

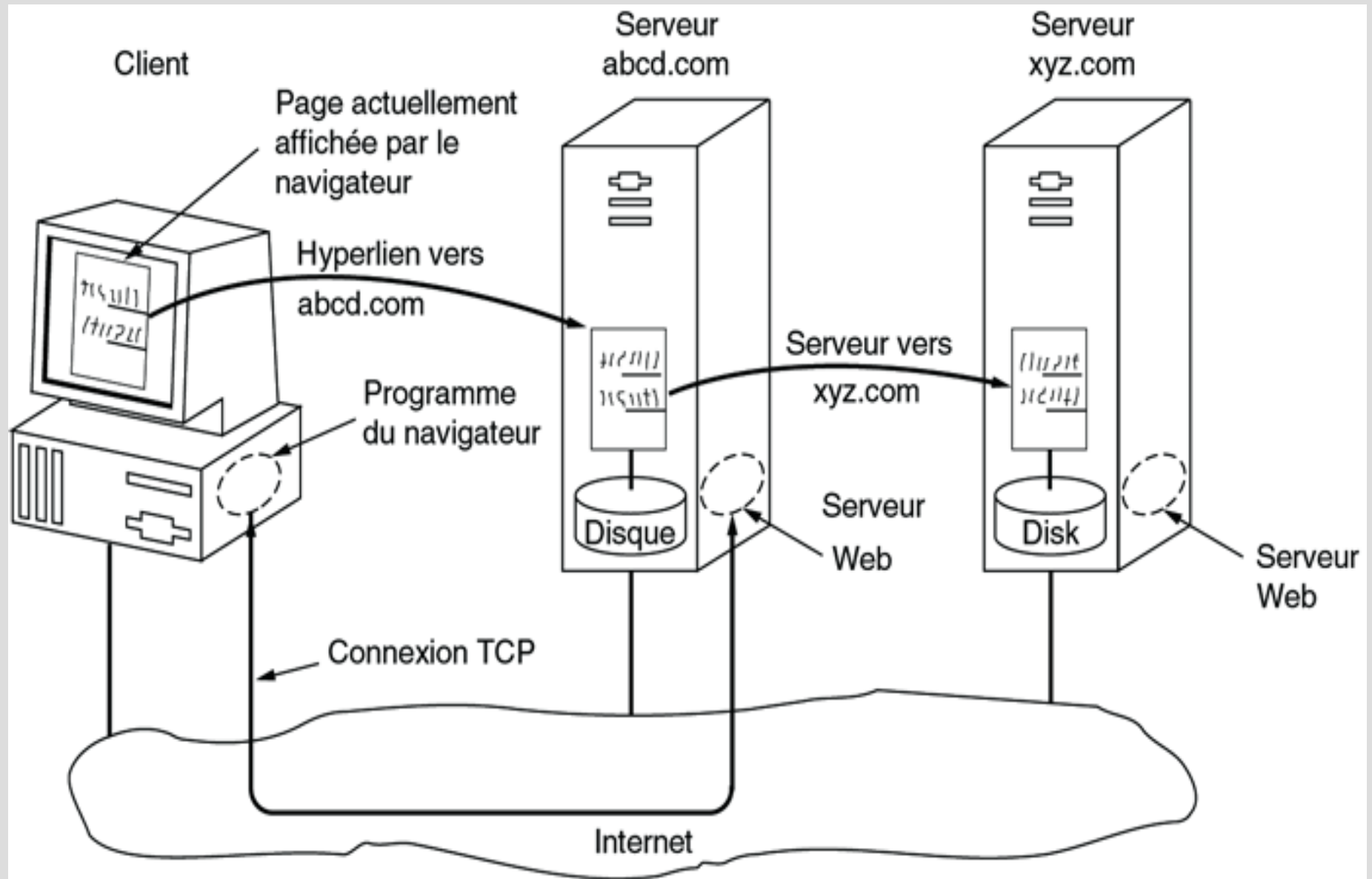
# Réseaux 1

## *Programmation Internet*

Damien Sauveron  
[damien.sauveron@unilim.fr](mailto:damien.sauveron@unilim.fr)  
<http://damien.sauveron.fr/>

