Lyon 1

Le protocole HTTP

Sylvain Brandel

Sylvain.brandel@liris.univ-lyon1.fr

`http://bat710.univ-lyon1.fr/~sbrandel`

M1 Informatique

MIF13 – Programmation web

Université Claude Bernard Lyon 1

UFR d'informatique

Sources

- Supports de cours de **Olivier Glück** (Lyon 1)
- Livres cités en bibliographie
- Le web

- Règle n°1 :
  - Rendre à César…

Les transparents et images proviennent essentiellement des sources citées

Objectifs

- Format des requêtes/réponses
- Durée de vie des connexions, Cookies
- Différentes versions de HTTP, Proxy
- Les requêtes clientes, les réponses du serveur
- Les en-têtes, les types MIME

Caractéristiques de HTTP

- HTTP : Hyper Text Transfer Protocol
- Protocole régissant le dialogue entre des clients Web et un serveur
- Fonctionne en mode client / serveur
- Une transaction HTTP contient
  - le type de la requête ou de la réponse (commande HTTP)
  - une en-tête
  - **une ligne vide**
  - un contenu (parfois vide)
- Très peu de type de requêtes / réponses
- Port standard : 80

Une transaction typique (1)

- Requête du client : client → serveur

1. demande du document `test.html`

```
GET /~sbrandel/test.html HTTP/1.1
```

2. envoi des informations d'en-tête : informer le serveur
   - configuration
   - documents acceptés

```
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: www710.univ-lyon1.fr
Accept: image/gif, image/jpeg
```

3. envoi d'une ligne vide (fin de l'en-tête)
4. envoi du contenu (vide dans cet exemple)

Une transaction typique (2)

- Réponse du serveur : serveur → client

5. code indiquant l'état de la requête

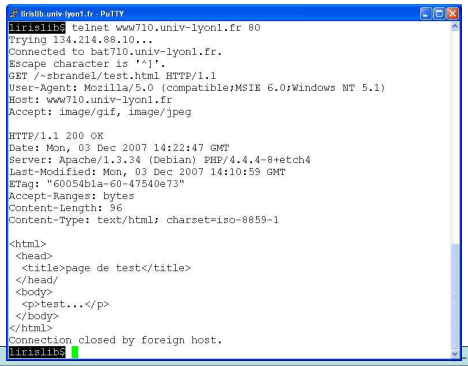
```
HTTP/1.1 200 OK
```

6. envoi des informations d'en-tête : informer le client
   - configuration du serveur
   - document demandé

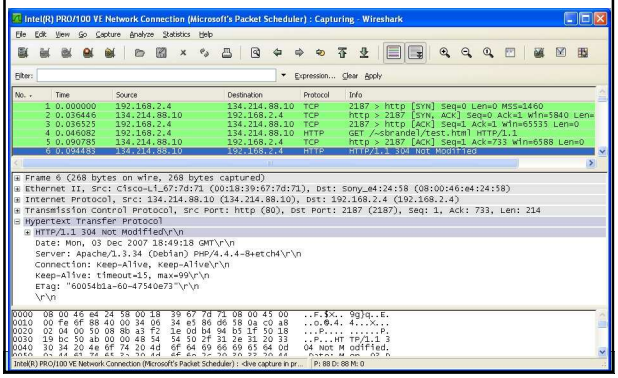
```
Date: Tue, 30 Sep 2008 06:11:28 GMT
Server: Apache/1.3.34 (Debian) PHP/5.2.1
Last-Modified: Tue, 30 Sep 2008 06:11:14 GMT
ETag: "600593b3-61-48e1c302"
Accept-Ranges: bytes
Content-Length: 97
Content-Type: text/html; charset=iso-8859-1
```

3. envoi d'une ligne vide (fin de l'en-tête)
4. envoi du contenu si la requête a réussi

Une transaction typique (3)



Une transaction typique (4)



