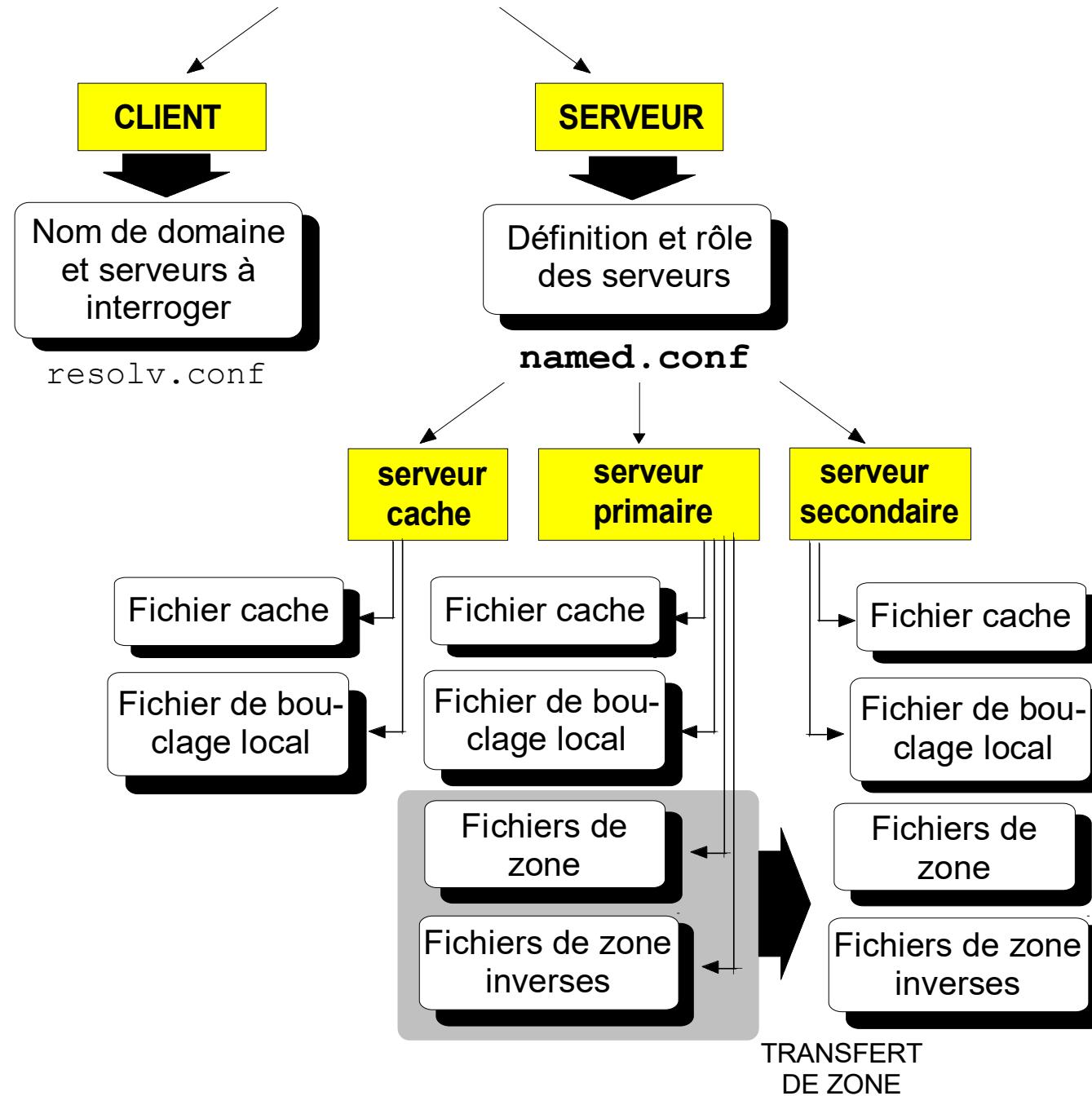
QUESTION SECTION

ANSWER SECTION

AUTHORITY SECTION

ADDITIONAL SECTION

# DNS





# Serveur DNS

- Trois possibilités:

- Serveur DNS cache:

- pas de zones gérées, mais seulement une mémoire cache et exécutent des requêtes récursives pour le compte de clients

- Serveur DNS secondaire:

- Intérêt tolérance aux pannes, décharge du primaire, transfert automatique des fichiers de zone

- Serveur DNS primaire:

- Configuration plus complexe, fichiers de zone, possibilité d'automatisation (dynamic dns)

# DNS - serveur cache UNIX

Répertoire où sont stockés les fichiers de zone

Fichier contenant les serveurs du domaine racine

```
o Configurer le fichier principal named.conf
options {
    directory "/var/named";
};

zone "." {
    type hint;
    file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "pz/127.0.0";
};

fichier /var/named/pz/127.0.0
```



# DNS – serveur primaire

```
options {
    directory "/var/named";
};

zone "." {
    type hint;
    file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "zone/127.0.0";
};

zone "land-5.com" {
    type master;
    file "zone/land-5.com";
};

zone "177.6.206.in-addr.arpa" {
    type master;
    file "zone/206.6.177";
};
```

FICHIERS  
DE ZONE



# DNS – enregistrements

RFC 1033 :

<nom> [ttl]<classe> <type> <données>

avec :

- <nom> : nom du domaine (ou d'une machine). Le symbole @ fait référence au domaine explicite spécifié par \$ORIGIN.
- [ttl] : champ optionnel donnant la durée de vie en secondes des informations du fichier. Par défaut, ttl = valeur définie dans le champ minimum de l'enregistrement SOA.
- <classe> : classe d'adresses de l'enregistrement de ressources. ex: IN = domaine Internet.
- <type> : décrit le type des données fournies à la suite. ex : SOA, NS, A, PTR, CNAME, MX et HINFO.
- <données> : sa structure dépend du champ <type>.



