

## La couche Application

---

Adlen Ksentini

## Introduction

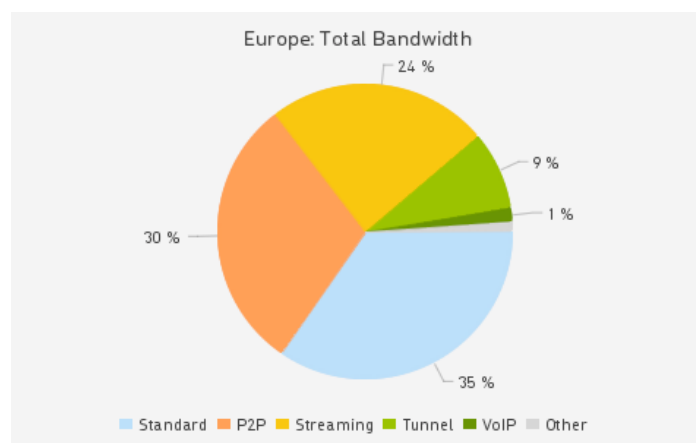
---

- But :
- Connaître les concepts et l'implémentation des protocoles applicatifs communiquant sur un réseau
  - Services offerts par la couche transport
  - Le concept Client/Serveur
  - Le concept Peer-to-Peer (p2p)
- Etude des protocoles applicatifs les plus populaires
  - HTTP
  - FTP
  - TELNET
  - SMTP / POP3 / IMAP
  - DNS
- Programmation des applications réseau
  - API Socket

## Quelques applications réseau

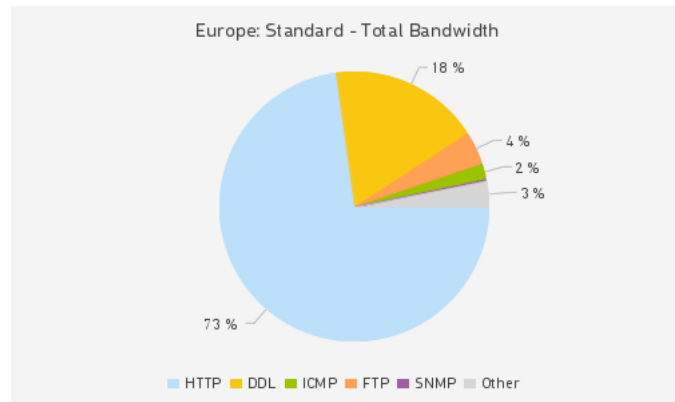
- E-mail
- Web
- Messagerie instantanée
- Contrôle à distance
- Partage de fichiers P2P
- Jeux en réseau (Second life, Warcraft)
- Streaming de vidéo (Youtube, Dailymotion)
- Téléphone sur Internet (Skype)
- Réseaux sociaux (Twitter, Facebook, Googl+)

## Distribution du trafic : Europe



Source : <http://www.internetobservatory.net/europe>

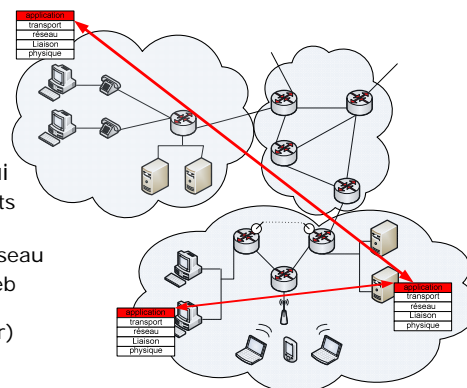
## Distribution du trafic



Source : <http://www.internetobservatory.net/europe>

## Création d'une application réseau

- Ecrire un programme qui
  - Fonctionne sur différents systèmes
  - Communique sur un réseau
  - Ex: Web, le serveur Web communique avec un programme (navigateur) client

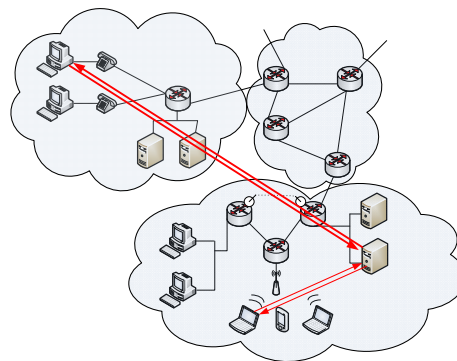


## Applications réseau : architectures

- Client-Serveur
- Peer-to-Peer
- Hybride entre Client/Serveur et P2P

## Architecture Client/Serveur

- Serveur
  - En écoute sur un hôte
  - Adresse IP (+ n° de port) fixe
- Client
  - Communique avec un serveur
  - Peut se connecter par intermittence
  - Ne communique pas avec un autre client
  - Peut avoir une adresse IP (+ n° de port) dynamique (non fixe)



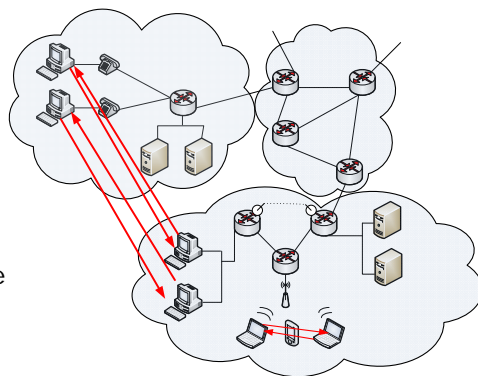
## Architecture Client/Serveur (2)

- Un serveur populaire peut rapidement être submergé par les requêtes
  - Ex. Serveur de recherche sur Internet (google) ou serveur de réseau social (facebook).
- Data center (ou ferme de serveurs)
  - D'une centaine à un millier de serveurs
  - Ex. Amazon, Google, Youtube,...



## Architecture pure P2P

- Pas de serveur
- Les systèmes terminaux communiquent directement
- Les pairs ("peers") sont connectés par intermittence et changent souvent d'adresse IP
- Ex: Gnutella (partage de fichiers) ou PPStream (IPTV)
- Av. : scalable à volonté
- Inc.: difficile à gérer



## Hybride entre Client/Serveur et P2P

- Napster
  - Transfert de fichiers P2P
  - La recherche de fichier est centralisée
    - Chaque peer enregistre un fichier au niveau d'un serveur
    - Chaque peer interroge le même serveur pour la recherche de contenu (fichier)
- Messagerie instantanée (Skype)
  - Chat (discussion) entre deux peers
  - La détection de présence et redirection vers un peer est centralisée
    - Chaque utilisateur s'enregistre auprès d'un serveur central
    - Un utilisateur contacte le serveur central pour récupérer l'adresse IP d'un autre utilisateur

## Les processus de communication

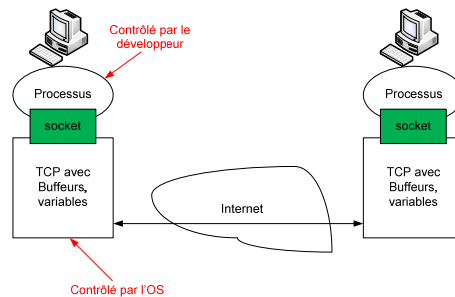
- Un **processus** est un programme qui s'exécute sur un hôte
- Deux processus communiquent dans un même hôte avec des **communications interprocessus** définies par l'OS
- Les processus sur différents hôtes communiquent en utilisant un **protocole applicatif (messages)**

