# Réseaux : Applications et couche application

Sławek Staworko

Univ. Lille3

8 décembre 2017

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

# Abstraction

Désormais, une vue au niveau applicatif, faisant abstraction des couches inférieures du réseau. Signifie que :

1. Deux ou plusieurs processus résidant sur des machines dans le réseau peuvent s'échanger des données par des messages qui sont acheminés de façon fiable en établissant une connexion (c'est-à-dire utilisant TCP) ou de façon moins moins fiable mais plus rapide, sans connexion (c'est-à-dire utilisant UDP).
2. Les machines sont identifiées par des adresses, les processus par des ports sur ces machines.

# Protocoles applicatifs

## Protocoles

Des processus qui échangent des messages doivent suivre des protocoles pour se ≪ comprendre ≫. Consistent en

- Des règles de syntaxe pour les messages
- De la sémantique pour l'interprétation des messages
- Des règles d'ordonnancement pour ces messages.

## Normalisation

- Il existe autant de protocoles que de programmes !
- La normalisation de ces protocoles assure l'interopérabilité.

## Architecture

- Client/Serveur ou
- Pair à pair.

# Architecture Client/Serveur I

## Les clients

- Tout processus qui demande un service sur le réseau à un autre processus (le serveur).
- Par extension, tout logiciel ou partie de logiciel qui lance ces processus.

## Les serveurs

- Répondre aux demandes de service des clients.
- Journaliser : consigner l'activité du service dans un fichier
- Persistence : assurer la continuité du dialogue avec un client.
- Autres : gestion de la charge, des erreurs, des pannes,. . .

# Architecture Client/Serveur II

## Entre les deux

- Infrastructure réseau,
- Proxys assurant le rôle de cache ou de filtre, de journalisation.
- Passerelles logicielles qui peuvent modifier le protocole.

## Remarques

- Clients et serveurs peuvent parfois se trouver sur une même machine.
- Services à destination d'un usager humain ou d'un autre processus.
- L'appel d'un service peut mettre en œuvre d'autres échanges client/serveurs.
- Aujourd'hui le navigateur semble devenir le client universel pour de nombreux services à destination d'un usager humain.

# Architecture Client/Serveur III

## Web

- Le serveur web distribue des pages web à des clients : les navigateurs, utilisés par des gens,
- mais aussi à des *crawlers* (robots) qui participent à l'indexation du web par les moteurs de recherche.
- Le serveur web peut interroger un serveur de messagerie pour obtenir les messages d'un utilisateur utilisant un webmail

## La suite

Petit parcours de quelques services, protocoles, normes.

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

# Stockage de fichiers

## Objectifs

- Mettre à disposition un espace de stockage.
- Collaborer sur ces données partagées
- Accéder à ces données depuis n'importe quel poste, de façon sûre (authentification, ...).
- Transparence : voir l'espace comme un simple disque local

## Différentes visions, conditions et contraintes

- Réseau local / Grand réseau
- Stockage / archive / sauvegarde / travail quotidien
- Accès rapide / récupération / collaboration / sécurité

# Sur le réseau local I

## Disque partagé

- Deux protocoles principaux.
- NFS (unix) : ancien et éprouvé.
- CiFS (Windows, disque partagé). Samba est une implantation sous unix.
- Autres : MacOSx Apple Filing Protocol (AFP), ...

## Matériel

- Architecture client/serveur : serveur de fichiers.
- Les NAS (Network-Attached Storage) fournissent en général tous les protocoles.
- Équipement en RAID : disques redondants et dupliquant l'information.

# Sur le réseau local II

## Niveau organisationnel

- Souvent désorganisé en travail collaboratif
- Adressage (identification des ressources, fichiers, répertoires) selon deux mondes
- Windows
  - Adressage local LFS Local File System `volume:\chemin\fichier` avec connexion d'un lecteur réseau
  - UNC Uniform Naming Convention `\\serveur\volume\chemin\fichier`
- Posix : Les autres (Unix/Apple/Linux)
  - Adressage local `/chemin/fichier` avec une opération de montage préalable.
  - `serveur:/chemin/fichier`
  - URLs avec le nom du service (ou protocole) `protocole://serveur/chemin/fichier` (Exemple `smb://machine/chemin/fichier` pour samba).

# Sur un grand réseau I

## Différents protocoles

- FTP : File Transfert Protocol. Le plus répandu, maintenant en désuétude. Messages non cryptés, sécurisation difficile, difficulté à traverser les NAT.
- scp / sftp : Alternative sécurisée reposant sur ssh (The secure shell). Non grand public à cause de la difficulté de compréhension de la cryptographie.
- rsync : Synchronisation de fichiers entre plusieurs machines. Très efficace et sûr et bien adapté à la sauvegarde. (Évolution dans dejadup).
- WebDAV Web-based Distributed Authoring and Versioning. Extension de HTTP qui permet une plus grande manipulation de ressources sur un serveur distant. Orienté collaboration.

# Sur un grand réseau II

## Cloud

- Services Web nombreux (Dropbox, UbuntuOne, Boxnet, GoogleDrive...)
- Protocoles souvent propriétaires.
- Alternatives libres : Owncloud, Sparkleshare permettant l'auto-hébergement.

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

# Adressage de ressources

## Ressources

*[. . .] toute chose ou entité susceptible d'être identifiée, nommée, [localisée] manipulée à travers ses représentations, par quelque moyen que ce soit, sur le Web en général ou dans n'importe quel système d'information utilisant les technologies du Web.*

*Wikipedia*

- Exemples : document HTML, image, vidéo, son, boîte aux lettres, . . .
- Identifiée : URI uniform resource identifier
- Nommée : URN uniform resource name (désigne une ressource où qu'elle soit, indépendamment de son lieu, même si elle a été déplacée).
- Localisée : URL uniform resource locator
- URI = URN ∪ URL ∪ . . .

# Les URLs

## Caractéristiques

- Adresse d'une ressource.
- Indique aussi comment y accéder.
- Peut être
  - statique : enregistrée à une certaine place dans dans un fichier, dans une arborescence.
  - dynamique : calculée par un programme à partir de paramètres.
- Peut être
  - absolue : indépendante « du lieu » d'où l'adresse est émise
  - relative : dépendante « du lieu » d'où l'adresse est émise
- Peut nécessiter une authentification.

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.

## Exemple

`http`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.

## Exemple

`http://`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom

## Exemple

`http://sam`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom

## Exemple

`http://sam:`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource

## Exemple

`http://sam:grat`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource

## Exemple

`http://sam:grat@`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine

## Exemple

`http://sam:grat@www`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.

## Exemple

`http://sam:grat@www.grappa.univ-lille3.fr`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.

## Exemple

`http://sam:grat@www.grappa.univ-lille3.fr:`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur

## Exemple

`http://sam:grat@www.grappa.univ-lille3.fr:80`

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms  des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms  des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms et valeurs des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=coucou
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms et valeurs des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=coucou&
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms et valeurs des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=coucou&p=machin
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms et valeurs des paramètres d'accès à la ressource.

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=coucou&p=machin#
```

# Composition d'une URL : cas HTTP

- Protocole : Par quel moyen interpréter, résoudre l'adresse et transmettre les données.
- Le nom et le mot de passe pour accéder à la ressource
- Serveur : le serveur où se trouve la ressource. Un nom de machine et son domaine FQDN.
- Le port de l'application sur le serveur
- Le chemin vers la ressource
- Le nom du document/programme contenant/calculant la ressource
- Les noms et valeurs des paramètres d'accès à la ressource.
- L'emplacement dans le document

## Exemple

