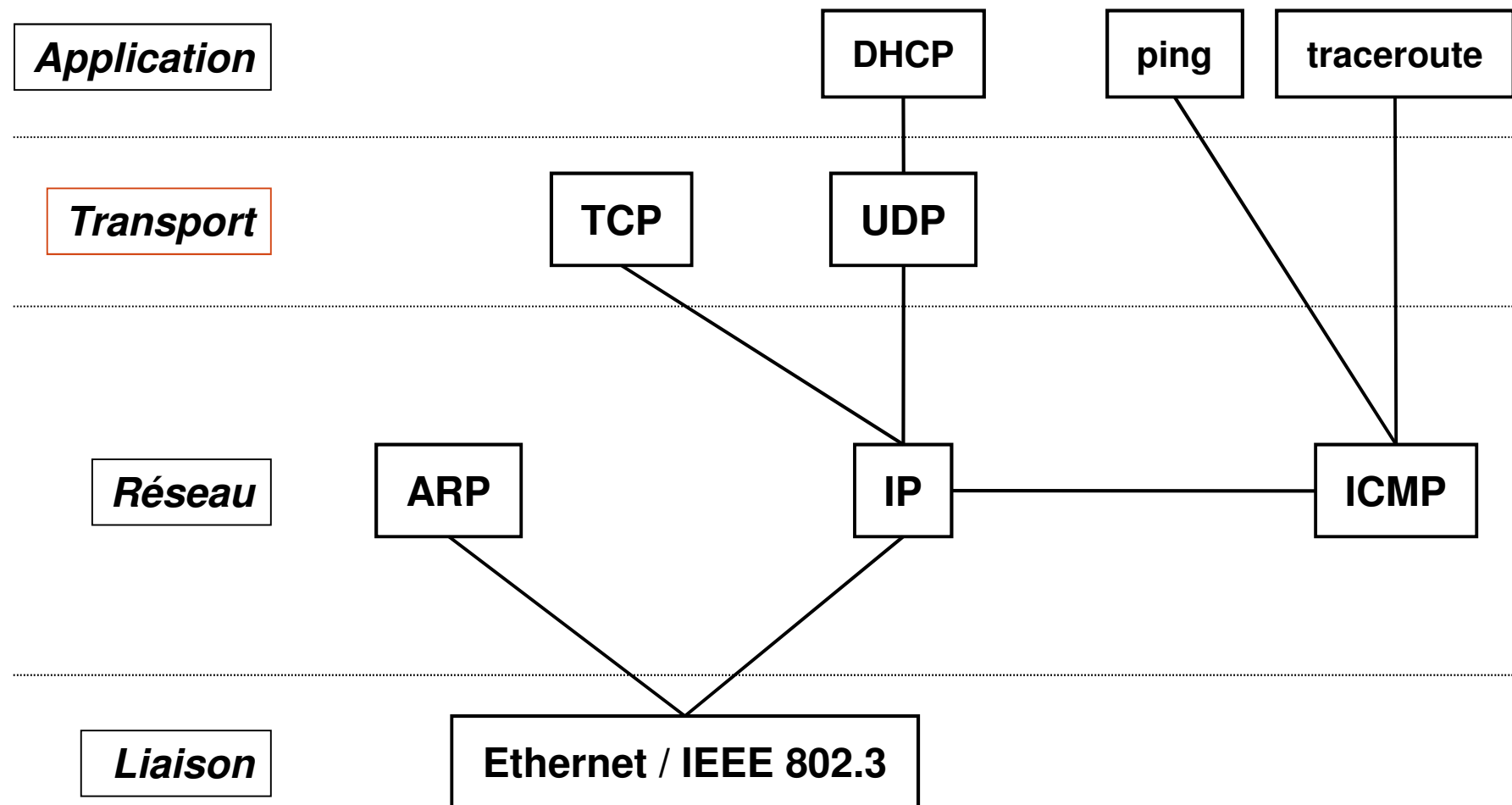


# VIII - Couche transport : TCP, UDP

# Couche 3 : architecture simplifiée



## UDP (User Datagram Protocol) (2)

- UDP est un protocole **non orienté connexion**, c'est-à-dire qu'il n'offre pas de fonction de contrôle du bon acheminement :
  - aucune vérification logicielle de la livraison des messages
  - pas de réassemblage des messages entrants
  - pas d'accusé de réception
  - aucun contrôle de flux
- Cependant, UDP offre l'avantage de nécessiter moins de bande passante que TCP. Il peut donc être intéressant d'utiliser ce protocole pour l'envoi de messages ne nécessitant pas de contrôle de qualité.

# Connexion UDP

- Un transfert UDP est caractérisé par :
  - @ IP source
  - @ IP destination
  - Port source
  - Port destination
- Connexion à usage unique
  - Le port client est rendu après utilisation
  - Le port serveur attend un autre client

# Applications/protocole qui utilisent UDP

- Le protocole HTTP
  - Surcouche de UDP
- Requête HTTP :
  - Le Client demande un port UDP
    - ➔ 1843
  - Le Client envoie datagramme
    - IPclient:1843 ➔ IPserveur:80
  - Le Serveur envoie une réponse (page Web)
    - IPserveur:80 ➔ IPClient:1843
    - Le port 1843 est rendu à la machine Client

# Le protocole TCP

- Transport Control Protocol
- Communication en **mode connecté**
  - Ouverture d'un canal
  - Communication Full-Duplex
  - Fermeture du canal
- La connexion sécurise la communication
  - Ordre garanti
  - Arrivée garantie

# Applications/protocoles qui utilisent TCP

- Quelques exemples
  - HTTP
  - SMTP
  - FTP
  - telnet, rlogin, ssh
- Proportion de trafic TCP dans l'Internet (McCreary et Claffy, 2000)
  - 91% des octets
  - 83% des paquets

# Qu'offre TCP aux applications ?

- Un canal transparent, sans erreurs et bidirectionnel (en mode *full-duplex*) qui transporte une séquence d'octets
  - Protocole de bout en bout et de point à point
  - « Flot d'octets » (*byte-stream*)
  - Service fiable
  - Service orienté connexion
- À chaque octet émis correspond un numéro de séquence
  - Numérotation des données envoyées  $\Rightarrow$  confirmation des données reçues



# Service fiable

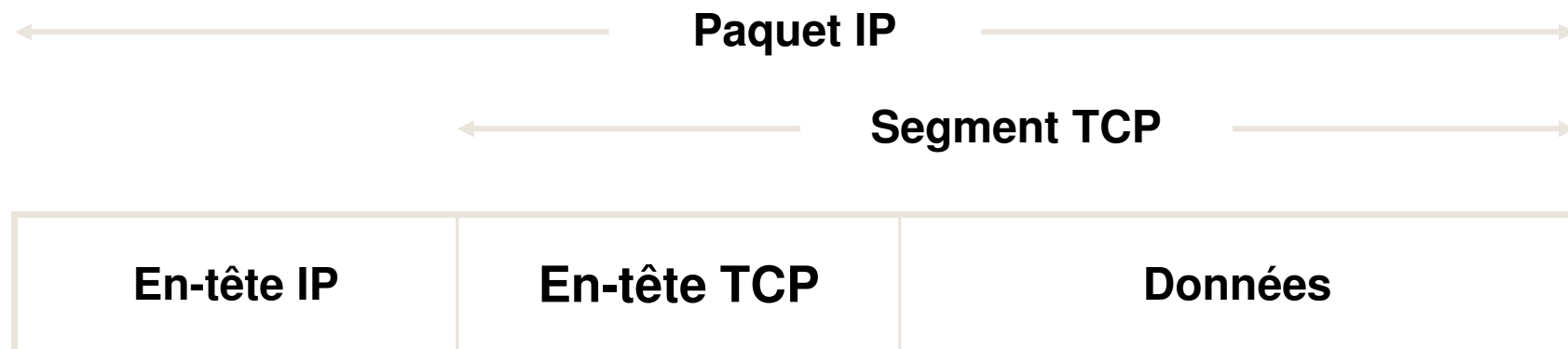
- TCP suppose que la couche inférieur (IP) *n'est pas* fiable
  - Acquittement (*acknowledgement*) des données reçues
  - Retransmission des données perdues
  - Contrôle de flux
  - Ordonnancement des données qui arrivent « dans le désordre »
  - Écartement des données dupliquées
  - Vérification de l'intégrité des données (*checksum*)

# Service orienté connexion

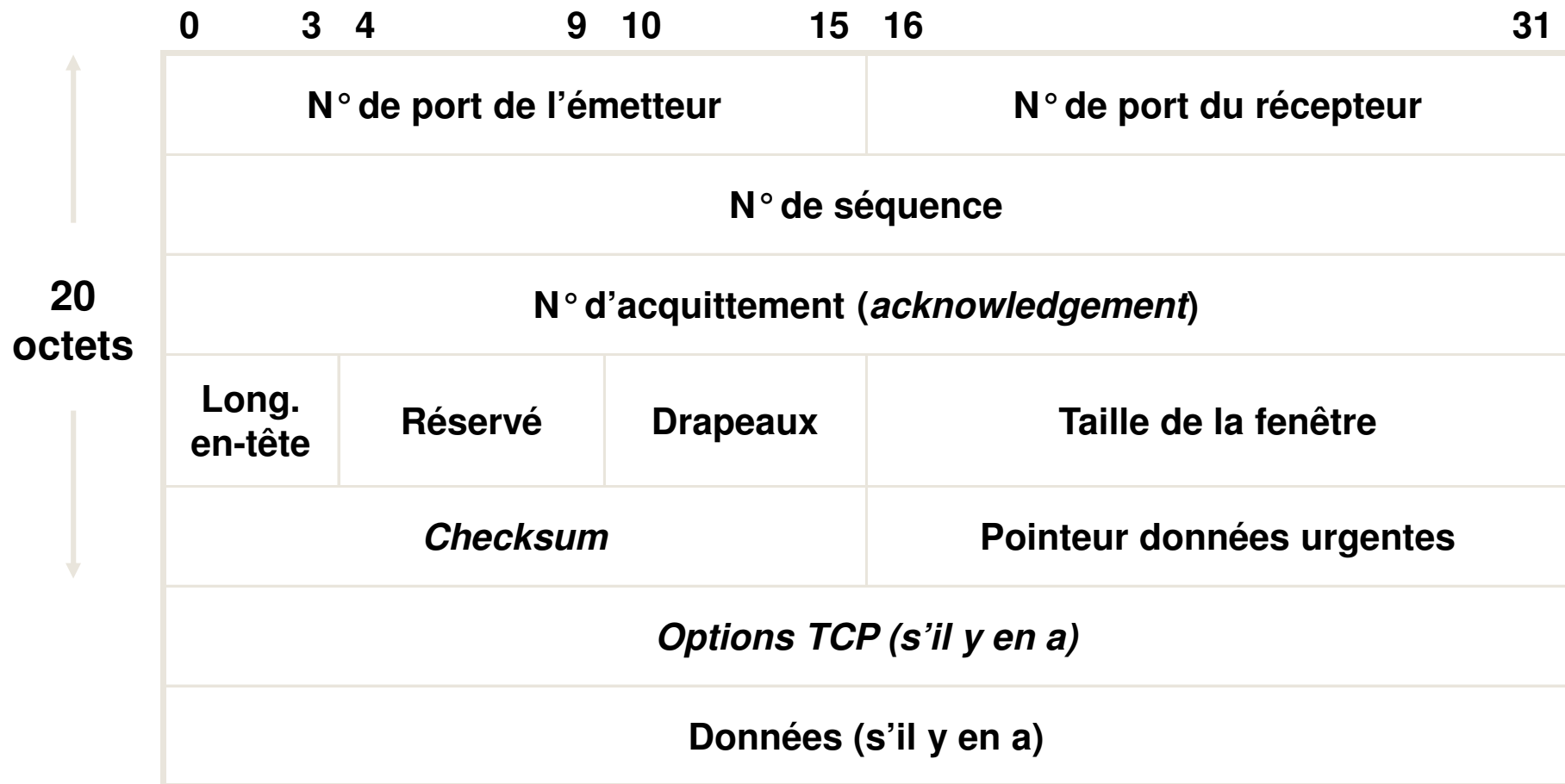
- Avant de pouvoir envoyer des données, il faut établir une connexion
  - Mécanisme de signalisation : *three-way handshake*
    - Analogie : appel téléphonique
- Phases typiques d'une connexion TCP
  - Etablissement (en 3 étapes SYN, SYN-ACK, ACK)
  - Échange de données
  - Fermeture

# Segments TCP et paquets IP

- Unité d'information (PDU) = segment
  - Les données de l'application sont coupées en blocs, transportés en segments TCP
  - Chaque segment TCP est « encapsulé » dans un paquet IP