# DNS – enregistrements

- SOA : associé à chaque zone DNS (Start of Authority)
- NS : Name Server liste les serveurs du domaine ( primaire et secondaires)
- A : associe un nom d'hôte à une adresse IP
- CNAME : définit un alias (dissimule le nom d'un serveur)
- MX : serveur de messagerie (si plusieurs, on spécifie la priorité par un entier)
- PTR : équivalents au type A pour la recherche inversée

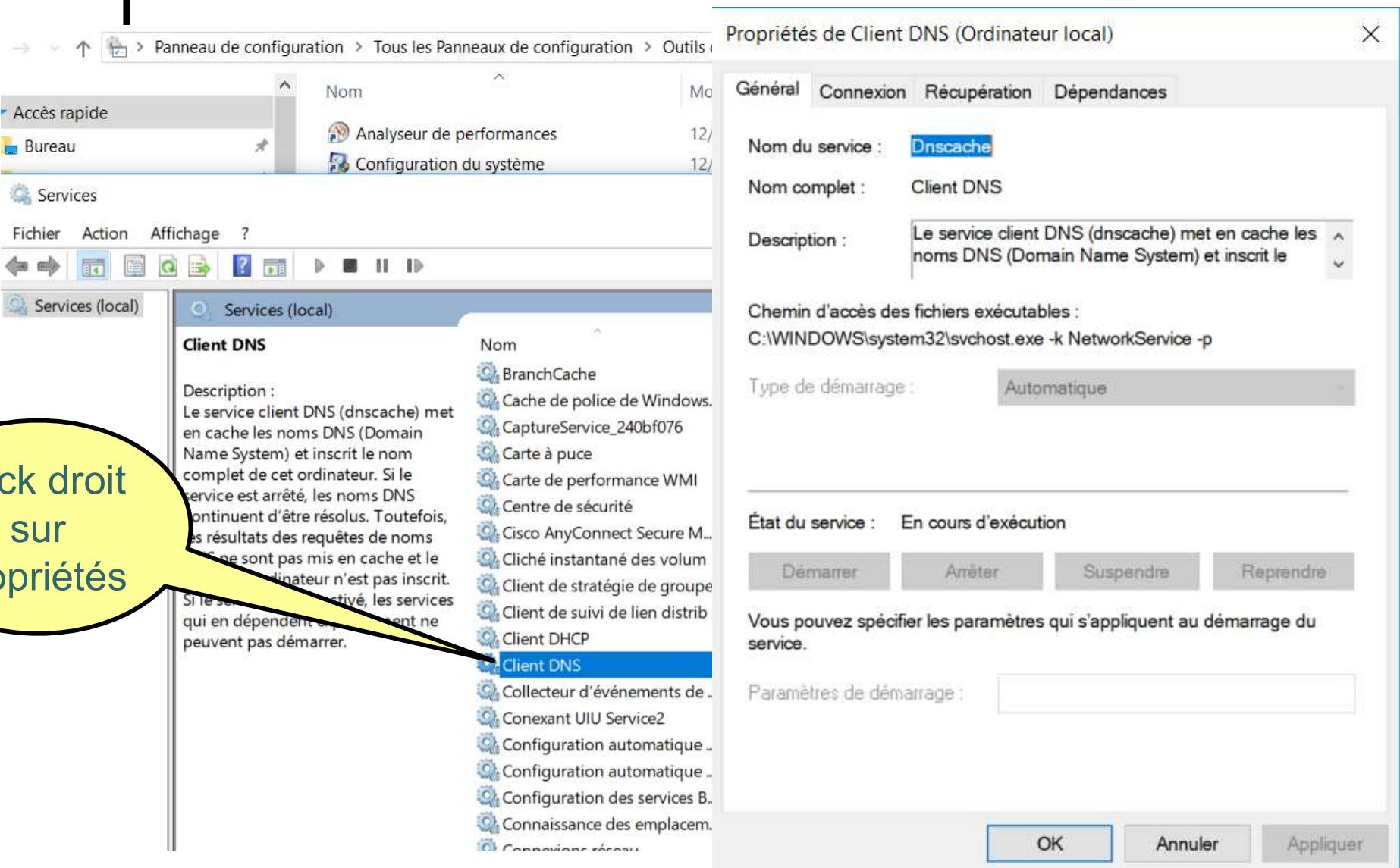


# Client DNS

- Sur Linux, deux fichiers de configuration sont associés au client :
  - **/etc/resolv.conf** permet de spécifier comment et à qui formuler les requêtes. Format :

```
search <domain_name>
Nameserver <adr_IP_server>
```
  - **/etc/host.conf** paramétrage du resolver (man host.conf), en particulier l'ordre de résolution  
order hosts,bind

# Client Windows 10





# La commande dig

- dig (domain information groper).
- envoi de requêtes uniques à un serveur DNS aux fins de tests ou d'écriture (non interactive).

