Technologie Web

Serveur Web et protocole HTTP

Alexandre Pauchet

INSA Rouen - Département ASI

BO.B.RC.18, pauchet@insa-rouen.fr

Plan

Historique (1/5)
Arpanet

- 1959-68 : le programme ARPA naît pendant la guerre froide

  La peur d'une guerre nucléaire

  - Faiblesse du système centralisé *versus* distribué
  - Proposition d'un maillage d'ordinateurs (1964, P. Baran)
  - $1^{ère}$ communication téléphonique entre 2 machines en 1965
- 1969 : ARPANET

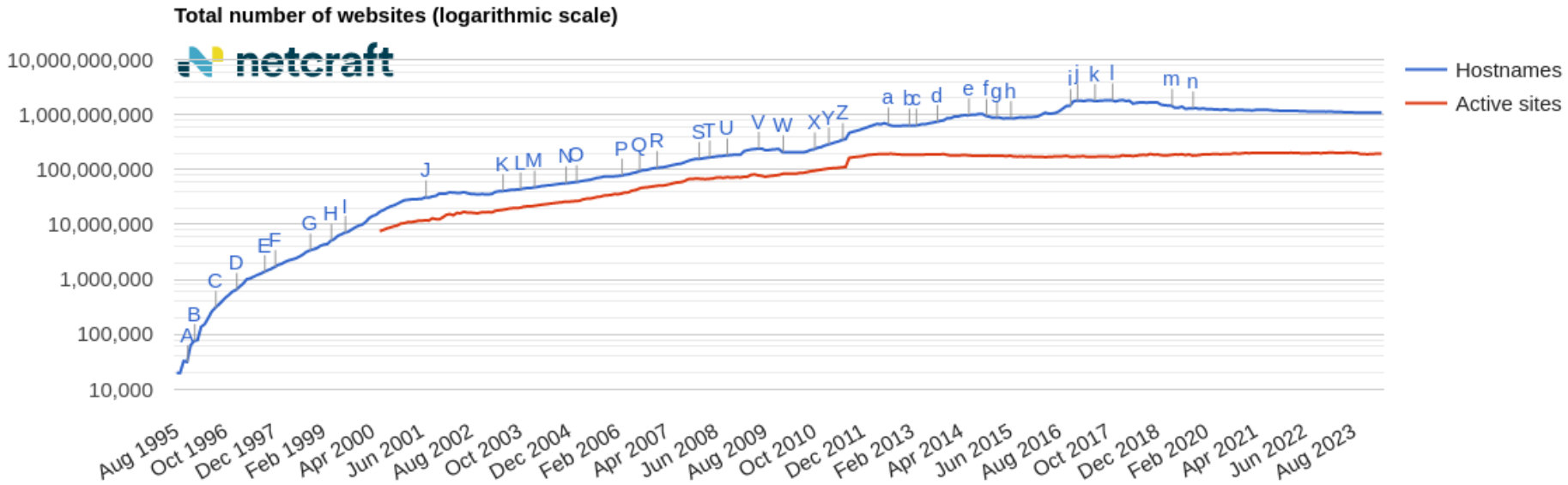
  1969 : 4 noeuds, 1971 : 15 nœuds, 1972 : 37 nœuds
- 1970-82 : ouverture sur le monde
  - Apparition du courrier électronique
  - Communications internationales (Angleterre, Norvège)
  - Apparition de TCP/IP (1974) plus puissant que NCP
- 1983 : TCP/IP adopté comme standard ARPANET $\Rightarrow$ Internet

Historique (2/5)
Internet/World Wide Web

- 1983-89 : expansion du réseau (*autoroutes de l'information*)
  - La NSF[1] effectue des progrès importants (réseau NFSNET)
  - Utilisation importante par les scientifiques
  - Réseaux hétérogènes (NCP et TCP/IP)
  - Fin officielle de ARPANET en 1989 (TCP/IP)
- 1990-97 : explosion d'internet
  - 1990, le physicien Tim Berners Lee (CERN) étend le concept de lien hypertexte à Internet
  - HyperText Markup Language (HTML) et HyperText Transfer Protocol (HTTP)
  - 1^{er} navigateur : NCSA Mosaic
  - 1995 ouverture au grand public (Netscape et Internet Explorer)
  - 1997 des dizaines de milliers de nœds dans plus de 42 pays

