Gilles.Roussel©univ-mlv.fr

HTTP/1.1

RFC 2068

Caractéristiques

- Niveau application
- Sans état
- Tout transfert de données
- Au dessus du protocole TCP
- Largement utilisé dans le *World Wide Web*
- Utilise les normes :
  - URI (*Uniform Resource Identifier*)
  - MIME (*Multipurpose Internet Mail Extensions*)

URL

RFC 1738 et 1808

- *Uniform Ressource Locators*
- Chaînes de caractères représentant une ressource accessible via un Internet
- Schémas
  - ftp, http, mailto, gopher, news
  - nntp, telnet, wais, file, prospero
- Schéma associé à :
  - un protocole particulier
  - une certaine interprétation de l'URL

URL (2)

- Schéma
  - composé de minuscule [a-z], de chiffres et des signes "+" et "-" (éventuellement encodés sous la forme "% hex").
  - Majuscules sont équivalentes aux minuscules
- Syntaxe générale d'une URL
  - `<schéma>:<partie dépendante du schéma>`

URL Internet

```
<schéma>://<user>:<passwd>@<machine>:<port>/<path>
```

- `<user>` : nom d'utilisateur utilisé uniquement par certains protocoles (exemple `ftp`)
- `<passwd>` : mot de passe pour `<user>`
- `<machine>` : nom de machine ou adresse IP
- `<port>` : port sur lequel se connecter, la plupart des schémas ont un port par défaut
- `<path>` : détail sur la façon d'accéder à la ressource

URL HTTP

```
http://<machine>:<port>/<path>?<searchpart>
```

MIME

- Utilisé pour typer les données retournées
  - `type/sous-type`
- Type et sous-type
  - text : html, plain, xml, ... (;charset=iso8859-1)
  - image : gif, jpeg...
  - audio : mp3, ...
  - video : avi, ...
  - multipart :
  - application : octet-stream, ...

HTTP Caractéristiques

- Protocole unidirectionnel
- Basé sur le paradigme de requête-réponse
- Utilisation d'un port par défaut : 80
- Messages HTTP = requête et réponse
- Messages organisés en lignes
  - Fin de ligne marquées par `\r\n`
    - Carriage return + End of line

Format général des messages

- une ligne de début
  - type de la requête ou de la réponse
- zéro ou n lignes d'en-têtes
  - paramétrage du transfert de données
  - format RFC 822 (`nom: valeur`)
- une ligne vide
  - fin des lignes d'en-tête
- éventuellement un corps de message
  - paramètres de requête ou données de la réponse

Récupérer des données

- La méthode GET
  - Permet la récupération d'une ressource localisée par un URL
  - Exemple en utilisant
    - `telnet www.vuibert.fr 80`
  - une requête ne contenant pas d'en-tête

```
GET http://www.vuibert.fr/som.htm HTTP/1.1¶
¶
```

La méthode GET

- `GET`
  - la méthode à appliquer à la ressource (toujours en majuscule)
- `http://www.vuibert.fr/som.htm`
  - l'URL de la ressource à récupérer
- `HTTP/1.1`
  - le protocole utilisé (toujours en majuscule), si pas de protocole HTTP/0.9
- ¶
  - indique la fin de la requête, le serveur peut envoyer sa réponse

La méthode GET (3)

- En réponse, le serveur retourne par exemple

```
HTTP/1.1 200 OK¶
Server: Netscape-Enterprise/3.5.1C¶
Date: Tue, 08 Jun 2004 10:01:11 GMT¶
Content-Type: text/html¶
Etag: "31bfe-9de-362eef8b"¶
Last-Modified: Thu, 22 Oct 2003 08:40:43 GMT¶
Content-Length: 2526¶
Accept-Ranges: bytes¶
¶
<!doctype ...>¶
<html>¶
... le code HTML du fichier som.htm ...¶
</html>¶
```

La méthode GET (4)

- `HTTP/1.1 200 OK`
  - ligne de statut
  - `HTTP/1.1` protocole utilisé pour la réponse
  - `200` code de statut
    - `1xx` réponse intermédiaire
    - `2xx` succès
    - `3xx` redirection
    - `4xx` erreur client
    - `5xx` erreur serveur
  - `OK` message informatif