**dig @<serveur> <domaine> <type\_requete>**

- <serveur> est la machine sur laquelle est exécuté le démon DNS à rechercher,
- <domain> est le domaine d'intérêt et
- <type\_requete> est un des enregistrements tels que A, ANY, MX, NS, SOA, HINFO, AXFR



# NFS, SMB



# Accès aux fichiers distants

- Principe :

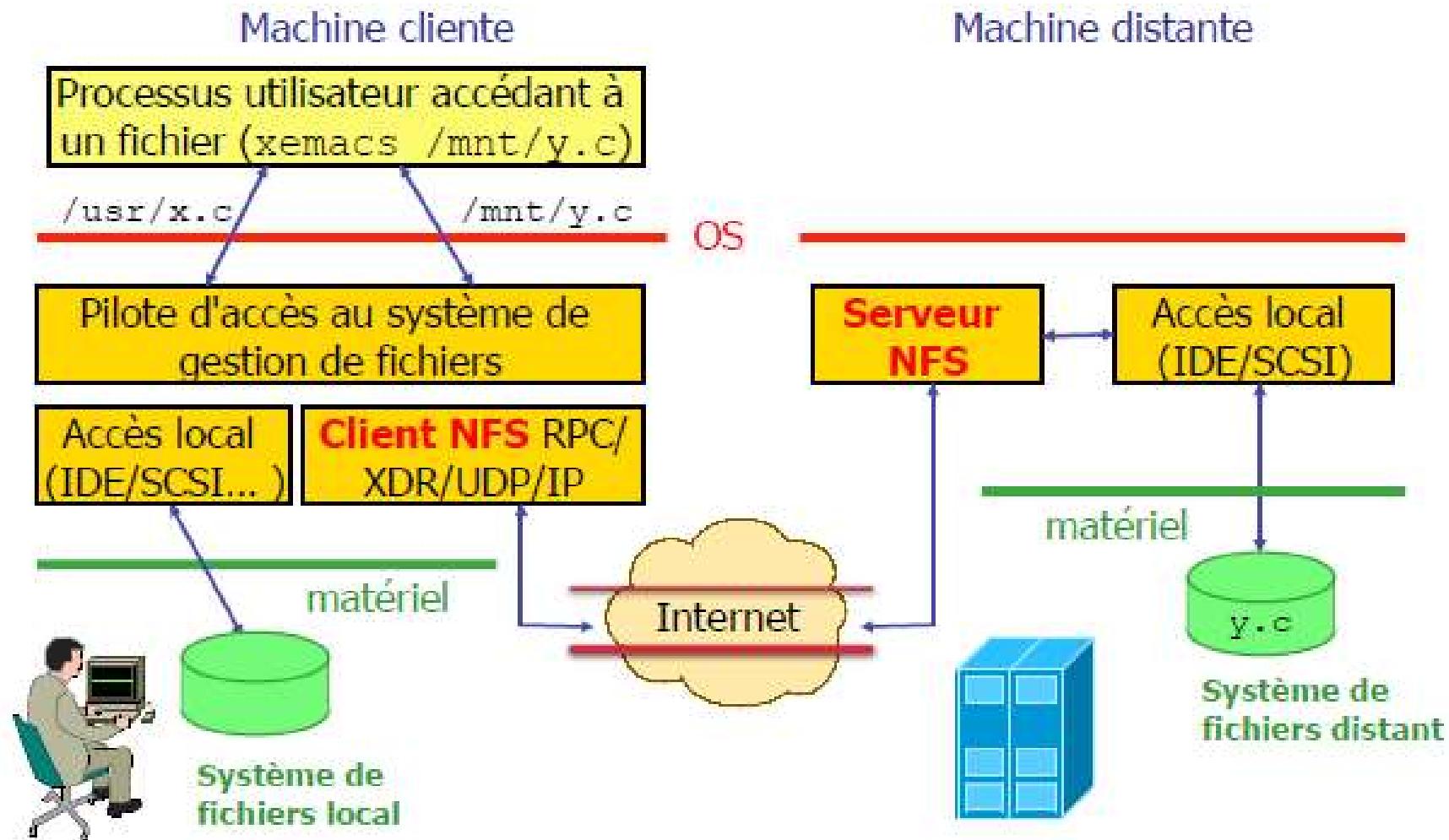
- tout se passe comme si le système de fichiers distant était local
- l'utilisateur peut éditer le fichier, le modifier, ... les modifications seront répercutées sur le système de fichiers distant

- Les deux principaux protocoles

- NFS : Network File System (Unix/Sun-RPC)
- SMB : Server Message Block (Microsoft)