1. National Science Foundation

Historique (3/5)
Nombre de sites Web



source : `http://www.netcraft.com`

Historique (4/5)
Logiciels disponibles

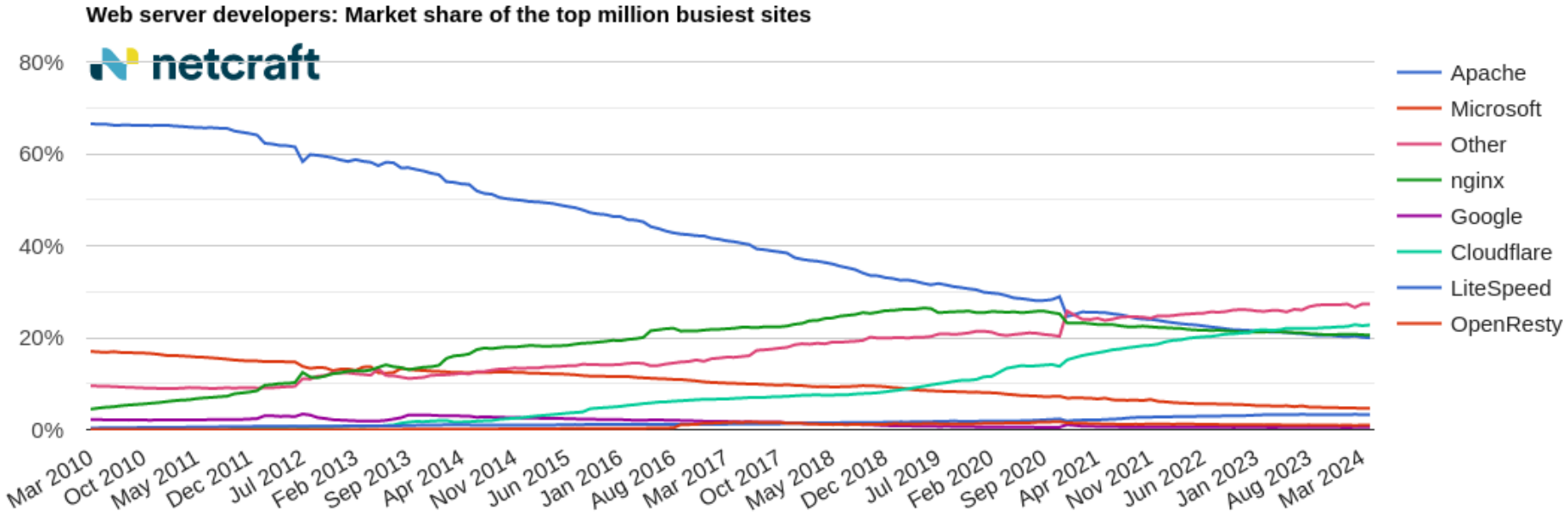
Clients
Netscape, Mozilla, Konqueror, Opera, Lynx, emacs, Internet Explorer

Serveurs
Apache, Internet Information Server (Microsoft), iPlanet (Netscape)

Developer	March 2017	Percent	April 2017	Percent	Change
Microsoft	704,000,530	39.99%	812,157,808	44.71%	4.73
Apache	383,707,112	21.79%	412,130,526	22.69%	0.90
nginx	350,540,372	19.91%	349,092,975	19.22%	-0.69
Google	18,849,171	1.07%	19,121,684	1.05%	-0.02