La méthode GET (5)

- différents en-têtes
  - majuscules et minuscules indifférenciées
  - `Date` date de génération du message
  - `Content-type` type MIME du corps du message
  - `Content-length` taille du corps du message
- ¶ indique la fin des en-têtes
- corps du message

Le corps du message

- Suite d'octets
- Format précisé par les en-têtes
  - `Content-Type` type des données du corps
  - `Content-Encoding` utilisé si les données du corps ont un codage particulier (par ex. compressées)
  - `Transfer-Encoding` utilisé si le serveur encode les données pour les transférer
- Préférences du client sur le format avec en-têtes `Accept` et `Accept-Encoding`

Le corps du message

- `Accept-Language` permet au client de préciser le langage préféré pour les documents
- `Content-Language` langage choisi par le serveur
- Tout message contenant un corps doit contenir un en-tête `Content-Type`
- Si aucun type n'est donné (`HTTP/0.9`)
  - essayer de déduire le type des données ou de l'URI
  - type `application/octet-stream`

Le corps du message

- La taille du corps déterminée
  - par la présence de l'entête `Content-Length`
  - soit par la fermeture de la connexion dans le cas d'un réponse
  - cas particulier du transfert par morceaux
    - `Transfer-Encoding: chunked`
    - suite de *chunks*
      - une ligne avec la taille en hexadécimal + `\r\n`
      - données
      - `\r\n`

Le corps du message

```
HTTP/1.1 200 OK¶
Date: Fri, 31 Dec 1999 23:59:59 GMT¶
Content-Type: text/plain¶
Transfer-Encoding: chunked¶
¶
1a¶
abcdefghijklmnopqrstuvwxyz¶
10¶
1234567890abcdef¶
0¶
¶
```

```
HTTP/1.1 200 OK¶
Date: Fri, 31 Dec 1999 23:59:59 GMT¶
Content-Type: text/plain¶
Content-Length: 42¶
¶
abcdefghijklmnopqrstuvwxyz1234567890abcdef¶
```

La méthode HEAD

- Similaire à GET mais pas de corps de message
- Par exemple pour vérifier la validité d'une page en cache

```
HEAD /welcome.html HTTP/1.1
Host: www.inria.fr
```

```
HTTP/1.1 200 OK
Date: Tue, 06 Jul 1999 09:47:41 GMT
Server:        /1.2.6
Last-Modified: Mon, 05 Jul 1999 15:08:28 GMT
ETag: "a9d8-1c17-3780ca6c"
Content-Length: 7191
Accept-Ranges: bytes
Content-Type: text/html
```

La méthode TRACE

- Réponse à une requête TRACE contient en corps de message la requête reçue par le serveur

```
TRACE / HTTP/1.1
Host: www.w3.org
```

```
HTTP/1.1 200 OK
Date: Tue, 06 Jul 1999 09:16:22 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
Content-Type: message/http
```

Autres méthodes

- `OPTIONS` demande d'information
- `PUT` installe une ressource sur le serveur
- `DELETE` détruit une ressource sur le serveur

Exemple

```
HEAD /Overview.html HTTP/1.1
Host: www.w3c.org
```

```
HTTP/1.1 301 Moved Permanently
Date: Thu, 01 Jul 1999 08:59:38 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
Location: http://www.w3.org/Overview.html
Content-Type: text/html
```

Exemple 2

```
HEAD /zorglub.html HTTP/1.1
Host: www.w3.org
```

```
HTTP/1.1 404 Not Found
Date: Thu, 01 Jul 1999 09:00:54 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
Content-Type: text/html
```

Exemple 3

```
Je voudrais Overview.html HTTP/1.1
Host: www.w3.org
```

```
HTTP/1.1 400 Bad Request
Date: Thu, 01 Jul 1999 09:09:32 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
Last-Modified: Thu, 05 Nov 1998 17:10:54 GMT
ETag: "2d1d66-3a9-3641dc1e"
Accept-Ranges: bytes
Content-Length: 937
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML
 ... document HTML expliquant l'erreur ...
</HTML>
```

Exemple 4

```
INCONNUE /Overview.html HTTP/1.1
Host: www.w3.org
```

```
HTTP/1.1 501 Method Not Implemented
Date: Thu, 01 Jul 1999 09:18:20 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
Allow: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH,
PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK, TRACE
Connection: close
```