**Processus Client** : le processus qui initialise la communication

**Processus Serveur** : le processus en attente d'une demande de communication

## Sockets

- Défini l'interface entre l'application et la couche transport, deux processus communiquent en émettant et en recevant des données via les sockets
- API :
  - Choix du protocole de transport
  - La possibilité de fixer les paramètres de transport



## Comment identifier un processus distant ?

- Adresse IP de l'hôte distant (32 bits pour IPv4 ou 128 bits pour IPv6)
- Numéro de port – permet de différencier les différents processus locaux sur une même machine, auquel le message doit être transmis
- Ex. :
  - Serveur HTTP : 80
  - Serveur de mail : 25
  - Liste des numéros de port connus (well known) <http://www.iana.org>

## Quel est le service de transport nécessaire à une application?

### Perte de données

- Certaines applications (ex., voix) peuvent tolérer des pertes
- D'autres applications (ex., ftp, telnet) nécessitent une fiabilité à 100%

### Bande passante (débit)

- Certaines applications (ex., multimedia) nécessitent une bande passante minimale
- D'autres applications (dites élastiques) utilisent toute la bande passante disponible

### Délai (Timing)

- Certaines applications (e.g., voix sur IP, jeux interactifs) nécessitent de faibles délais

## Besoin en service de transport de quelques applications connues

Application	Pertes	Bande passante	Sensibilité temp.
Transfert de fichier	sans perte	élastique	non
e-mail	sans perte	élastique	non
Web	tolérant	élastique	non
Audio/video en temps/réel	tolérant	audio: 5 Kbit/s-1 Mbit/s	oui, qq 100 ms
Audio/video enregistré	tolérant	video: 10 Kbit/s-5 Mbit/s	oui, qq s
Jeux interactifs	tolérant	similaire	oui, qq 100 ms
Appli. financière	sans perte	Quelques Kbit/s, élastique	oui et non



## Services offerts par les protocoles de transport Internet

### Service TCP:

- *Orienté connexion*: une connexion est établie entre le client et le serveur
- *Transport fiable et illimité de données num.* entre un processus émetteur et un p<sup>us</sup> récepteur
- *Contrôle de flux* : l'émetteur ne submerge pas le récepteur
- *Contrôle de congestion* : réduit le débit de l'émetteur quand le réseau est congestionné
- *Ne propose pas* de garantie de délai ou de bande passante minimale

### Service UDP:

- Transport limité et non fiable de données numériques
- Ne propose pas de connexion, de fiabilité, de contrôle de flot, de contrôle de congestion, de garantie temporel ou de bande passante

Aucun des deux protocoles ne chiffrent les messages

## Applications Internet : application, protocoles de transport

Application	Protocole applicatif	Protocole de transport
E-mail	SMTP [RFC 5321]	TCP
Accès distant	TELNET [RFC 854]	TCP
Web	HTTP [RFC 7230]	TCP
Transfert de fichier	FTP [RFC 959]	TCP
Streaming multimedia	RTP [RFC 3550]	UDP ou TCP
Voix sur IP	SIP ou skype (propriétaire)	UDP

## Protocole applicatif

---

- Type des messages à échanger
  - Ex. : requête ou réponse
- Syntaxe du message
  - Les champs composant un message (ordre, taille, code, etc.)
- Sémantique du message
  - Le sens des infos. contenues dans les champs du message
- Règles
  - Quand et comment envoyer et répondre à un message
- Protocoles du domaine publique
  - Définis dans des RFCs
  - Ex. : HTTP, SMTP
- Protocoles propriétaires
  - Ex. : Kazaa, Skype

## World Wide Web et HTTP

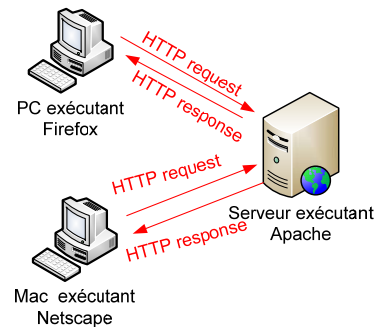
---

- World Wide Web (WWW) : « le cœur de l'Internet »
- Page Web:
  - Contient des “documents” (documents)
    - Un simple fichier HTML, une image JPEG ou GIF, une applet JAVA, une animation FLASH
  - Identifié par une URL (Uniform Resource Locator)
- La plupart des pages Web contiennent :
  - Du texte HTML
  - Documents référencés par le biais de leur URL
- L'URL a deux composants : nom d'hôte du serveur et le chemin d'accès au document  
[www.ifsic.univ-rennes1.fr/M1\\_crypto/pic.gif](http://www.ifsic.univ-rennes1.fr/M1_crypto/pic.gif)

## HTTP

HTTP: HyperText Transfer Protocol

- Couche applicative Web
- Définit le modèle client/serveur
  - *client*: le navigateur qui demande, reçoit, affiche les documents Web
  - *serveur*: serveur Web envoie les réponses (souvent le contenu des documents demandés) aux requêtes
- HTTP 1.0: RFC 1945
  - Jusqu'à 1997
- HTTP 1.1: RFC 2068
  - À partir de 1998
  - Compatible avec la version 1.0



## HTTP (suite)

- Le navigateur web : agent de l'utilisateur (UserAgent)
  - Client HTTP
  - Par ex. Chrome, Internet Explorer, Firefox/Mozilla
- Le serveur web
  - Serveur HTTP
  - Par ex. Apache, Microsoft Internet Information Server (IIS), Netscape Enterprise Server

## HTTP (suite)

### Utilise les services de transport TCP :

- Le client initialise une connexion TCP (crée une socket) avec le serveur, port 80
- Le serveur accepte la demande de connexion TCP du client
- Les messages HTTP (protocole applicatif) sont échangés entre le navigateur web (client HTTP) et le serveur Web
- La connexion TCP est fermée

### HTTP est « sans état »

- Le serveur ne maintient aucune information au sujet des requêtes précédentes des clients

### Les protocoles gardant un "état" sont complexes !

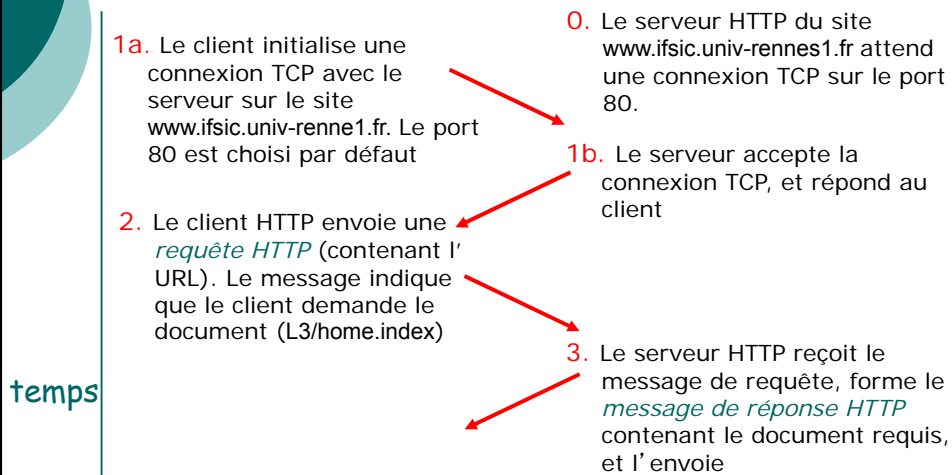
- L'historique doit être gardé
- Si le serveur ou le client s'arrête de fonctionner les états peuvent être inconsistants