# En-tête d'un segment TCP



# Drapeaux de l'en-tête TCP

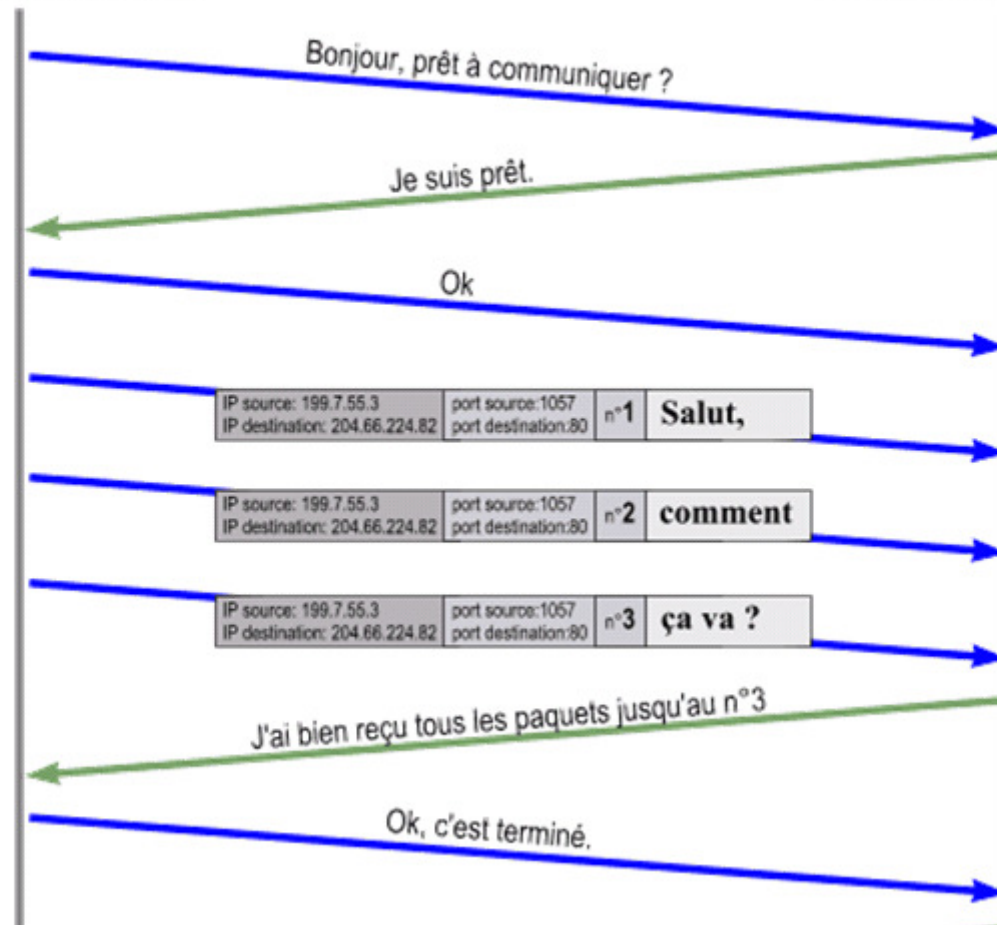
- Utilisés pour la signalisation

<b>SYN</b>	<b>Synchroniser numéros de séquence aux deux extrémités de la connexion</b>
<b>FIN</b>	<b>Le TCP émetteur du segment n'a plus de données à envoyer</b>
<b>ACK</b>	<b>Le champ « Numéro d'acquittement » contient une valeur valable</b>
<b>RST</b>	<b>« Réinitialiser » (interrompre) la connexion</b>
<b>PSH</b>	<b>Le TCP récepteur du segment doit passer les données à l'application le plus rapidement possible</b>
<b>URG</b>	<b>Le champ « Pointeur données urgentes » est valable</b>

# Etablissement de connexion en 3 phases : SYN, SYN-ACK, ACK

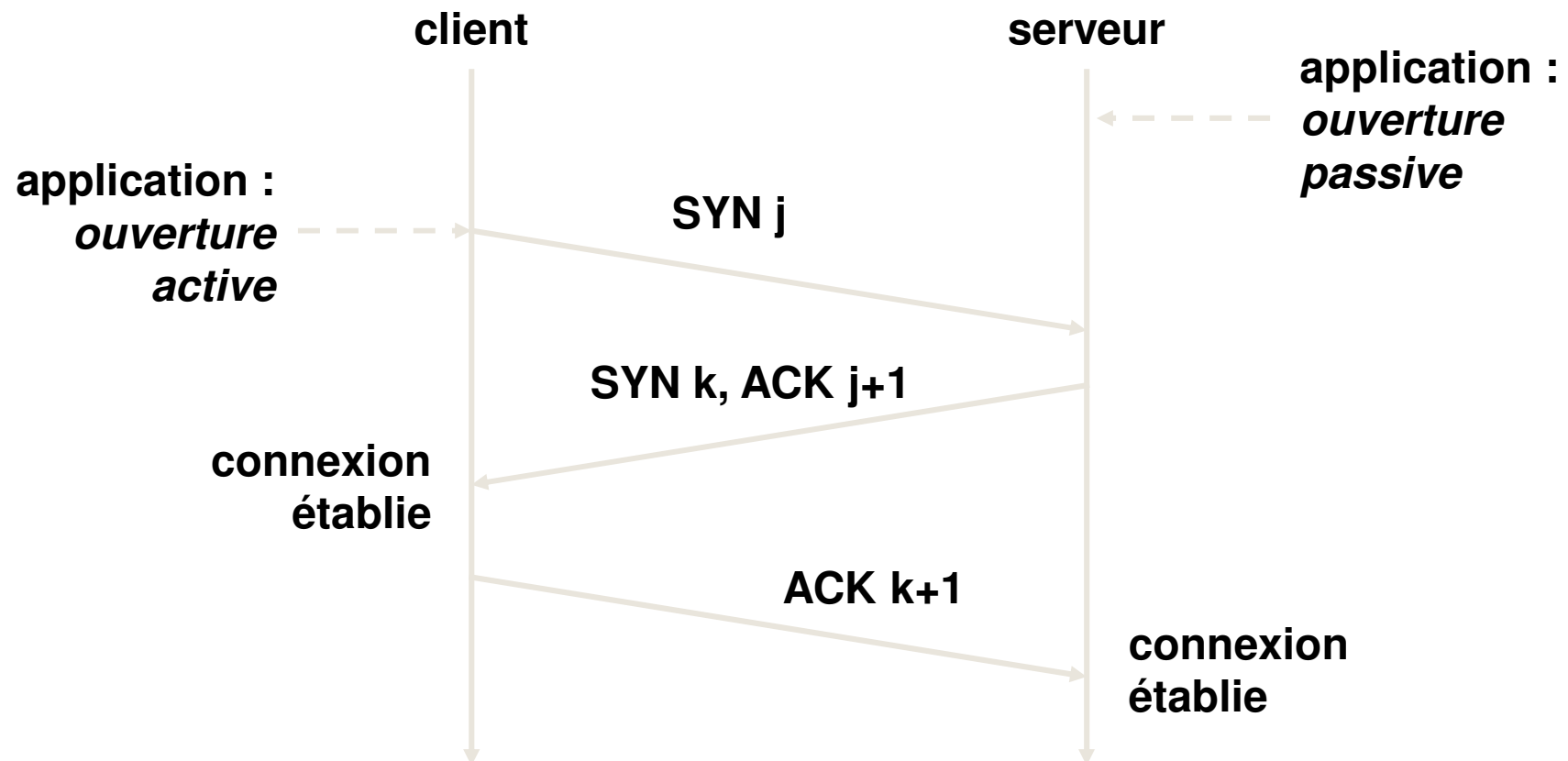
ordinateur 199.7.55.3

ordinateur 204.66.224.82



<http://sebsauvage.net>

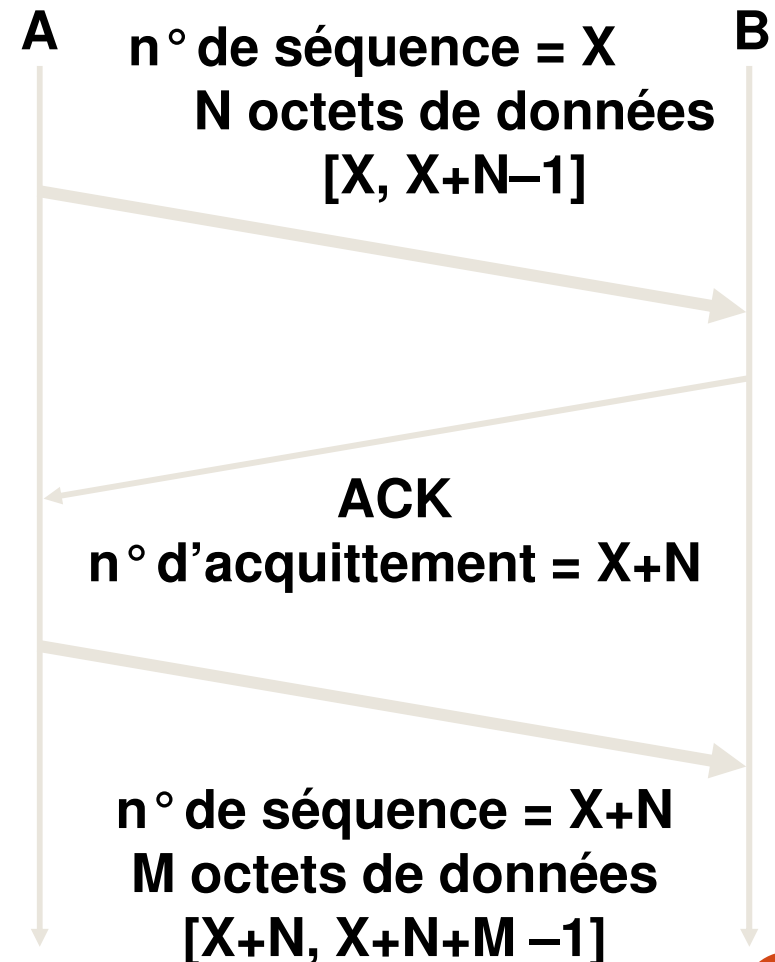
# Ouverture de la connexion



Trois segments pour ouvrir la connexion: *three-way handshake*

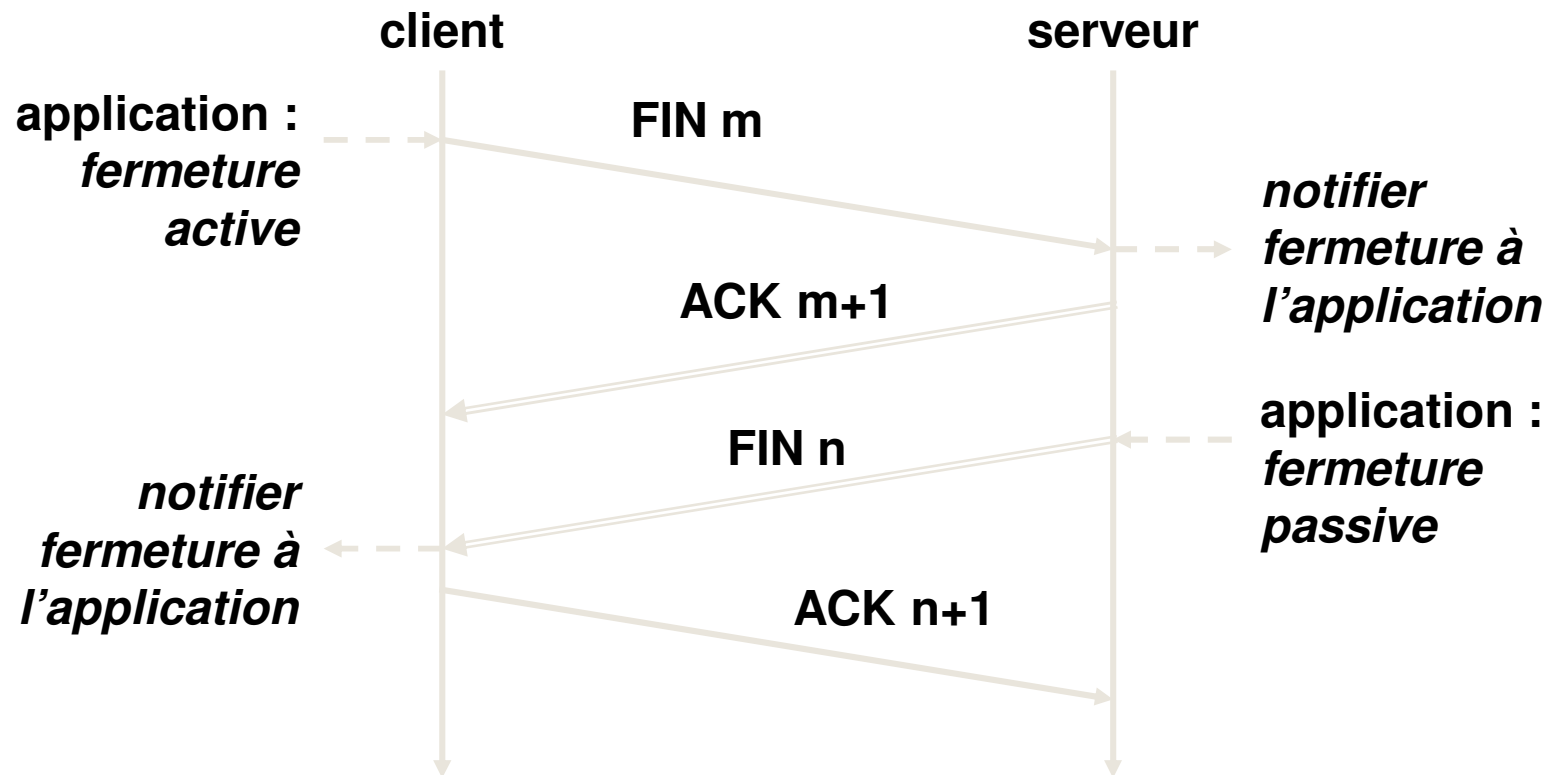
# N° de séquence et d'acquittement

- N° de séquence =  
premier octet de données du  
segment
- N° d'acquittement =  
prochain octet que l'émetteur de  
l'ACK s'attend à recevoir
- Connexion *full-duplex*
  - Chaque extrémité de la connexion  
enregistre le n° de séquence pour  
chaque direction





# Fermeture de la connexion



**Quatre segments pour fermer la connexion  
(à cause de la « demi-fermeture »)**

# Mécanisme simple : *stop-and-wait*

## ■ Émetteur

1. Envoyer un paquet
2. Attendre confirmation
3. S'il n'y a pas de confirmation dans un temps T :
  - a. retransmettre
  - b. retour à (2)
4. Retour à (1)

## ■ Récepteur

1. Attendre un paquet
2. Envoyer confirmation
3. Retour à (1)

## ■ Avantages

- Transport fiable
- Difficile de saturer le récepteur...