Format des requêtes / réponses

- Format des requêtes
  - type de la requête (`METHOD, URL, version HTTP`)
  - en-tête
  - une ligne vide
  - un contenu éventuel
- Format des réponses
  - code de la réponse (`version HTTP, code, description`)
  - en-tête
  - une ligne vide
  - le contenu de la réponse

Durée de vie des connexions

- HTTP 1.0 (RFC 1945)
  - dès que le serveur a répondu à une requête, il ferme la connexion HTTP
- HTTP 1.1 (RFC 2068)
  - par défaut, la connexion est maintenue tant que le serveur ou le client ne décide pas de la fermer (`connection: close`)
- HTTP est un protocole sans état
  - aucune information n'est conservée entre deux connexions
  - permet au serveur HTTP de servir plus de clients en un temps donné (gestion légère des transactions)
  - pour conserver des informations entre deux transactions, il faut utiliser un cookie, des champs cachés d'un formulaire...

Cookies

- HTTP : protocole sans état
  - →cookie : moyen pour le serveur de stocker des informations chez le client
- Cookie
  - chaîne de caractères url-encodée de 4ko max stockée sur le disque dur du client
  - informations associées à un ensemble d'URL, utilisées lors de toute requête vers l'une de ces URL
- Les cookies permettent de
  - propager un code d'accès : évite une authentification lors de chaque requête
  - identification dans une base de données
  - fournir des éléments statistiques au serveur : compteurs de pages visitées...

Installation d'un cookie sur le client

- Directive `Set-Cookie` dans l'en-tête de la réponse HTTP (envoyée lors de la première connexion)

```
Set-Cookie: nom=valeur; expires=date; path=chemin_accès;
            domain=nom_domaine; secure
```

  - `nom=valeur` : contenu du cookie, sans espace, point-virgule et virgule (seul champ obligatoire)
  - `expires` : devient invalide après la date d'expiration
  - `path=/pub` : cookie est valable pour toutes les requêtes dont l'URL contient `/pub`
  - `domain` : nom de domaine (associé au serveur) pour lequel le cookie est valable
  - `secure` : le cookie n'est valable que lors d'une connexion sécurisée

Utilisation d'un cookie par le client

- Avant chaque requête, le client vérifie dans sa liste de cookies s'il y en a un qui est associé à cette requête
- Si c'est le cas, le client utilise la directive `Cookie` dans l'en-tête de la requête HTTP

```
Cookie: nom1=valeur1; nom2=valeur2; ...
```

- Le serveur peut insérer plusieurs directives `Set-Cookie`
- Dans la première spécification des cookies :
  - un client peut stocker un maximum de 300 *cookies*
  - un maximum de 20 cookies par domaine est permis
  - la taille d'un *`cookie`* est limitée à 4Ko

Différentes versions de HTTP (1)

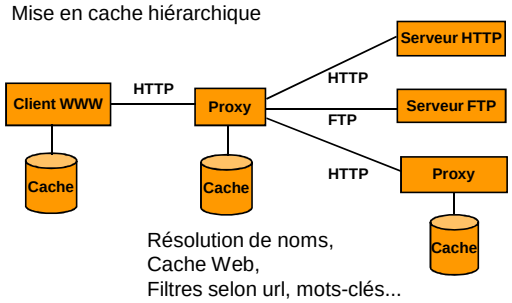
- Version d'origine : HTTP 0.9
  - Une seule méthode : GET
  - Pas d'en-têtes
  - Une requête = une connexion TCP
- Amélioration en 2 étapes
  - HTTP 1.0 :
    - introduction des en-têtes (échange de "méta" info)
    - nouvelles possibilités : utilisation de caches, méthodes d'authentification...
  - HTTP 1.1 :
    - mode connexions persistantes par défaut
    - introduction des serveurs virtuels
      → la directive `Host` dans la requête est nécessaire

Différentes versions de HTTP (2)