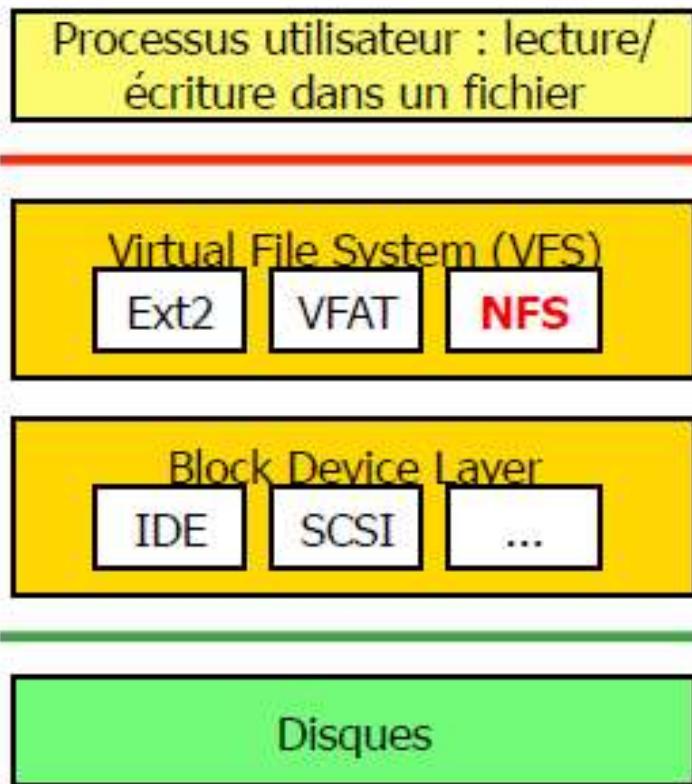


# NFS : principe





# NFS : principe



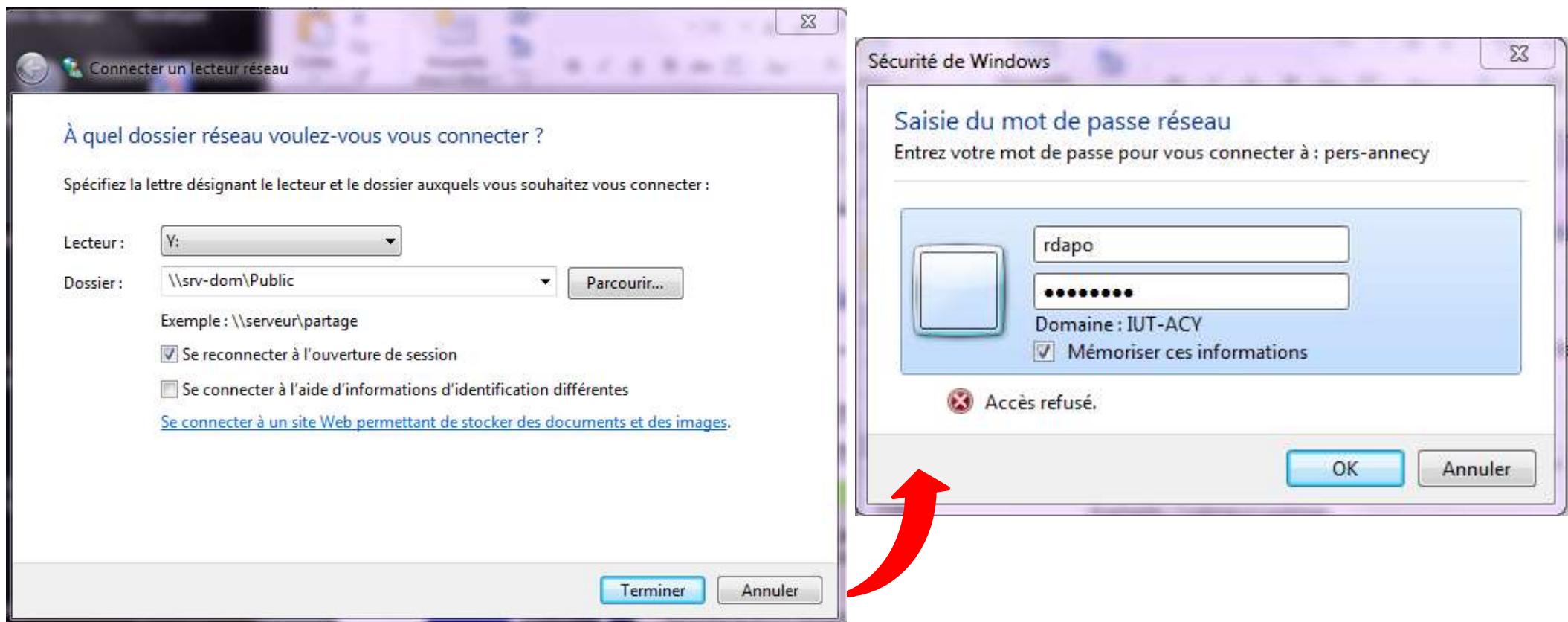
- Manipule des chemins, descripteurs, déplacements
- Manipule des Files, Dentries, Inodes, déplacements
- Masque les différences à l'application (API uniforme, ...)
- Manipule des blocks
- Matériel-dépendant



# NFS : caractéristiques

- Le serveur NFS est sans état : entre deux requêtes, pas de mémorisation :
  - des accès précédents à un fichier donné,
  - des fichiers ouverts, ...
- Protocole sans état :
  - en cas de crash du serveur NFS
    - permet de simplifier son redémarrage
    - transparent pour le client : pas besoin de réitérer certaines requêtes en cas de crash