## ■ Inconvénient

- Utilisation très inefficace de la bande passante

# Contrôle de flux dans TCP

- « Fenêtre glissante »
  - **Idée** : l'émetteur peut envoyer des données sans avoir reçu de confirmation de tout ce qu'il a déjà émis... tant que le récepteur peut absorber les nouvelles données !
- Piloté par le récepteur
  - Champs de l'en-tête TCP
    - *Taille de la fenêtre* (16 bits)
    - *N° de ACK* (32 bits)

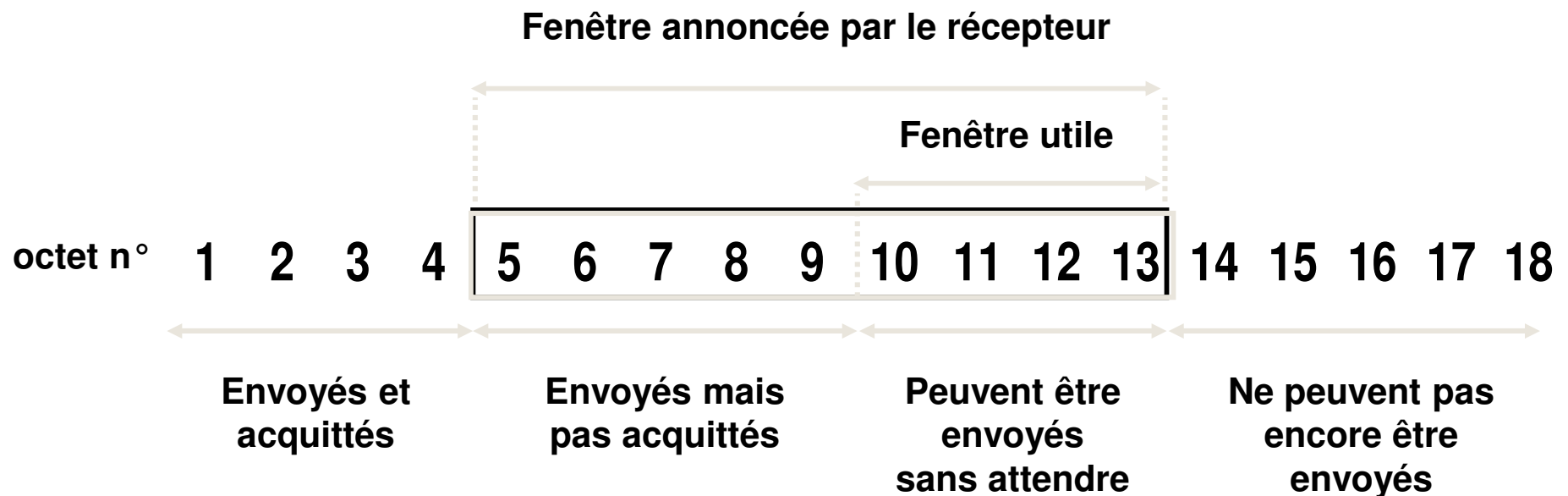
# Fenêtre glissante : évolution dans le temps

- Le récepteur confirme l'arrivée de données  $\Rightarrow$  la fenêtre se ferme
  - Champ *N° de ACK*
- Le récepteur lit des données (acquittées)  $\Rightarrow$  la fenêtre s'ouvre
  - Champ *Taille de la fenêtre*
- Fermeture + ouverture = la fenêtre se déplace « vers la droite » (glisse)

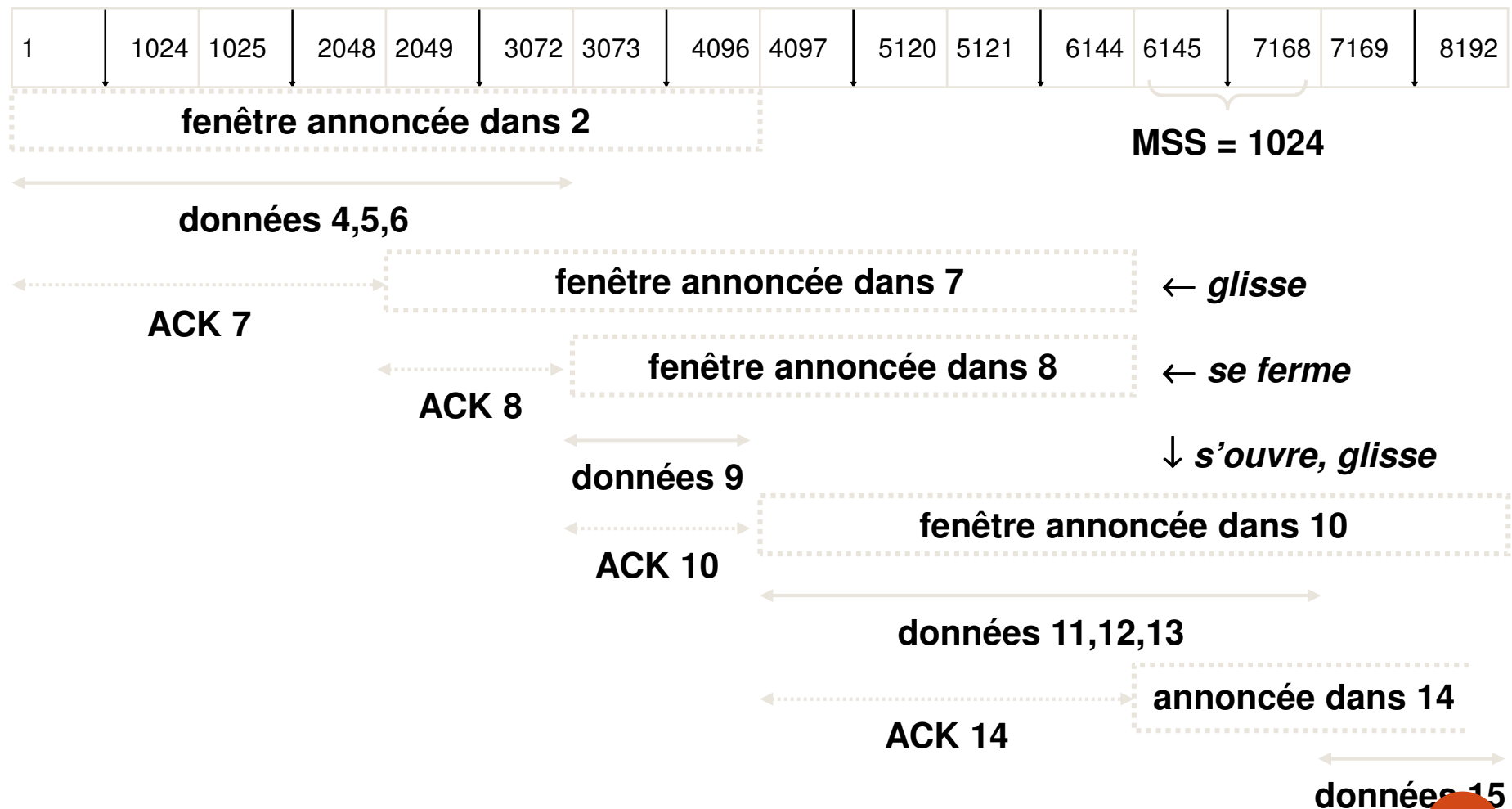


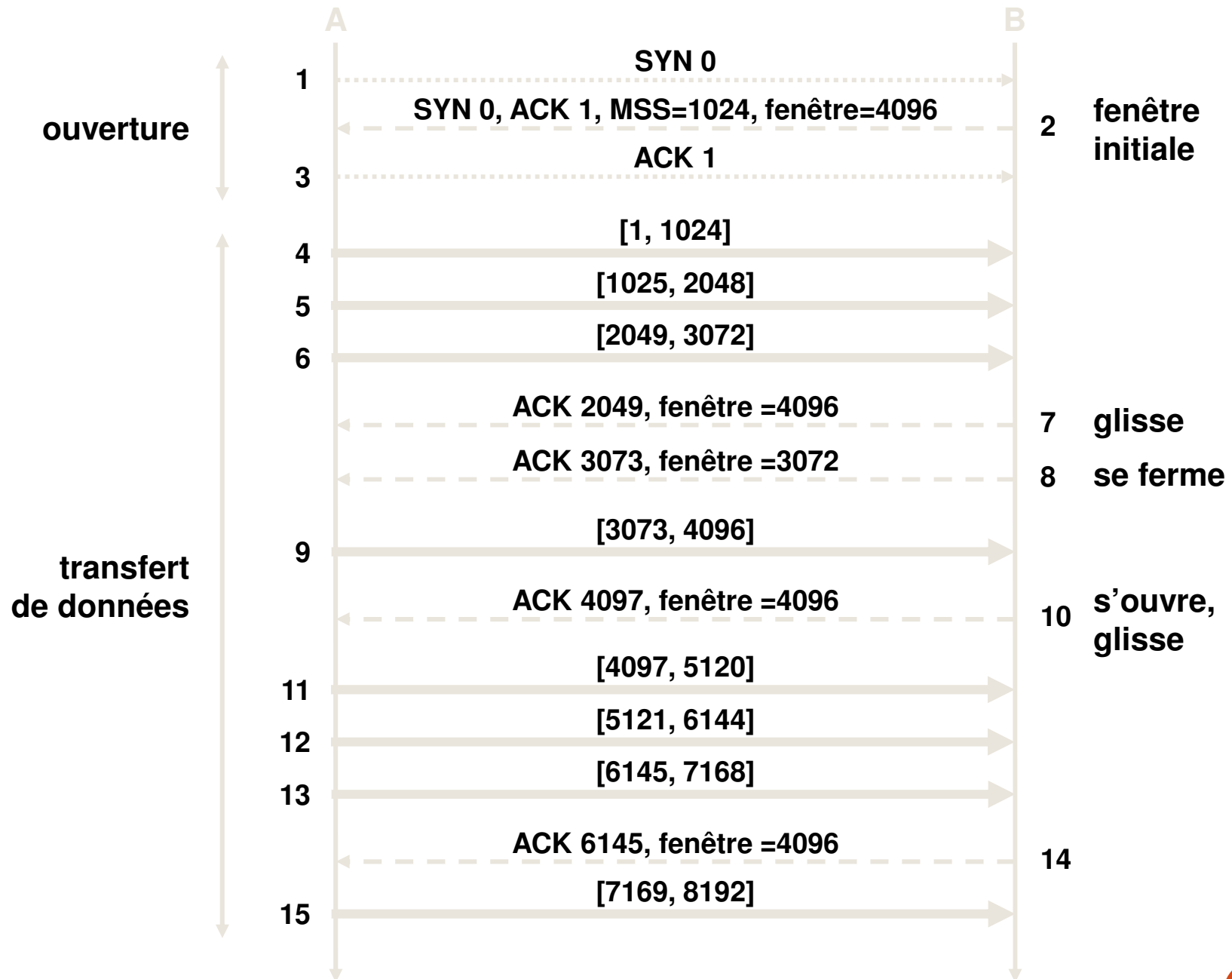
# Principe de la fenêtre glissante

- Chaque TCP annonce le nombre d'octets qu'il est disposé à recevoir, à compter du n° dans le champ ACK



# Fenêtre glissante : exemple (cf. schéma précédent)





# IX: Les ports



# Les ports

- Un port est un point d'entrée à un service sur un équipement (pc, serveur,...) connecté à un réseau.
- **A partir d'une seule adresse IP, vous pouvez faire plusieurs requêtes**

# L'utilité des ports

- Programmes TCP/IP exécutés simultanément sur Internet.
- Chacun de ces programmes travaille avec un protocole, toutefois l'ordinateur doit pouvoir distinguer les différentes sources de données.
- Chacune de ces applications se voit attribuer une adresse unique sur la machine, codée sur 16 bits : **un port** (la combinaison *adresse IP + port* est alors une adresse unique au monde, elle est appelée socket).
  - **L'adresse IP** identifie de façon unique un ordinateur sur le réseau
  - **Le numéro de port** indique l'application à laquelle les données sont destinées.
- Lorsque l'ordinateur reçoit des informations destinées à un port, les données sont envoyées vers l'application correspondante.

# L'utilité des ports

- Il y a 65536 ports (16 bits) :
  - **De 0 à 1023 : ports reconnus** ou réservés («**Well Known Ports**»). De manière générale, réservés aux processus système (démons) ou aux programmes exécutés par des utilisateurs privilégiés.
  - **De 1024 à 49151 : ports enregistrés** («**Registered Ports**»).
  - **De 49152 à 65535 : ports dynamiques et/ou privés**».
- Grâce au port, combiné à une adresse IP, on détermine de façon unique une application qui tourne sur une machine donnée.