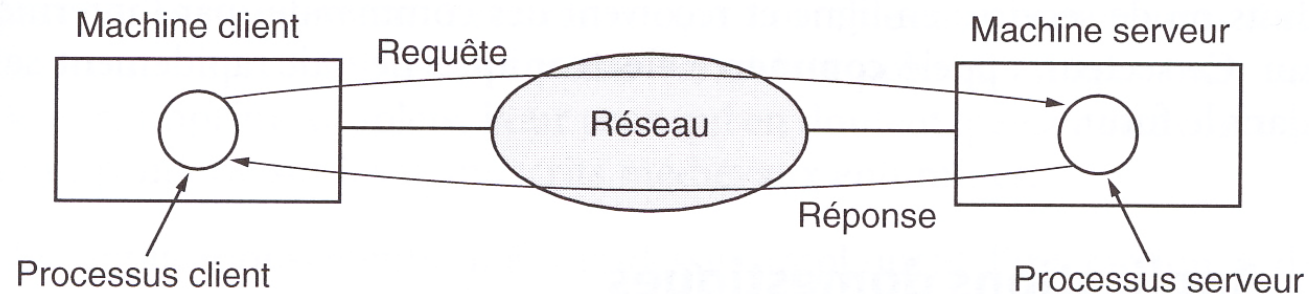
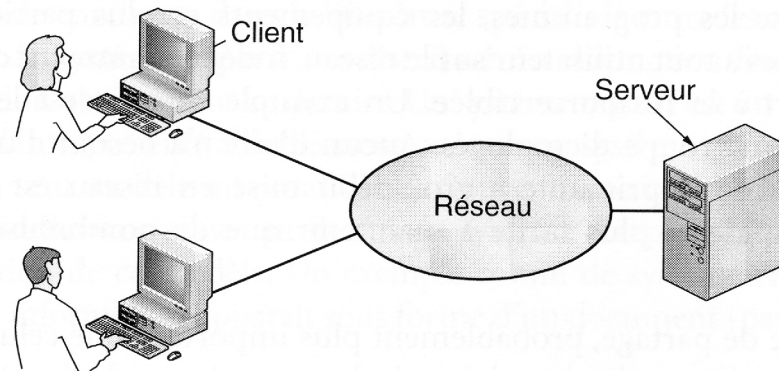


# Hyperlien



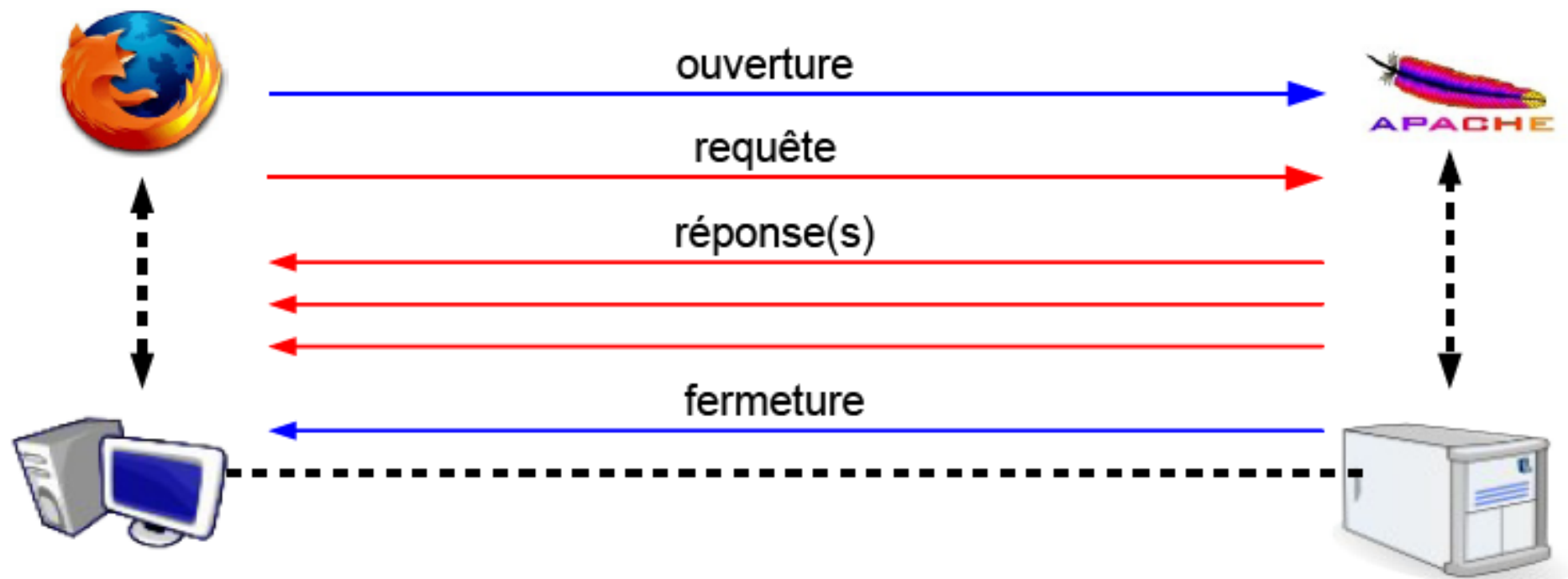
HTTP

# Modèle Client-Serveur



# Connexion à un serveur WEB

- déroulement:
  - connexion sur le port 80/TCP du serveur
  - requête du client (GET)
  - réponse du serveur (ou erreur)
  - fermeture de la connexion