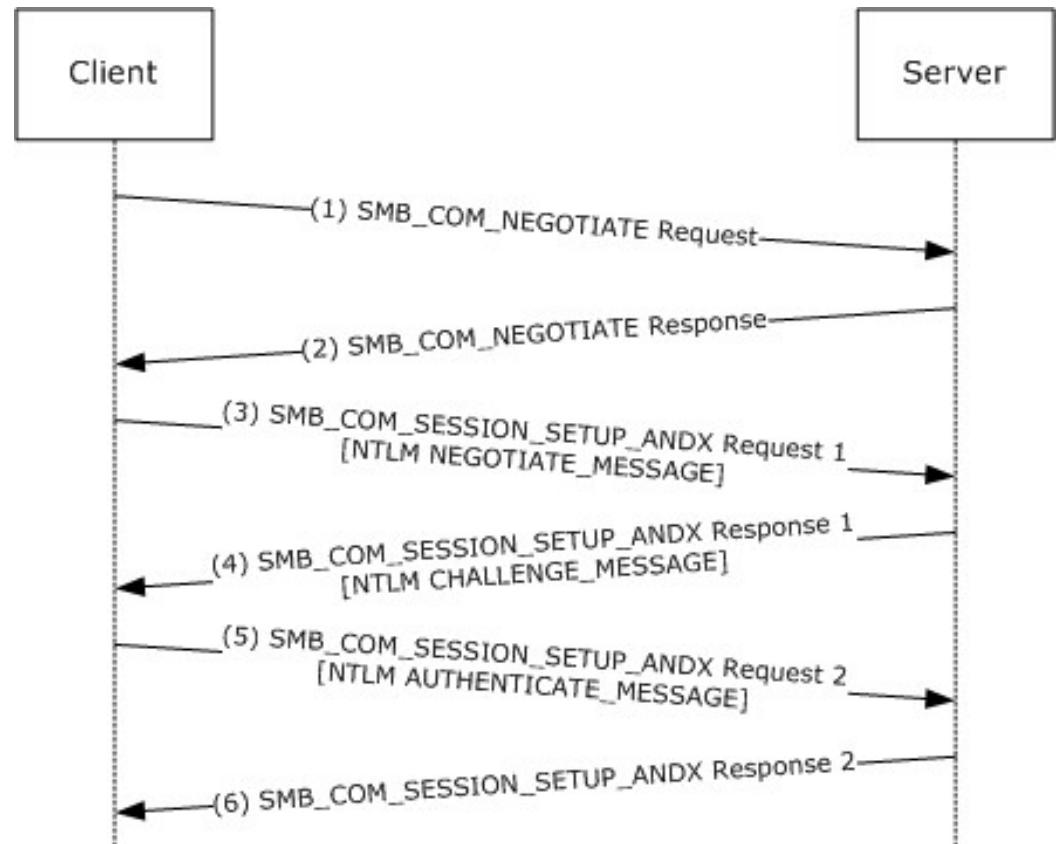
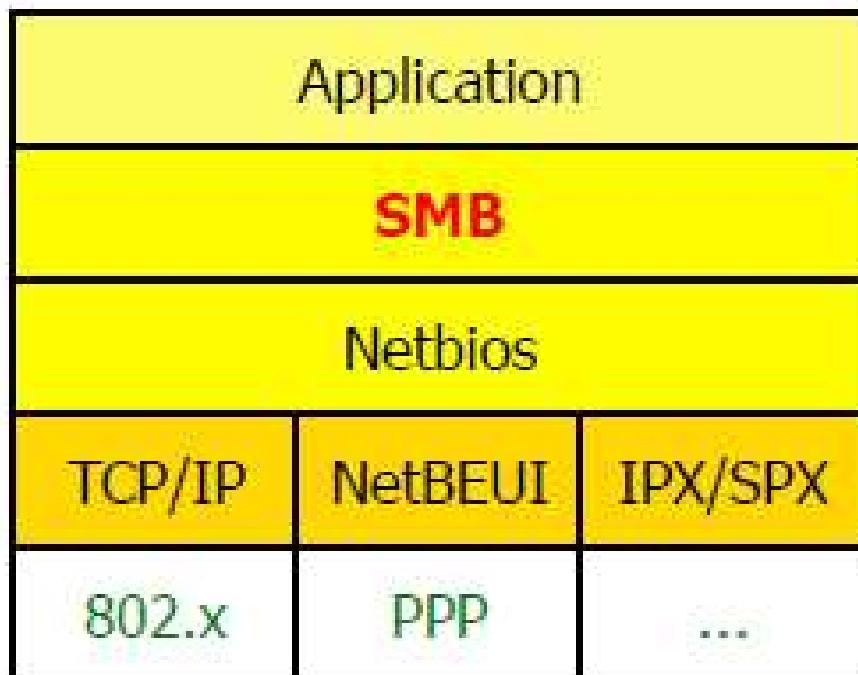
# SMB (Server Message Block)



- Protocole de Microsoft et Intel permettant le partage de ressources (disques, imprimantes...) à travers un réseau



# SMB : principe





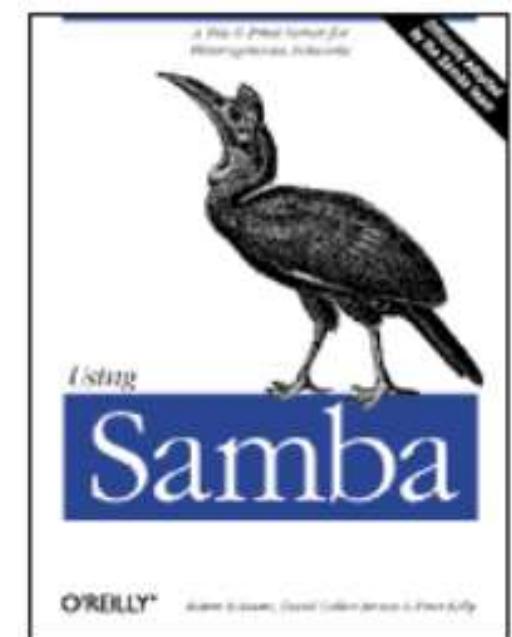
# SMB : sécurité

- Deux niveaux de protection des accès :
  - au niveau de chaque utilisateur : basé sur le nom des utilisateurs (user, passwd), permet de gérer l'accès aux ressources et aux éléments d'une ressource
  - au niveau de chaque ressource : un mot de passe commun à tous les utilisateurs est associé à une ressource pour y autoriser l'accès



# Samba

- Samba : implémentation de SMB sous Unix qui permet un partage de ressources entre les mondes Unix et Windows
- Le serveur Samba sur la machine Unix émule un domaine SMB





# SMTP



# Messagerie électronique

- 4 composants principaux :
  - des agents utilisateurs
  - des serveurs de mail
  - un protocole de transfert de mail : Simple Mail Transfer Protocol (SMTP)
  - un protocole d'accès à la boîte aux lettres (POP, IMAP, ...)