## Connexions HTTP

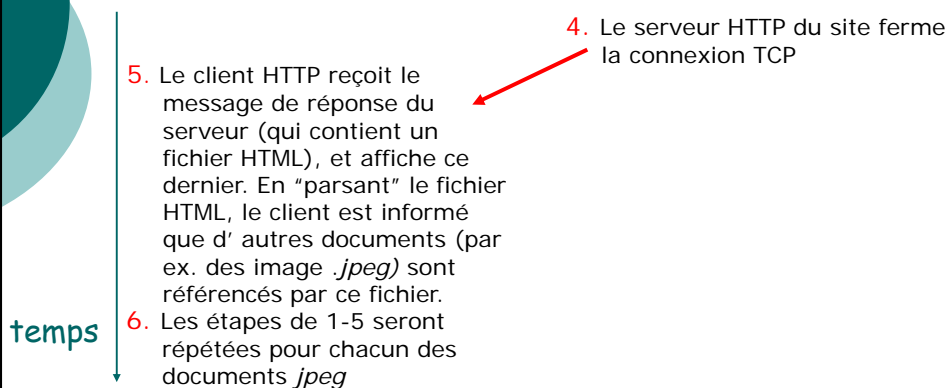
- Connexion HTTP non-persistante
  - Au plus, un document est transmis avec une connexion TCP
  - HTTP/1.0 utilise ce type de connexion
- Connexion HTTP persistante
  - Plusieurs documents peuvent être transmis avec une seule connexion TCP (entre le client et le serveur)
  - HTTP/1.1 utilise ce type de connexion par défaut

## HTTP non-persistant

L'utilisateur consulte l'URL `www.ifsic.univ-rennes1.fr/L3/home.index`



## HTTP non-persistant (suite)



- o La page affichée peut être différente suivant le navigateur
- o HTTP spécifie uniquement les messages à échanger et leur format (pas le format du contenu : c-à-d. pas le format des documents) !

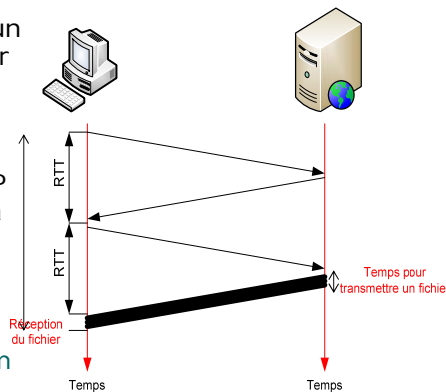
## Modélisation du temps de réponse

**RTT (Round Time Trip)** : le temps nécessaire pour qu'un paquet fasse un aller-retour

Temps de réponse :

- 1 RTT pour établir la connexion TCP
- 1 RTT pour la requête HTTP et les quelques octets de la réponse HTTP
- Temps de transmission du document

Délai total =  $2 \cdot \text{RTT} + \text{Temps\_transmission}(\text{document})$



## HTTP persistant

HTTP non-persistant

- HTTP/1.0
- 2 RTTs sont nécessaires pour lire chaque document
- Le serveur interprète les requêtes, répond et ferme la connexion TCP

HTTP persistant

- HTTP/1.1 (par défaut)
- Une seule connexion TCP est ouverte vers le serveur
- Le client envoie une requête pour tous les documents requis dès qu'ils sont référencés dans le fichier HTML

Persistant sans pipelining

- Le client envoie une nouvelle requête qu'après la réception de la réponse de la précédente requête
- Un RTT pour chaque document

Persistant avec pipelining

- HTTP/1.1 (par défaut)
- Le client envoie une nouvelle requête sans attendre la réponse de la requête précédente
- Les requêtes sont (peuvent être) envoyées en même temps
- Les réponses sont (peuvent être) envoyées en même temps

## Les messages HTTP : *request*

- Deux types de messages HTTP : *request*, *response*
- Message HTTP :
  - Lignes de caractères
  - Codage ASCII (compréhensible par les humains)

Ligne de requête  
(commandes  
GET, POST, HEAD)

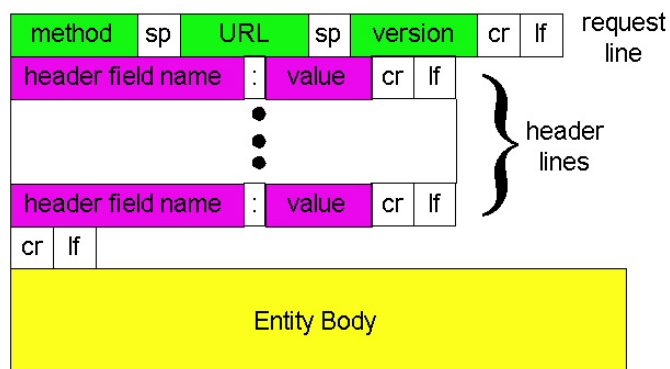
Lignes  
d'entête

Le retour chariot,  
indique la fin  
du message

```
GET /homedir/page.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr
```

<retour chariot>, <saut de ligne>

## Format du message Requête HTTP



## Forme de la demande

---

- Avec la méthode **POST**
  - La page web se présente sous la forme d'un formulaire
  - Envoie des données définissant la demande **dans le corps du message**
- Avec la méthode **GET**
  - Envoie des données définissant la demande **dans l'URL**

## Types de méthodes

---

### HTTP/1.0

- GET
- POST
- HEAD
  - Demande les informations sur le document et non le document lui-même
  - L'en-tête uniquement

### HTTP/1.1

- GET, POST, HEAD
- PUT
  - « Uploade » le document placé dans corps du message dans le document spécifié par l'URL
- DELETE
  - Supprime le document spécifié par l'URL



## Les messages HTTP : *response*

Ligne de réponse  
(protocole,  
code de statut,  
état)

HTTP/1.1 200 OK

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998 .....

Content-Length: 6821

Content-Type: text/html

Lignes  
d'entêtes

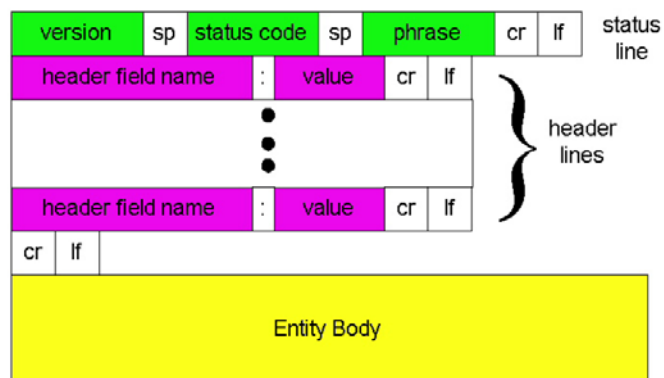
Données, par ex.,  
le fichier html

data

data data data

data ...