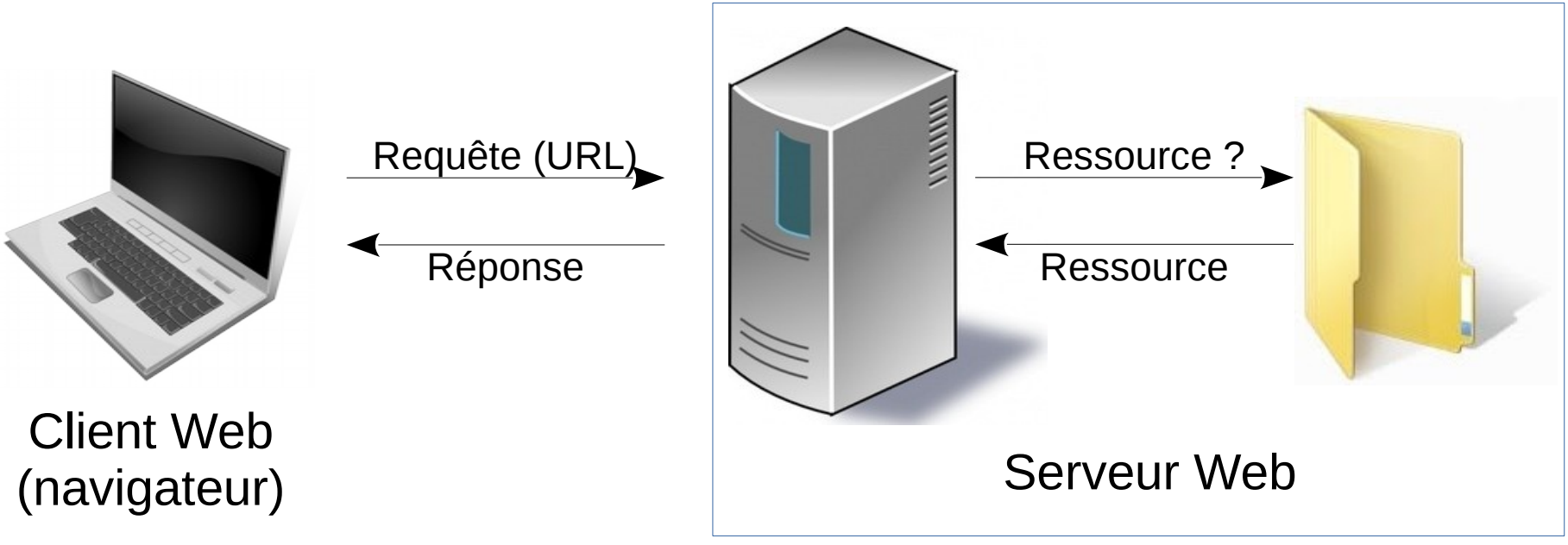
Sources: http://www.netcraft.com

Historique (5/5)
Répartition des serveurs sur le marché



source : `http://www.netcraft.com`

Principes de fonctionnement (1/8)
La base du Web



- Architecture Client/Serveur
- Nécessité d'un protocole de communication : HTTP

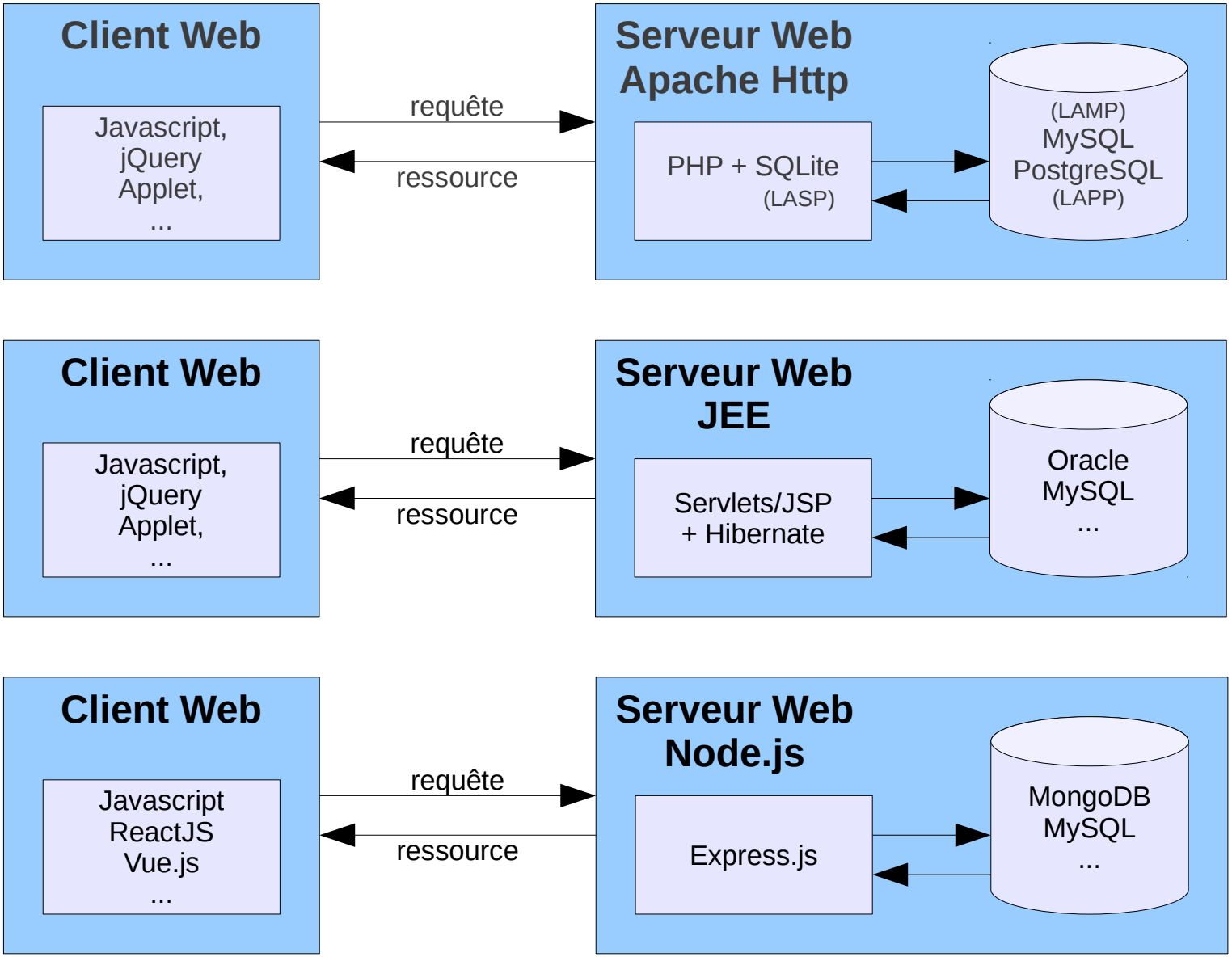
Principes de fonctionnement (2/8)

Différents types de ressources

- Ressources statiques : HTML, images, son, vidéos
- Ressources dynamiques
  - Côté client : applet (Java), Javascript/JQuery, Plugin, ActiveX, ...
  - Côté serveur : CGI, servlets/JSP, scripts serveur (php), scripts Javascript (node.js)

Principes de fonctionnement (3/8)

3 grandes familles d'architecture Full Stack



Principes de fonctionnement (4/8)
URL, URN et URI

- URL : Uniform Ressource Locator
  - Spécification de la localisation d'une ressource de manière unique
- URN : Uniform Ressource Name
  - Mécanisme de nommage des ressources
  - `urn:<Namespace>:<SpecificString>`
  - `Namespace` : identificateur de nommage (ex : isbn)
  - `SpecificString` : chaîne de caractères spécifique désignant la ressource de manière unique
- URI : Uniform Resource Identifier
  - `URI = URL + URN`
  - En pratique, la forme d'URI la plus utilisée est l'URL