# Les ports reconnus

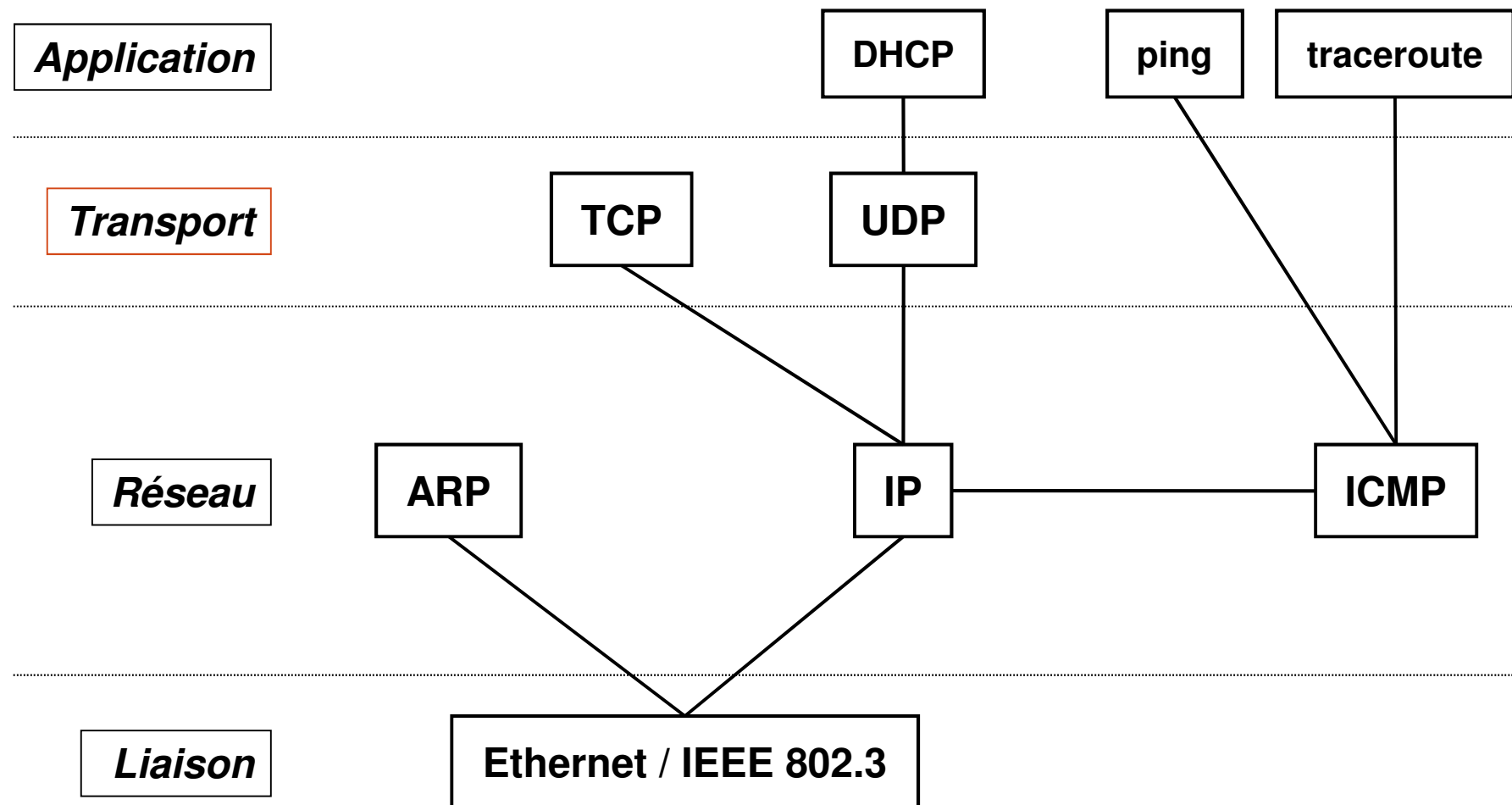
- **21** : FTP
- **22** : SSH
- **23** : Telnet
- **25** : SMTP
- **53** : Domain Name System
- **68** : DHCP
- **80** : HTTP
- **110** : POP3
- ...

# Ports client-serveur

- Un serveur possède des numéros de port fixes auxquels l'administrateur réseau a associé des services.
- Du côté du client, le port est choisi aléatoirement parmi ceux disponibles par le système d'exploitation. Ainsi, les ports du client ne seront jamais compris entre 0 et 1023 (qui sont les *ports reconnus*).

# X – Protocoles applicatifs et d'administration réseaux

# Couche 3 : architecture simplifiée



# XI - Applications réseaux

DHCP, ftp, http, telnet ssh....



# DHCP : Attribution d'adresses IP

- Pour l'attribution dynamique des adresses IP, on utilise des protocoles qui attribuent des IP aux hôtes :
  - **BOOTP** : Ce protocole permet à un équipement de récupérer son adresse IP au démarrage. L'émetteur envoie un message de broadcast (255.255.255.255) reçu par le serveur qui répond lui aussi par un broadcast contenant l'adresse MAC de l'émetteur ainsi qu'une IP.
  - **DHCP** : Remplaçant de BOOTP, il permet l'obtention dynamique d'IP. Lorsqu'un ordinateur entre en ligne, il communique avec le serveur qui choisit une adresse et l'attribue à l'hôte. Avec le protocole DHCP, il est également possible pour un ordinateur de récupérer sa configuration complète (adresse, masque de sous réseau, etc.)

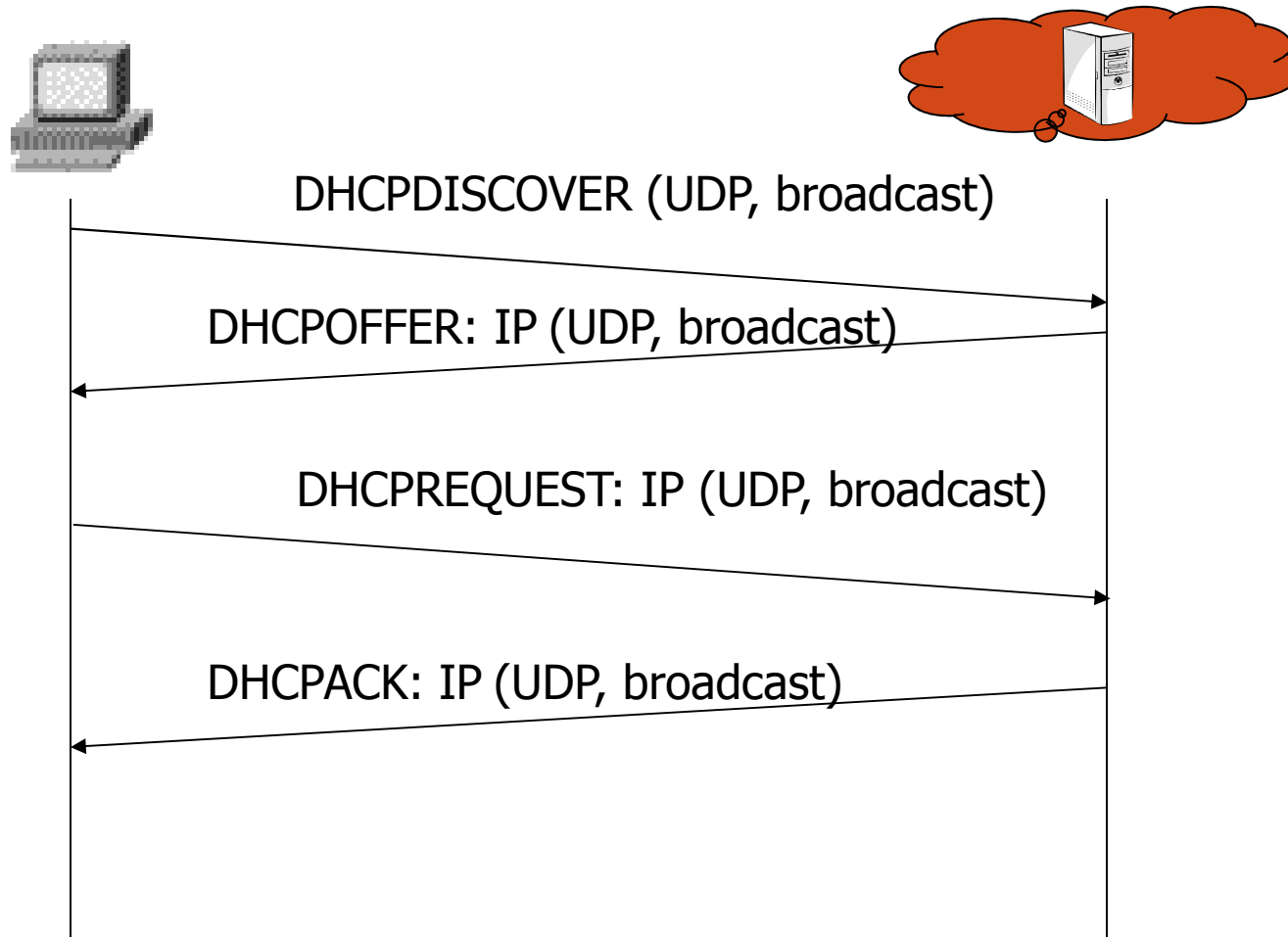
# Attribution d'adresses IP

- On distingue 2 méthodes d'attribution des adresses IP pour les hôtes :
  - **Statique** : chaque équipement est configuré avec une adresse unique
  - **Dynamique** : on utilise des protocoles qui attribue dynamiquement les adresses IP dès la connexion à partir d'un pool d'adresses.

# DHCP (*Dynamic Host Configuration Protocol*)

- Pour connecter une machine à l'Internet, DHCP configure dynamiquement :
  1. Une adresse IP unique dans le réseau local et appartenant au même réseau logique que toutes les autres machines du réseau,
  2. L'adresse IP d'une passerelle qui permet d'accéder à l'extérieur,
  3. L'adresse IP d'un serveur DNS pour pouvoir résoudre les noms des hôtes,
  4. un masque de sous réseau, le même pour tous les hôtes du réseau.

# Modèle de fonctionnement

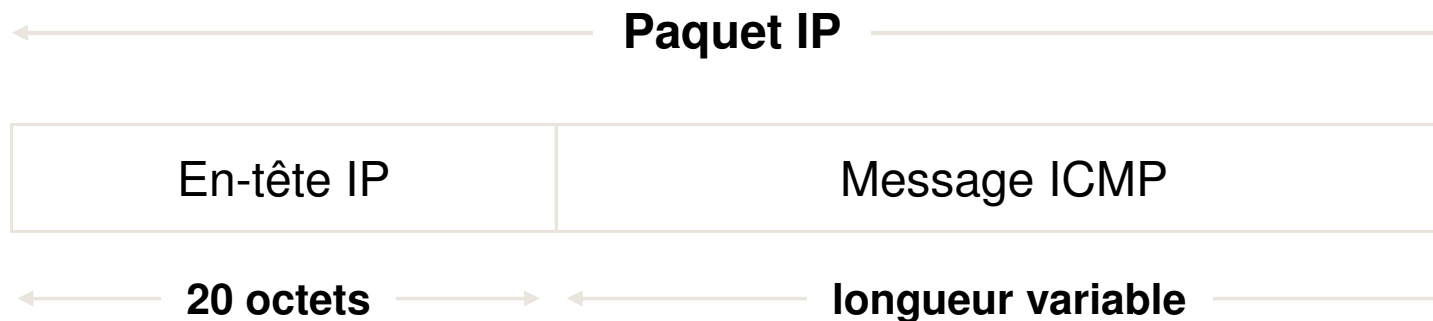


# Messages DHCP










- DHCPDISCOVER : localiser les serveurs DHCP disponibles
- DHCPOFFER : réponse du serveur à un paquet DHCPDISCOVER, contenant les premiers paramètres
- DHCPREQUEST : requête diverse du client pour, par exemple, prolonger son bail
- DHCPACK : réponse du serveur qui contient des paramètres et l'adresse IP du client
- DHCPNAK : réponse du serveur pour signaler au client que son bail est échu ou si le client annonce une mauvaise configuration réseau
- DHCPDECLINE : le client annonce au serveur que l'adresse est déjà utilisée
- DHCPRELEASE : le client libère son adresse IP
- DHCPINFORM : le client demande des paramètres locaux

# Protocole ICMP

- *Internet Control Message Protocol* (RFC 792)
- But : échange de messages d'erreur et de demande d'information
  - Traités soit par IP, soit par une couche supérieure
- Niveau 3, mais encapsulé dans des paquets IP
  - Champ *Protocole* = 1



# Quelques types de messages ICMP

Type	Code	Description	Demande	Erreur
0	0	Réponse à une demande d'écho		
3		Destination non accessible :		
	0	Réseau inaccessible		
	1	Station inaccessible		
	2	Protocole inaccessible		
	3	Fragmentation nécessaire mais bit DF = 1		
	4	Port inaccessible		
		etc.		
8	0	Demande d'écho		
11		La durée de vie (TTL) a atteint 0 :		
	0	Durant le transit		
	1	Durant le réassemblage		

# Commande *ping*

- Basée sur les messages ICMP de type 8 (*echo request*) et 0 (*echo reply*)
  - Réception d'un message type 8  $\Rightarrow$  émission d'un message type 0
- Format des messages

0	7	8	15	16	23	24	31
Type (0 ou 8)		0		Checksum			
Identificateur				Numéro de séquence			
(données optionnelles)							

**La réponse contient une copie des champs *Identificateur*,  
*N° de séquence* et les données optionnelles**



# ping

```
Z:\>ping www.google.fr
```

```
Envoi d'une requête 'ping' sur www.l.google.com [209.85.135.103] avec 32 octets de données :
```

```
Réponse de 209.85.135.103 : octets=32 temps=31 ms TTL=241  
Réponse de 209.85.135.103 : octets=32 temps=32 ms TTL=241  
Réponse de 209.85.135.103 : octets=32 temps=32 ms TTL=241  
Réponse de 209.85.135.103 : octets=32 temps=32 ms TTL=241
```