Exemple 5

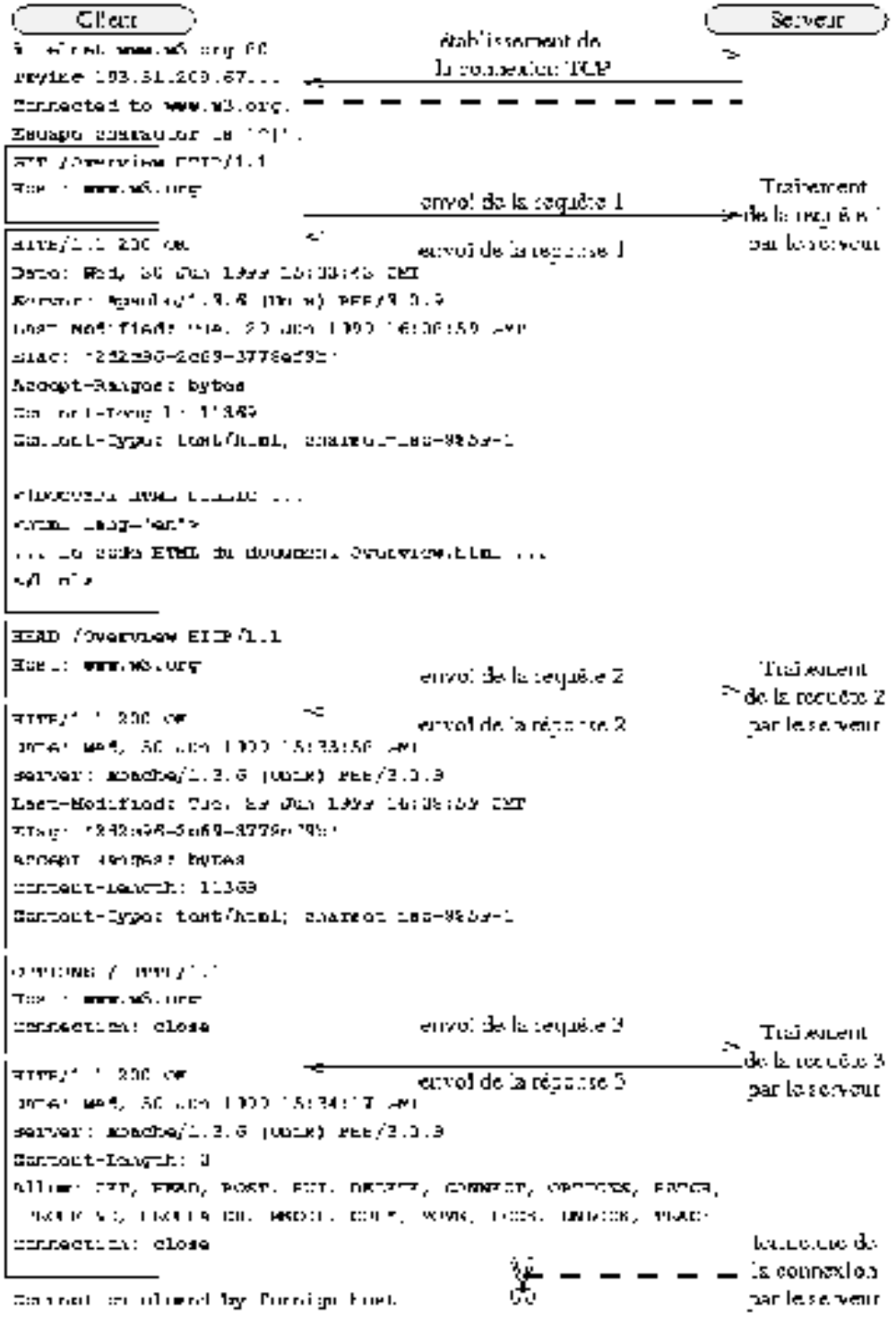
```
GET /Overview.html HTTP/1.1
Host: www.w3.org
If-Modified-Since: Wed, 30 Jun 1999 00:00:00 GMT
```

```
HTTP/1.1 304 Not Modified
Date: Thu, 01 Jul 1999 10:49:30 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
ETag: "2d2c96-2c69-3778ef9b"
```