## Format du message Response HTTP



## Code de réponse de HTTP

---

- Dans la première ligne de la réponse

### 200 OK

- La requête a réussi et le document demandé est dans le corps du message

### 301 Moved Permanently

- Le document demandé a changé définitivement de place, son nouvel emplacement est donné dans la suite du message

### 400 Bad Request

- La requête est erronée

### 404 Not Found

- Le document demandé n'est pas disponible sur le serveur

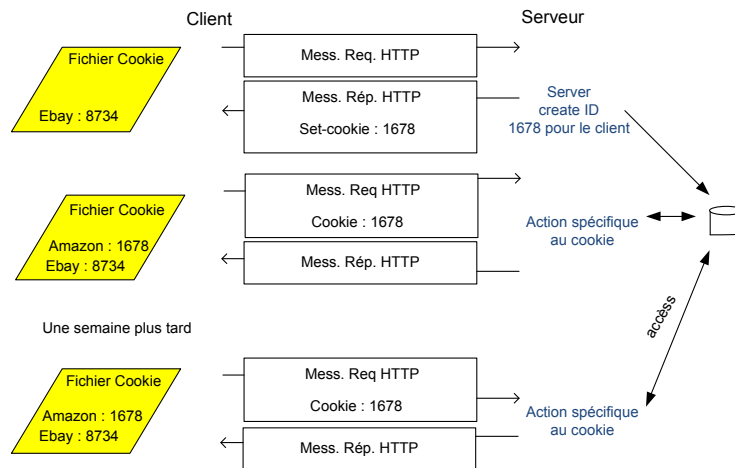
### 505 HTTP Version Not Supported

## Sauvegarde d'états entre clients et serveurs – Les cookies

---

- Beaucoup de sites emploient les cookies
  - Identifier les clients
  - Personnaliser le contenu vis-à-vis des clients
- Quatre composants
  - Une ligne d'en-tête cookie dans la réponse HTTP
  - Une ligne d'en-tête cookie dans la requête HTTP
  - Le **cookie est sauvegardé sur la machine du client**, il est géré par le navigateur du client
  - Une base de données des habitudes des usagers
- Par ex. :
  - Toto visite un site de e-commerce pour la première fois
  - A l'initialisation de la requête HTTP, le site crée un unique identifiant ID et l'insère dans le cookie et une base de données pour une future identification de Toto
    - Toto a accès à Internet toujours du même PC
    - Il se connecte une fois par semaine en moyenne

## Cookies (suite)



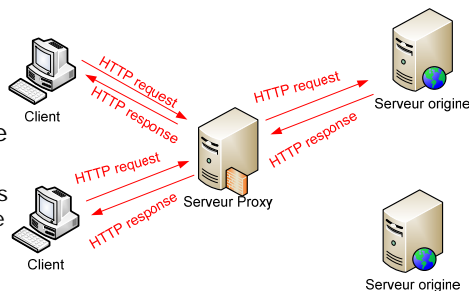
## Cookies (suite)

- Fonctionnalités associées à l'utilisation de cookies
  - Autorisation
  - Achats en ligne (l'exemple du panier)
  - L'état d'une session Web
- Par contre
  - Les cookies permettent aux sites web de connaître vos habitudes sur le net
    - Les moteurs de recherche
    - Les publicitaires

## Cache Web – Serveur Proxy

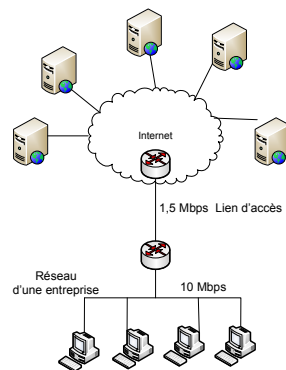
**But:** satisfaire la requête du client rapidement sans utiliser le serveur origine

- L'utilisateur configure son navigateur : accès au Web via le cache
- Le client envoie toutes ses requête HTTP vers le cache web
  - Si le document est dans le cache, on le retourne
  - Sinon on demande au serveur initial, on mémorise la réponse et on répond ensuite à la requête
- Le cache web a une double fonction : client et serveur



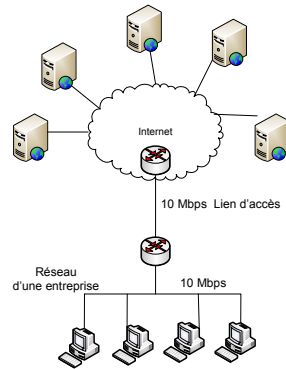
## Cache Web : exemple

- **Suppositions**
  - Taille moyenne d'un document = 100.000 bits
  - Fréquence moy. des requêtes HTTP vers les serveurs web = 15 req/s
  - RTT du routeur local vers n'importe quel serveur = 2 s
- **Conséquences**
  - Débit des requêtes : 1,5 Mbit/s
  - Utilisation du LAN = 15%
  - Utilisation du lien d'accès = 100%
  - Délai total = délai sur Internet + délai du lien d'accès + délai sur le LAN = 2 s + **des minutes !** + (<1) milliseconde



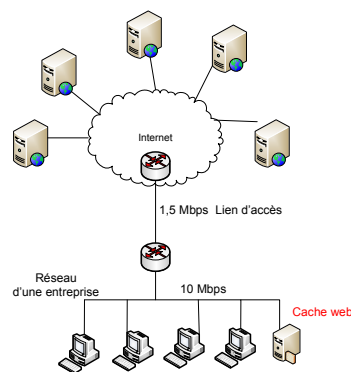
## Cache Web : exemple (suite)

- Une solution
  - Augmenter le débit du lien d'accès, 10 Mbit/s.
- Conséquences
  - Utilisation du LAN = 15%
  - Utilisation du lien d'accès = 15%
  - Délai total = délai sur Internet + délai d'accès + délai sur le LAN = 2 sec + qq millisecondes + milliseconde
- Solution très couteuse



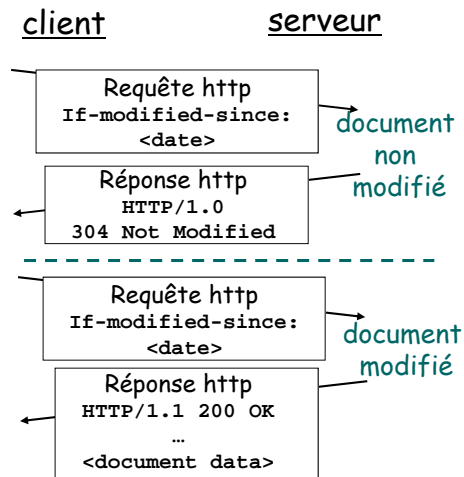
## Cache Web : exemple (suite)

- Installer un cache Web
  - On suppose que le cache satisfait 40% des requêtes
- Conséquences
  - 60% des requêtes sont traitées par le serveur origine
  - Utilisation du lien d'accès réduite à 60%, impliquant des délais un peu plus faible (on suppose 10 ms)
  - Délai total = délai sur Internet + délai d'accès + délai sur le LAN =  $0,6 * 2 \text{ s} + \text{qq millisecondes} < 1,2 \text{ s}$

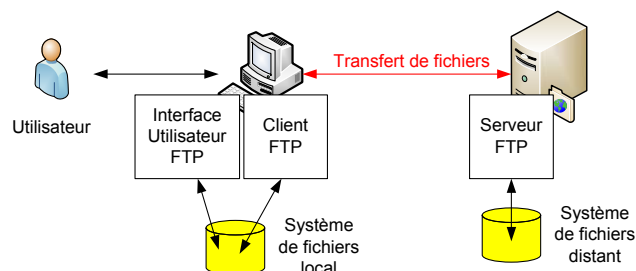


## GET conditionnel

- Objectif : ne pas envoyer un document que le client a déjà dans son cache
- Client: spécifie la date de la copie en cache dans la requête http
  - **If-modified-since: <date>**
- Serveur: le corps de la réponse est vide si la copie en cache est à jour  
**HTTP/1.0 304 Not Modified**