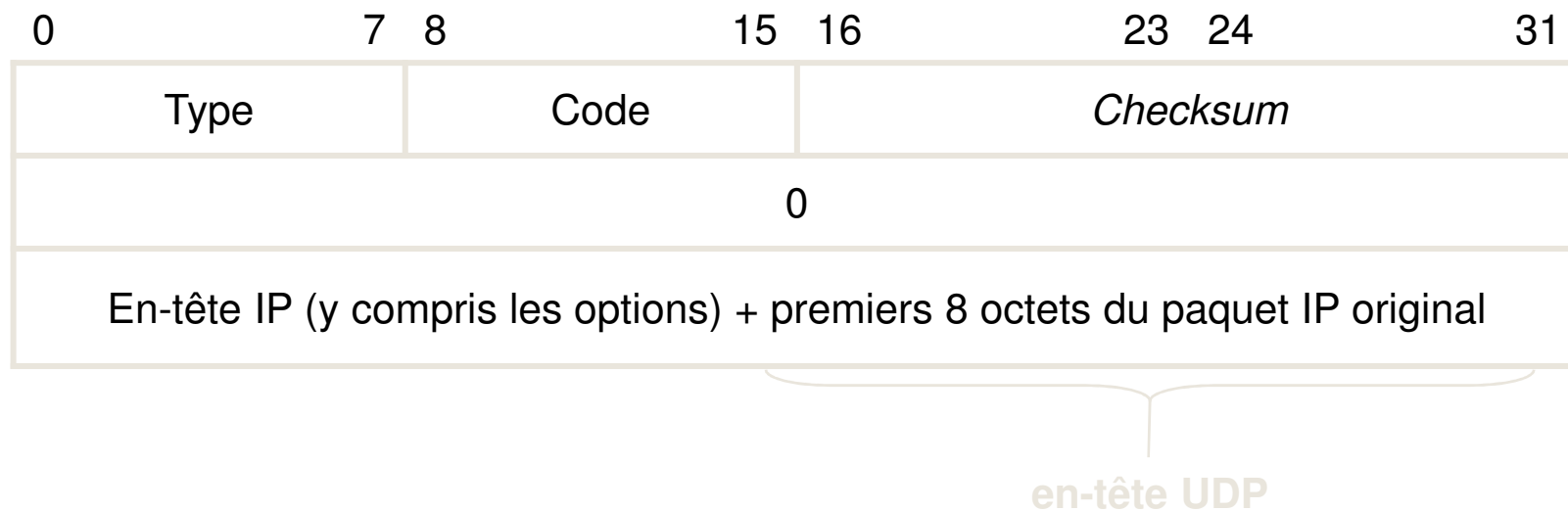
```
Statistiques Ping pour 209.85.135.103:
```

```
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),  
Durée approximative des boucles en millisecondes :  
    Minimum = 31ms, Maximum = 32ms, Moyenne = 31ms
```

```
Z:\>
```

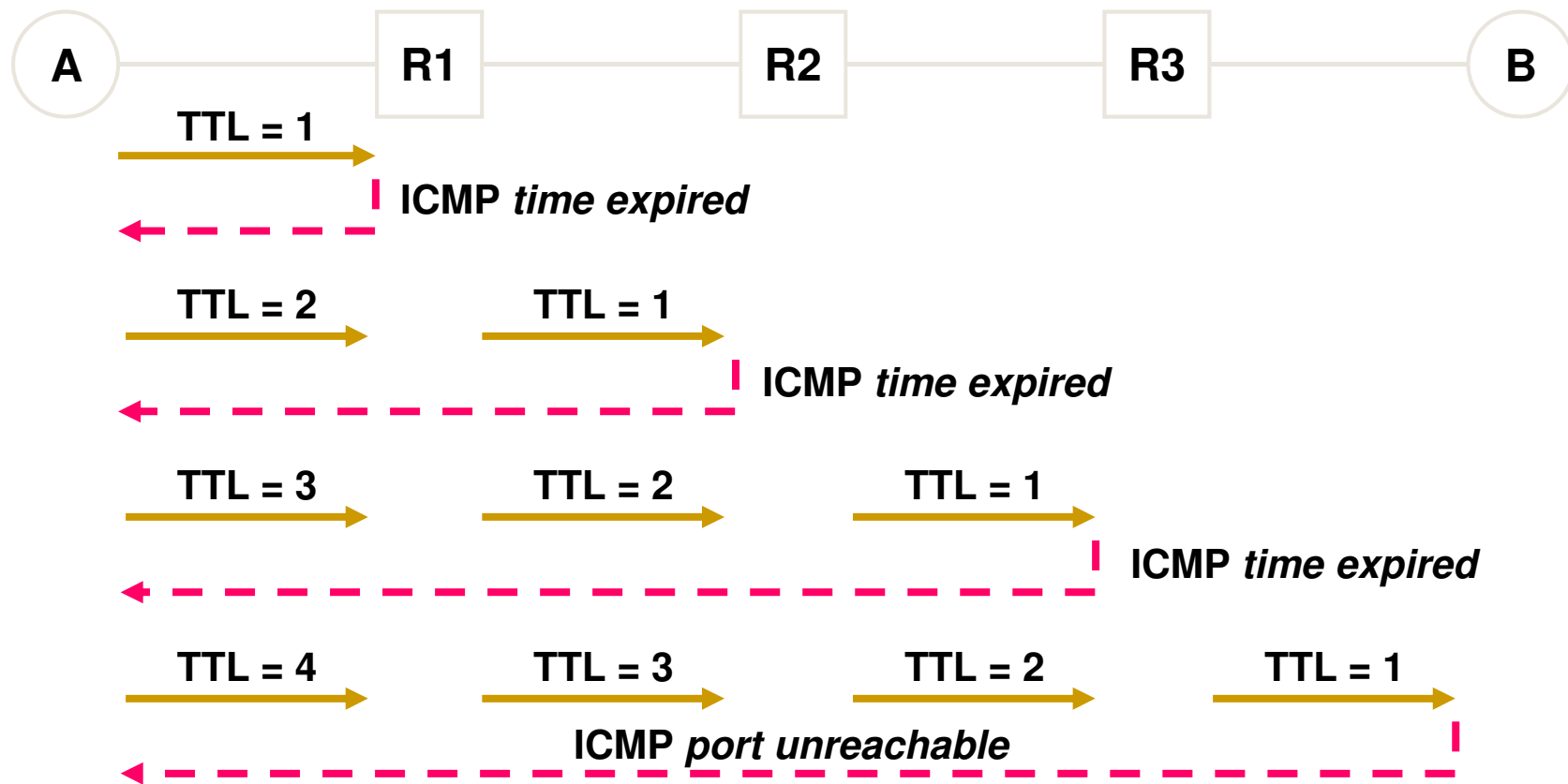
# tracert

- Basé sur les messages ICMP de type 11 / code 0 (*time exceeded*) et type 3 / code 4 (*port unreachable*)
  - Envoi de datagrammes UDP qui déclenchent ces messages d'erreur ICMP
- Format des messages



# *tracert* : fonctionnement

Sens A → B : datagrammes UDP (avec *N° port destination* ≥ 33434)



# tracert

```
Z:\>tracert www.google.fr

Détermination de l'itinéraire vers www.l.google.com [209.85.135.103]
avec un maximum de 30 sauts :

  1      1 ms      <1 ms      <1 ms      sophia-p32.inria.fr [138.96.32.250]
  2      <1 ms      <1 ms      <1 ms      nice-g3-0-60.cssi.renater.fr [193.51.181.138]
  3      15 ms      15 ms      15 ms      marseille-pos4-0.cssi.renater.fr [193.51.179.245]
1]
  4      15 ms      15 ms      15 ms      montpellier-pos2-0.cssi.renater.fr [193.51.179.2
42]
  5      15 ms      15 ms      15 ms      lyon-pos14-0.cssi.renater.fr [193.51.179.221]
  6      14 ms      14 ms      14 ms      nri-b-pos9-0.cssi.renater.fr [193.51.179.13]
  7      15 ms      15 ms      15 ms      TELEGLOBE-FRANCE-INTERNATIONAL.sfinx.tm.fr [194.
68.129.242]
  8      15 ms      15 ms      15 ms      if-6-0-7.core1.PU1-Paris.teleglobe.net [80.231.7
9.14]
  9      24 ms      24 ms      24 ms      if-2-0-0.core2.FR1-Frankfurt.teleglobe.net [80.2
31.65.65]
10     32 ms      35 ms      35 ms      12.icore1.FR1-Frankfurt.teleglobe.net [80.231.65
.6]
11     24 ms      25 ms      25 ms      195.219.180.30
12     25 ms      49 ms      25 ms      209.85.249.178
13     31 ms      31 ms      31 ms      209.85.248.248
14     31 ms      36 ms      31 ms      72.14.239.46
15     38 ms      35 ms      33 ms      72.14.239.58
16     32 ms      31 ms      32 ms      mu-in-f103.google.com [209.85.135.103]

Itinéraire déterminé.
Z:\>_
```

Z:\>ipconfig /all

Configuration IP de Windows

Nom de l'hôte . . . . . : BOUGEOTTE  
Suffixe DNS principal . . . . . : inria.fr  
Type de nœud . . . . . : Inconnu  
Routage IP activé . . . . . : Non  
Proxy WINS activé . . . . . : Non  
Liste de recherche du suffixe DNS : inria.fr  
inria.fr  
inria.fr

Carte Ethernet Connexion au réseau local 2:

Suffixe DNS propre à la connexion : inria.fr  
Description . . . . . : Broadcom NetXtreme 57xx Gigabit Controller #5  
Adresse physique . . . . . : 00-1C-23-11-2B-5C  
DHCP activé . . . . . : Oui  
Configuration automatique activée . . . . . : Oui  
Adresse IP . . . . . : 138.96.241.88  
Masque de sous-réseau . . . . . : 255.255.0.0  
Passerelle par défaut . . . . . : 138.96.112.250  
Serveur DHCP . . . . . : 138.96.0.23  
Serveurs DNS . . . . . : 138.96.0.11  
138.96.0.10  
Bail obtenu . . . . . : mardi 4 décembre 2007 16:23:53  
Bail expirant . . . . . : mercredi 5 décembre 2007 04:23:53

Carte Ethernet Connexion réseau sans fil:

Suffixe DNS propre à la connexion : inria.fr  
Description . . . . . : Intel(R) PRO/Wireless 3945ABG Network Connection #2  
Adresse physique . . . . . : 00-1B-77-C8-1C-FE  
DHCP activé . . . . . : Oui  
Configuration automatique activée . . . . . : Oui  
Adresse IP . . . . . : 193.51.208.195  
Masque de sous-réseau . . . . . : 255.255.255.0  
Passerelle par défaut . . . . . : 193.51.208.129  
Serveur DHCP . . . . . : 138.96.0.26  
Serveurs DNS . . . . . : 138.96.0.10  
138.96.0.11  
Bail obtenu . . . . . : mardi 4 décembre 2007 18:18:47  
Bail expirant . . . . . : mardi 4 décembre 2007 18:30:22

Carte Ethernet Connexion au réseau local:

Statut du média . . . . . : Média déconnecté  
Description . . . . . : Bluetooth Personal Area Network  
Adresse physique . . . . . : 00-1A-6B-C4-7D-9C

# Nslookup – interrogation du serveur de noms

```
Z:\>nslookup
Serveur par défaut : dns2-sop.inria.fr
Address: 138.96.0.11

> www.google.fr
Serveur : dns2-sop.inria.fr
Address: 138.96.0.11

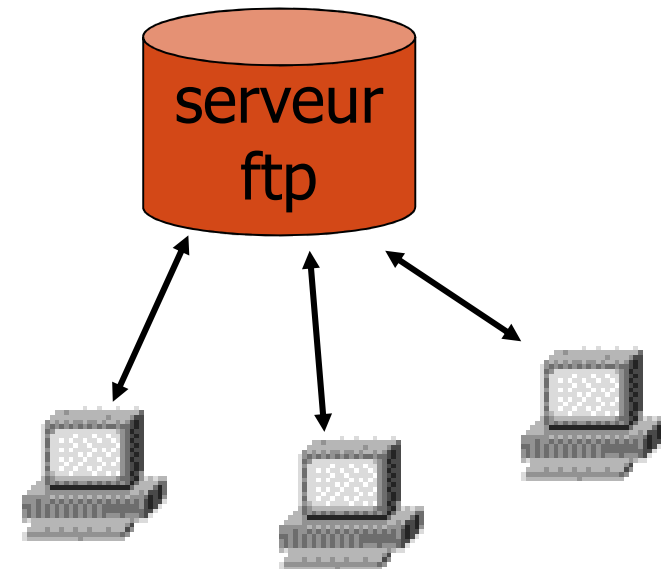
Réponse ne faisant pas autorité :
Nom : www.l.google.com
Addresses: 209.85.137.147, 209.85.137.99, 209.85.137.104
Aliases: www.google.fr, www.google.com

> 134.59.20.121
Serveur : dns2-sop.inria.fr
Address: 138.96.0.11

Nom : iut-sg-web.unice.fr
Address: 134.59.20.121
```

# FTP (*File Transfer Protocol*)

- Permet le transfert de fichiers d'une machine (serveur) vers une autre (client).
- Utilité :
  - stockage de fichiers
- Utilisation
  - directement par l'utilisateur
  - par d'autres applications