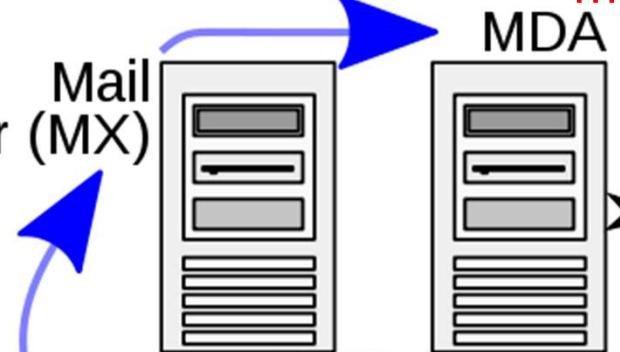


# SMTP

MUA : Mail User Agent, client de messagerie (Ex : ThunderBird),

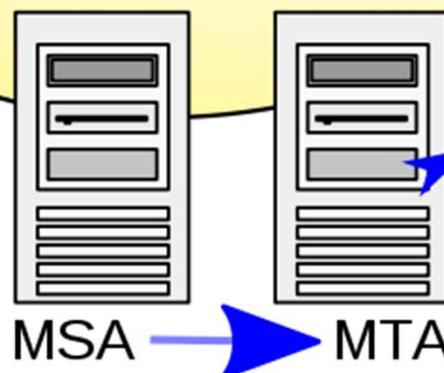


exchanger (MX)



MDA : Mail Delivery Agent, service de distribution des mails

MTA : Mail Transfert Agent, envoi des mails entre les serveurs

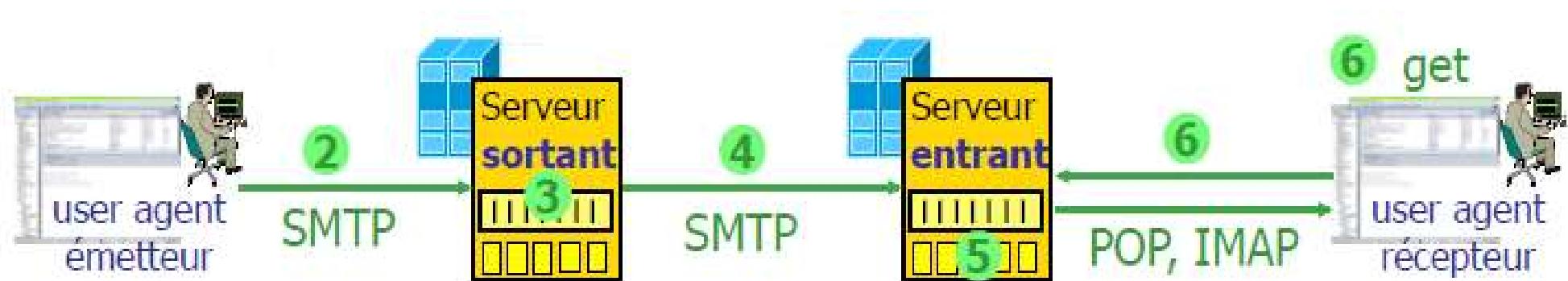




# Courier électronique

Les protocoles d'accès : consulter sa boîte aux lettres (après authentification)

- POP3 : Post Office Protocol v3 [RFC 1939] autorisation (agent ↔ server) et téléchargement
- IMAP4 : Internet Message Access Protocol v4 [RFC 3501] manipulation de messages stockés sur le serveur
- HTTP (Webmail) : Hotmail , Yahoo





# POP3

- Post Office Protocol
- écoute sur le port 110 d'un serveur
- Objectif : permettre à l'utilisateur de relever son courrier depuis un hôte qui ne contient pas sa boîte aux lettres.
- dialogue entre le logiciel de messagerie (MUA) et la boîte aux lettres de l'utilisateur sur le serveur



# IMAP4

- Internet Message Access Protocol, version 4
- écoute sur le port 143 d'un serveur
- protocole de récupération de mails plus évolué que POP3
- possibilité de pouvoir lire uniquement les objets des messages (sans le corps)
- les mails restent stockés dans des dossiers sur le serveur



# Webmail

- Agent utilisateur = Web browser
- Utilise le protocole HTTP (ou HTTPS) pour communiquer avec les serveurs SMTP/IMAP
  - le serveur HTTP exécute des scripts qui utilisent
    - le protocole IMAP pour communiquer avec le serveur IMAP et manipuler les messages distants de l'utilisateur
    - le protocole SMTP pour traduire une demande d'envoi d'un message de la part de l'utilisateur



# Protocole SMTP

- RFC 821
- Transfert serveur émetteur - serveur récepteur
- 3 phases de transfert
  - handshaking (établissement de la connexion)
  - transfert d'un ou plusieurs messages
  - fermeture de la connexion
- connexions **persistentes** (les messages à destination du même serveur transiteront tous sur la même connexion TCP)



# Commandes SMTP

Commande	Description
<b>HELO</b> nom_client	identifie le client SMTP ; établit la connexion
<b>MAIL</b> From: <@exp>	identifie l'expéditeur du message
<b>RCPT</b> To: <@dest>	désigne le destinataire du message
<b>DATA</b>	indique le début du message (en-tête+corps)
<b>QUIT</b>	termine la connexion
<b>NOOP</b>	pas d'opération ; force le serveur à répondre
<b>RSET</b>	réinitialisation de la saisie de données (DATA)



# Sockets TCP/IP



# Structure des sockets

- **socket** : famille des interfaces programmables supportant les applications réseau ou API (*Application Programming Interface*).
- **but** : deux programmes d'application, l'un s'exécutant sur le système local et l'autre sur un système distant communiquent entre eux grâce à des **primitives standardisées**