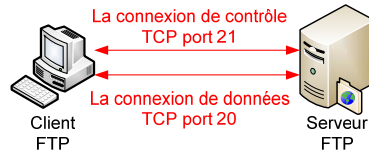
## FTP : le protocole de transfert de fichier



- Transfert de fichiers entre (deux) machines
  - Le transfert peut être ternaire (C : A=>B)
- Basé sur le modèle Client/Serveur
  - Le client : l'utilisateur initialisant la connexion
  - Le serveur : la/les machines distantes
- FTP : RFC 959
- Le serveur FTP écoute sur le port 21

## FTP: la connexion de contrôle et la connexion de données

- Le client FTP contacte le serveur sur le port 21
  - Utilisation de TCP
- Le client obtient les autorisations, échangées sur la connexion de contrôle
- Le client navigue dans le système distant en envoyant des commandes sur la connexion de contrôle
- Dès que le serveur reçoit une demande de transfert de fichier, il ouvre une connexion pour les données sur un nouveau port TCP
- Dès que le fichier est téléchargé, le serveur clos la connexion de données



- Le serveur ouvrira une nouvelle connexion de données pour le transfert d'un nouveau fichier
- La connexion de contrôle est dite **Hors bande**
- Le serveur FTP maintient l'état de la connexion :
  - Le répertoire courant, la dernière connexion ouverte

## FTP en pratique

```
CLI$ ftp ftp://ftp.free.fr
...
220 Welcome to ProXad FTP server
230 Login successful.
...
ftp> cd mirrors/ftp.ubuntu.com/dvd/current
250 Directory successfully changed.
ftp> ls
...
-rw-rw-r-- 1 ftp ftp 1468051456 Jun 12 2012 quantal-dvd-i386.iso
...
226 Directory send OK.
ftp> get quantal-dvd-amd64.iso
local: quantal-dvd-amd64.iso remote: quantal-dvd-amd64.iso
150 Opening BINARY mode data connection for quantal-dvd-amd64.iso (1476702208 bytes).
100% |*****| 1408 MIB 4.04 MIB/s 00:00 ETA
226 File send OK.
1476702208 bytes received in 05:48 (4.04 MIB/s)
ftp> quit
```

## FTP : commandes et réponses

### Ex. de commandes

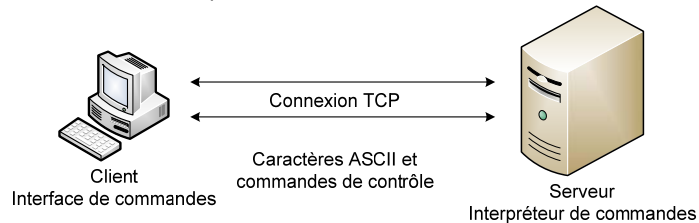
- Envoie des commandes (en code ASCII, 7-bits) sur la connexion de contrôle
- **USER login**
- **PASS mot\_passe**
- **LIST**
  - liste les fichiers présents dans le répertoire courant
- **RETR fichier**
  - récupère un fichier
- **STOR fichier**
  - sauvegarde un fichier sur l' hôte distant

### Ex. de réponses

- Statut et phrases (comme HTTP)
- **331 Username OK, password required**
- **125 Data connection already open, transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

## TELNET

- TELNET : protocole pour la commande d'une machine à distance (accès distant à l'interpréteur de commandes). Hist. : « terminal distant virtuel »
- telnet => logiciel qui utilise le protocole TELNET
- TELNET
  - permet de connecter un client (système composé d'un écran et d'un clavier) à un serveur (interpréteur de commandes) à l'aide d'une connexion TCP





## TELNET

---

- Port 23
- Données et contrôle sur la même connexion TCP
- Network Virtual Terminal (NVT)
- Négociation des options entre le client et le serveur (jeu des caractères, mode ligne ou caractère, etc.)

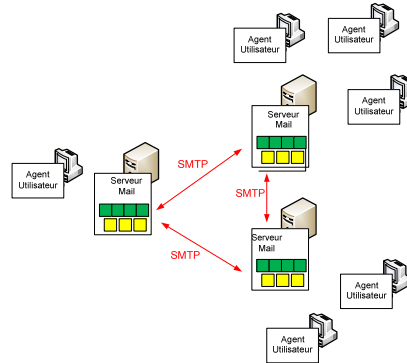
## SSH

---

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

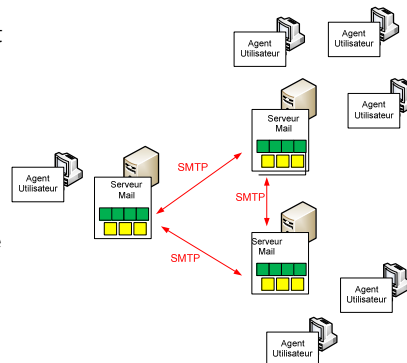
## La messagerie électronique

- Trois composants majeurs
  - L'agent de l'utilisateur (user agent)
  - Le serveur de messagerie
  - Le protocole SMTP (Simple Mail Transfer Protocol)
- Agent de l'utilisateur (User Agent)
  - Composer, envoyer, lire des emails
  - Ex. : Mail, Outlook, Thunderbird



## Messagerie électronique

- Serveur de messagerie
  - Boîte d'emails : contient les messages à destination d'un utilisateur
  - File d'attente des messages : contient les messages SMTP en cours d'émission
  - Le protocole SMTP : est utilisé entre serveurs de messagerie pour l'envoi d'emails
    - Client : serveur SMTP envoyant un email
    - Serveur : serveur SMTP destinataire de l'email

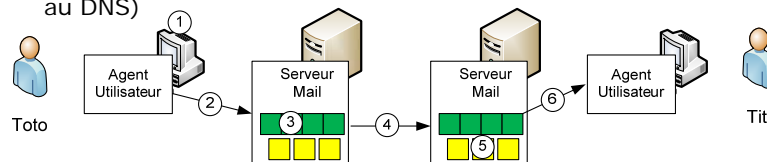


## Messagerie électronique : SMTP

- Basé sur TCP pour l'envoi fiable de messages électroniques (emails) du client vers le serveur, sur le port 25
- Transfert direct entre serveurs SMTP
- Trois phases pour la session SMTP
  - Etablissement de la session SMTP
  - Transfert des messages SMTP
  - Clôture de la session SMTP
- Interaction commandes / réponse
  - **Commandes** : ASCII texte
  - **Réponses** : Statut de la commande
- Les messages sont codés en lignes de caractères en ASCII sur 7 bits

## Scénario : email de Toto vers Titi

- 1- Toto utilise son UA pour composer un email à destination de [titi@univ-rennes1.fr](mailto:titi@univ-rennes1.fr)
- 2- L'UA de Toto envoie un email vers son serveur de messagerie par défaut
- 3- Le serveur de messagerie utilisé par Toto ouvre une connexion SMTP/TCP avec le serveur de messagerie de Titi (ce serveur a été trouvé grâce au DNS)
- 4- Le serveur de messagerie de Toto envoie le message sur la connexion TCP
- 5- Le serveur de messagerie de Titi place le message dans la boîte d'emails de Titi
- 6- Titi demande à son UA de lire le message