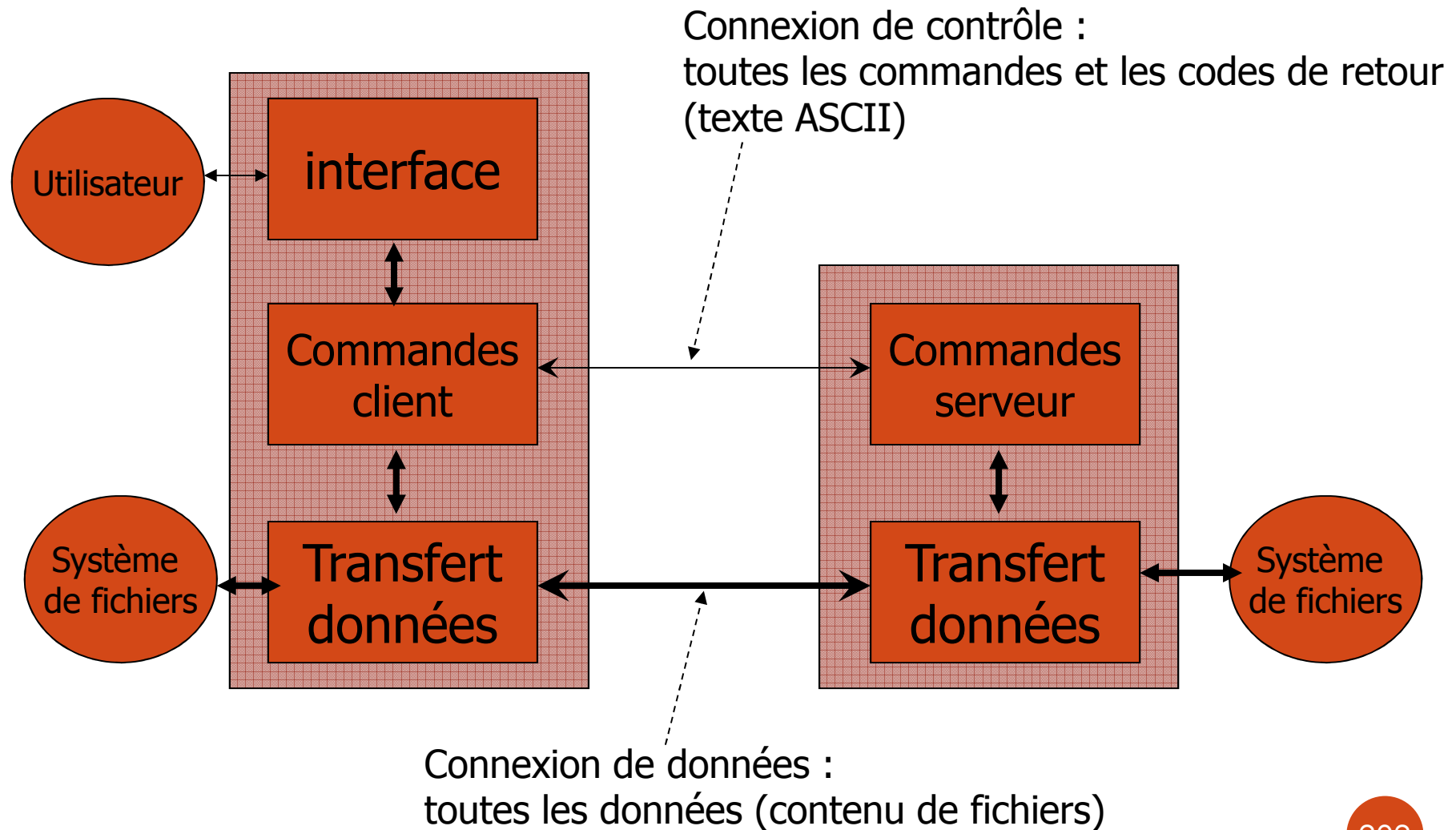


# Commandes

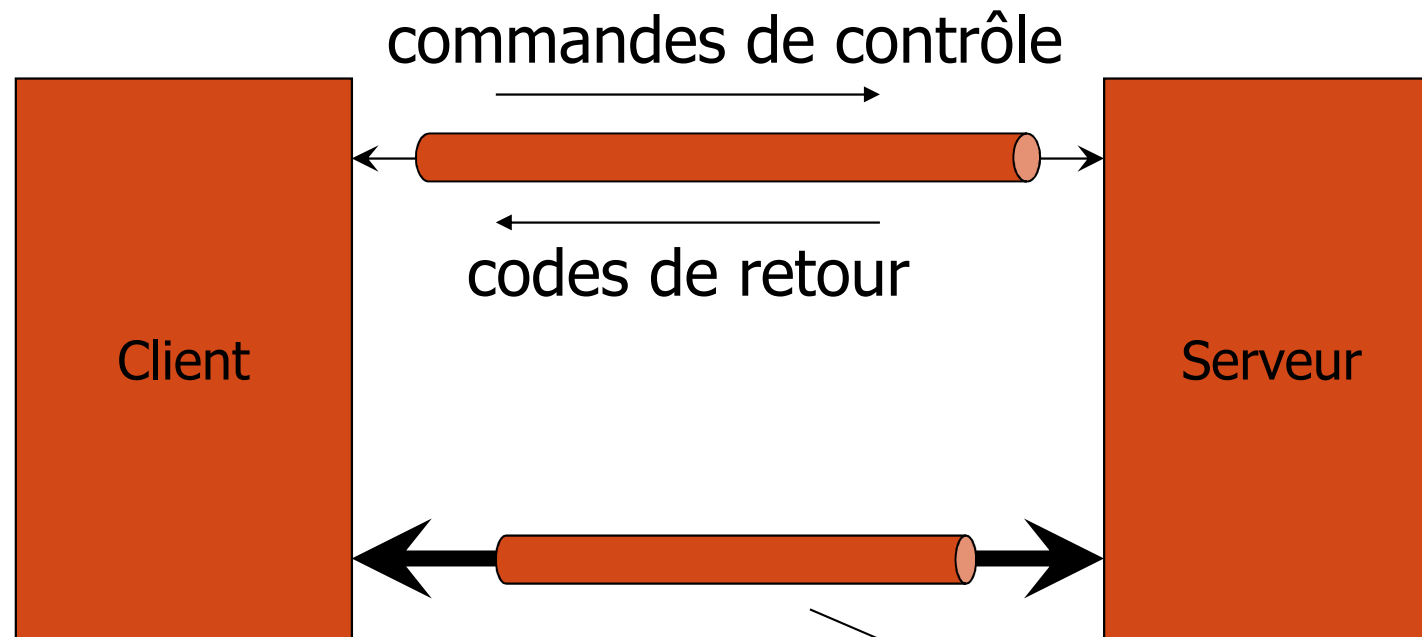
- get, mget
- put, mput
- lcd, cd
- bin, ascii
- quit, bye
- ls
- help
- ...



# Modèle de fonctionnement



# Modèle de fonctionnement



## Commandes de contrôle

Access : USER, PASS, CWD, QUIT

Transfert : PORT, PASSV, MODE

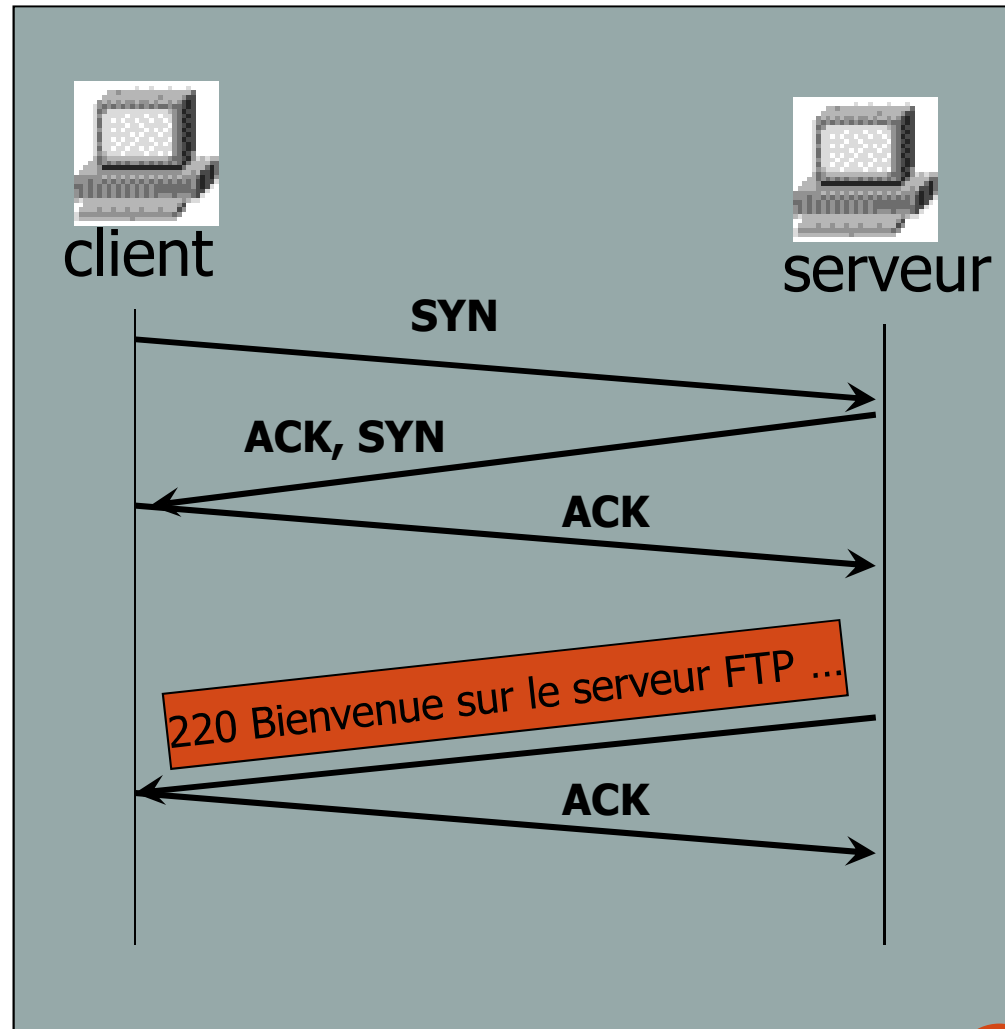
Service : STOR, RETR, LIST

Chaque transfert :  
nouvelle connexion TCP

# Exemple d'utilisation

## Interface utilisateur

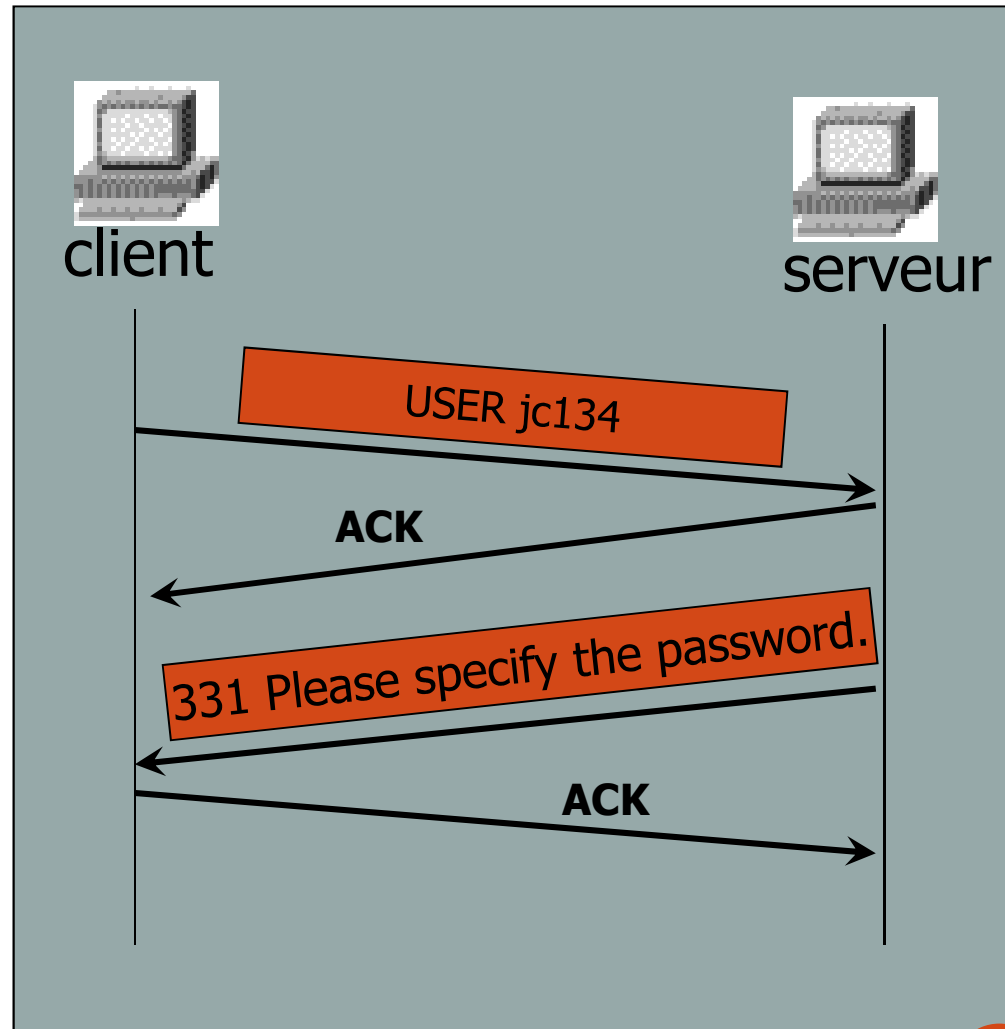
```
ftp  
>open servername
```



# Exemple d'utilisation

## Interface utilisateur

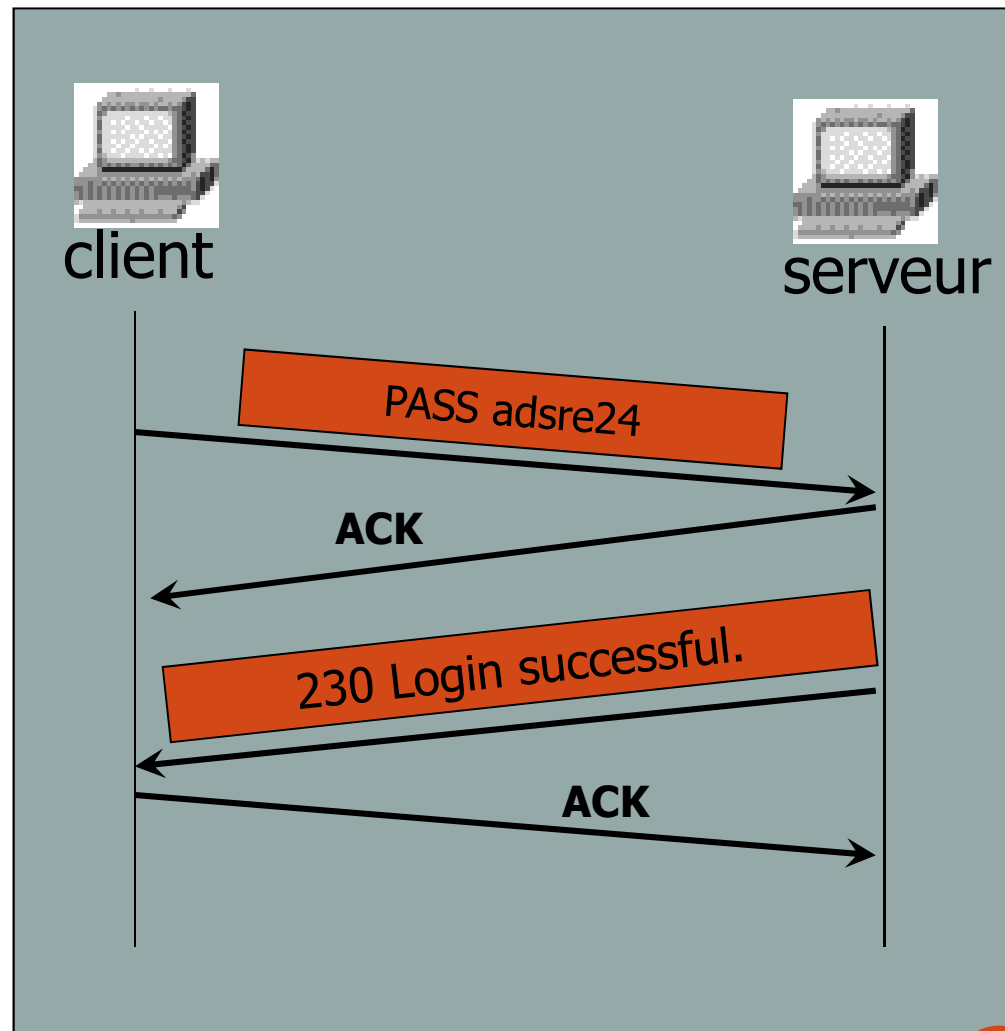
```
ftp  
>open servername  
Bienvenue sur le  
serveur FTP ...  
>user jc1234
```