Principes de fonctionnement (5/8)
URL : Uniform Ressource Locator

Format

```
<protocole>://<serveur>:<port>/<chemin>/<ressource>
```

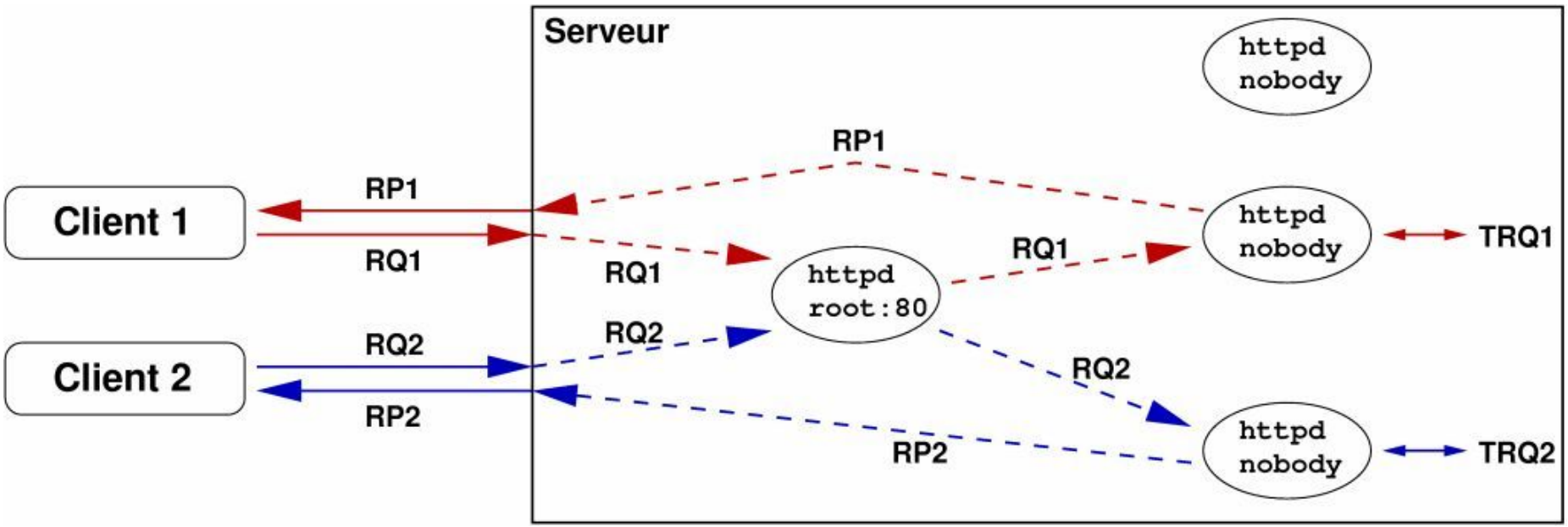
Remarque : certains caractères doivent être encodés par % suivi de leur valeur hexadécimale en ISO Latin ou ASCII (ex : doc#2.html ⇒ doc%232.html).

Exemples :

```
http://www.linux-mandrake.com:80/fr/index.html
http://asi.insa-rouen.fr/enseignants/~apauchet/
ftp://ftp.debian.fr.org/
sftp://apauchet@insa-rouen.fr/
file://home/cours/
mailto:pauchet@insa-rouen.fr
telnet://user:password@host:port
```

Principes de fonctionnement (6/8)