```
http://sam:grat@www.grappa.univ-lille3.fr:80/~staworko/exemple
/complique/de/programme.php?param=coucou&p=machin#ici
```

# Les URNs

## Objectifs

- Définir un nom normaliser pour identifier et non localiser une ressource.
- L'idée est d'associer une correspondance entre URNs et localisations.
- Finalement assez peu utilisé en pratique.

## La forme d'une URN

`urn:NID:NSS`

- NID (Namespace Identifier). Espace de noms enregistré dans l'IANA.
- NSS (Namespace specific String) Grossièrement le nom dans cet espace.

## Exemples

- `URN:ISBN:...`
- `URN:ISSN:...`

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

## Communication sur internet

- Typologie des contenus
- HTTP : distribution de pages hypertexte
- SMTP/POP/IMAP : échange de courriers électroniques
- FTP : échange de fichiers
- WebDAV : échange avancé de fichiers
- XMPP : communication instantanée
- SIP, H323, RTP,. . . : pour la VOIP, le streaming,. . .

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
   - HTTP
   - FTP
   - SMTP/IMAP/POP
6. Le navigateur WEB
7. Dans les nuages

# Les content-type

## Objectifs

- Méta-donnée principale donnant le type d'un contenu.
- Permet de signaler aux lecteurs de la donnée (clients d'un service) la nature de ce qu'ils reçoivent
- Permet d'associer une application particulière en fonction de ce type.
- Apparue d'abord pour les contenus des emails : attachements, etc.
- Étendue ensuite à de nombreux protocoles dont HTTP.
- Standardisation dans Multipurpose Internet Mail Extensions (MIME).

# MIME

## Les entête comprennent

- La version de MIME : `MIME-Version`
- Typage : `Content-type` sous la forme type/sous-type. Voir par exemple le fichier `/etc/mime.types`. Peut spécifier un contenu composé de plusieurs parties (`multipart/mixed`).
- Encodage du transfert à travers le réseau : `Content-Transfer-Encoding`. Par défaut 7 bits, et des encodages en 7 bits par les algorithmes `quoted-printable` ou `base64`. Pour les serveurs modernes, les transferts 8bits sont acceptés.
- Identification : `Content-ID` numéro d'identification d'une partie.
- Structuration (parfois considéré comme désuet) : `Content-Disposition`. Le contenu doit être interprété comme un document attaché (`attachment`) ou affiché dans le client (`inline`).

# Example MIME

```
...
MIME-Version: 1.0
Content-Type: multipart/mixed;
   boundary="----=_NextPart_000_0005_01CCE650.9AC585B0"
...
This is a multi-part message in MIME format.

------=_NextPart_000_0005_01CCE650.9AC585B0
Content-Type: multipart/alternative;
   boundary="----=_NextPart_001_0006_01CCE650.9ACA40A0"
...
------=_NextPart_001_0006_01CCE650.9ACA40A0
Content-Type: text/plain; charset="Windows-1252"
Content-Transfer-Encoding: quoted-printable

Bonjour =E0 tous,
...
```

# Example MIME

```
Bonjour =E0 tous,
...
------=_NextPart_001_0006_01CCE650.9ACA40A0
Content-Type: text/html; charset="Windows-1252"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META content=3D"text/html; charset=3Dwindows-1252" =
http-equiv=3DContent-Type>
...
------=_NextPart_000_0005_01CCE650.9AC585B0
Content-Type: image/jpeg; name="IMG.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="IMG.jpg"

/9j/4AAQSkZJRgABAAEBLAEsAAD/4SQoRXhpZgAASUkqAAgAAAAKAA8BAgAGAAAAhgA
...
```

# Plan

5. Typologie des contenus
   - HTTP
   - FTP
   - SMTP/IMAP/POP

# HTTP I

## Hypertext Transfert Protocol

L'invention de Sir Tim Berners Lee.

- Clients : navigateurs (IE, Firefox, Chrome Safari, Opera, Konqueror, lynx, ...) voir un diagramme.
- Clients : robots, crawlers, aspirateurs, ...
- Serveurs : Apache, IIS, ...

# HTTP II

## Principes

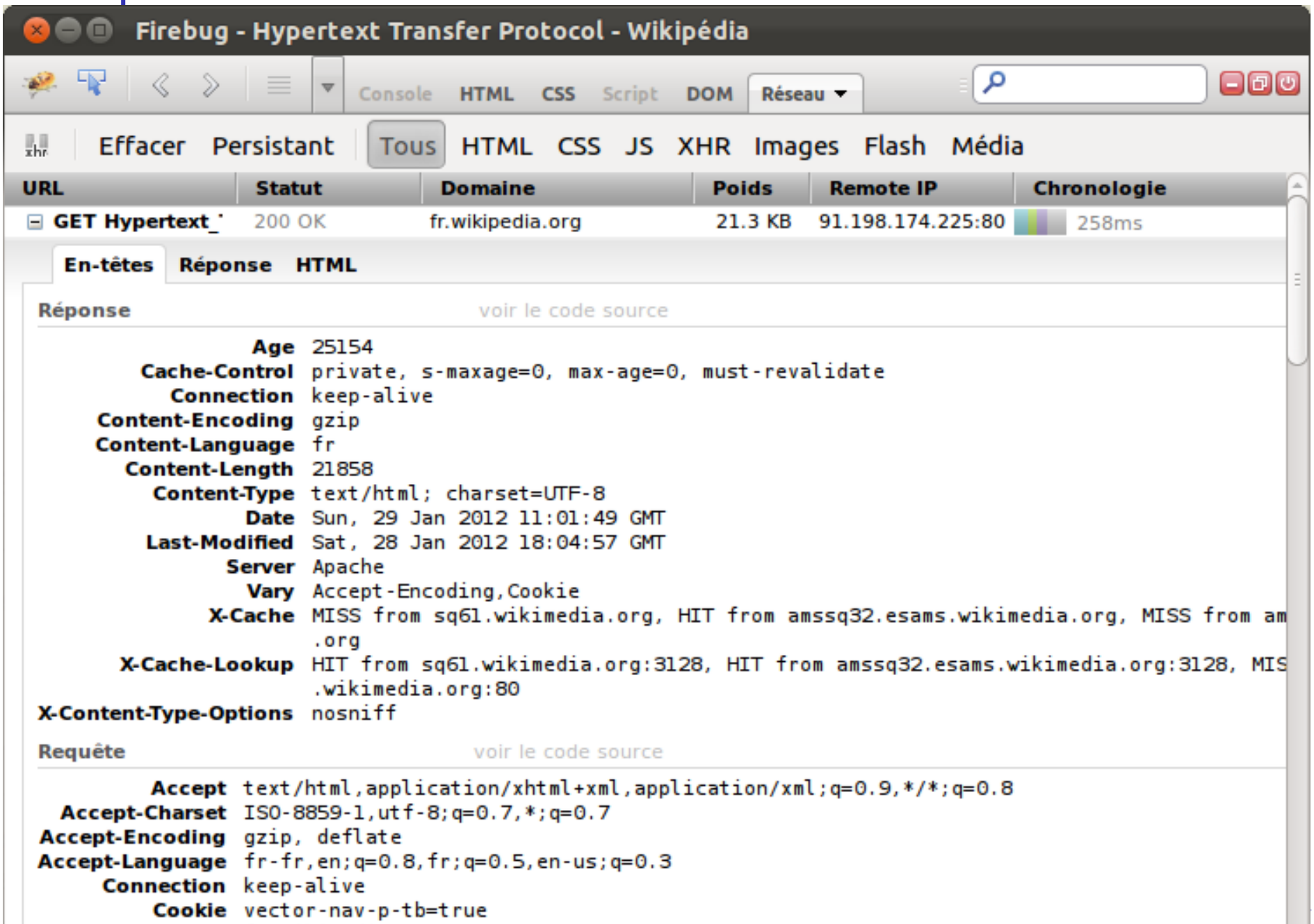
- Les messages sont composés d'un entête et d'un corps séparés par une ligne blanche.
- Les entêtes comprennent des méta-données dont par exemple les content-type (types de contenus) et encoding (encodages) du corps, le user-agent (client), le referer (d'où il vient), le host (la machine demandée), ...
- Les messages sont des requêtes et réponses à ces requêtes.
- Les requêtes sont des commandes au serveur
- Les réponses aux requêtes comportent un code de retour. (404 : non trouvé, 200 Ok, 301 déplacé,...)

# HTTP III

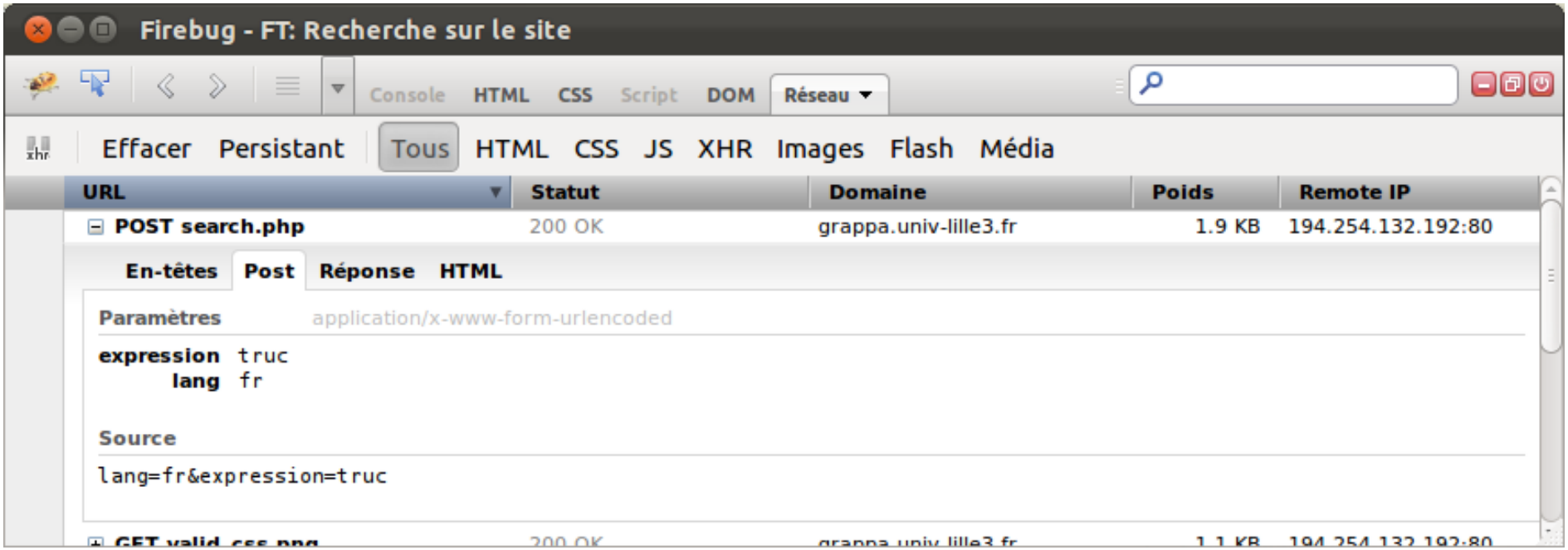
## Principales commandes

- GET : Permet d'obtenir une ressource. Utilisé dans la barre d'URL du navigateur.
- POST : Permet d'envoyer des données au serveur. Utilisé dans certains formulaires HTML.
- HEAD : Demander les méta-données uniquement.

# Exemple de GET



# Exemple de POST



# Exemple de petit client