## Exemple d'interaction SMTP

---

```
S: 220 univ-rennes1.fr
C: HELO quelquepart.fr
S: 250 hello quelquepart.fr, pleased to meet you
C: MAIL FROM: toto@quelquepart.fr
S: 250 toto@quelquepart.fr... Sender ok
C: RCPT TO: titi@univ-rennes1.fr
S: 250 titi@univ-rennes1.fr
C: DATA
S: 354 Enter mail, end with "." on a line itself
C: Blablablalbla
C: Blablalbla
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 univ-rennes1.fr closing connection
```

## SMTP (fin)

---

- SMTP utilise une connexion persistante
- SMTP impose l'utilisation de lignes de caractères codés en ASCII sur 7 bits (Corps et En-têtes SMTP)
- Le serveur détermine la fin du message SMTP grâce à la suite CRLF.CRLF
- Comparaison avec HTTP :
  - HTTP : technologie **Pull**, le client demande le document
  - SMTP : technologie **Push**, le client envoie le document
  - Les deux utilisent des commandes/réponses codés par des lignes de caractères, et des codes retours (Status)

## Format du message électronique

- o Format des messages électroniques (RFC 822 )

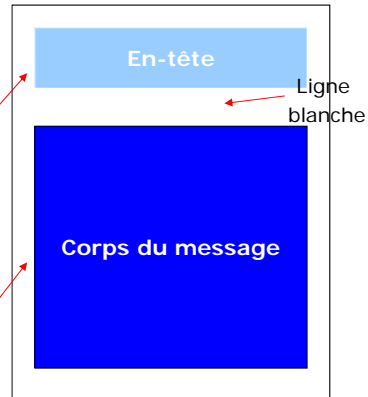
- Différent du protocole pour l'échange de messages électroniques, c.-à-d. SMTP

- o Entête du message élect. : des lignes d'en-têtes

- To : <vers>
- From : <de>
- Subject : <objet>  
(différent de la commande SMTP)

- o Corps du message élect.

- Le message uniquement en lignes de caractères ASCII



## Format du message : l'extension multimédia

- o MIME (Multipurpose Internet Mail Extensions) : Extension pour le multimédia (RFC 2045, 2056)
- o Lignes d'en-tête pour déclarer le type du contenu MIME

Version MIME

Méthode utilisée pour encoder les données

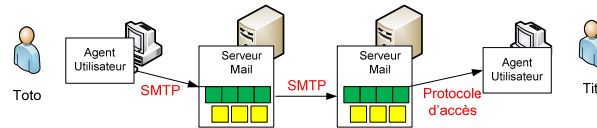
Type et sous-type des données multimédia

Données encodées

```
From : toto@crepes.fr
To : titi@hamburger.edu
Subject : Image d'une crepe
MIME-Version : 1.0
Content-Transfer-Encoding : base64
Content-Type : image/jpeg

base64 encoded data .....
.....
.....
..... base64 encoded data
```

## Protocoles d'accès à la messagerie



- SMTP : Transmission vers et stockage sur le serveur de messagerie du destinataire
- Protocole d'accès à la messagerie : récupérer les messages du serveur du destinataire
  - POP : Post Office Protocol [RFC 1939]
    - Authentification (agent-> serveur) et téléchargement
  - IMAP : Internet Mail Access Protocol [RFC 1730]
    - Manipulation des messages sur le serveur
  - HTTP : Hotmail, Yahoo Mail, Gmail, etc.

## POP3 et IMAP

### POP3

- Mode par téléchargement et suppression
  - Pas de possibilité de retrouver le message si on change d' UA
- Mode par téléchargement et sans suppression
  - Possibilité d'avoir une copie d'email sur chaque client utilisé
- POP3 est sans état

### IMAP

- Maintien des messages sur le serveur
- Permettre aux utilisateurs d'organiser leur boîte aux lettres en répertoire
- IMAP maintient des états

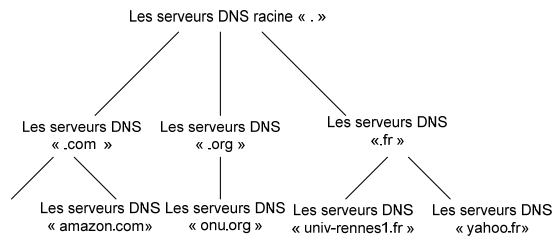
## DNS : Domain Name System

- Une personne => plusieurs identifiants
  - Numéro de Sécurité Social, nom, passeport ..
- Hôtes et routeurs sur Internet
  - Une adresse IP (32 bits) pour identifier le destinataire d'un paquet
  - « nom de domaine », par ex. « www.yahoo.fr » utilisé par les humains
- Q : comment trouver une correspondance entre un nom et une adresse IP ?
- Système de résolution de nom de domaine (DNS)
  - Une BD mondiale, distribuée et hiérarchique
  - Fiable et scalable :
    - Plusieurs serveurs de nom
- Protocole de niveau applicatif
  - Hôtes et serveurs de nom communiquent pour résoudre des noms de hôtes
    - Nom de domaine et Adresse IP correspondante

## Services DNS

- Résolution
  - Nom de hôte → Adresse IP  
text.esams.wikimedia.org. IN A 91.198.174.232
- Alias
  - Nom d'alias (canonique)  
fr.wikipedia.org. IN CNAME text.wikipedia.org.
  - Nom pour les serveurs d'email  
wikimedia.org. IN MX 10 mchenry.wikimedia.org.
- Résistance aux pannes et équilibrage de charge (load balancing)
  - Plusieurs adresses IP peuvent correspondre à un seul nom de domaine
- Contrôle de l'accessibilité/censure ! : gestion dynamique des enregistrements
- Authentification des enregistrements : DNSSEC

## DNS hiérarchique



Un client veut récupérer l'@IP de [www.univ-rennes1.fr](http://www.univ-rennes1.fr)

- Interrogation d'un des serveurs racines pour trouver le serveur DNS du domaine *.fr*
- Le client interroge le serveur DNS de *.fr* afin de trouver le serveur DNS du domaine *univ-rennes1.fr*
- Le client interroge le serveur DNS de *univ-rennes1.fr* pour obtenir l'@IP de *www.univ-rennes1.fr*

## DNS Roots

- Ils sont contactés par les serveurs DNS, qui n'arrivent pas à résoudre directement une requête
- Un serveur de noms du domaine « . » (dit racine)
  - Connait un serveur faisant autorité sur chaque domaine TLD
  - Renvoie ce résultat au serveur de noms local

13 serveurs dans  
le monde





## Les TLD (Top Level Domain) et les serveurs *authoritative* (autorité)

---

- Les serveurs de noms TLD : responsables des domaines de noms com, org, net, edu, tv, ..., et ceux représentant les pays fr, es, uk, dz
  - Par ex., le .fr est géré par l'AFNIC
- Les serveurs de noms faisant autorité : les serveurs de noms d'une entreprise, d'un institut, ou d'une organisation
  - Peuvent être gérés par l'entreprise ou l'organisation
  - Contiennent les enregistrements des noms appartenant au domaine de l'entreprise