- Intérêt des connexions persistantes
  - exemple d'une page d'accueil avec 5 images
    HTTP 0.9 : 6 connexions/déconnexions TCP/IP
    HTTP 1.1 : 1 seule connexion TCP/IP
- Intérêt d'un cache : amélioration des performances
  - les pages qui sont le plus souvent demandées sont conservées dans un cache
    → soulage le réseau
    → accès plus rapide
  - peut être utilisé localement ou par l'intermédiaire d'un serveur relais (*proxy*)

Proxy / cache web

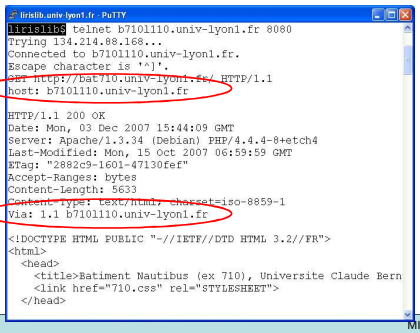
- Mise en cache hiérarchique



Résolution de noms,
Cache Web,
Filtres selon url, mots-clés...

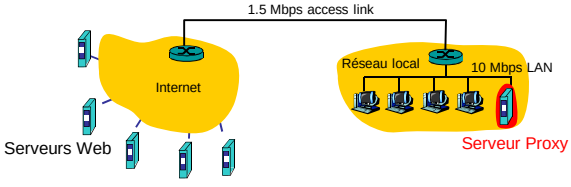
Proxy / cache web

- Proxy Apache



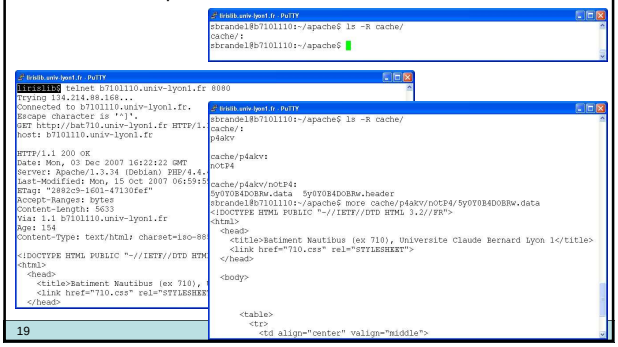
Proxy / cache web

- Hypothèse : le cache est proche du client
- Réduction du temps de réponse
- Réduction du débit vers les serveurs distants



Proxy / cache web

- Cache web Apache



Les requêtes du client

- Rappel : Format d'une requête
  - une commande HTTP (METHOD), une URL qui identifie la ressource demandée, la version de HTTP
  - l'en-tête
  - une ligne vide
  - éventuellement corps de la requête (contenu)
- Méthode GET
- Méthode POST
- Méthode HEAD
- D'autres méthodes (pas toujours supportées par les serveurs)

La méthode GET

- La méthode standard de requête d'un document
  - récupérer un fichier, une image...
  - activer un script CGI en lui transmettant des données
- Le corps (contenu) de la requête est toujours vide
- Réponse du serveur :
  - une ligne décrivant l'état de la requête
  - un en-tête
  - une ligne vide
  - contenu demandé
- En cas d'échec
  - le contenu de la réponse décrit la raison de l'échec : fichier non présent, problème de droit

La méthode GET et les CGI

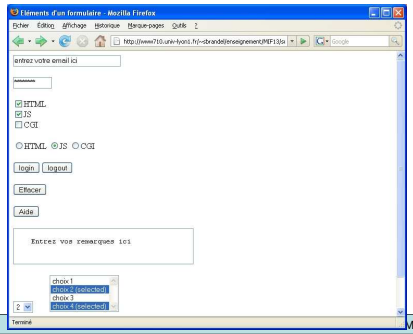
- Transmission des données dans l'URL après un ?
- Les champs sont séparés par un &

```
GET /cgi-bin/prog.cgi?email=toto@site.fr&pass=toto&s=login HTTP/1.1
```