# Sockets headers

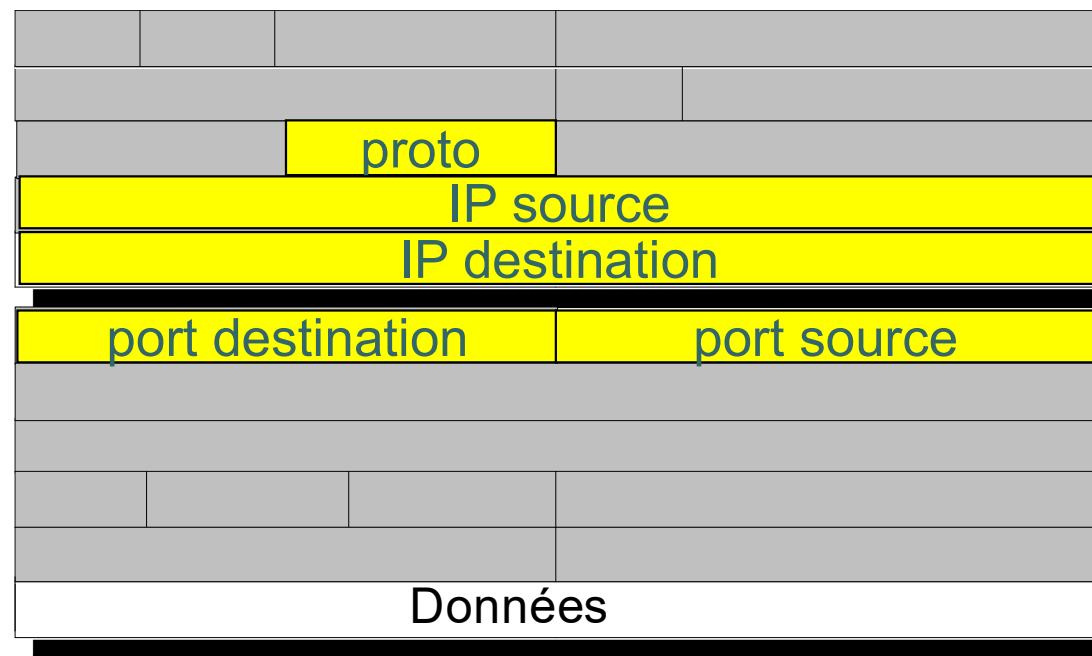
Couche application  
Couche transport  
Couche réseau