# Le modèle client/serveur et le protocole HTTP “Hyper Text Transfer Protocol”

## Un système de requêtes/réponses

La base de la consultation de documents sur le Web est le mécanisme de **requête/réponse** :

### Pour le client web :

- il reçoit une URL,
- il analyse l'URL pour en déduire l'adresse de la machine désignée par l'URL et le nom du document à demander
- il se connecte sur la machine hébergeant la ressource de la manière indiquée dans l'URL (ftp, http...)
- après une connexion réussie, il réclame le document
- en général, la connexion est interrompue.

*Netscape Navigator, Opera, Internet Explorer... sont des navigateurs web.*

### Pour le serveur Web :

- il traite différentes natures de document, mais **uniquement** pour un accès suivant le protocole **HTTP**
- il attend, en général sur le port 80, la connexion d'un client
- il reçoit une requête
- il analyse la requête afin de déterminer le document demandé
- il vérifie les autorisations d'accès pour le document (sous Unix, le serveur Web se comporte comme un utilisateur ayant des droits d'accès restreint, en particulier sur les autres fichiers que ceux qu'il met en consultation)
- il retourne le document demandé au client
- en général, il termine sa connection avec le client

*Apache est le serveur web le plus employé et le plus sûr.*

# HTTP

- *HyperText Transfert Protocol*
  - protocole pour le transport des données HTML (ou autres)
  - protocole pour la soumission de formulaires
  - modèle client / serveur
- Protocole simple
  - connexion / question / réponse / déconnexion
  - mode « lignes de caractères »
    - telnet www.
  - optimisation (HTTP/1.1): *KeepAlive*
    - série de plusieurs requêtes sur une connexion TCP

# Le protocole HTTP (“Hyper Text Transfer Protocol”) RFC 1945

## Caractéristiques

C'est un protocole texte : échange de lignes de commandes au format ASCII 7bits

Il utilise une transmission entre le client et le serveur de type **TCP**.

Il est **très simple**, ce qui explique sa popularité et sa facilité de mise en œuvre.

## Versions

HTTP/0.9 version de base avec requête/réponse le document est renvoyé directement

HTTP/1.0 version normalisée (RFC 1945) avec comme amélioration, l'ajout :

- d'en-tête pour la description des ressources échangées (utilisation du format MIME RFC822),
- des informations supplémentaires envoyées par le client (format de données supporté ou désiré, description de la version et de la marque du navigateur...)

HTTP/1.1 ajout de connexions persistantes entre le client et le serveur en vue de l'échange de plusieurs ressources par l'intermédiaire de la même connexion (transfert des différents éléments d'un même document composite).

## Dialogue client/serveur

Le protocole HTTP se caractérise par l'échange d'une requête et d'une réponse.

Ces requêtes et réponses ont un schéma similaire :

- une **ligne initiale** (ligne de **commande** pour la requête ou **d'état** pour la réponse)
- aucune ou plusieurs lignes d'en-tête
- une ligne « **vide** »
- un message optionnel (un fichier, des données pour la requête ou des résultats de la requête)