- Ici, trois champs du formulaire sont transmis dans la requête
- Le mot de passe est transmis en clair
- Permet de conserver dans un bookmark les données saisies dans le formulaire
- L'URL a une taille limitée (4Ko)

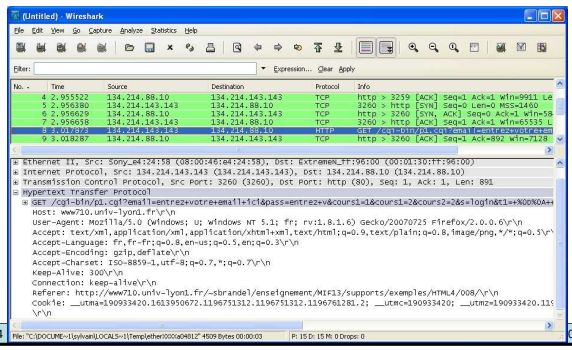
La méthode GET et les CGI

- Exemple de formulaire



La méthode GET et les CGI

```
<form name="f1" method="GET" action="/cgi-bin/p1.cgi">
```



La méthode POST

- Transmission des données dans le corps de la requête

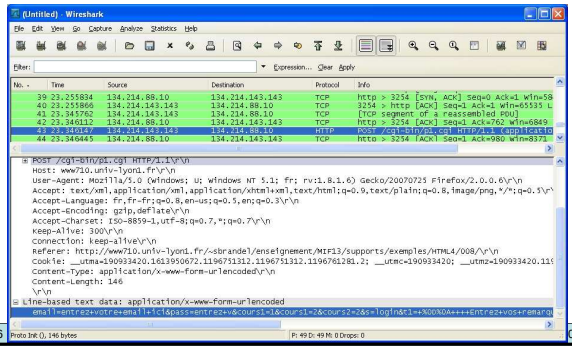
```
POST /cgi-bin/prog.cgi HTTP/1.1
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: localhost
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 36

email=toto@site.fr&pass=toto&s=login
```

- Le mot de passe est toujours transmis en clair

La méthode POST et les CGI

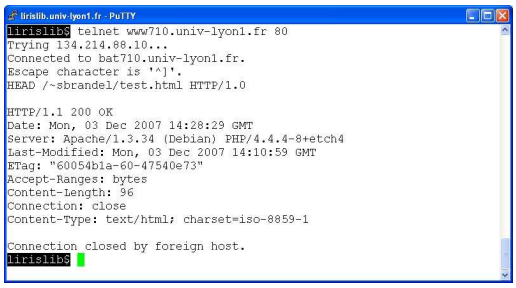
```
<form name="f1" method="POST" action="/cgi-bin/p1.cgi">
```



La méthode HEAD (1)

- Identique à GET mais permet de récupérer uniquement l'en-tête relatif à un document
- Corps de la requête toujours vide
- Pour un document, récupérer
  - date de dernière modification (caches, JavaScript)
  - taille (estimation du temps d'arrivée du document)
  - type (le client peut sélectionner le type de documents qu'il accepte)
- Récupérer le type du serveur
  - permet de faire des requêtes spécifiques selon le type du serveur
- Remarque : le serveur ne fournit pas nécessairement toutes ces informations

La méthode HEAD (2)



Autres requêtes clientes

- PUT : stocker le corps de la requête sur le serveur à l'URL spécifiée
- DELETE : suppression du document spécifié par l'URL
- OPTIONS : renvoie la liste des méthodes autorisées par le serveur
- TRACE : la corps de la requête entrante est renvoyée au client (débogage)
- ...