```
user@machine:~$ host www.google.fr
www.google.fr is an alias for www-cctld.l.google.com.
www-cctld.l.google.com has address 173.194.66.94
user@machine:~$ python
>>> from socket import *
>>> s= socket(AF_INET,SOCK_STREAM)
>>> c = s.connect(("173.194.66.94",80))
>>> s.send("GET /\n\n")
>>> d = s.recv(1000)
```

# Côté Serveur

## Observation des logs

- Les contenus des journaux sont hautement paramétrables.
- Ils peuvent conserver les informations dans les entêtes des échanges `GET`, `POST`, etc.
- Ils peuvent aussi ajouter des informations provenant de l'environnement (temps,...)

## Extrait logs d'apache

```
172.20.110.253 - - [10/Feb/2012:16:43:56 +0100]
    "GET /~jmary/wordpress/ HTTP/1.0" 200 2102
    "http://masterid.pedago.local/~jmary/?C=N;O=A"
    "Mozilla/5.0 (Windows NT 5.1; rv:9.0.1) Gecko/20100101 Fi
127.0.0.1 - - [10/Feb/2012:17:00:01 +0100]
     "GET /drupal6/cron.php HTTP/1.1" 200 616 "-"
     "curl/7.18.2 (i486-pc-linux-gnu) libcurl/7.18.2
            OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.
```

# Côté Serveur

## Charge, pannes

- Plusieurs processus sont lancés pour répondre à un nombre important de requêtes

## Table des processus sous unix : processsus apache

```
26664 ?        S<s    0:07 /usr/sbin/apache2 -k start
22419 ?        S<     0:49  \_ /usr/sbin/apache2 -k start
24324 ?        S<     0:42  \_ /usr/sbin/apache2 -k start
26633 ?        S<     0:29  \_ /usr/sbin/apache2 -k start
31101 ?        S<     0:27  \_ /usr/sbin/apache2 -k start
32632 ?        S<     0:22  \_ /usr/sbin/apache2 -k start
32637 ?        S<     0:22  \_ /usr/sbin/apache2 -k start
  704 ?        S<     0:17  \_ /usr/sbin/apache2 -k start
  918 ?        S<     0:16  \_ /usr/sbin/apache2 -k start
 2053 ?        S<     0:08  \_ /usr/sbin/apache2 -k start
 2284 ?        S<     0:09  \_ /usr/sbin/apache2 -k start
```

# Côté Serveur

## Association entre adresses et ressources

- La configuration du serveur contient ces associations et autres droits et autorisations
- Elle peut inclure aussi des règles de réécriture des adresses : simplifier les adresse, masquer les appels de programmes,...

```
RewriteEngine on
RewriteRule ^page/([^/\.]+)/?$ programme.php?p=$1 [L]

Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Allow from all
</Directory>
```

# Les deux méthodes pour un formulaire

## Les formulaires HTML

- Permettent d'écrire des ≪ écrans de saisie ≫ en HTML.
- Un formulaire est associé à une action
- L'action est le plus souvent une URL
- L'action peut être contrôlée par un script en javascript.

## Description en HTML

- Balise `FORM` et à l'intérieur des balises spécifiques pour les zones de saisie :
- `INPUT`
- `TEXTAREA`
- `OPTIONS`
- ... Voir la recommandation du W3C

# Exemples

## Formulaire d'interrogation de google

- Le formulaire doit comprendre une zone de texte avec un nom `q`
- La balise `action` a `http://www.google.fr/search` comme `action`.
- Essayez avec l'attribut `Method` à `POST` ou `GET`.

## Formulaires

- Faire un formulaire qui demande un nom et une couleur préférée parmi rouge, vert et bleu, un bouton d'envoi et un bouton de réinitilisation du formulaire. Utiliser des labels.
- Ajouter un choix de régime végétarien oui ou non.
- Ajouter une zone de texte pour laisser un message dans le livre d'or
- Remplacer le bouton d'envoi par une image cliquable.
- Ajouter un bouton d'envoi de fichier et modifier la balise `FORM` en conséquence.

# Plan

5 Typologie des contenus
- HTTP
- FTP
- SMTP/IMAP/POP

# FTP : transfert de fichiers

## Objectifs

- Transfert de fichiers de (download) ou vers (upload) un serveur.

## Adresses des ressources

`ftp://utilisateur:motDePasse@Serveur:port_ftp`

Le plus souvent le port n'est pas mentionné (21 par défaut). En pratique :

`ftp://utilisateur@Serveur`

# FTP : transfert de fichiers

## Points positifs

- Adapté pour transférer des gros fichiers ou des arborescences.
- Très adapté/utilisé pour mettre à jour ou installer des pages web.

## Points négatifs

- Protocole peu sécurisé dans sa version de base.
- Nécessite l'ouverture de nombreux ports : pas apprécié par les gestionnaires de réseau.
- Transferts non cryptés.
- Parfois peu connu et donc pas trop accessible aux utilisateurs de base (à l'inverse des années 90).

# Le protocole

## Généralités

- Inventé dans les années 70.
- anonyme (parfois avec login *anonymous* et mot de passe une adresse email) ou authentifié.
- Permet de créer des répertoires, supprimer et déplacer des fichiers, ...
- Une connexion sur le port 21 en général pour le contrôle, une autre sur un autre port pour le transfert des données.
- Mode actif ou passif

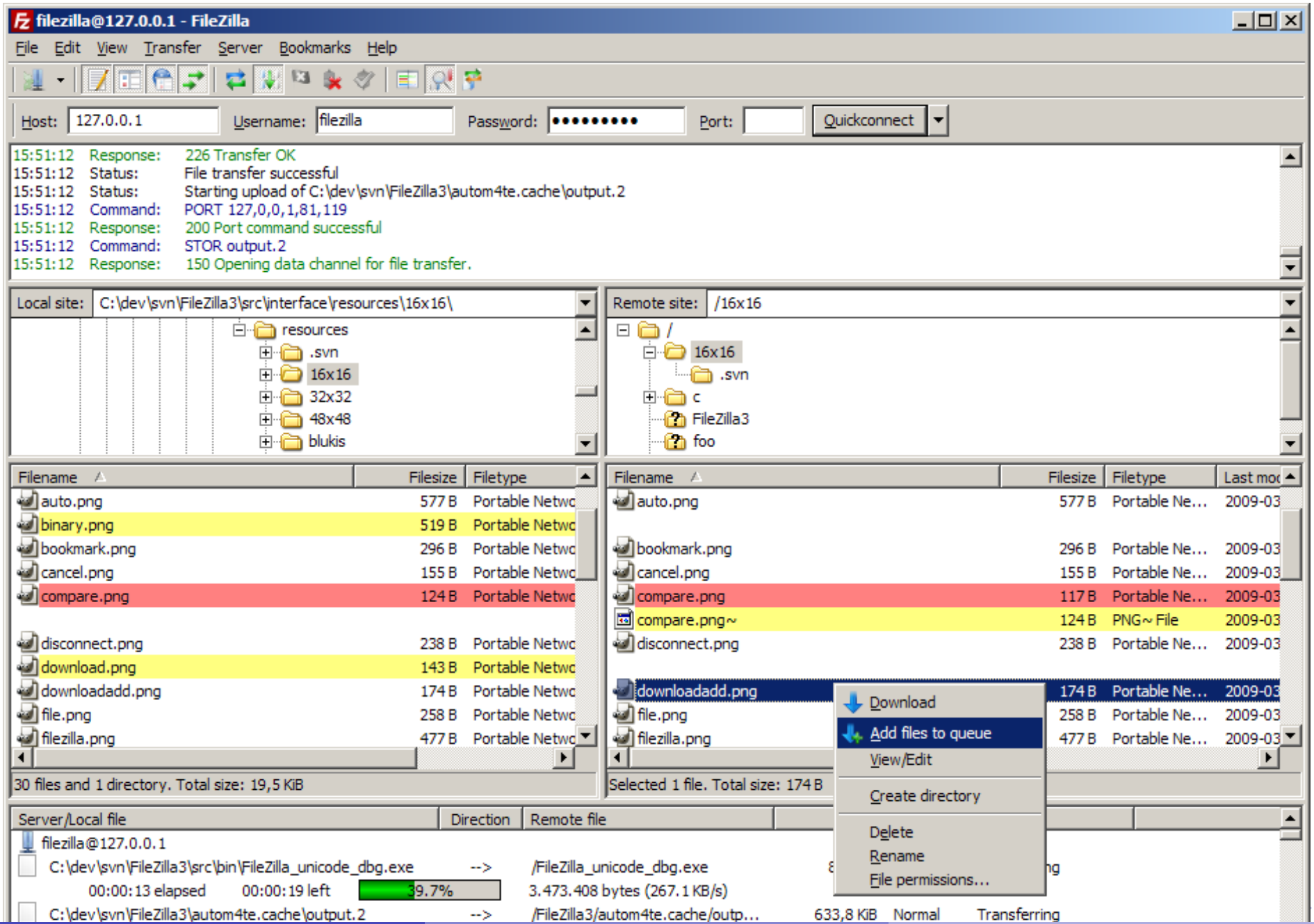
# Logiciels FTP

- En ligne de commandes `ftp`, `ncftp`

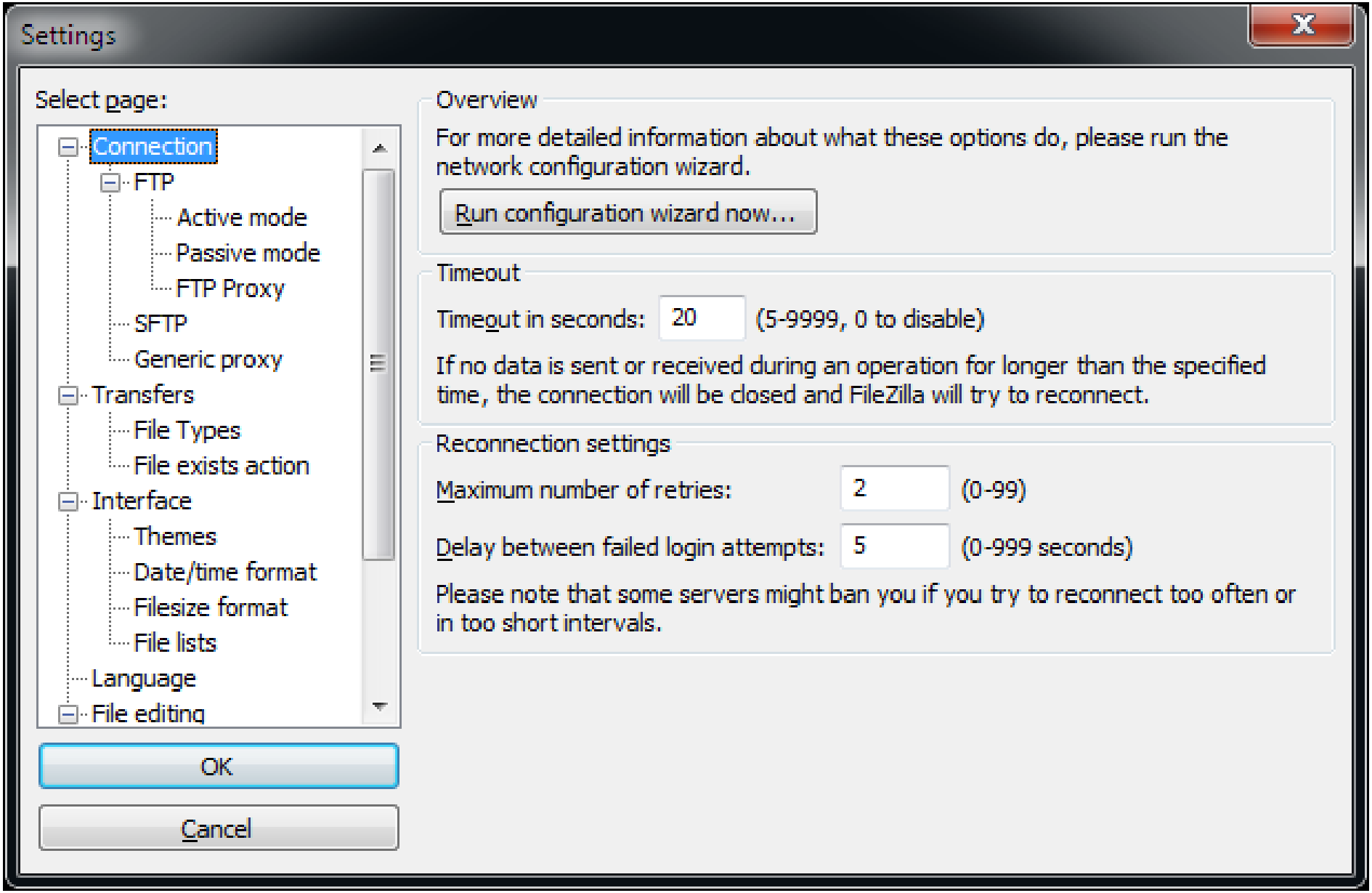
```
staworko@stokrotka:~$ ftp staworko.hd.free.fr
Connected to staworko.hd.free.fr.
220 Serveur de mise a jour des pages perso de Free.fr vers
Name (staworko.hd.free.fr:staworko):
```

- Avec interface graphique : `filezilla`, ...
- Intégré dans le navigateur : Natif pour le download, mais aussi sous forme d'extension (`FireFTP`).
- Intégré dans l'interface graphique système d'exploitation : sous linux Gnome (nautilus) ou KDE (konqueror).
- Intégré dans des langages de programmation comme PHP, ....

# Filezilla


filezilla@127.0.0.1 - FileZilla
File Edit View Transfer Server Bookmarks Help
Host: 127.0.0.1 Username: filezilla Password: ••••••••• Port: Quickconnect
15:51:12 Response: 226 Transfer OK
15:51:12 Status: File transfer successful
15:51:12 Status: Starting upload of C:\dev\svn\FileZilla3\autom4te.cache\output.2
15:51:12 Command: PORT 127,0,0,1,81,119
15:51:12 Response: 200 Port command successful
15:51:12 Command: STOR output.2
15:51:12 Response: 150 Opening data channel for file transfer.
Local site: C:\dev\svn\FileZilla3\src\interface\resources\16x16\
resources
.svn
16x16
32x32
48x48
blukis
