TP programmation réseau

Module FAR

Polytech Montpellier – IG3

David Delahaye

Objectifs

- Écrire un serveur et un client HTTP.
- HTTP = HyperText Transfer Protocol.
- Protocole de communication client-serveur développé pour le Web.
- Protocole de la couche application (la boucle est bouclée dans notre voyage initiatique !).
- Fonctionne sur n'importe quelle connexion fiable, dans les faits on utilise le protocole TCP (port 80).

delahaye@herbrand: ~ — Echange de fichiers en utilisant protocole HTTP

```
delahaye@herbrand:~$ telnet www.lirmm.fr 80
Trying 193.49.104.233...
Connected to saturne.lirmm.fr.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.lirmm.fr

HTTP/1.1 200 OK
Date: Thu, 23 Feb 2017 17:58:05 GMT
Server: Apache
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu, 23 Feb 2017 17:58:09 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
X-Powered-By: eZ Publish
Served-by: www.lirmm.fr
Content-language: fr-FR
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8

6607
<!DOCTYPE html>
<html lang="fr-FR">
<head>

<title>Page d'accueil
             /         Lirmm.fr
             /
                - LIRMM</title>    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="/extension/lirmm_site/design/li
rmm_design/stylesheets/reset.css"/>
```

port de connection

← pour chercher fichier (par défaut index.html)

⚠ caractère de fin de ligne : \r\n

6607 → nb d'octets qu'il va transférer

Méthodes HTTP

- Commandes spécifiant un type de requête.
- Requête = le client demande au serveur d'effectuer une action.
- L'action concerne une ressource identifiée par l'URL qui suit le nom de la méthode.
- Plusieurs méthodes : GET, POST, HEAD (les plus courantes, mais 9 méthodes au total).

Méthode GET

A envoyer au serveur via socket

- Un entête et pas de corps dans la requête GET.
- Entête (HTTP 1.1) :
  - Host : précise le site web concerné par la requête. (Obligé ←)
  - User-Agent : indique la signature du programme effectuant la requête.
  - Connection : keep-alive (si connexion persistante) ou close (si connexion fermée après réponse). (→ durée dizaine de secondes au cas où nvelle demande au serveur)

Réponse à la méthode GET

- Première ligne : status ; plusieurs codes : 200 (OK), 404 (page non trouvée), etc.
  ↳ pb côté serveur
- Date : moment auquel le message est généré.
- Server : indique le serveur HTTP.
- Content-Type : indique le type MIME de la ressource.
  ↳ pour qualifier les données (image ...)
- Content-Length : indique la taille en octets de la ressource envoyée.

delahaye@herbrand: ~

```
delahaye@herbrand:~$ telnet www.lirmm.fr 80
Trying 193.49.104.233...
Connected to saturne.lirmm.fr.
Escape character is '^]'.
GET /~delahaye/ HTTP/1.1
Host: www.lirmm.fr

HTTP/1.1 200 OK
Date: Thu, 23 Feb 2017 18:49:37 GMT
Server: Apache/2.2.22 (Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1
Vary: Accept-Encoding
Content-Length: 6627
Content-Type: text/html
Via: 1.1 www.lirmm.fr
Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--

        Design by TEMPLATED
        http://templated.co
        Released for free under the Creative Commons Attribution License

        Name        : Coffeelike
        Version     : 1.0
        Released    : 20130222

-->
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
```

→ cherche le fichier racine dans ~delahaye (index.php)

Rq : pour récupérer des images dans une page web, il faut faire d'autres requêtes.

Exercice n°1 : le client

- Le client doit demander l'URL à l'utilisateur (domaine et chemin jusqu'au fichier désiré ; si aucun fichier indiqué, on veut « index.html »), ainsi que le port de connexion (par défaut, on dira 80). → port fixe
- On doit ensuite trouver l'IP du domaine indiqué.
- Puis on ouvre une connexion TCP sur l'IP correspondante avec le port indiqué.
- On envoie la requête GET sur la connexion ouverte.
- Enfin, on récupère la réponse.

Exercice n°1 : le client

- Avec la réponse récupérée, on pourra :
  - L'afficher simplement (sans l'interpréter) ;
  - La transmettre à un navigateur qui pourra l'interpréter et l'afficher avec la mise en forme.
- Test du client :
  - Le tester avec le serveur Appache du LIRMM (hôte : www.lirmm.fr).

Exercice n°2 : le serveur

- Le serveur doit pouvoir être lancé sur n'importe quel port (demandé à l'utilisateur au démarrage), ainsi qu'avec un répertoire de référence quelconque (là où on va chercher les fichiers).
- Il doit pouvoir gérer plusieurs connexions simultanément (par défaut, on sera en mode de connexion « close »). On utilisera « fork » dans un premier temps.
- Il répondra uniquement aux requêtes GET dans un premier temps (les autres requêtes seront ignorées).

Exercice n°2 : le serveur

- Test du serveur :
  - Avec le client précédemment écrit ;
  - Avec un navigateur (soit en local, soit à partir d'une autre machine).
- On donnera une autre version du serveur avec des « threads » au lieu d'utiliser « fork ».