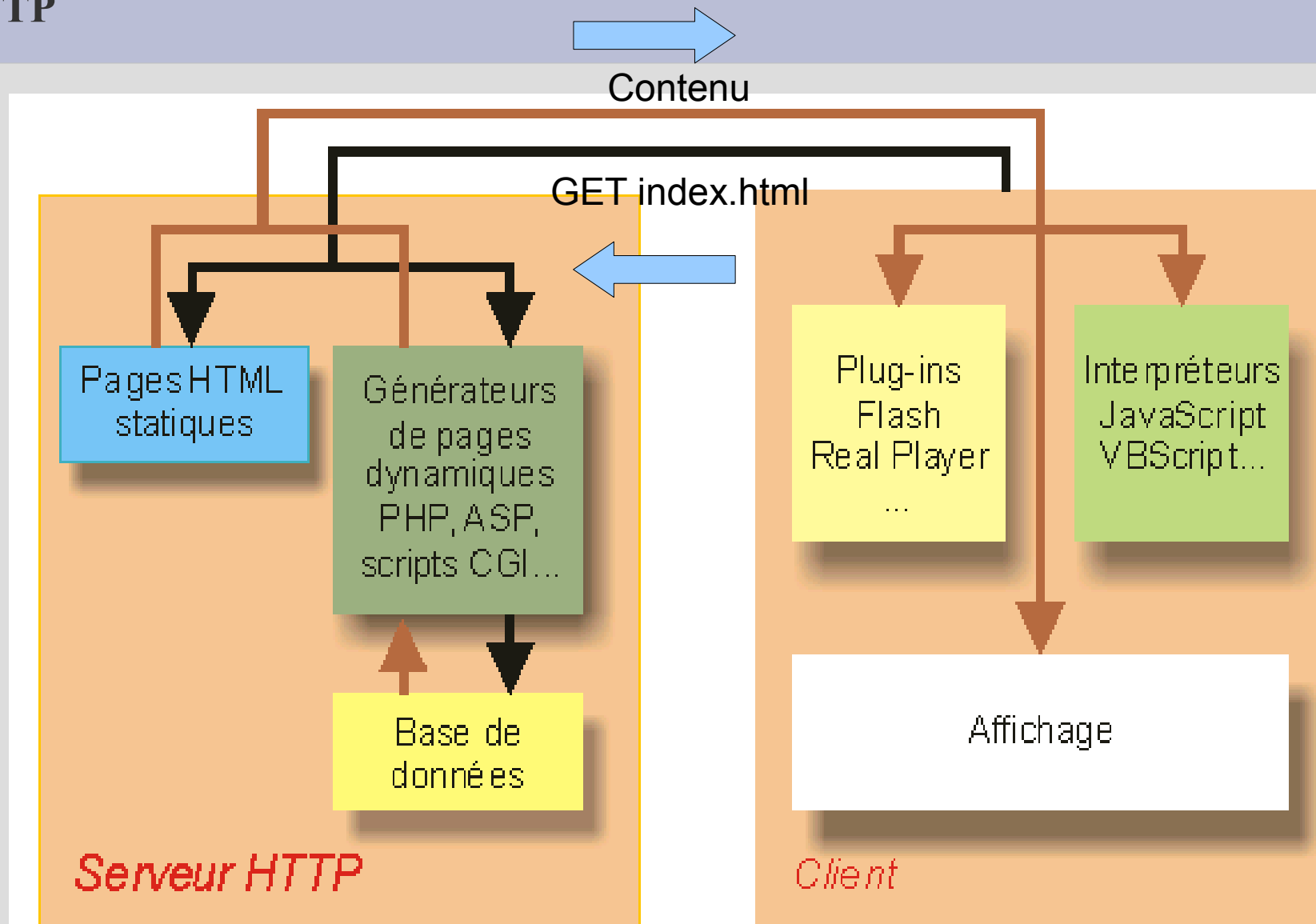
<ligne initiale (différente pour la requête ou la réponse)>

En-tête1: value1

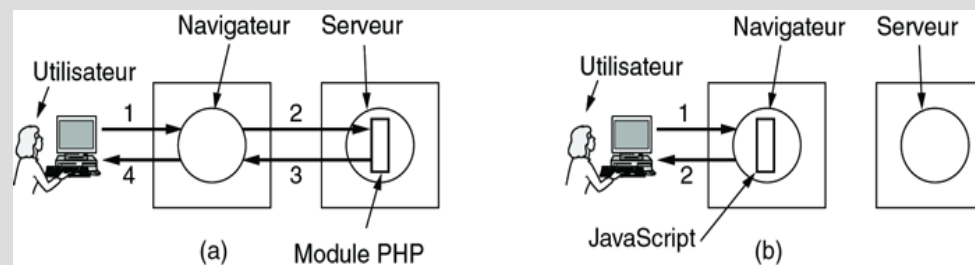
En-tête2: value2

<message optionnel, pouvant disposé d'une en-tête, suivie d'un corps constitué d'octets sur 7 ou 8bits>

# HTTP



## Notion de plugins



# Méthodes de la requête HTTP

- GET: accès direct à une ressource pointée par l'URI
- HEAD: accès direct aux entêtes
  - *i.e.* sans le corps de la réponse
- POST: soumission de données (formulaire) à une URI
- PUT: envoie du corps de la requête à une URI
- DELETE: suppression de données pointées par l'URI
- OPTIONS: retourne les méthodes HTTP supportées
- TRACE: renvoie la requête
  - debug: permet de visualiser les effets des proxies