# Exemple d'utilisation

## Interface utilisateur

```
ftp
>open servername
Bienvenue sur le
serveur FTP ...
>user jc1234
Password: adsre24
Login successful.
>
```

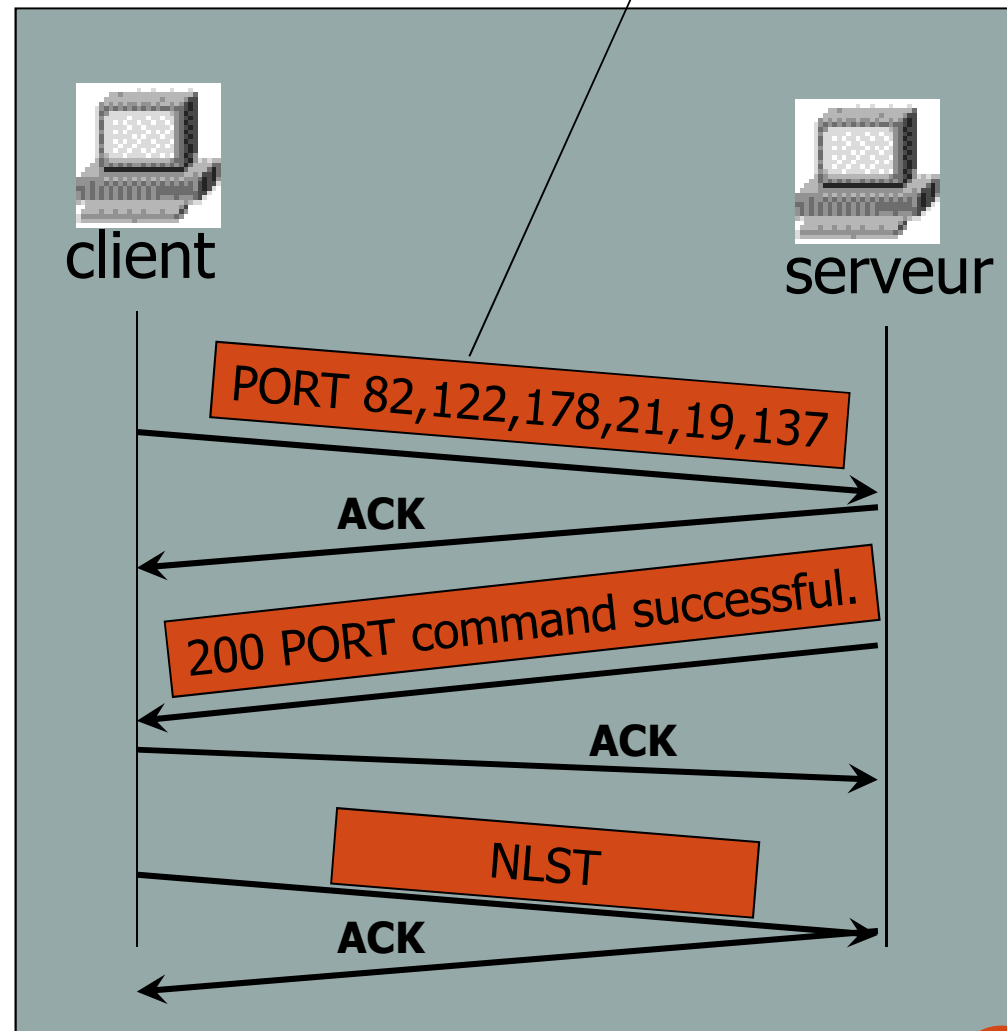


# Exemple d'utilisation

Le client 82.122.178.21 écoute sur  
le port  $19 \times 256 + 137 = 5001$

## Interface utilisateur

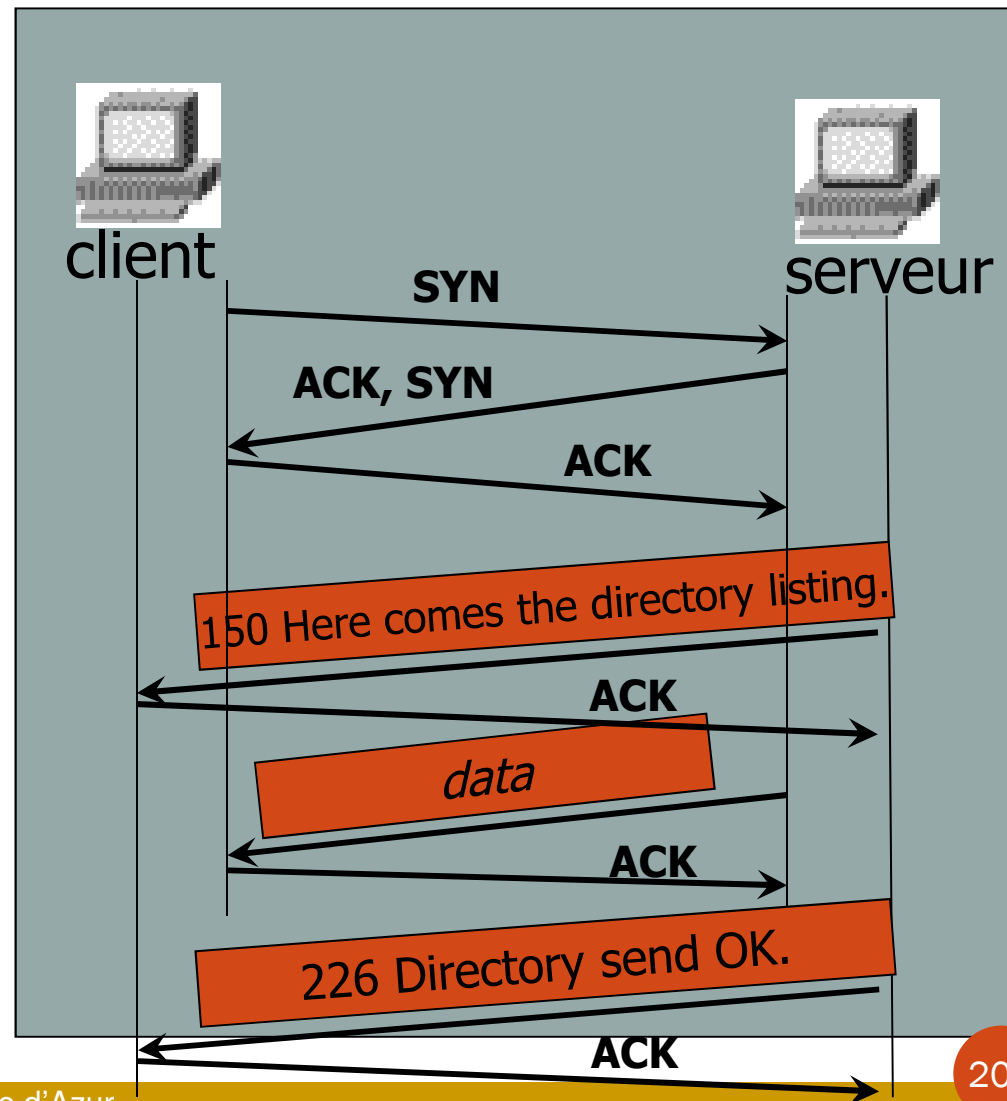
```
ftp
>open servername
Bienvenue sur le
serveur FTP ...
>user jc1234
Password: adsre24
Login successful.
>ls
```



# Exemple d'utilisation

Interface utilisateur

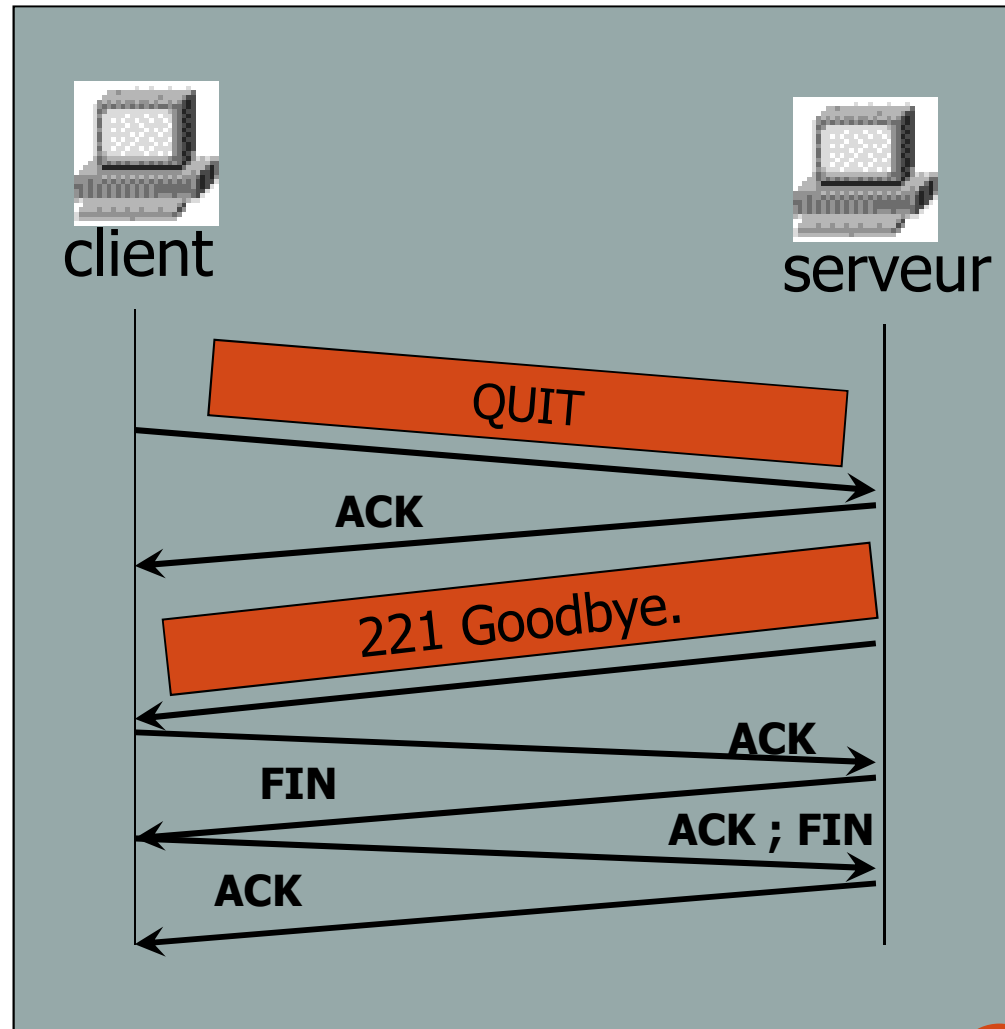
```
...  
>ls
```



# Exemple d'utilisation

Interface utilisateur

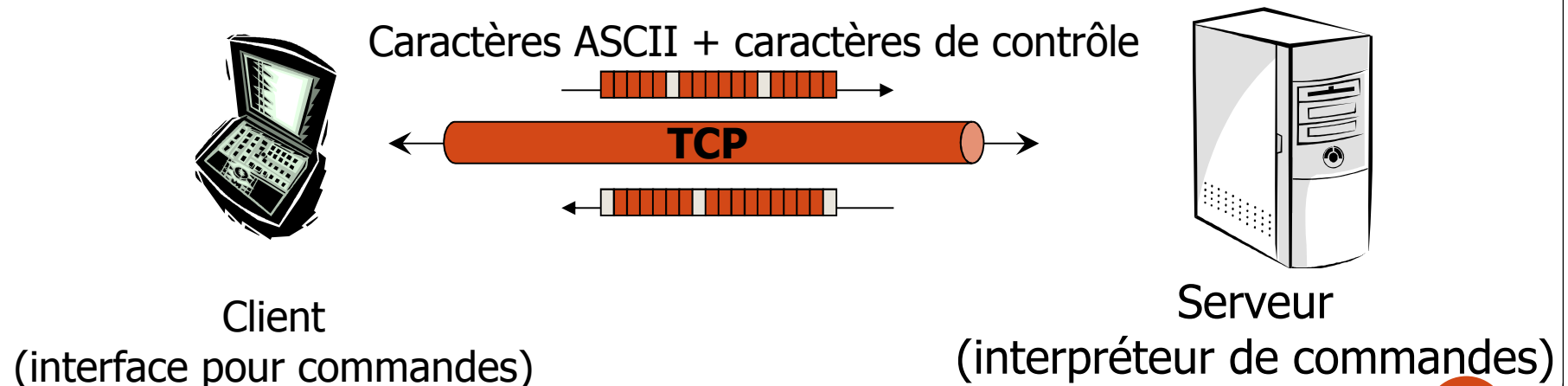
...  
>bye





# TELNET

- TELNET = protocole réseau
- telnet = logiciel qui utilise le protocole TELNET
- TELNET
  - permet de connecter un client (système composé d'un affichage et d'un clavier) à un serveur (interpréteur de commandes)



# TELNET

- Port 23
- Données et contrôle sur la même connexion TCP
- Network Virtual Terminal (NVT)
  - Représentation virtuelle d'un terminal générique (clavier standard, taille d'écran standard, etc.)
- Négociation des options entre le client et le serveur (jeu des caractères, mode ligne ou caractère, etc.)