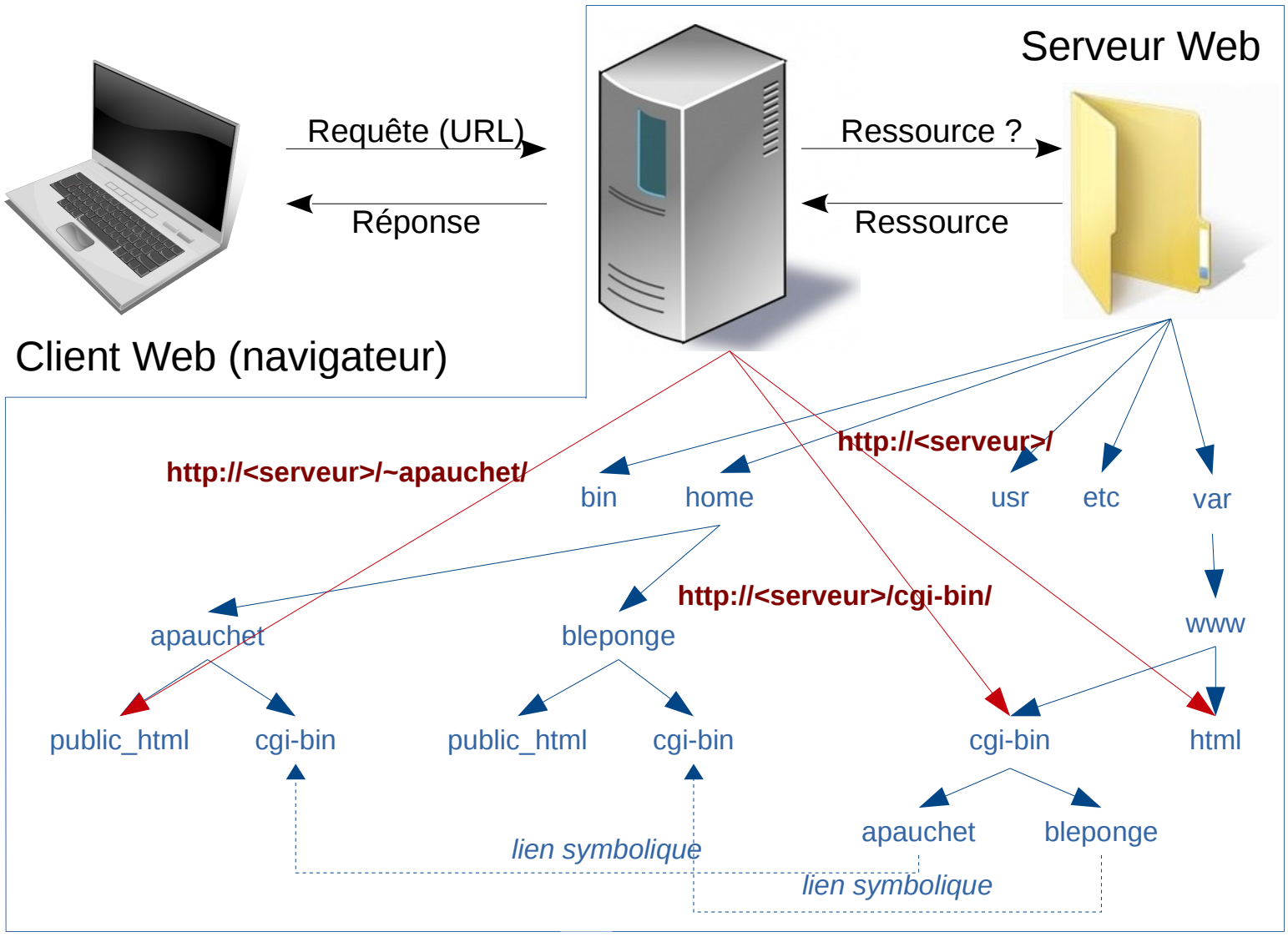
Fonctionnement d'un serveur HTTP



- Serveur : application qui écoute un port de communication
- Port standard : 80 (Apache HTTP), 8080 (Serveur web JEE)
- Serveur "maître" : utilisateur `root` écoute le port standard
- Serveurs "esclaves" : créés par le maître (propriétaire différent)
- Réception d'une requête :
  1. le maître reçoit la connexion
  2. le maître crée un esclave et lui transmet le canal de communication
  3. l'esclave traite la requête et retourne le résultat

Principes de fonctionnement (7/8)

Exemple : système de fichiers Apache HTTP



Principes de fonctionnement (8/8)

Exemple : système de fichiers Apache HTTP

`http://<serveur>/`	racine du serveur
`http://<serveur>/fic.html`	`fic.html` à la racine
`http://<serveur>/cgi-bin`	répertoire des scripts `CGI`
`http://<serveur>/~arthur`	la "homepage" de Arthur
`http://<serveur>/~arthur/cgi-bin`	les scripts `CGI` de Arthur
`http://<serveur>/cgi-bin/arthur`	les scripts `CGI` de Arthur

Remarque : Interdiction d'accéder aux fichiers

- de configuration du serveur
- extérieurs au serveur WEB (système)

Le protocole HTTP (1/13)
Présentation de HTTP

- `http://www.w3.org/`
- Protocole orienté caractères ⇒ `telnet host 80` ou `netcat host 80`
- Non sécurisé (par opposition à HTTPS)
- HTTP 0.9, protocole très simple
- HTTP 1.0 (rfc1945)
  - ajout du n° de version, du statut
  - apparition des entêtes (`user-agent`)
  - les cookies (simulation de session)
- HTTP 1.1 (rfc2616 [2]), version actuelle
  - persistance des connexions
  - méthodes `PUT`, `DELETE`, ...
- HTTP NG, en standby
  - gestion des sessions

2. `http://www.w3.org/Protocols/rfc2616/rfc2616.html`

Le protocole HTTP (2/13)
Exemples de requête HTTP 1.1

Le fichier `phrase.txt` est placé à la racine d'un serveur Apache Http, tournant sur la machine cliente.

```
> netcat localhost 80
GET /phrase.txt HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Date: Wed, 15 Jul 2009 13:08:49 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT
ETag: "31c06d-1c-46eae8cd55a40"
Accept-Ranges: bytes
Content-Length: 28
Content-Type: text/plain

Voici un exemple de phrase.
```

Le protocole HTTP (3/13)
Exemples de requête HTTP 1.1

```
> telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /phrase.txt HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Date: Tue, 14 Jul 2009 18:28:30 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT
ETag: "31c06d-1c-46eae8cd55a40"
Accept-Ranges: bytes
Content-Length: 28
Content-Type: text/plain

Voici un exemple de phrase.
Connection closed by foreign host.
```