telnet (port 23)  
TCP (6)  
196.0.0.10

En-tête  
IP

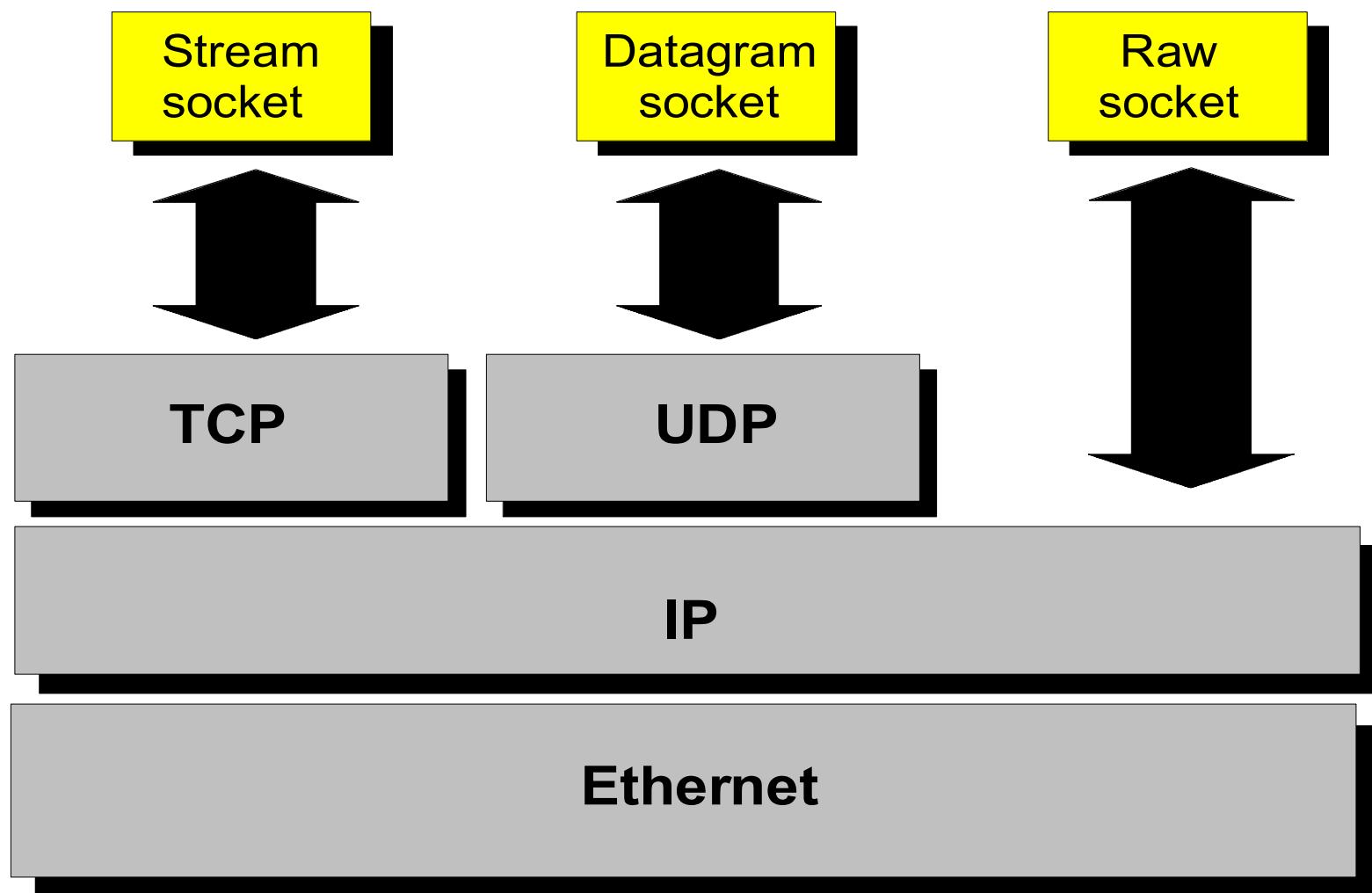
En-tête  
TCP

Données





# Types de sockets

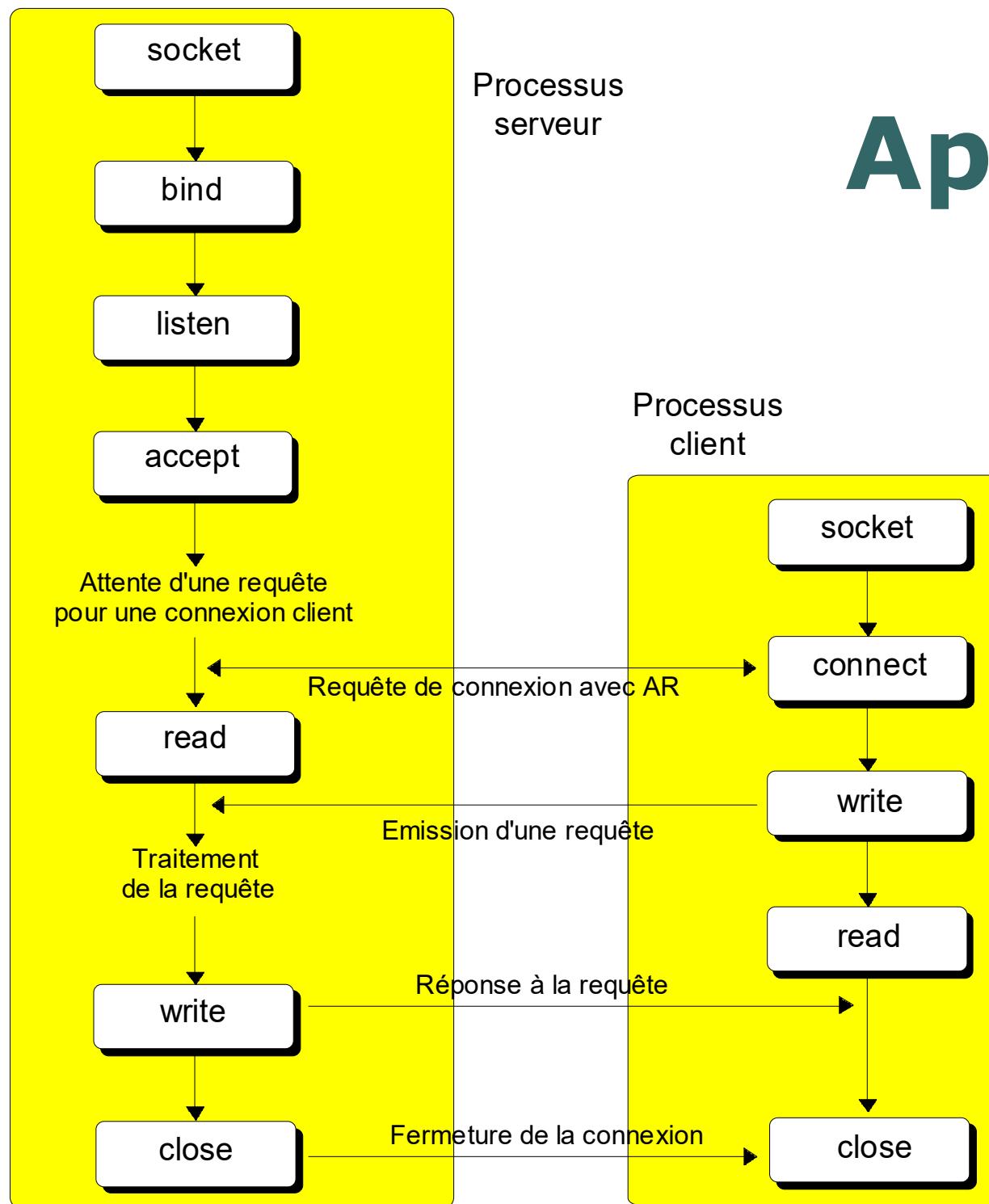




## Structure de socket

- Appels système aux sockets : référence à un pointeur sur une structure de données associée à la socket.
- Les informations contenues dans cette structure identifient de manière unique une socket.

# Appels



Appel système serveur	Appel système client	Informations connexion vues du serveur	Informations connexion vues du client
<i>socket()</i>	<i>socket()</i>	<div style="border: 1px solid black; padding: 5px;">           protocole            port local            adresse IP locale            port distant            adresse IP distante         </div>	<div style="border: 1px solid black; padding: 5px;">           protocole            port local            adresse IP locale            port distant            adresse IP distante         </div>
<i>bind()</i>		<div style="border: 1px solid black; padding: 5px;">           protocole            port local            adresse IP locale            port distant            adresse IP distante         </div>	
<i>accept()</i>		<div style="border: 1px solid black; padding: 5px;">           protocole            port local            adresse IP locale            port distant            adresse IP distante         </div>	
	<i>connect()</i>		<div style="border: 1px solid black; padding: 5px;">           protocole            port local            adresse IP locale            port distant            adresse IP distante         </div>



# Conclusion

- Dans l'architecture client/serveur le client requiert un service sur un serveur ⇒
  - Quel service? ⇒ numéro de port serveur
  - Sur quel serveur? ⇒ adresse IP serveur
- Contrainte : le serveur doit être actif en permanence et à l'écoute sur son port serveur
- Combien de clients peuvent être gérés simultanément? ⇒ nombre prédéfini (`listen()`)
- Comment traiter plusieurs clients? ⇒ `fork()`