# Requête client HTTP (1/2)

- Une requête contient:

<méthode> <URI> HTTP/<version>

[ <champs d'entête> : <valeur> ]

\* ligne blanche \*

[ corps de la requête pour méthode POST ]

- Exemple – connexion au site <http://www.lesite.org>:

GET / HTTP/1.1

Host: www.lesite.org

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1) Gecko/20060601 Firefox/2.0  
(Ubuntu-edgy)

# Requête client HTTP (2/2)

- Exemple (suite):

```
Accept: text/xml, application/xml,  
       application/xhtml+xml, text/html; q=0.9,  
       text/plain; q=0.8, image/png, */*; q=0.5
```

```
Accept-Language: en-us,en;q=0.5
```

```
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

```
Keep-Alive: 300
```

```
Connection: keep-alive
```

```
*   une ligne blanche   *
```

# URL / URI

- Adresser de manière unique une ressource

|                                                                                 |             |               |                |
|---------------------------------------------------------------------------------|-------------|---------------|----------------|
| <u>protocole</u>                                                                | <u>hôte</u> | <u>chemin</u> | <u>ancree</u>  |
| <code>http://user:pass@example.com:992/animal/bird?species=seagull#wings</code> |             |               |                |
| <u>login / password</u>                                                         |             | <u>port</u>   | <u>requête</u> |

- Une URL (ou adresse) est:
  - **absolue** (chemin complet)
  - ou **relative** (par rapport à la position actuelle)
- Exemples:
  - `ftp://somehost/resource.txt`
  - `mailto:kartoch@kartoch.com`
  - `../../../../resource.txt`
  - `/relative/URI/with/absolute/path/to/resource.txt`

# Le protocole HTTP — Description d'une transaction — suite

## La ligne de commande dans le cas de la requête

Ces commandes sont divisés en trois parties séparées par des espaces :

- une **méthode**
- le **chemin d'accès** de la ressource
- la **version** du protocole HTTP utilisé

|        |                                                    |
|--------|----------------------------------------------------|
| GET    | Récupération d'un document                         |
| HEAD   | Consultation de renseignement sur un document      |
| PUT    | Rangement d'un document sur le serveur             |
| POST   | « Ajout » à une ressource des informations données |
| DELETE | Suppression d'un document                          |
| LINK   | Définition d'une connexion entre deux documents    |
| UNLINK | Suppression d'une connexion entre deux documents   |

*Le chemin d'accès (ou requête URI "Uniform Resource Identifier") est la partie qui suit l'adresse du serveur.*

*La version du protocole est indiquée sous la forme « HTTP/x.x » en majuscule.*

Exemple :       GET /chemin/d/acces/au/fichier/index.html HTTP/1.0

## La ligne d'état dans le cas de la réponse

Elle est décomposée en trois parties séparées par des espaces :

- la **version** du protocole HTTP
- un **code d'état** pour donner le résultat de la requête
- une **phrase** expliquant la raison du code d'état

Le code d'état est un entier sur trois chiffres, dont le premier chiffre donne la catégorie générale :

- 1xx message d'information seulement
- 2xx indication du succès
- 3xx redirection du client vers une autre URL
- 4xx indication d'une erreur au niveau du client
- 5xx indication d'une erreur au niveau du serveur

## Le protocole HTTP — Description d'une transaction — suite

### Les lignes d'en-tête

Ces lignes donnent des informations à propos de la requête ou de la réponse.

Elles sont au format général introduit dans la RFC 822, c-à-d. « Nom-En-tête: valeur ».

*Les noms d'en-têtes peuvent être minuscules ou majuscules.*

*Il peut y avoir des espaces et des tabulations entre le « : » et la valeur.*

*Une ligne d'en-tête commençant par un espace ou une tabulation est associé à la ligne précédente :*

*EN\_TÊTE1: un\_nom\_tres\_long, un\_autre\_nom\_tres\_long  
est équivalent à :*

*EN\_TÊTE1:      un\_nom\_tres\_long,  
                  un\_autre\_nom\_tres\_long*

Il existe de nombreux en-têtes (HTTP/1.0 16 en-têtes, HTTP/1.1 46 en-têtes).

Au niveau d'une requête :

- "Host:" qui est obligatoire pour indiquer le nom du serveur que l'on veut utiliser

*Dans le cas de l'hébergement de plusieurs sites Web sur une même adresse IP, il permet de distinguer le site auquel on veut avoir accès en fonction de l'alias de la machine que l'on a utilisé :*

*193.50.185.1 correspond à www.un\_site.un\_domaine.fr et www.un\_autre\_site.un\_domaine.fr*

- "From:" donne l'adresse électronique de la personne ayant fait la requête
- "User-Agent:" permet d'identifier le logiciel qui fait la requête

*Exemple : User-agent: Mozilla/0.9*

Au niveau d'une réponse :

- "Server:" permet d'identifier le logiciel serveur
- "Last-modified:" permet de donner la date d'actualisation de la ressource (intéressant pour un cache...)

# Réponse serveur HTTP

- Une réponse contient:

HTTP/<version> <Status> <Commentaire status>

Content-Type: <type MIME du contenu>

[ <champs d'entête> : <valeur> ]

\* ligne blanche \*

[ Document ]

- Exemple:

HTTP/1.1 200 OK

Content-Type: text/html; charset=ISO-8859-1

Cache-Control: private

Date: Tue, 02 Jan 2007 14:06:31 GMT

Server: GWS/2.1

...

# Code de statut HTTP

- **1XX:** Informationnel
- **2XX:** Succès
  - 200: OK
- **3XX:** Redirection
  - 301: Redirection permanente
  - 307: Redirection temporaire
- **4XX:** Erreur Requête
  - 400: requête invalide
  - 402: non autorisé
  - 404: ressource non trouvée
- **5XX:** Erreur Serveur

# Méthodes GET et POST

- Passage de paramètres des formulaires:

- GET: paramètres passés avec l'URL