Le protocole HTTP (4/13)
Exemples de requête HTTP 1.1

Le protocole HTTP (5/13)
Exemples de requête HTTP 1.1

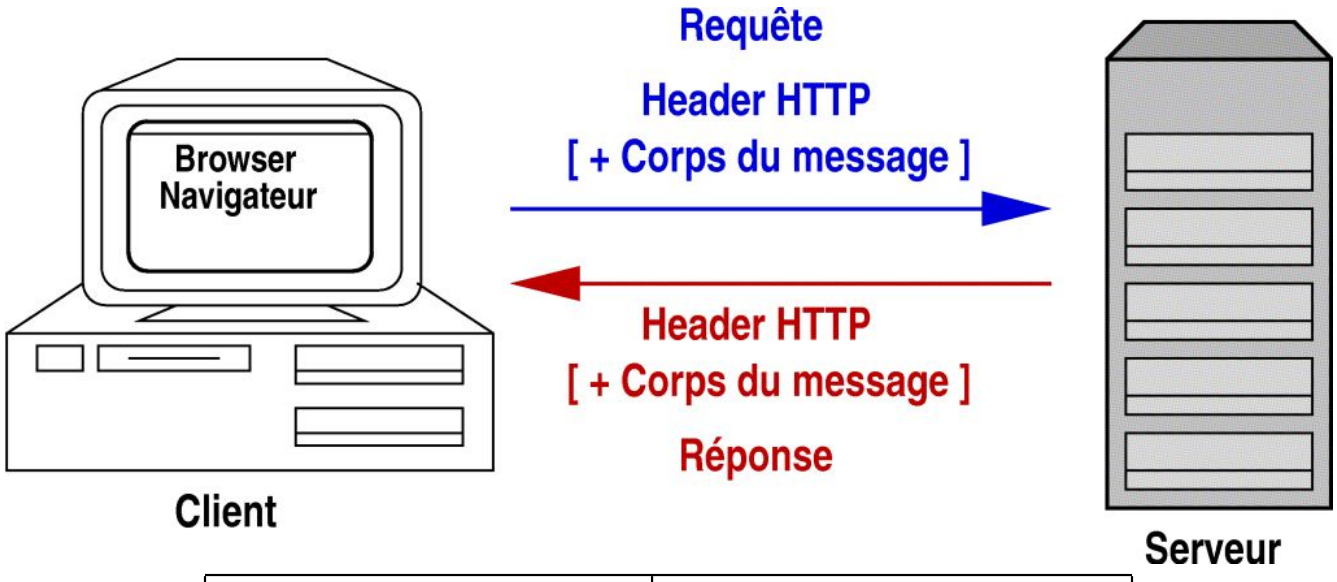
```
netcat localhost 80
GET /index.html HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Date: Wed, 15 Jul 2009 13:17:53 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Last-Modified: Tue, 14 Jul 2009 18:27:21 GMT
ETag: "31c072-2d-46eae96d8d440"
Accept-Ranges: bytes
Content-Length: 45
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>
```

Le protocole HTTP (6/13)

Requête et Réponse



Requête	Réponse
Request line	Status line
General header	General header
Request header	Response header
Entity header	Entity header
CRLF	CRLF
Message body	Message body

Le protocole HTTP (7/13)

Requête : Request-Line

Request-Line

```
METHODE URI [HTTP-Version]
```

Les méthodes

- `OPTIONS` : demande les méthodes utilisables sur l'URI
- `GET` : demande les informations et les données de l'URI
- `POST` : envoie de données (ex : formulaire) traitées par l'URI
- `HEAD` : demande uniquement les informations sur l'URI
- `PUT` : enregistre le corps de la requête à l'URI
- `DELETE` : supprime les données pointées par l'URI
- `TRACE` : retourne ce qui a été envoyé par le client ($\simeq$ echo)

Par défaut la version utilisée est la 1.0

Le protocole HTTP (8/13)
Requête/Réponse : General header

- `Cache-Control` : définit la politique de cache pour la ressource
- `Date` : date du message
- `Pragma` : utilisé pour spécifier des comportements aux serveurs intermédiaires (proxy)
- `Transfer-Encoding` : types de transformations appliquées au corps du message
- `Via` : indique les intermédiaires par lesquels est passée la requête

- `Connection` : paramètre de gestion de la connexion (ex : `Connection: close`)
- `Upgrade` : spécifie quels autres protocoles supporte le client

Le protocole HTTP (9/13)

Requête : Request header

- `Accept` : types de médias acceptés (ex : `Accept: text/html`)
- `Accept-Charset` : spécifie les jeux de caractères acceptés
- `Accept-Encoding` : spécifie les types de transformations (compressions) du message acceptés
- `Accept-Language` : spécifie les langues acceptées
- `From` : e-mail de l'utilisateur du client (nécessite accord)
- `Host*` : spécifie le serveur (et le port) pour la requête
- `If-Modified-Since`, `If-Unmodified-Since` : requête conditionnelle sur la dernière date de modification de l'URI
- `Range` : précise la portion de données de la ressource
- `Referer` : spécifie l'URI à l'origine de la requête
- `User-Agent` : contient l'identifiant du navigateur client

Le protocole HTTP (10/13)
Requête/Réponse : Entity header

- `Allow` : liste les méthodes autorisées
- `Content-Encoding` : indique l'encodage utilisé pour la ressource (complément au type de média du `Content-Type`)
- `Content-Language` : définit la langue utilisée
- `Content-Length` : taille du corps du message
- `Content-Location` : donne la véritable URI de la ressource si celle-ci a été trouvée grâce à une autre URI
- `Content-Range` : donne la plage de données récupérées sur la totalité de la ressource
- `Content-Type` : le type du média (ex : `text/html; charset=ISO-8859-1`)
- `Expires` : date d'expiration de la ressource
- `Last-Modified` : date de dernière modification

Le protocole HTTP (11/13)