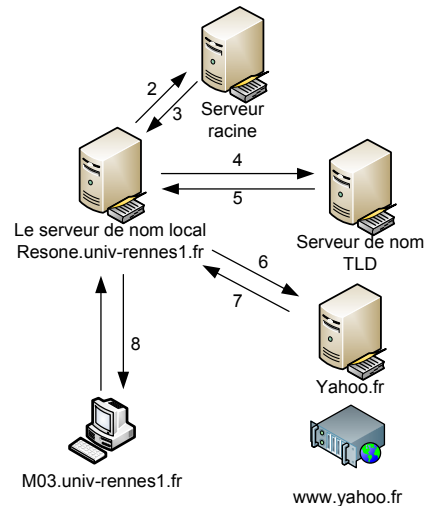
## Serveur de noms local

---

- Chaque FAI ou Institution (Université) à son serveur de noms local
  - Le serveur de noms par défaut
- Il reçoit les requêtes de résolution de noms émises par les hôtes de son domaine
  - Il fonctionne comme un proxy, il transfère les requêtes vers le système DNS
- Il reçoit les requêtes de résolution de noms des hôtes et serveurs distants
  - Il répond à ces requêtes
  - Il connaît le nom et l'adresse des hôtes de son domaine

## Exemple

- Une machine (m03.univ-rennes1.fr) veut résoudre le nom www.yahoo.fr



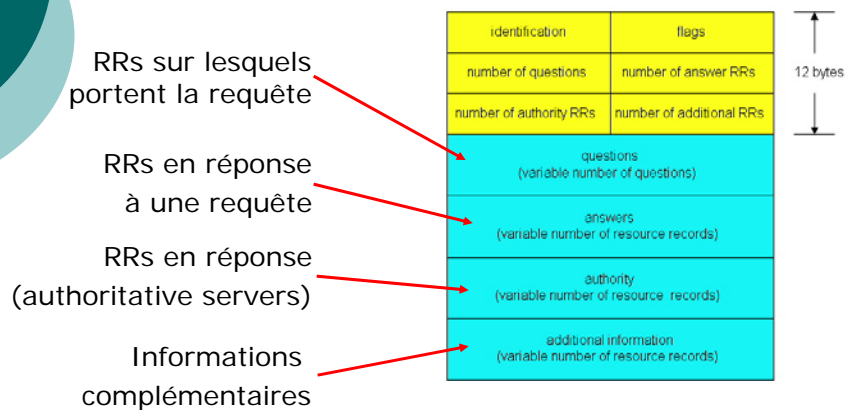
## DNS : le système de cache

- Une réponse DNS peut être mise en cache
  - Chaque entrée du cache a une durée de vie
  - Les informations sur les serveurs des domaine TLD sont naturellement rapidement mises en cache au niveau de tous les serveurs locaux
    - Eviter les requêtes vers les serveurs de noms du domaine de noms racine
- RFC 2136

## DNS

- Stockage distribuée des données : Resource records (RR)
  - Format RR : (nom, valeur, type, ttl)
- Type A
  - Nom : nom d'une machine
  - Valeur : l'adresse IP de la machine
- Type CNAME
  - Nom : nom d'une machine
  - Valeur : le nom canonique de la machine
- Type MX
  - Nom : nom de domaine
  - Valeur : le nom du serveur de mail associé au nom de domaine

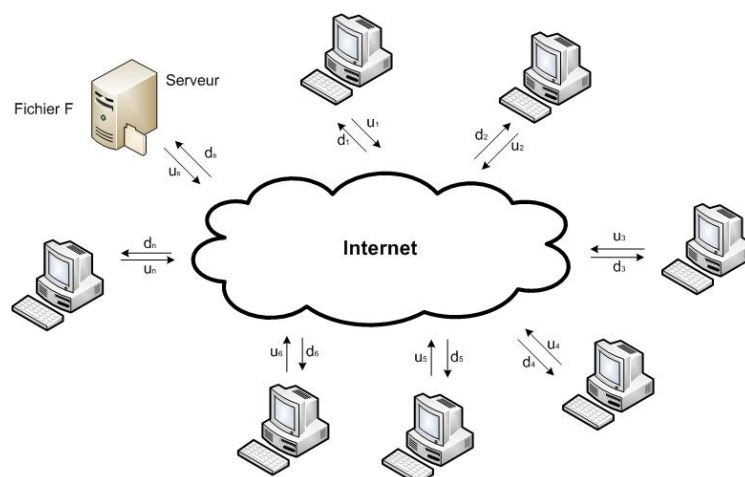
## DNS format message



## Architecture P2P

- Partage de fichier : Bitorrent, Gnutella
- Téléphonie sur IP : Skype

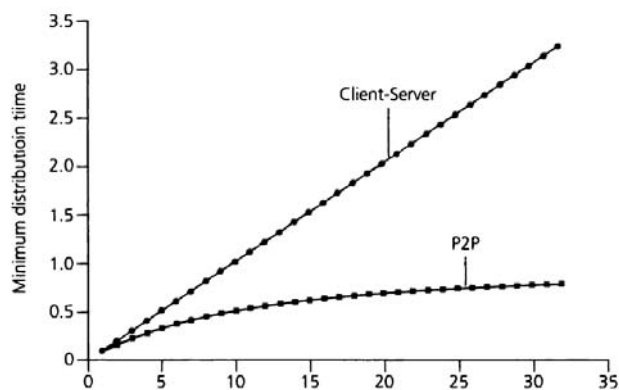
## Partage de fichier : client/serveur vs p2p



## Partage de fichiers : client/serveur vs p2p

- Architecture client/serveur
  - Temps (minimum) pour satisfaire la demande de N clients :  $D_{cs} \geq \max(N.F/u_s, F/d_{\min})$
- Architecture p2p
  - Initialement, uniquement le serveur a le fichier
  - Temps (minimum) pour satisfaire la demande de N clients :  $D_{p2p} \geq \max(F/u_s, F/d_{\min}, N.F/u_s + u_1 + \dots + u_N)$

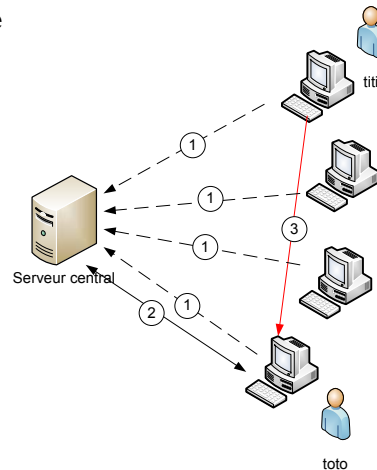
## Partage de fichiers : client/serveur vs p2p



- On suppose que tous les peers ont la même capacité d'envoi
- $F/u = 1$  heure,  $u_s = 10u$ .

