The HTTP protocol

Encodage

Qu'est-ce qu'un byte? Qu'est-ce qu'un caractère?

Représenter une séquence de caractères par une séquence de bytes

Représenter une structure arborescente par une séquence de caractères

Représenter un document par une structure arborescente

La vie d'une requête

`http://{host}/{rest}`

1. Contacter le serveur DNS pour obtenir l'adresse IP de `{host}`
2. Se connecter (par TCP/IP) à cette adresse IP
3. Envoyer la requête:

```
GET /{rest} HTTP/1.1
Host: {host}
```

4. Recevoir une réponse, telle que

```
HTTP/1.0 200 OK
Content-Type: text/html

<html><head>...</head><body>...</body></html>
```

DNS - Domain Name Service

Système distribué mondial

Table associative hiérarchique

serveurs racines responsables du domaine `.`

Autres serveurs responsables de `.ca.`, puis de `.umontreal.ca.`

Usage de réplication et de "*caches*" pour supporter la charge

Plusieurs infos par nom (A, AAAA, MX, NS, ...)

Un nom peut correspondre à un autre nom ou une liste d'adresses IP

Requêtes qui viennent de FORM

```
GET /{rest}?{params} HTTP/1.1
```

Requêtes en *lecture* seulement!

Les paramètres ont la forme:

```
{params}    =    {key1}={val1}&{key2}={val2}...
```

Encodage d'une table associative en un séquence de bytes

Le `Host:{host}` est une autre table associative

Requêtes POST

Pour les cas où on veut avoir un *effet* sur le serveur

```
POST /{rest} HTTP/1.1
Host: {host}
Content-Type: {type}

{content}
```

Pas de limite de taille

Contenu pas visible dans votre browser ou votre historique

Browser sait qu'il ne faut pas le renvoyer!

Type populaires: `application/x-www-form-urlencoded` et `multipart/form-data`

HTTPS

HTTP trafic est librement lisible en chemin

HTTPS utilise la cryptographie pour cacher l'information en transit

Un serveur HTTP peut en cacher un autre

HTTPS utilise la cryptographie pour authentifier le serveur

Protocol TLS Transport Layer Security:

authentification et établissement de clé

Attaques MITM

Cookies

Reçus dans les réponses HTTP via:

```
Set-Cookie: {key}={value}; {attributs}
```

Renvoyés par

```
Cookie: {key1}={value1}; {key2}={value2}; ...
```

lors des envois suivants

Attributs: `Expires`, `Domain`, `Secure`, `Path`

Formulaires

La base de l'interaction en "Web 1.0"

Utilisés pour:

- Spécifier une requête de recherche d'information
- Fournir une interactivité
- Envoyer de l'information à un serveur

Types de formulaires:

- Recherche, commande, inscription, ajout à un panier, sondage, menu, login, ...

Élément FORM

Regroupe et organise des éléments d'entrée

Attributs:

- `action`: Indique la destination de la requête, sous forme d'URL
- `method`: Spécifie le type de requête.
  - `GET`: requête en lecture seulement
  - `POST`: envoi d'information qui affecte l'état du serveur

Éléments d'entrée

Attribut `name`, unique (par formulaire), utilisé comme nom d'argument

Beaucoup de types différents:

- Boîtes de textes: `input` avec `type="text"`
- Boutons: `input` avec `type="button"`
- Check box: `input` avec `type="checkbox"`
- Listes déroulantes: `select+option`
- Grosses boîtes de texte: `textarea`

Exemple de formulaire

```
<form method="get">
 <input type="hidden" name="rep"
        value="bugit" />
 Show issue:
 <input type="text" name="issue"
        placeholder="{number or name}" />
</form>
```

Envoie une requête de la forme:

```
GET /{script}?rep=bugit&issue={mytext} HTTP/1.1
Host:  {host}
```

Pas de bouton “submit”, mais `RET` dans le texte envoie le formulaire

Exemple (suite)

attribut `action="{URL}"` utilise la même “page” par défault

Peut résulter en une requête telle que

```
GET /{script}?rep=bugit&issue=h%26%23215 HTTP/
Host: {host}
```

Common Gateway Interface

Standard original pour générer des pages dynamiquement

Chaque requête `http://{host}/{script}?{args}`
exécute le programme `{script}`

⇐ Le programme reçoit une description de la requête

dans son *environnement*

⇒ Le programme renvoie la réponse

au format attendu par HTTP ou presque

CGI environnement

`REQUEST_METHOD` indique si c'est un `GET` ou `POST`

`QUERY_STRING` contient les *arguments*

`SCRIPT_NAME` contient le `/{script}`

`HTTP_HOST` contient le nom d'hôte utilisé

`HTTP_USER_AGENT` contient le nom du browser

...

CGI output

Format de sortie du protocole HTTP sauf:

Pas de première ligne genre `HTTP/1.0 200 OK`

À la place: `Status:`

Et `Location:`