Remote site: /16x16
/
16x16
.svn
c
FileZilla3
foo
Filename Filesize Filetype
auto.png 577 B Portable Netwo
binary.png 519 B Portable Netwo
bookmark.png 296 B Portable Netwo
cancel.png 155 B Portable Netwo
compare.png 124 B Portable Netwo
disconnect.png 238 B Portable Netwo
download.png 143 B Portable Netwo
downloadadd.png 174 B Portable Netwo
file.png 258 B Portable Netwo
filezilla.png 477 B Portable Netwo
30 files and 1 directory. Total size: 19,5 KiB
Filename Filesize Filetype Last mo
auto.png 577 B Portable Ne... 2009-03
bookmark.png 296 B Portable Ne... 2009-03
cancel.png 155 B Portable Ne... 2009-03
compare.png 117 B Portable Ne... 2009-03
compare.png~ 124 B PNG~ File 2009-03
disconnect.png 238 B Portable Ne... 2009-03
downloadadd.png 174 B Portable Ne... 2009-03
file.png 258 B Portable Ne... 2009-03
filezilla.png 477 B Portable Ne... 2009-03
Selected 1 file. Total size: 174 B
Download
Add files to queue
View/Edit
Create directory
Delete
Rename
File permissions...
Server/Local file Direction Remote file
filezilla@127.0.0.1
C:\dev\svn\FileZilla3\src\bin\FileZilla_unicode_dbg.exe --> /FileZilla_unicode_dbg.exe
00:00:13 elapsed 00:00:19 left 39.7% 3.473.408 bytes (267.1 KB/s)
C:\dev\svn\FileZilla3\autom4te.cache\output.2 --> /FileZilla3/autom4te.cache/outp... 633,8 KiB Normal Transferring

# Filezilla

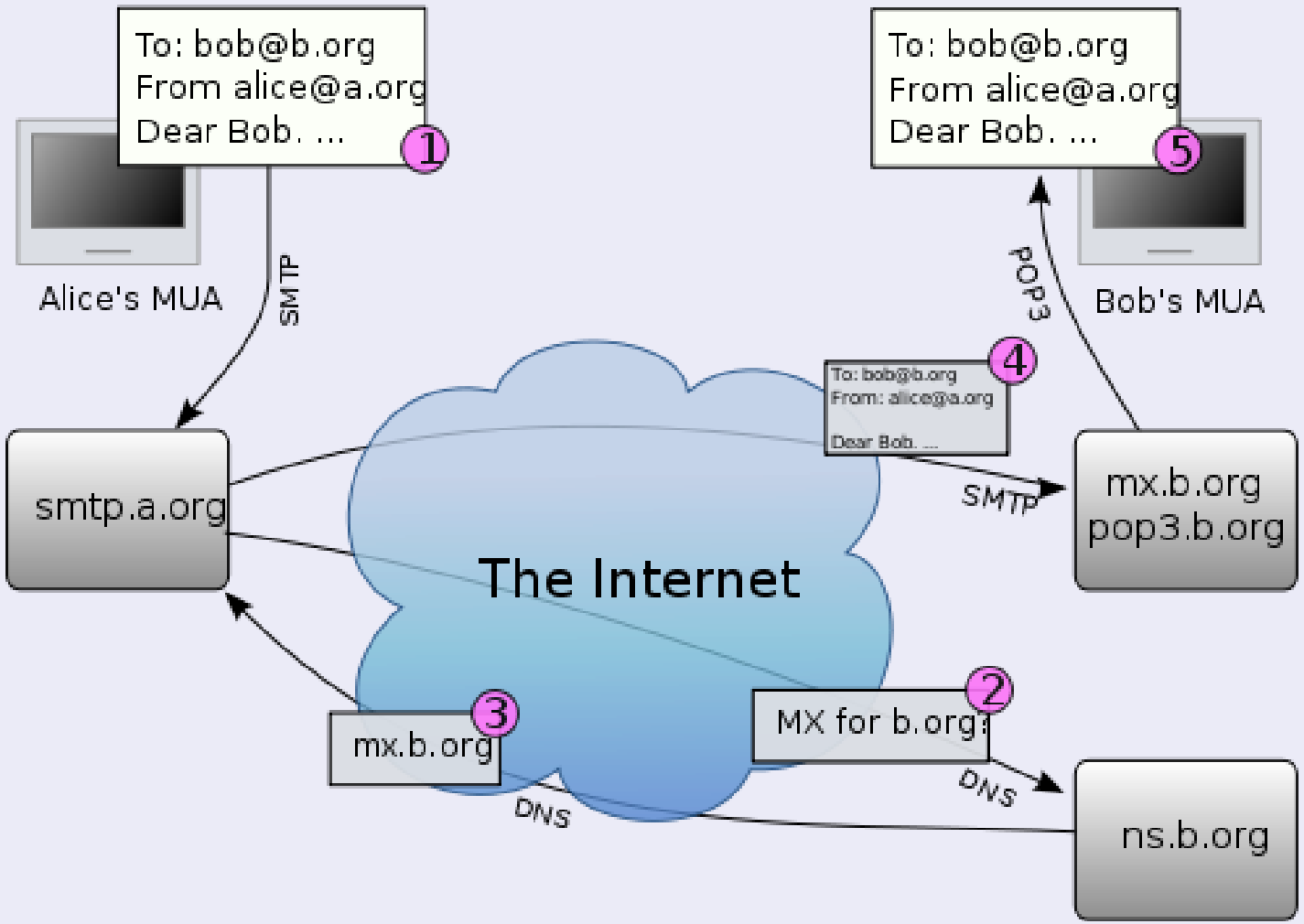

Settings
Select page:
Connection
FTP
Active mode
Passive mode
FTP Proxy
SFTP
Generic proxy
Transfers
File Types
File exists action
Interface
Themes
Date/time format
Filesize format
File lists
Language
File editing
OK
Cancel
Overview
For more detailed information about what these options do, please run the network configuration wizard.
Run configuration wizard now...
Timeout
Timeout in seconds: 20 (5-9999, 0 to disable)
If no data is sent or received during an operation for longer than the specified time, the connection will be closed and FileZilla will try to reconnect.
Reconnection settings
Maximum number of retries: 2 (0-99)
Delay between failed login attempts: 5 (0-999 seconds)
Please note that some servers might ban you if you try to reconnect too often or in too short intervals.

# Plan

5 Typologie des contenus
- HTTP
- FTP
- SMTP/IMAP/POP

# Le mail

## Globalement



# Le mail

## Les adresses

- `user@fqdn`. NB : les adressent ne font pas référence aux serveurs (machines) mais aux domaines.
- URI de la forme `mailto:user@fqdn`
- Les serveurs de DNS fournissent le nom et adresses des serveurs associés aux domaines.

## Plusieurs protocoles

- SMTP : pour l'envoi de messages
- POP ou IMAP : pour la réception
- La DNS : pour obtenir les serveurs de domaines.

# La forme des messages I

## Entête et corps

- Comme `HTTP`, les messages ont une entête, contenant des méta-données et un corps pour le contenu.
- Corps et entête sont séparés par une ligne blanche.

## Principales méta-données

- Certaines sont proposées à la saisie lors de la composition des messages : **From** : Émetteur ; **To** : destinataires ; **Cc** : Carbon Copy, destinataires en copie ; **Bcc** : Blind Carbon Copy, destinataires en copie cachée ; **Subject** : sujet.
- D'autres sont intégrées lors de l'envoi ou la transmission. Les méta-données MIME ; `Received` : liste des serveurs qui ont transmis le mail ; `X-mailer` nom du client de mail ; `Message-Id` pour les fils de discussion ; ...

# La forme des messages II

## Le corps

- Dépend des méta-données MIME.
- Contient les attachements.

## SMTP

- Protocole très simple, non sécurisé. Pas d'authentification. Port 25.
- Généralement des limitations contre le spam : exclure les connexions venant de machines hors du même réseau.
- Versions sécurisées SMTPS

# SMTP