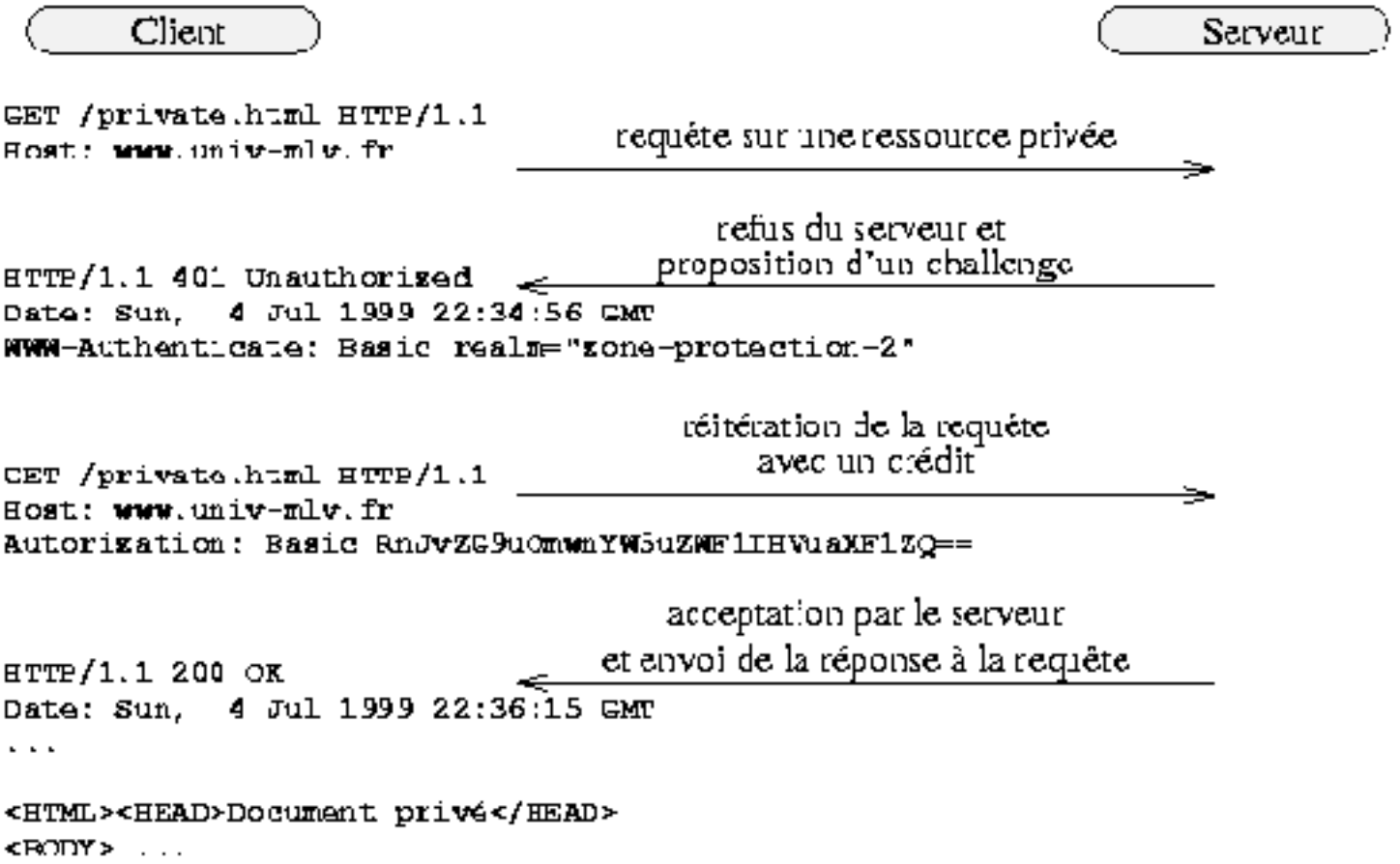
Exemple 6

```
GET /Overview.html HTTP/1.1
Host: www.w3.org
If-None-Match: "2d2c96-2c69-3778ef9b"
```

```
HTTP/1.1 304 Not Modified
Date: Thu, 01 Jul 1999 11:58:52 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.9
ETag: "2d2c96-2c69-3778ef9b"
```