Les réponses du serveur

- Les codes de réponse
  - version HTTP, code de statut, description du code

```
HTTP/1.1 200 OK
HTTP/1.1 404 Not Found
```

- code=entier sur 3 chiffres classé selon des catégories
  - 100-199 : message d'information
  - 200-299 : succès de la requête cliente
  - 300-399 : la requête n'est pas directement serviable, le client doit préciser certaines choses
  - 400-499 : échec de la requête dû au client
  - 500-599 : échec de la requête dû au serveur (échec CGI)

Quelques en-têtes de requêtes

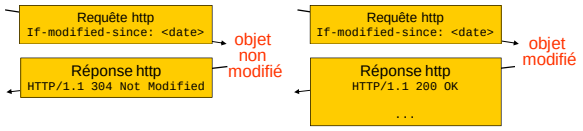
- Identification du client
  - `From` : adresse mail du client
  - `Host` : serveur, **obligatoire en HTTP1.1**
  - `Referer` : URL d'où l'on vient
  - `User-Agent`
- Préférences du client
  - `Accept` : liste des types MIME acceptés
  - `Accept-Encoding` : compress, gzip...
  - `Accept-Langage`
  - `Accept-Charset`

Quelques en-têtes de requêtes

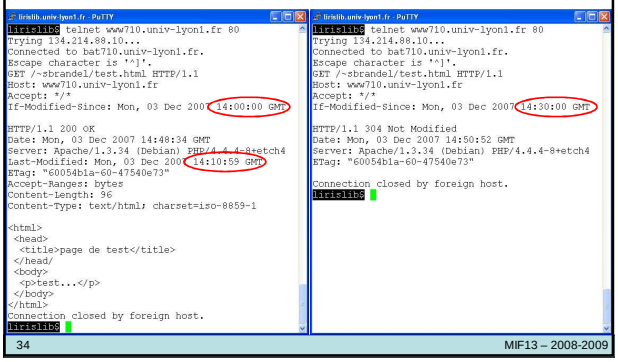
- Information pour le serveur
  - `Autorization` (`username:passwd` encodé en base64)
  - `Cookie`
- Conditions sur la réponse
  - `If-Modified-Since` : utile pour les caches
  - `If-Unmodified-Since`
  - `If-Match` (`Etag`)

Quelques en-têtes de requêtes

- Objectif : ne pas envoyer un objet que le client a déjà dans son cache
- Problème : les objets contenus dans le cache peuvent être obsolètes
- Le client spécifie la date de la copie cachée dans la requête http : `If-modified-since: <date>`
- la réponse du serveur est vide si la copie cachée est à jour



Quelques en-têtes de requêtes



Quelques en-têtes de réponses

- Contenu du document
  - `Content-Type` : type MIME du document
  - `Content-Length` : barre de progression du chargement
  - `Content-Encoding`, `Content-Location`, `Content-Langage`
- Document lui-même
  - `Last-Modified` : date de dernière modification
  - `Allow` : méthodes autorisées pour ce document
  - `Expires` : date d'expiration du document
- En-tête générales
  - `Date` : date de la requête
  - `Server` : type du serveur

Transfert par morceaux (HTTP/1.1)

- La réponse peut être envoyée en plusieurs morceaux
- Cas des CGI : le serveur ne peut pas toujours déterminer la longueur totale de la réponse

```
Transfer-Encoding: Chunked
```

- Chaque morceau est constitué d'une ligne :
  - taille du morceau en hexadécimal
  - données
- Après les morceaux, une ligne :
  - 0 (zéro)
  - éventuellement des en-têtes supplémentaires

Les types MIME

- MIME : Multi-purpose Internet Mail Extensions
- Echange de fichiers multimédias entre machines quelconques en spécifiant le type du fichier
- Composition
  - type général : `text`, `image`, `audio`, `video`, `application`...
  - sous-type : `image/gif`, `image/jpeg`, `application/pdf`, `application/rtf`, `text/plain`, `text/html`

Les types MIME

- Les commandes MIME ont été intégrées dans HTTP1.0
- En perpétuelle évolution
- La machine cliente doit ensuite associer l'exécution d'une application à chaque type MIME
- Le serveur positionne `Content-type` à partir de l'extension du document demandé (`/etc/mime.types`)