```
staworko@fitis:~/ telnet smtp.free.fr 25
Trying 2a01:e0c:1:1599::10...
Connected to smtp.free.fr.
Escape character is '^]'.
220 smtp1-g21.free.fr ESMTP Postfix
HELO toto
250 smtp1-g21.free.fr
MAIL FROM: perenoel@lune.org
250 2.1.0 Ok
RCPT TO: xxx.xxx@xxx.xx
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Test

Allo la terre

.

250 2.0.0 Ok: queued as 158B4940126
QUIT
221 2.0.0 Bye
```

# Réception

## POP : Post Office Protocol

- Le plus simple, permet d'obtenir la liste des messages sur le serveur, les télécharger, les supprimer sur le serveur.
- Peu adapté aux pratiques modernes, par exemple nombreux clients différents pour un même utilisateur.
- Existe des versions sécurisées, mais initialement non sécurisé.

## IMAP

- Le plus courant aujourd'hui.
- Les messages, dossiers, etc restent sur le serveur.
- Permet une utilisation hors-ligne : clients et serveurs se synchronisent.

# Clients I

## Clients lourds

- Outlook, Thunderbird, mais il en existe des dizaines.
- Tendent vers des clients de Groupware (contacts, agendas, tâches, ...) et lecture de flux RSS, newsgroups,...
- Permet de lire son mail en mode déconnecté
- Permet d'intégrer de nombreuses boîtes aux lettres (différentes adresses email).
- Permet de gérer plusieurs identités.
- Permet de filtrer les messages à leur arrivée.
- Permet un traitement des indésirables personnalisé.

# Clients II

## Clients légers

- Tous les webmails.
- Tendance qui s'impose grâce aux technologies du web (Ajax).
- Rattrapent les clients lourds en termes de fonctionnalité.
- Peuvent être aussi installés en réseau d'entreprise.

# Exemples

# Intégration d'outils

## Groupware

- Messagerie classique
- Calendrier
- Tâches
- Partage de documents
- Messagerie instantanée
- Contacts

## Logiciels

- Exchange (Microsoft)
- Zimbra (VMWare, Opensource)
- SoGo (OpenSource)

# Usages

## Netiquette

- Éviter les messages avec une longue liste de destinataires (spam)
- Éviter les messages trop gros.
- Éviter les signatures trop longues
- Éviter les hoaxes.

## Difficultés organisationnelles

- Excès de messages, difficultés de traitement, hiérarchisation des priorités,
- Burning out, addiction,
- Règles parfois données : éviter le trop grand nombre de CC, les accusés de réception inutiles...
- Mais reste un très bon outil quand il est utilisé correctement.

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

# Le navigateur

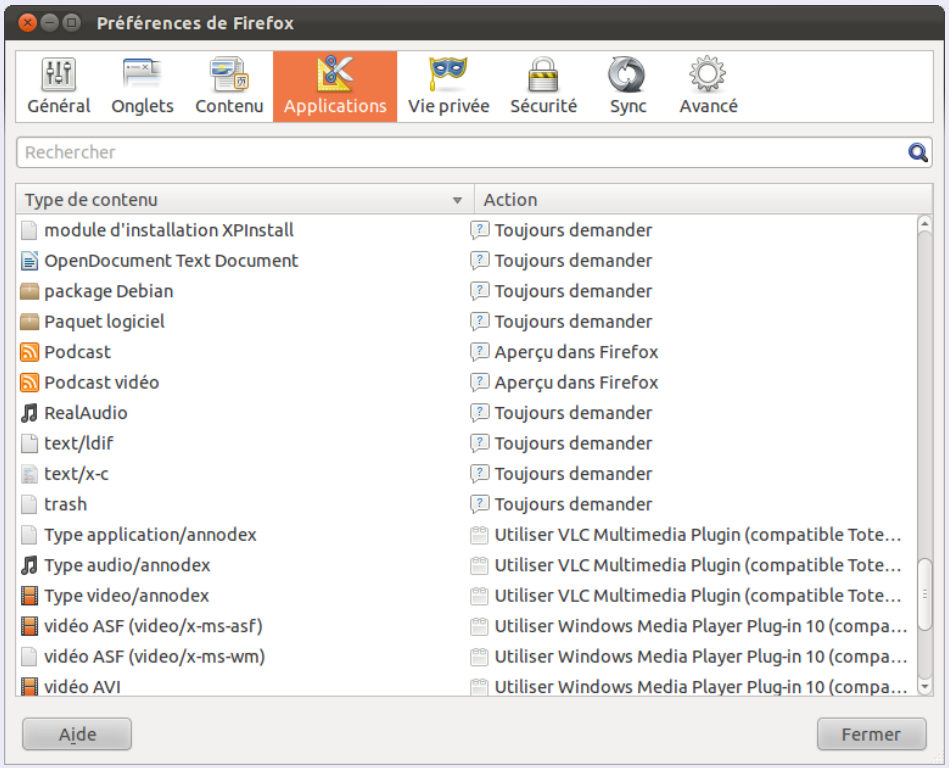
## Objectifs

- De l'outil de navigation dans un réseau de pages hypertexte, ...
- au *client universel* pour les applications modernes, ...
- à une *machine virtuelle* pour le développement d'applications :
  - Chrome devient Chrome OS
  - La plateforme XUL de firefox avec ses nombreuses extensions et applications indépendantes : Songbird, Miro, Uploadr de Flikr,...

# Types mimes et applications

## Objectifs

- Configurer le comportement du navigateur en fonction du type de contenu (`content-type`) qu'il reçoit de la part du serveur.
- Lancement d'application dédiées, intégration dans le navigateur,...



# Cache et historique

## Objectifs du cache

- À chaque donnée téléchargée est associée une date d'expiration (date limite de validité).
- Le cache conserve les données et évite de télécharger à nouveau des données si elles ne sont pas périmées.
- Le cache permet donc d'accélerer les temps d'affichage.

## Objectifs de l'historique

- L'historique est basée sur l'activité de l'utilisateur
  - un ensemble de page (ensemble de données) visitées,
  - un ensemble de recherches effectuées,
  - de sessions ouvertes,...
- La conservation de l'historique permet de :
  - naviguer dans l'historique : boutons suivant, précédent
  - ré-afficher un onglet fermé,
  - restaurer une session au démarrage du navigateur,...

# Les cookies I

## Objectifs

- Permet de stocker sur le client l'état du service et de renvoyer cet état au serveur quand c'est nécessaire (à sa demande)
- Utilisé par exemple pour les mécanismes d'authentification, la personnalisation des services, le suivi (tracking), ...

## Caractéristiques

- Les cookies sont stockés sur le client
- Ils peuvent être effacés par l'utilisateur
- Ils ont une durée de validité (expiration) qui peut être très variable : éphémère ou persistant.
  - pour un échange, une visite de site : cookie sans date de validité, existant jusqu'à la fermeture du navigateur
  - pour plusieur jours, semaines ou plus,... : cookie avec date de validité. Il est supprimé après cette date.

# Les cookies II

## Définition

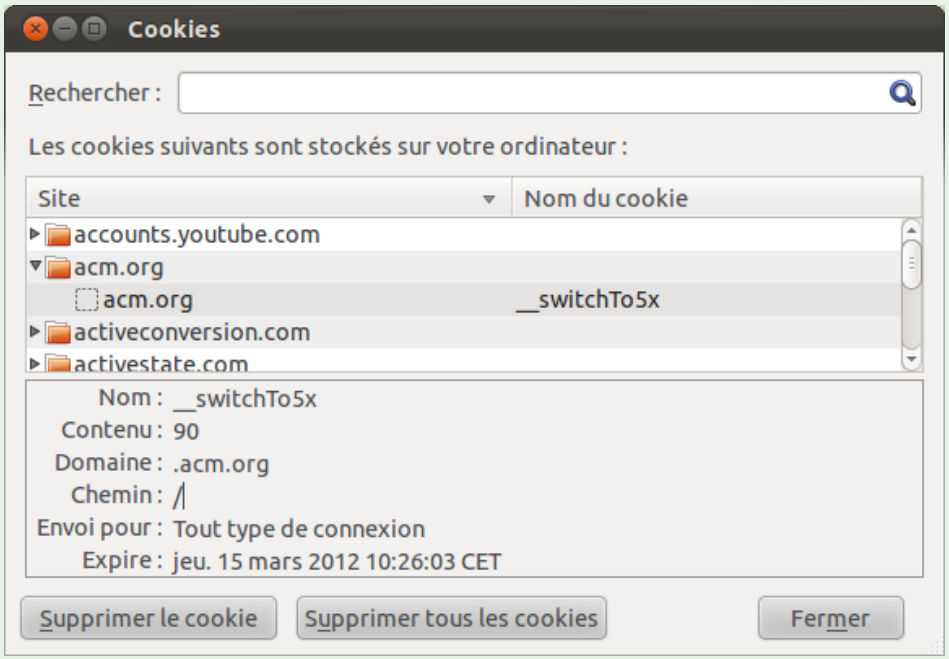
- Donnée avec quelques méta-données associées à un service.
- La valeur de la donnée est atomique : un texte ou une valeur numérique

## Méta-données

- Nom du cookie (nom de la donnée)
- Domaine du serveur qui fournit le service associé.