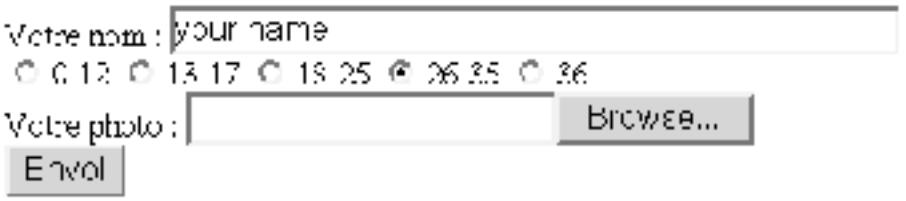


L'authentification



Envoyer des données

- Les formulaires
  - permettent d'ajouter à une page HTML des zones de saisie pour l'utilisateur



Formulaire 2

```
<form enctype="application/x-www-form-urlencoded"
      action="http://igm.univ-mlv.fr/~roussel" method="GET" >
 Votre nom :
 <input type="text" size="40" name="user" value="your name"><br
 Votre age : <input type="radio" name="age" value="0-12"> 0-12
 <input type="radio" name="age" value="13-17"> 13-17
 <input type="radio" name="age" value="18-25"> 18-25
 <input type="radio" name="age" value="26-35" checked> 26-35
 <input type="radio" name="age" value="36-"> 36-<br>
 Votre photo :
 <input type="file" name="photo" size="20" accept="image/*">
 <input type="hidden" name="customerid" value="c2415-345-8563">
 <br>
 <input type="submit" name="test" value="Envoi">
</form>
```

Formulaire (3)

- type=text
- type=checkbox
- type=radio
- type=submit
- type=image
- type=reset
- type=file
- type=hidden

Formulaires 4

- Menus au moyen de SELECT

```
<SELECT NAME="flavor">
   <OPTION VALUE=a>Vanilla
   <OPTION VALUE=b>Strawberry
   <OPTION VALUE=c>Rum and Raisin
   <OPTION VALUE=d>Peach
</SELECT>
```

Envoyer des données (2)

- GET avec application/x-www-form-urlencoded

```
http://www.url.org/path?user=your+name&age=26-35&photo=%2Fhome%
2Fens%2Froussel%2Fgilles.gif&customerid=c2415-345-
8563&test=Envoi
```

- POST

```
POST http://www.url.org/path HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 115

user=your+name&age=26-35&photo=%2Fhome%2Fens%2Froussel%2F-
gilles.gif&customerid=c2415-345-8563&test=Envoi
```

Envoi de fichiers

- encotype=multipart/form-data

```
POST /sendfile HTTP/1.1
Host: server
Content-Type: multipart/form-data; boundary=ABCDEF
Content-Length: 372

--ABCDEF
Content-Disposition: form-data; name="fichier1"; filename="test.txt"
Content-Type: text/plain

Test de fichier
--ABCDEF
Content-Disposition: form-data; name="fichier2"; filename=""
Content-Type: application/octet-stream


--ABCDEF
Content-Disposition: form-data; name="ID"

roussel
--exemple
Content-Disposition: form-data; name="submit"

Envoyer
--ABCDEF--
```

Sauver un état

- URL *rewriting*
  - modifier les liens retournés pour ajouter de l'info
- *Cookie*
  - en-tête ajouté par le serveur lors de sa réponse HTTP
  - stocké par le client (si l'utilisateur le permet)
  - client envoie dans en-tête chaque fois qu'il accède à certains sites

Cookies 2

- Format serveur

```
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH;
domain=DOMAIN_NAME; secure
```

- Plusieurs entêtes `Set-Cookie` peuvent être envoyées par le serveur.

Cookies (3)

- Format client

```
Cookie: NAME1=VALUE1; NAME2=VALUE2 ...
```

- Envoi chaque fois que le client demande un URL correspondant au `domain` et au `path` du *cookie*

Virtual host

- Possibilité d'héberger plusieurs sites sur une même machine
  - Utilisation de plusieurs adresses pour une même machine
  - Utilisation de plusieurs noms qui sont obtenus *via* l'en-tête `Host`