Réponse : Status-Line

Status-Line

`HTTP-Version Status-Code Reason-Phrase`

- `Status-Code` : code numérique représentant le succès où l'échec de la requête
- `Reason-Phrase` : texte expliquant le `Status-Code`

5 classes de `Status-Code`

- `1XX` : Information
- `2XX` : Succès
- `3XX` : Redirection
- `4XX` : Erreur client
- `5XX` : Erreur serveur

Le protocole HTTP (12/13)

Réponse : les Status-Code

Code	Signification	Code	Signification	Code	Signification
100	Continue	101	Switching Protocols		
200	OK	201	Created	202	Accepted
203	Non-Authoritative Information	204	No Content	205	Reset Content
206	Partial Content				
300	Multiple Choices	301	Moved Permanently	302	Moved Temporarily
303	See Other	304	Not Modified	305	Use Proxy
400	Bad Request	401	Unauthorized	402	Payment Required
403	Forbidden	404	Not Found	405	Method Not Allowed
406	Not Acceptable	407	Proxy Authentication Required	408	Request Time-out
409	Conflict	410	Gone	411	Length Required
412	Precondition Failed	413	Request Entity Too Large	414	Request-URI Too Large
415	Unsupported Media Type				
500	Internal Server Error	501	Not Implemented	502	Bad Gateway
503	Service Unavailable	504	Gateway Time-out	505	HTTP Version not supported

Le protocole HTTP (13/13)

Réponse : Response header

- `Accept-Ranges` : informe l'acceptation des requêtes `Range` par le serveur
- `Location` : redirige la requête vers une autre URI (ex : `Status-Code` : 3XX)
- `Server` : indique le type du serveur web répondant à la requête

Négociation de contenu (1/6)

Principe

Le client exprime ses capacités
Le serveur doit alors retourner la ressource la plus adéquate

Les types de spécification possibles :

- Langue
- Type MIME
- Charset (encodage des caractères)
- Encodage (compression, format d'image, *etc.*)

Négociation de contenu (2/6)

Exemple de négociation sur la langue

Les fichiers `info.txt.en` et `info.txt.fr` sont placés à la racine du serveur Apache Http.

```
GET /info.txt HTTP/1.1
Host: localhost
Accept-Language: fr;q=1,en;q=0.5

HTTP/1.1 200 OK
Date: Wed, 15 Jul 2009 13:50:02 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Content-Location: info.txt.fr
Vary: negotiate,accept-language
TCN: choice
Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT
ETag: "31c070-15-46eae8cd55a40;46ebeabae2180"
Accept-Ranges: bytes
Content-Length: 21
Content-Type: text/plain
Content-Language: fr

Ceci est du francais.
```

Négociation de contenu (3/6)

Exemple de négociation sur la langue

```
GET /info.txt HTTP/1.1
Host: localhost
Accept-Language: en;q=1,fr;q=0.5

HTTP/1.1 200 OK
Date: Wed, 15 Jul 2009 13:50:34 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Content-Location: info.txt.en
Vary: negotiate,accept-language
TCN: choice
Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT
ETag: "31c06f-19-46eae8cd55a40;46ebeabae2180"
Accept-Ranges: bytes
Content-Length: 25
Content-Type: text/plain
Content-Language: en

These are english words.
```

Négociation de contenu (4/6)
Exemple de négociation sur fichier absent

```
GET /info.txt HTTP/1.1
Host: localhost
Accept-Language: de

HTTP/1.1 200 OK
Date: Thu, 06 Sep 2012 13:48:44 GMT
Server: Apache/2.2.22 (Ubuntu)
Content-Location: info.txt.en
Vary: negotiate,accept-language,Accept-Encoding
TCN: choice
Last-Modified: Thu, 06 Sep 2012 13:30:33 GMT
ETag: "4000b7-19-4c9087ee99040;4c90880b353c0"
Accept-Ranges: bytes
Content-Length: 25
Content-Type: text/plain
Content-Language: en

These are english words.
```

Négociation de contenu (5/6)

Exemple de négociation multi-critères

Les fichiers `info.txt.en`, `info.txt.fr`, `info.html.en` et `info.html.fr` sont placés à la racine du serveur Apache Http.

```
GET /info HTTP/1.1
Host: localhost
Accept-Language: fr;q=1,en;q=0.7
Accept: text/plain;q=0.5,text/html;q=1

HTTP/1.1 200 OK
Date: Thu, 06 Sep 2012 14:04:04 GMT
Server: Apache/2.2.22 (Ubuntu)
Content-Location: info.html.fr
Vary: negotiate,accept,accept-language,Accept-Encoding
TCN: choice
Last-Modified: Thu, 06 Sep 2012 14:01:12 GMT
ETag: "400176-1d-4c908ec867a00;4c908ec867a00"
Accept-Ranges: bytes
Content-Length: 29
Content-Type: text/html
Content-Language: fr

Ceci est en français.(HTML)
```

Négociation de contenu (6/6)

Exemple de négociation multi-critères

Les fichiers `text.txt.en`, `text.txt.fr` et `text.html.en` sont placés à la racine du serveur Apache Http.