- Chemin sur ce serveur. Permet de déterminer en quelques sortes l'application serveur qui demande ce cookie.
- Expiration.
- Type de connexion : sécurisée ou non.

# Les cookies III

## Exemple



# Les cookies IV

## Techniquement

- Il est échangé dans l'entête des messages HTTP.
- Le serveur envoie une instruction : `Set-Cookie: name=value` et suivi de méta-données
- Le client envoie un champ `Cookie: name=value` pour indiquer la valeur du cookie au serveur quand le domaine et le chemin correspondent.
- La correspondance : même domaine ou sous-domaine du domaine du cookie ; même chemin ou chemin plus spécifique.

# Les cookies V

## Démonstration

- Consulter la page `http://www.grappa.univ-lille3.fr/~tommasi/cours/reseau/cookie.php`
- Recharger plusieurs fois la page
- Regarder les méta-données du cookie déposé.
- Supprimer le cookie et essayer à nouveau de charger la page.

# Les cookies VI

## Vie privée

- Le cookie permet de tracer les internautes.
- Les cookies persistants augmentent cette capacité de traçabilité sur plusieurs jours, voire plusieurs mois.
- Mais la traçabilité permet de rendre service.
- Une page web peut être composée par des différentes sources de domaines différents (balises `src` dans les images, les cadres, ... : chacune peut poser des cookies.
- Un script en `JavaScript` peut aussi lire des cookies : menace de vol (HTTPOnly évite cela).

# Les cookies VII

## Comment réagir ?

- Doit-on autoriser les cookies ?
- Doit-on autoriser les cookies persistants ?
- Doit-on autoriser des cookies de site différents de celui affiché sur la barre d'url ?
- Doit-on autoriser des cookies de domaine très large (de premier niveau par exemple) ?

# Les sessions I

## Constat

- Le navigateur est devenu un client universel pour utiliser des services à travers Internet.
- Les services évolués demandent de nombreux échanges, faisant évoluer l'état de la transaction entre client et serveur : session.
- Ces échanges sont asynchrones, peuvent être interrompus pour un temps parfois assez long
- L'état courant d'une transaction (la session), peut être qualifié par de nombreuses données : données d'authentification et d'identification, articles dans un panier, mémorisation de recherches effectuées, préférences d'utilisation, etc.
- L'état complet ne peut être échangé uniquement par le mécanisme des cookies.

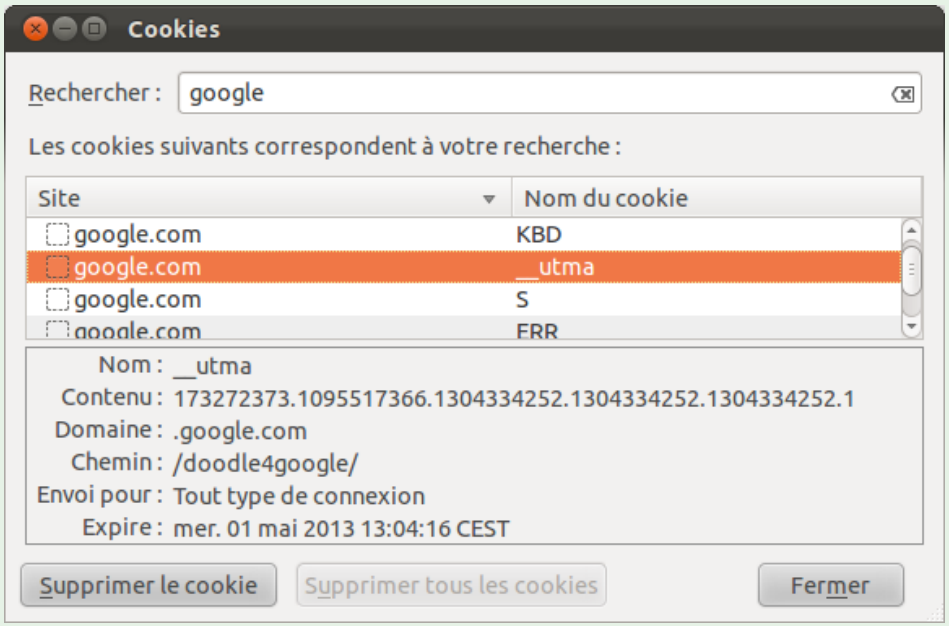
# Les sessions II

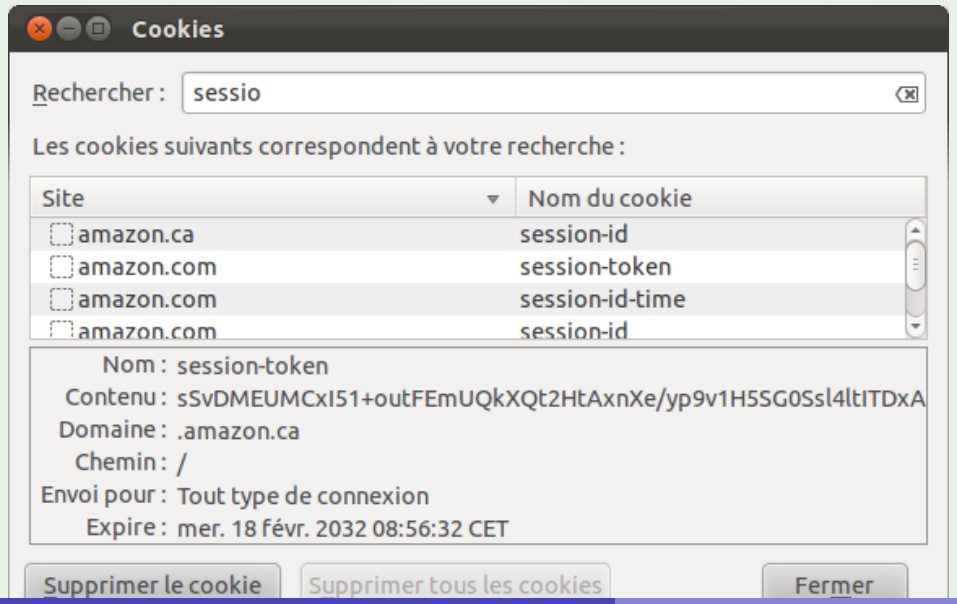
## Solutions

- Les données de session sont stockées sur le serveur
- Un numéro unique (numéro de session) permet d'associer un client avec ces données de session.
- Le numéro de session est échangé soit par un cookie (le plus souvent), soit dans un `GET`, donc dans l'URL.

# Les sessions III

## Deux cookies de session



# Exemple de session

## Essayez :

- `http://www.grappa.univ-lille3.fr/~tommasi/cours/reseau/session.php`
- Constater que le cookie de session ne contient pas la valeur du nom saisit dans le formulaire.
- Si on saisit `Joe` dans le formulaire on trouvera sur le serveur un fichier avec un nom unique de la forme (par exemple) `/var/lib/php5/sess-664c26cc87e83c5ce5d0d7357be3bdcd` contenant (par exemple, selon le programme serveur) : `nom|s:3:"Joe";`
- Détruire le cookie de session pour vérifier le fonctionnement.

# Signets I

## Objectifs

Marques-page, favoris, raccourcis internet, bookmarks, hotlists, . . .

- Garder en mémoire des URLs.
- Accès rapide à des sites fréquemment visités.

## Solutions

- Dans le navigateur.
- Utilisation d'une application web spécifique (social bookmarking).

## Différents types de signets sur le navigateur

- Signets dynamiques (live bookmarks) : titres de flux RSS
- Signets javascript (bookmarklets) : permettent d'effectuer une action.

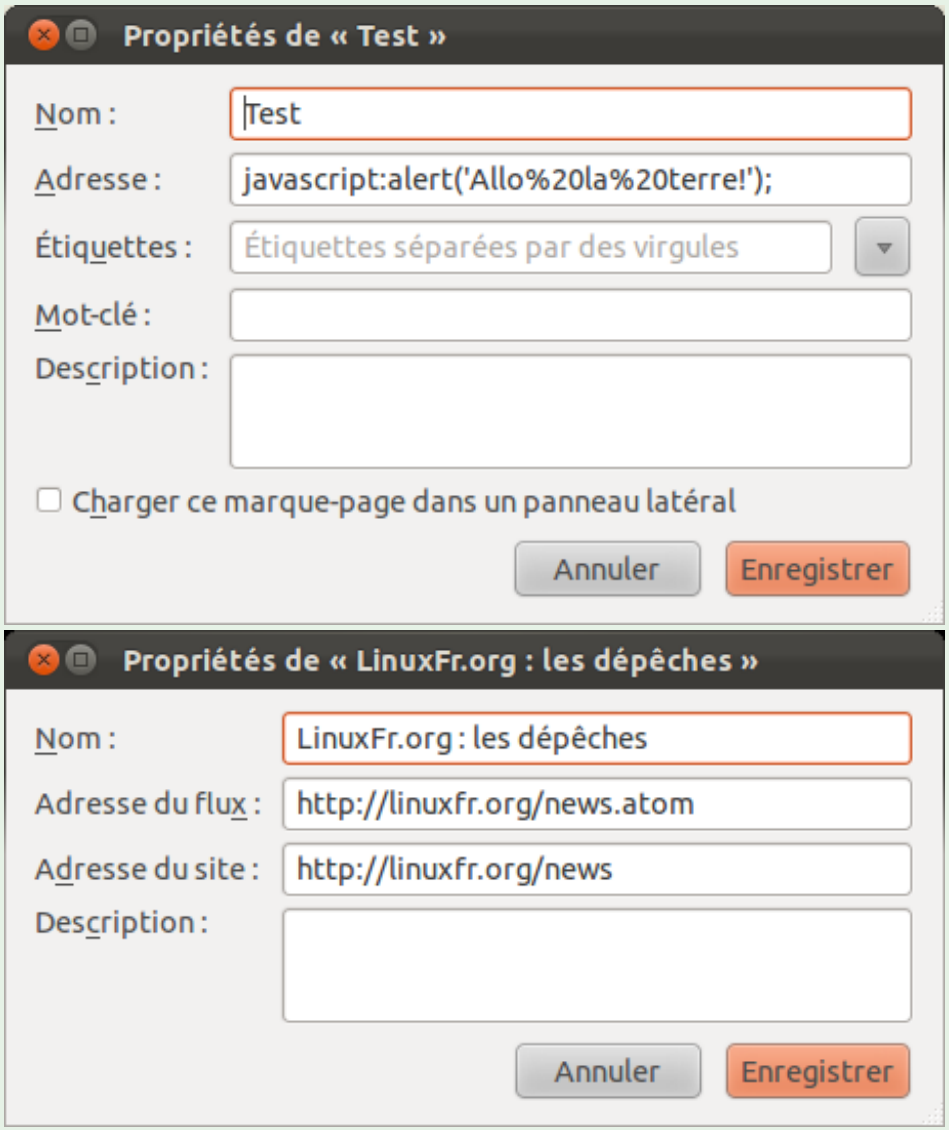
# Signets II

## Organisation dans le navigateur (firefox)

- Rangement par dossier
- Association d'étiquettes
- Affichage de signets, dossiers ou étiquettes sur la barre de signets.

# Signets III

## Essayez de reproduire ces signets



# Vie privée I

## Que penser si des tiers : programmes ou utilisateurs

- consultent l'historique ou le cache
- examinent les cookies
- consultent les mots de passes stockés ou les formulaires pré-remplis,
- voient les signets

## Pour les utilisateurs : navigation privée

Mode de navigation où sont désactivés :

- l'historique,
- le remplissage automatique des formulaires

# Vie privée II

## Pour les applications

- Choisir un bon navigateur qui implante des fonctions de contrôle
- Utiliser des extensions spécifiques (*e.g.* noscript).

# Extensions

## Quelques extensions importantes

- Scrapbook : sauvegarde de pages pour une consultation hors ligne.
- Zotero : gestionnaire de données bibliographiques.
- Itsalltext : permet de saisir de zones de texte dans un formulaire grâce à un éditeur externe (moins de risque de perte de données).