```
GET /get?param1=val1&param2=val2 HTTP/1.1
```

```
User-Agent: Mozilla/5.0...
```

```
* une ligne blanche *
```

- POST: paramètres

```
GET /post HTTP/1.1
```

```
User-Agent: Mozilla/5.0...
```

```
* une ligne blanche *
```

```
param1=val1&param2=val2
```

- Les caractères sont encodés sur 7 bits (x-www-form-urlencoded)

# Entêtes HTTP (générales)

- Utilisées dans une requête client ou une réponse serveur
  - Cache-Control: politique de cache
  - Connection: maintenance de la connexion
  - Date: **date** actuelle
  - Pragma, Trailer, Transfer-Encoding, Upgrade, Via, Warning

# Entêtes HTTP (requête client) (1/2)

- Utilisées dans une requête client:
  - Accept, Accept-Charset, Accept-Encoding, Accept-Language: liste des médias, ensemble de caractères, encodage (gzip) et langues acceptables pour la réponse
    - le paramètre q permet de spécifier un ordre de préférence
  - Cookie: retourne le cookie associé au serveur
  - Expect: fonctionnalités du serveur attendues par le client
  - Host: permet de spécifier l'hôte contacté
    - permet le *virtual hosting* (plusieurs sites sur une adresse IP)

# Entêtes HTTP (requête client) (2/2)

- `If-Modified-Since`, `If-Unmodified-Since`: charger un document s'il a été modifié / n'a pas été modifié depuis une date
- `If-Match`, `If-None-Match`: charger un document si son **Etag** contient / ne contient pas une séquence
- `Referer`: permet au serveur de savoir d'où « vient » le client
- `User-Agent`: précise le logiciel (*agent*) client
- `Authorization`, `From`, `Max-Forwards`, `Proxy-Authorization`, `Range`, `TE`, `If-Range`

# Entêtes HTTP (réponse serveur)

- Utilisées dans une réponse serveur:
  - Age: « âge » du document renvoyé
  - ETag: « entity tag »
    - identifiant du document retourné
  - Location: redirection
    - utilisé avec les code HTTP 201 et 3XX
  - Server: indique le serveur et les logiciels associés
  - Set-Cookie: création / mise-à-jour du cookie
  - Accept-Ranges, ETag, Proxy-Authenticate, Retry-After, Vary, WWW-Authenticate

# Entêtes HTTP (entité)

- Meta-information à propos de la ressource
  - Allow: méthodes supportées
  - Content-Encoding, Content-Language, Content-Length, Content-MD5, Content-Range, Content-Type: information sur le document transporté
  - Expires: date d'expiration du document (utilisé par le cache)
  - Last-Modified: dernière modification
  - ...

# Internationalisation

- Language: **langue du document**
  - en, fr, de, ja, zh...
- Charset: **table de caractères**
  - par défaut: ISO-8859-1 (Latin I)
  - préférable: UTF-8
  - Autres:
    - ISO-8859-2 (hongrois, polonais)
    - ISO-8859-5 (russe, bulgare, polonais)
    - ISO-8859-7 (grec)
    - Shift\_JIS, ISO-2022-JP, EUC-JP (japonais)
    - Big5 (chinois simplifié)
    - GB2312 (chinois traditionnel - Taiwan)

# Exemple de requete

GET /~poulingeas/index.htm HTTP/1.1

Accept: image/gif, image/jpeg, \*/\*

Accept-Language: fr

Host: www.msi.unilim.fr

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

# Exemple de réponse

Le serveur fournit alors la réponse suivante :

HTTP/1.1 200 OK

Date: Tue, 18 Jan 2005 15:25:00 GMT

Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2

Last-Modified: Thu, 04 Nov 2004 11:20:11 GMT

Content-Length: 361

Connection: close

Content-Type: text/html; charset=iso-8859-1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
```

```
"http://www.w3.org/TR/html4/frameset.dtd">
```

```
<html>
```

```
<head>
```

```
<title>
```

```
Site de Patrick Poulingeas
```

```
</title>
```

```
</head>
```

```
<frameset cols="18%,82%">
```

```
<frame src="choix.htm" name="index" frameborder="0" scrolling="yes"
noresize>
```

```
<frame src="present.htm" name="Debut" frameborder="0">
```

```
</frameset>
```

```
</html>
```