# Commandes TELNET

- Le caractère avec le code 255 signale une commande (IAC, Interpret As Command)
- Commandes :
  - 247 (EL, Erase Line)
  - 246 (EC, Erase Character)
  - 243 (IP, Interrupt Process)
  - ...

# Exemple connexion telnet

```
Z:\users>telnet if-4433.insa-lyon.fr
```

```
Login: SP1321
```

```
Password: *****
```

```
*=====
```

```
Bienvenue à Microsoft Telnet Server.
```

```
*=====
```

```
C:\>netstat
```

```
Connexions actives
```

Proto	Adresse locale	Adresse distante	Etat
TCP	if-4433:telnet	localhost:4342	TIME_WAIT
TCP	if-4433:telnet	localhost:4352	ESTABLISHED
TCP	if-4433:4352	localhost:telnet	ESTABLISHED
TCP	if-4433:4143	servif-baie.insa-lyon.fr:microsoft-ds	ESTABLISH
ED			
TCP	if-4433:4145	cs27.msg.dcn.yahoo.com:5050	ESTABLISHED
TCP	if-4433:4146	baym-cs17.msgr.hotmail.com:1863	ESTABLISHED
TCP	if-4433:4170	servif-impr.insa-lyon.fr:netbios-ssn	ESTABLISHE
D			
TCP	if-4433:4306	csiges9.insa-lyon.fr:993	ESTABLISHED

```
C:\>exit
```

```
Perte de la connexion à l'hôte.
```

```
Z:\users>
```

# Aspects concernant l'utilisation du protocole TELNET

- Beaucoup des serveurs TCP ne parlent pas TELNET
  - telnet peut être utilisé, mais pas de commandes de contrôle
  - Exemple :
    - telnet [www.wanadoo.fr](http://www.wanadoo.fr) 80
- Informations envoyées en clair
  - Nom d'utilisateur + mot de passe

# SSH

- Secure Shell
- Alternative à TELNET
- Protocole permettant une communication sécurisée entre le client et le serveur - les données sont cryptées
- Port 22

# HTTP (*HyperText Transfer Protocol*)

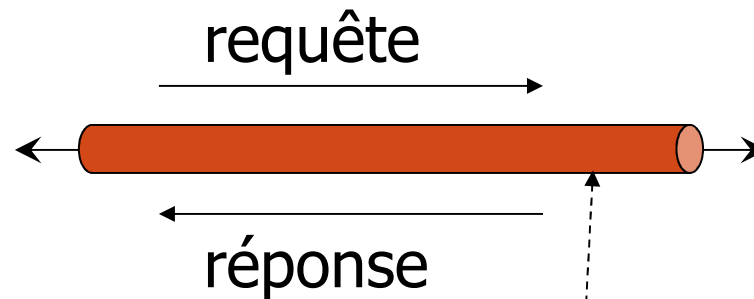
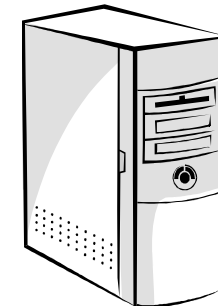
- Le protocole de communication standard pour le Web
- Port 80
- Deux versions utilisées :
  - http 1.0
  - http 1.1
- Références :
  - HTTP 1.0 : <http://www.faqs.org/rfcs/rfc1945.html>
  - HTTP 1.1 : <http://www.faqs.org/rfcs/rfc2616.html>

# Modèle de fonctionnement

Client HTTP  
(Navigateur Web)



Serveur HTTP  
(serveur Web)

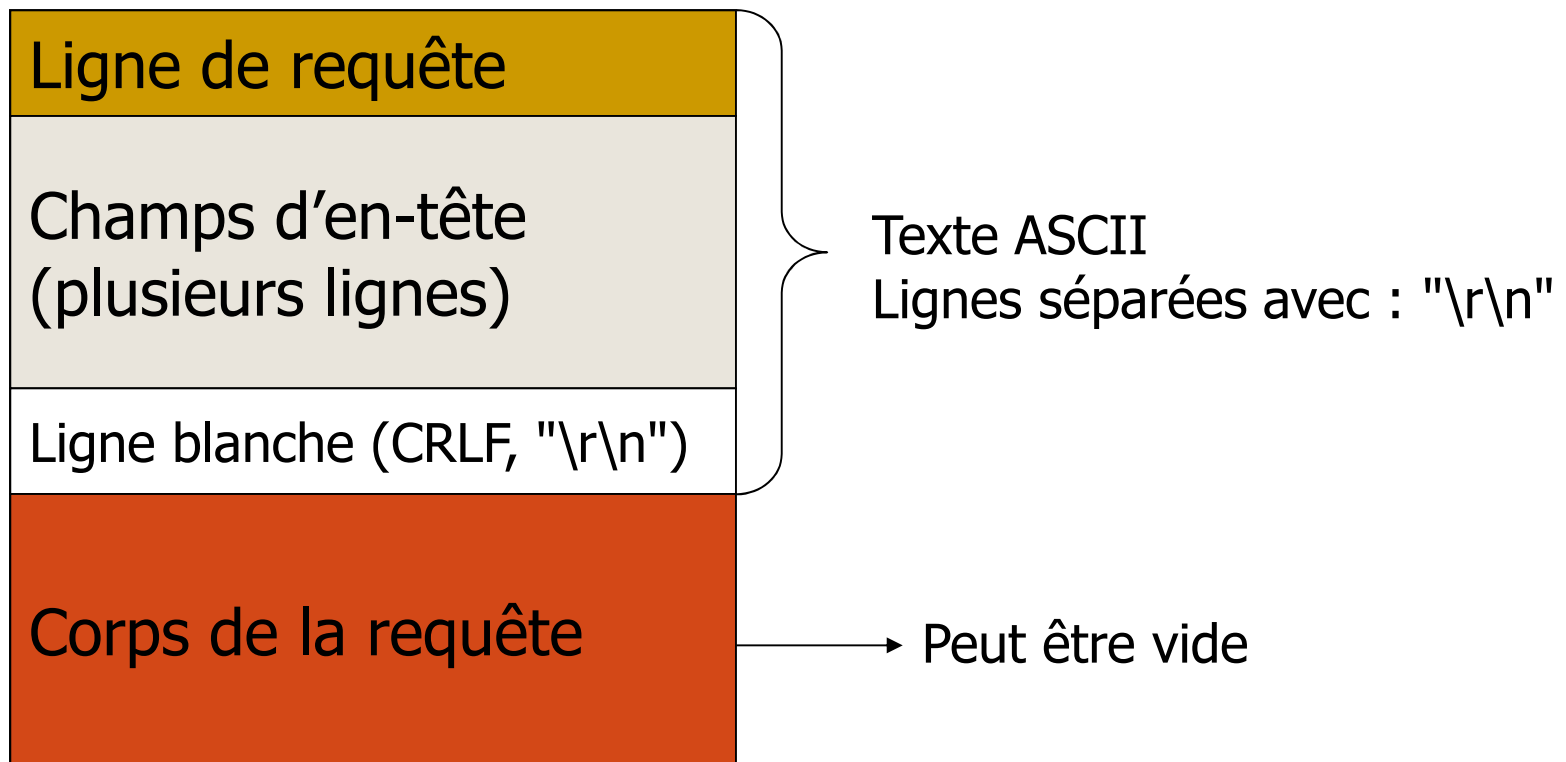


Transport  
(d'habitude TCP)

HTTP 1.0 : une requête/réponse par connexion  
HTTP 1.1 : plusieurs ...



# Requête HTTP – format général



# La ligne de requête

- *Méthode URI protocole CRLF*
- Protocole :
  - HTTP/1.0
  - HTTP/1.1
- Méthode :
  - GET : Requête de la ressource
  - HEAD : Requête de l'en-tête de la ressource
  - POST : Envoi de données à un programme
  - PUT : Envoi de données à l'URL spécifié
  - DELETE : Suppression de la ressource

# Requête HTTP - exemple

**GET /index.htm HTTP/1.1**

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,  
application/x-shockwave-flash, application/vnd.ms-excel,  
application/vnd.ms-powerpoint, application/msword, \*/\*

Accept-Language: fr

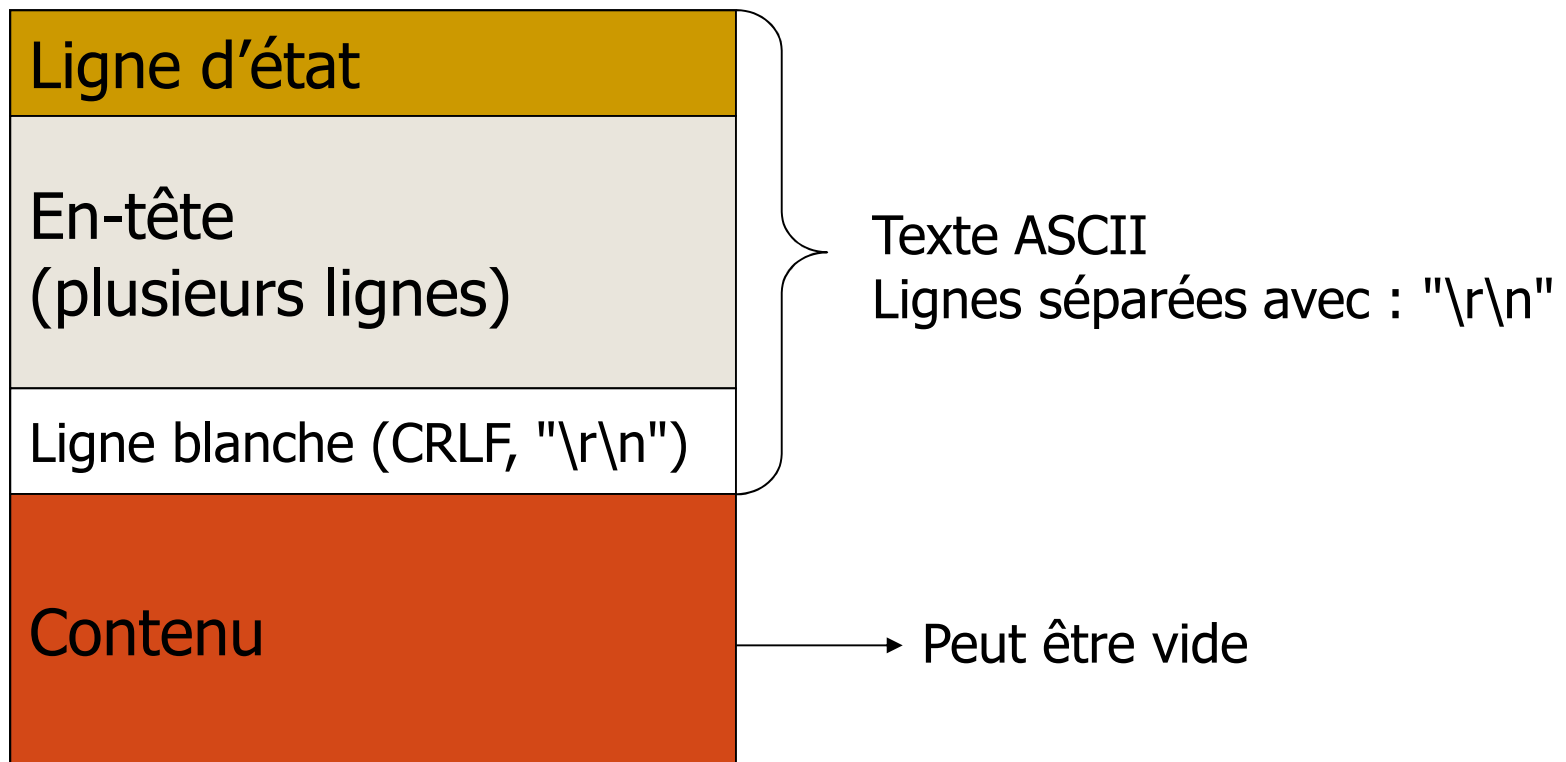
Accept-Encoding: gzip

User-Agent: Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.1;  
SV1; .NET CLR 1.1.4322; .NET CLR 1.0.3705)

Host: www.reuters.com

Connection: Keep-Alive

# Réponse HTTP



# Réponse HTTP - exemple

HTTP/1.1 200 OK

Server: Netscape-Enterprise/4.1

Date: Mon, 15 Nov 2004 10:42:39 GMT

Content-type: text/html

Content-Length: 1234

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<link rel="stylesheet" href="reuters.css" TYPE="text/css">

<link rel="stylesheet" href="reutersTables.css" TYPE="text/css">

<SCRIPT LANGUAGE="JavaScript" SRC="/utilities.js"></script>

...

# Modèle de communication

## HTTP 1.0

Chaque site Web : plusieurs fichiers  
-> Problème de performance

Pour chaque requête :

- Réaliser une connexion TCP
- Envoyer la requête
- Recevoir la réponse
- Libérer la connexion

## HTTP 1.1

Réaliser une connexion TCP

Pour chaque requête :

- Envoyer la requête
- Recevoir la réponse

Libérer la connexion

# Serveur HTTP Proxy

- Fonctionnalités :
  - Cache : garde en mémoire les pages les plus souvent visitées
  - Filtrage, suivi de connexions