## P2P centralisé

- Le fonctionnement de « *Napster* »
- 1) Lorsqu'un peer se connecte, il envoie au serveur central, son IP, et le contenu à partager avec les autres peers
- 2) Toto interroge le serveur pour le fichier « fich1 »
- 3) Toto demande le fichier « fich1 » à Titi



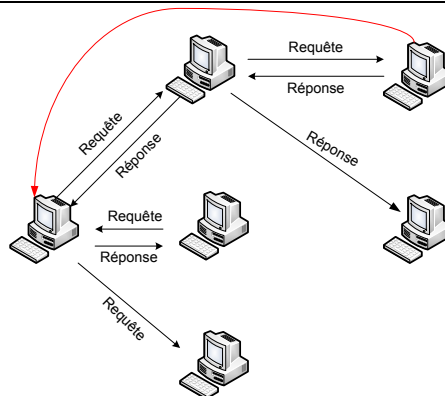
## P2P centralisé : problèmes

- Point central : pas de tolérance aux pannes
- Congestion autour du serveur
- ⇒ Le transfert de fichier est décentralisé, mais la localisation du contenu est centralisée

## P2P décentralisé : Gnutella

- Complètement distribué
  - Pas de serveur central
- Protocole publique
- Le réseau overlay: graphe
  - L'existence d'une connexion TCP entre un peer X et Y est représentée par un arc dans le réseau overlay
  - Le réseau overlay : les arcs et les peers
  - Un arc n'est pas un lien physique
  - En moyenne, chaque peer a moins de 10 voisins

## Le protocole Gnutella



- Requêtes envoyées via les connexions TCP
- Chaque peer diffuse la requête
- La réponse (@IP) est renvoyé sur le même chemin que la requête

## Recherche du contenu ?

---

- Organisation en enregistrement d'une base de données (BD) : nom du contenu, adresse IP du détenteur de ce contenu
- Difficile à mettre en place dans un environnement décentralisé (p2p)
- Solution : distribuer la BD sur l'ensemble des peers
  - Chaque peer contiendra une partie de la BD
  - Chaque peer interroge la BD pour une paire de clé particulière

## Exemple de recherche sur la BD distribuée

---

- Toto recherche la dernière distribution Linux, il interroge la BD distribuée.
- La DB sait que Titi est responsable de la clé « Linux ».
- Alors la DB demande à Titi d'envoyer l'adresse des détenteurs du contenu « Linux »



## Solution DB distribu 

---

- o Solution na ve : r partir al atoirement les cl s sur tous les peers.
- o Chaque peer interroge l'ensemble des peers pour conna tre les d tenteurs du contenu.
- o Solution tr s couteuse.
  
- ✓ Distributed Hash Table (DHT)

## Distributed Hash Table (DHT)

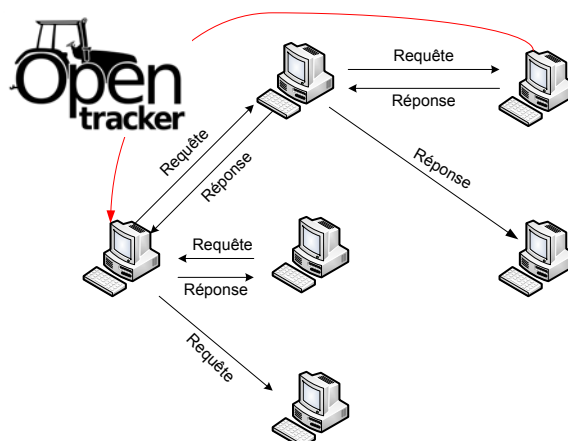
---

- o Affecter un identifiant   chaque peer, entre 0 et  $2^{n-1}$
- o Utiliser une fonction sp ciale qui associe un nom de contenu avec une valeur
  - Fonction de hachage : ex. MD5, SHA
- o Associer le contenu avec l'identifiant le plus proche.
- o Ex.
  - On suppose que  $n=4$ , donc les identifiants de peer ainsi que les valeurs (cl s) sont dans l'intervalle  $[0,15]$ .
  - On suppose que 7 peers sont dans le r seau ayant comme identifiant 1,3,5,7,8,10,13 et on cherche le contenu « Linux » dont la valeur produite par la fonction de hachage est 12.
  - Le peer le plus proche de 12 est 13.

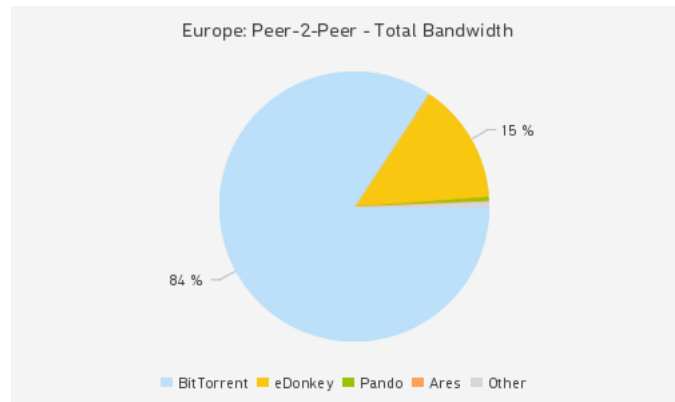
## P2P : BitTorrent

- Protocole et client BitTorrent (représentait 30% du trafic du cœur du réseau en 2011)
- Pas de recherche de contenu
  - Un fichier .torrent, contient un ensemble d'informations (tracker) pour le téléchargement de contenus
  - Obtenu à partir d'un serveur web
- Un torrent => une session de transfert d'un seul contenu entre peers
- Deux types de peers
  - Seeds :
    - proposent le téléchargement du contenu en entier
    - restent connectés
  - Leechs :
    - proposent le téléchargement d'un chunk (morceau)
    - téléchargent un chunk qu'ils n'ont pas encore
    - volatiles

## BitTorrent



## P2P stat.

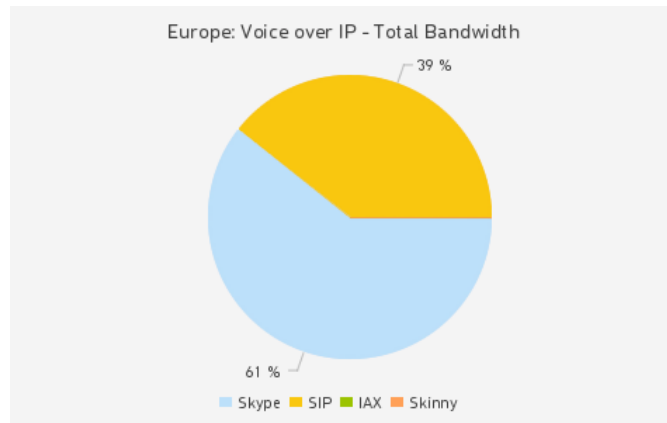


Source : <http://www.internetobservatory.net/europe>

## Skype VoIP basé sur P2P

- Créé par le concepteur de KaZaa
- Racheté par eBay à 2,6 milliards de dollars puis par Microsoft
- Protocole propriétaire, tous les messages sont chiffrés
- Notion de superNode (super nœud)
  - Elu en fonction de leur grande capacité
  - Permet de traverser le NAT
  - Maintient la liste des connectés

## Skype vs VoIP



Source : <http://www.internetobservatory.net/europe>

## Couche Application : sommaire

- Les protocoles : HTTP, FTP, SMTP, DNS, P2P, etc.
- Le modèle d'échanges Client/Serveur
  - Le client demande un service (des informations)
  - Le serveur répond avec les données, un statut (code)
- Le format des messages :
  - En-tête du message : des champs contenant des informations de contrôle sur les données
  - Corps du message : les données
- Quelques points importants :
  - Message de contrôle et les messages de données
  - Avec état ou bien sans état
- Transfert fiable versus transfert non fiable
- Présentation du contenu : HTML/XML, MIME