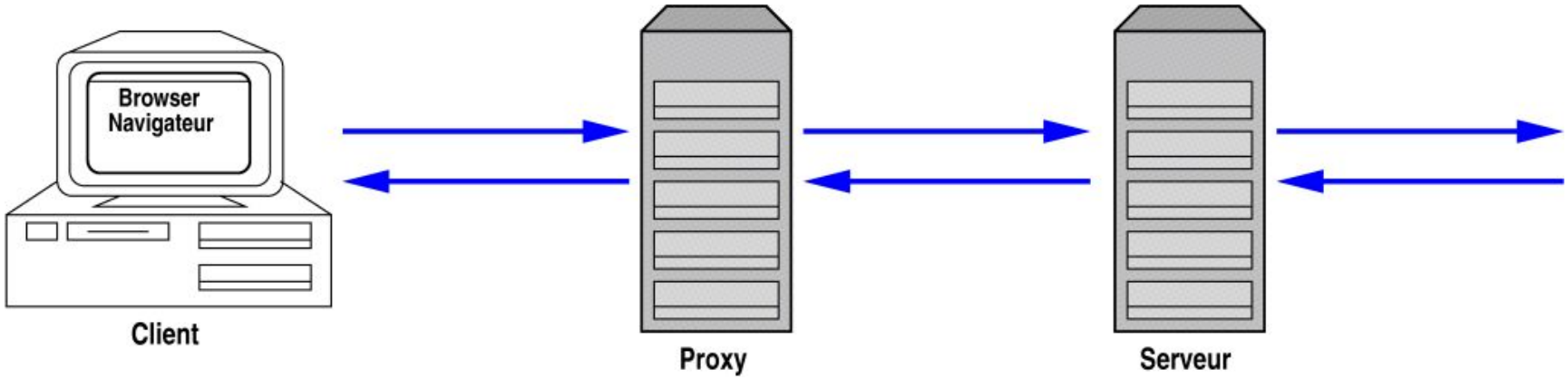
```
GET /text HTTP/1.1
Host: localhost
Accept-Language: fr;q=1,en;q=0.2
Accept: text/plain;q=0.7,text/html;q=0.8

HTTP/1.1 200 OK
Date: Thu, 06 Sep 2012 14:22:20 GMT
Server: Apache/2.2.22 (Ubuntu)
Content-Location: text.html.en
Vary: negotiate,accept,accept-language,Accept-Encoding
TCN: choice
Last-Modified: Thu, 06 Sep 2012 14:01:06 GMT
ETag: "40017a-20-4c908ec2aec80;4c909190ccac0"
Accept-Ranges: bytes
Content-Length: 32
Content-Type: text/html
Content-Language: en

These are english words. (HTML)
```

HTTP et proxys (1/3)

Utilisation d'un proxy



HTTP et proxys (2/3)

Deux grands types de proxy

Le proxy “classique”

Il s’agit d’un relais pouvant servir de “filtre” ou de “firewall”

Le proxy cache

Il archive les pages, lors d’une requête.

S’il possède déjà la page, il la renvoie, sinon il va la chercher.

HTTP et proxys (3/3)
Protocole HTTP et proxy cache

- le client effectue une requête (GET) via un proxy cache
- le proxy vérifie s'il dispose de la page demandée
- si oui
  - le proxy vérifie la date d'expiration de l'URI (Expires)
  - interroge le serveur de l'URI afin de comparer les dates
    - de dernière modification de l'URI dans le cache
    - de dernière modification de l'URI sur le serveur

    Utilisation des méthodes `GET` ou `HEAD` et des directives `If-Modified-Since` ou `Last-Modified`
  - si l'URI du cache est à jour alors le proxy retourne la ressource de son cache
- le proxy récupère la ressource du serveur, l'archive et la retourne au client.

Limites (1/3)
Déclaration d'un formulaire (rappel)

L'élément `<form> ... </form>` déclare un formulaire

Les attributs :

- `action` : URL spécifiant le traitement des données (script, *etc.*)
- `method` : spécifie la méthode d'acheminement des données (`GET` par défaut ou `POST`)
- `enctype` : spécifie, pour un envoi en `POST`, la méthode d'encodage
  - `application/x-www-form-urlencoded` : (valeur par défaut). Tous les caractères sont encodés avant d'être envoyés
  - `multipart/form-data` : aucun caractère n'est encodé.
  - `text/plain` : seuls les espaces sont remplacés par des '+'.

Limites (2/3)
Gestion des sessions : les cookies

Le protocole Http 1.1 ne gère pas les sessions
$\Rightarrow$ Alternative utilisée par les navigateurs : **cookies**

- Fichiers textes stockés sur le disque dur du client
- Durée de vie limitée, fixée par le site visité
- Problème avec les vieux navigateurs : il est possible pour un serveur de récupérer des cookies d'un client dont il n'est pas à l'origine

Limites (3/3)
Http sécurisé : Https

Le protocole Http n'est pas sécurisé
$\Rightarrow$ Alternative à HTTP : **HTTPS**

- 'S' pour secured
- Combinaison de HTTP avec SSL ou TLS
- Vérification de l'identité d'un site par un certificat d'authentification
- Garantie confidentialité et intégrité des données envoyées par l'utilisateur (ex : formulaires)
- Port par défaut : 443.

Sources

- http://www.netcraft.com
- http://www.w3.org/
- http://apache.org/
- http://stielec.ac-aix-marseille.fr/cours/caleca/http/index.html