- AdBlock : suppression de publicités.
- NoScript : interdiction très paramétrée de l'usage de Javascript.
- Greasemonkey : automatisation de tâches répétitives dans le navigateur.
- Firebug, WebDeveloppper UserAgent Switcher,... outils de développement pour le webmaster.

# Plan


1. Introduction
2. Service de fichiers
3. Adressage de ressources : URI, URL, URN,...
4. Principaux protocoles
5. Typologie des contenus
6. Le navigateur WEB
7. Dans les nuages

# Dans les nuages

## Définition

- Cloud computing en anglais.
- Déporter sur des serveurs distants tout ou partie du système automatisé d'informations : données, applications...
- C'est une forme d'externalisation.
- Nuage pour traduire le fait qu'on ne sait pas où données et applications se trouvent physiquement.
- Accès par un navigateur web le plus souvent.

## Origines

- Idées très anciennes (années 60)
- Réalisation récente : Amazon vers 2006. Motivée par une refonte technologique et la location de ressources non utilisées.
- Clouds privés dès 2008 grâce à Eucalyptus.

# Trois couches de services I

## IaaS

- Infrastructure as a Service.
- On loue des ressources : capacité de calcul, capacité de stockage.
- Souvent une location de machine (physique ou virtuelle)
- Le locataire est responsable de tout : installation et maintenance d'un système, puis des programmes, etc.

# Trois couches de services II

## PaaS

- Platform as a Service.
- On loue des services de base pour installer ensuite des applications finales pour l'utilisateur
- Exemple : machine plus le trio Apache/PHP/MySQL pour réaliser un serveur Web.
- Le locataire est déchargé de la maintenance et l'installation de la plateforme.
- Il ne s'occupe que des applications.
- Le dimensionnement est souvent réalisé automatiquement, à la demande (ajout de disque, mémoire, bande passante, ...) pour répondre à la charge.

# Trois couches de services III

## SaaS

- Software as a Service
- Location d'une solution logicielle complète : suite bureautique, solution de messagerie, CMS, ECM, ....
- Parfois juste un seul service : authentification, commentaires, ou un assemblage par mashups...
- Le locataire ne s'occupe que de l'administration de cette application.
- En général un prix en fonction (entre autres) du nombre d'utilisateurs.

# Cloud privés et publics

## Plusieurs modèle de déploiement

- Clouds publics : service offert à tous.
- Clouds communautaires : pour un secteur d'activité (droit, santé,...)
- Clouds privés : réservés à une seule entreprise pour des raisons de sécurité essentiellement.
- Clouds hybrides mélant parties publiques et privées.

## Mashups

- assemblage de petits composants sur une page web pour réaliser un application complète.
- Les composants sont souvent des services issus du cloud (public)

# Protocoles mis en jeu

## Intégration avec le SI par les webservices

- Basée sur une architecture logicielle REST (Representational state transfer),
- utilisant SOAP (Simple Object Access Protocol) comme protocole d'échange de données, lui-même reposant sur
- HTTPS, version sécurisée de HTTP
- Les données elle même sont représentée par leur URI et au format XML ou JSON.

# Acteurs I

## Infrastructures et plateformes

- Amazon : EC2, et de nombreuses solutions : `http://aws.amazon.com/fr/`.
- OpenSource : RedHat `http://www.redhat.com/solutions/cloud-computing/`, Canonical `http://www.ubuntu.com/cloud`, ... avec Eucalyptus et OpenStack. ,
- Microsoft : Azur : `http://www.windowsazure.com/fr-fr/`
- SalesForce, un pionnier `http://www.salesforce.com/platform/`
- Google : `https://code.google.com/intl/fr/appengine/`
- et de nombreux autres...

# Acteurs II

## Softwares

- Tous les éditeurs y passent. Secteurs privilégiés :
- Bureautique (Google apps, Office web apps, Lotus d'IBM,...)
- CRMs (salesforces, SugarCRM, ...)
- ECMs (Nuxeo,...)
- Stockage de données

## Systèmes d'exploitation

- JoliCloud,
- EyesOs,
- Chome OS,
- Boot2Gecko etc...

# Acteurs III

## Données

- Ubuntu One,
- OwnCloud ...
- DropBox,
- Box.net
- Apple Icloud,
- Google drive,

# Enjeux

## Ce qui change avec le cloud

- Les prix : entrée plus facile, pas d'investissement lourd.
- Indépendance vis à vis du matériel : PC, tablettes,. . .car reposant sur les techniques du web.
- Indépendance vis à vis des services informatiques internes, mais... dépendance par rapport au fournisseur de service.
- Passage à l'échelle : le redimensionnement est plus facile
- Sécurité et confidentialité : peu rassurant à cause de l'ouverture sur le web, mais...
- Questions juridiques sur la localisation du service et des données.
- Questions organisationnelles : moins de maîtrise sur ses propres données et applications.

# Exercice

## Des mashups

Consituter une page web avec de nombreux services (identité, social, recherche, flux, ...)