# DEMO RECUPERATION D'UNE IMAGE ET VIRTUAL-HOST

# Session Tracking (1/3)

- HTTP: protocole *stateless*
  - Le protocole ne maintient pas d'information sur l'utilisateur dans les requêtes
  - comment suivre un utilisateur sur le site ?
  - comment garder un état de l'utilisateur côté serveur ?
  - comment reconnaître un utilisateur lors de son retour sur le site ?
- Nécessité d'avoir une notion de session
  - le serveur génère un identifiant de session unique pour identifier l'utilisateur
  - comment le transmettre / recevoir ?

# Session Tracking (2/3)

- Première solution: champ HIDDEN
  - entrée dans un formulaire caché à l'utilisateur mais transmis lors des requêtes GET/POST
  - Défauts:
    - Uniquement utilisable dans les formulaires
    - Ambiguïté dans le cas de retours arrières de l'utilisateur
- Deuxième solution: réécriture URL
  - Ajout de l'identifiant de session, dans les URLs de la page:
    - <http://www.lesite.org/command?session=1234567890>
  - Défauts:
    - Nécessite de générer les pages avec toutes les URLs modifiés
    - cohabitation avec Javascript

# Session Tracking (3/3)

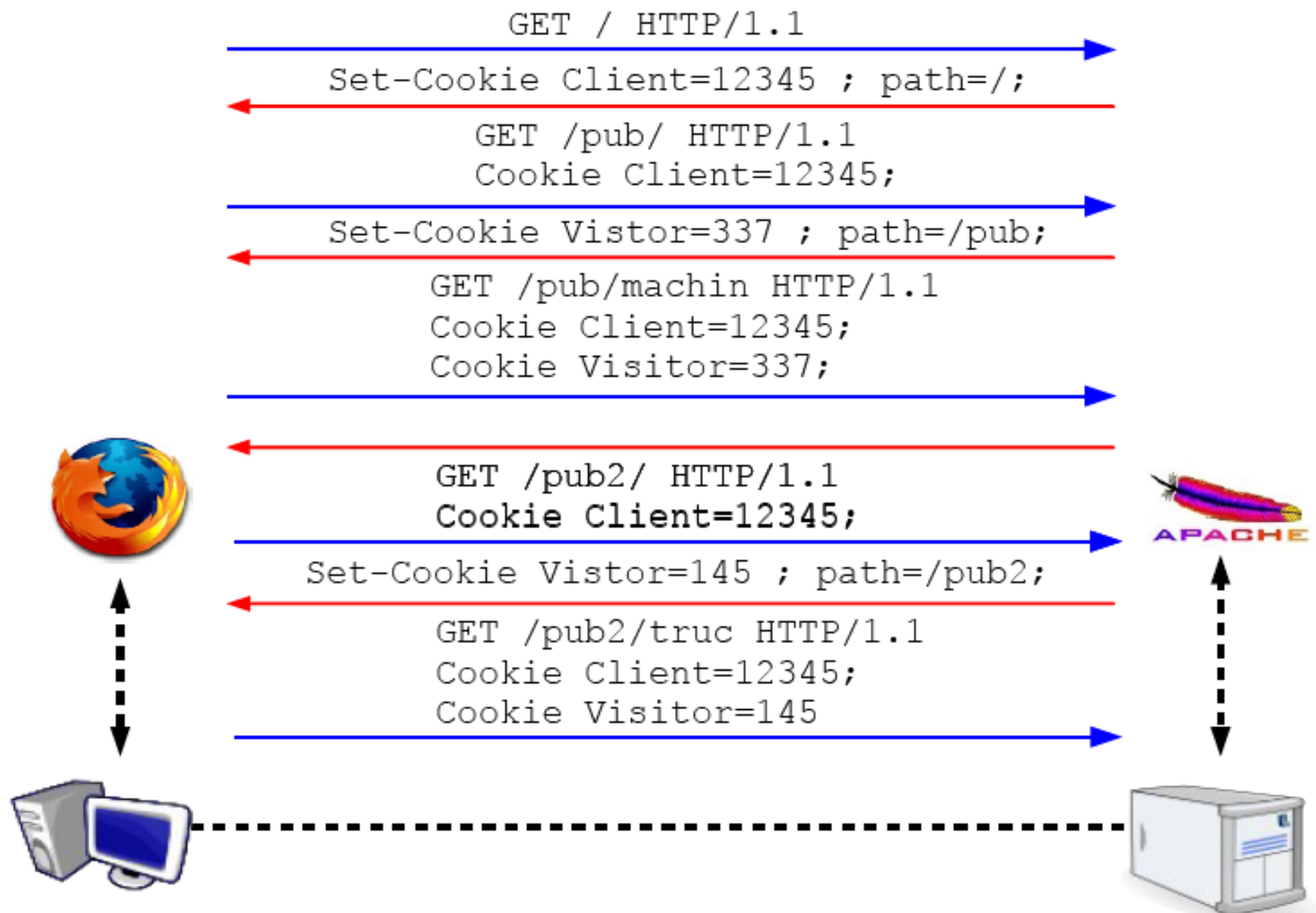
- Troisième solution: suivi de connexion SSL
  - Une session HTTPS est identifié par un numéro unique
  - défauts:
    - connexion entièrement avec SSL
    - relation langage serveur / identifiant SSL
- Quatrième solution: les cookies
  - permet d'associer un couple clé / valeur associé à un site
  - stocké chez le client
  - création / mise-à-jour par le serveur (`Set-Cookie`)
  - envoyé à chaque requête par le client (`Cookie`)
  - accessible par JavaScript dans la page HTML

# Cookies

- Attributs:
  - un nom
  - une valeur
  - un domaine (domain)
  - un chemin (path)
  - une date d'expiration
  - un drapeau *discard*
  - un drapeau *secure*
  - un numéro de version
- Un cookie est envoyé en fonction du triplet domaine / chemin / *secure*



# Cookies: exemple



## Le protocole HTTP — Description d'une transaction — suite

### Le message optionnel transmis après les lignes d'en-tête

En général, il correspond à la ressource elle-même dans le cas d'une réponse.

Dans le cas d'une requête, ce message peut contenir une ressource à mettre à jour (commande PUT par exemple) ou bien des informations saisies par l'utilisateur.

Ce message dispose de sa propre en-tête, exprimée sous forme de lignes d'en-tête :

- "Content-Type" pour donner la nature de cet élément au format MIME
- "Content-Transfer-Encoding" pour indiquer quel format est utilisé pour le transférer
- "Content-Length" pour donner la taille de l'élément.

### Dans le cas d'un message joint à une requête, c'est un champ obligatoire !

Il permet u serveur de lire *exactement* le nombre d'octets concernant l'élément, avant de passer à l'élaboration de la réponse.

### Cas particuliers de certaines commandes

- Dans le cas de la méthode "**HEAD**", seul l'en-tête de la réponse est envoyée(juste la ligne d'état et les en-têtes correspondant au serveur mais pas le message optionnel).

Cette méthode permet de savoir si une ressource a été modifiée et doit être de nouveau récupérée.

- La commande "**POST**" est employée pour envoyer des données de l'utilisateur.

Elle utilise pour décrire le message joint à la requête, les en-têtes "Content-Length" et "Content-Type".

*En général, ces informations sont fournies en entrée d'un programme et la réponse du serveur correspond à la sortie de ce programme.*

*L'URI employée est souvent le nom du programme à déclencher.*

*La réponse est souvent construite de manière **dynamique**.*

### Mode spécial de transfert du message optionnel

Le mode « en morceaux », "Content-Transfer-Encoding: chunked", permet de transmettre un message optionnel crée par morceaux et dont on ne connaît pas la taille totale a priori. (on donne à chaque fois en hexadécimal sur deux chiffres suivi d'un point-virgule la taille du morceau suivi du morceau).

# Le langage HTML

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<p>Hello world.
</p>
<p>
</p>
</body>
</html>
```

