LFM

LES PROTOCOLES TCP/IP

et

L'USAGE DES RFCs

SOMMAIRE :

Les protocoles TCP/IP et les RFCs

Les protocoles permettent à un serveur de dialoguer avec ses clients. Ils décrivent la manière d'échanger requêtes et réponses entre le serveur et le client.

Les protocoles doivent être indépendants d'une architecture matérielle ou logiciel, ou d'un fabricant… C'est à dire ouverts. Ils sont accessibles à tous, et leur développement s'effectue par consensus, et non par décision autoritaire, ou unilatérale. Comme ils sont ouverts, n'importe qui et libre de développer des produits qui utilisent les spécifications de ces protocoles.

La plupart de ces protocoles ouverts sont publiés dans les RFC : *Requests For Comments*. Ces documents sont le produit de groupes de travail, ils sont ouverts à améliorations, et contiennent un panel très large d'informations intéressantes et utiles dans la pratique.

Sites Internet de référence :

- IETF : http://www.ietf.org/
- IETF – RFCs : http://www.ietf.org/rfc.html
- IESG : http://www.ietf.org/iesg.html
- Internet Society – RFC Editor : http://www.rfc-editor.org/
- FAQS – Archives RFCs : http://www.faqs.org/rfcs/
- World Wide Web Consortium – RFC2616 : http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html

Etude de quelques protocoles

http : Hypertext Transfert Protocol

Présentation

Pour tester le protocole http, nous allons consulter le RFC2616 (mis à jour par les RFC 7230 à 7237). Pour plus de confort, nous utiliserons la version html du site de l'IETF et celle du site du W3C.

Le RFC fait 176 pages, mais heureusement, il y a un sommaire. Dirigeons nous vers la section 5, elle explique comment dialoguer avec le serveur :

> The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.
>
> Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Pour que ce soit clair, voici ce qui est inscrit : pour consulter une page, il faut :

- taper la méthode, i.e. ce que l'on veut faire, suivi d'un espace,
- puis de l'URI, i.e. l'adresse et le chemin de la ressource, encore un espace,
- puis la version de protocole utilisée (0.9, 1.0 ou 1.1), et un retour à la ligne CRLF (carriage return line feed).

La méthode la plus courante est la méthode GET qui permet d'obtenir une ressource. Suite à une méthode GET, on reçoit l'en-tête (HEAD, dépend du serveur) et le corps (BODY, le document lui même). Procédure :

- Ligne de requête + CRLF,
- Lignes de définition de champs optionnelles suivies d'un CRLF,
- CRLF pour mettre le serveur au travail.

Tests

Nous allons donc procéder à des tests afin de mettre en œuvre ce protocole http.

Avec *Windows*, il est possible d'utiliser le programme gratuit *Putty* (taper *putty* dans un moteur de recherche, aller sur la page de téléchargement (*download*), et le récupérer …).

Il faut se mettre en mode *Raw*, port *80* (à moins que votre serveur http écoute un autre port, 80 est le port standard pour le protocole http !). Taper l'adresse, par exemple *www.vije.net*, puis sauvegarder la session sous un nom adéquat. Enfin cliquer *Open*.

Lorsqu'une session est terminée,

- avec un clic droit dans la barre de programme, et en choisissant New session, on peut ouvrir une nouvelle session dans la même fenêtre,
- avec un clic droit dans la barre de programme, et en choisissant Duplicate session, on ouvre une nouvelle fenêtre en utilisant la même définition de session.

Avec *Linux*, il suffit d'ouvrir une fenêtre de terminal (menu KDE *Système*), et taper : telnet <*adresse*> [*port*]

Protocole http 1.1, le standard depuis la fin des années 90. Il y a deux points cruciaux :

- il faut fournir le champ hôte (*Host*),
- la session n'est pas automatiquement fermée par le serveur, car le protocole http 1.1 suppose la session persistante, et la conserve ouverte un certain temps, jusqu'à ce que le délai sans requête configuré dans le serveur se soit écoulé (timeout). On peut éviter cela avec le champ Connection : close ! Cela a été introduit pour permettre à un client ayant chargé une page de pouvoir en charger les ressources (css, images, script, etc …) sans avoir à rouvrir une session pour chaque ressource.

Exemple 1 avec http 1.1 (si vous taper trop lentement, la session se fermera, recommencez plus vite !) :

- Terminal : *telnet moodle.lyceestendhal.it 80*
- Taper : *GET / http/1.1 <CRLF>Host: moodle.lyceestendhal.it <CRLF><CRLF>*

  Nous obtenons la page d'accueil par défaut du site lsmi.it : *moodle.lyceestendhal.it/*index.php

Exemple 2 :

- Terminal : *telnet moodle.lyceestendhal.it 80*
- Taper : *GET /index.php http/1.1 <CRLF>Host: moodle.lyceestendhal.it <CRLF><CRLF>*

  Nous obtenons la même page d'accueil du site lsmi.it : *moodle.lyceestendhal.it/*index.php

Vous voilà initié avec votre premier protocole. Libre à vous de tester les autres méthodes et champs …

SMTP : Simple Mail Transfert Protocol

Nous allons aller un peu plus vite maintenant ! Cet exemple ne fonctionne qu'à partir du lycée où d'un endroit enregistré sur un compte du site smtp2go.com

Exemple à faire : *taper les lignes suivantes (C=client) et relever la réponse du serveur (S=serveur) :*

- telnet mail.smtp2go.com 25

  S :
- C : HELO mail.smtp2go.com <CRLF>

  S :
- C : MAIL FROM: <*votre adresse email ou une inventée*><CRLF>

  S :
- C : RCPT TO: <*votre adresse email*><CRLF>

  S :
- C : RCPT TO: <*adresse email d'un autre apprenant*><CRLF>

  S :
- C : DATA<CRLF>

  S :
  - C : DATE: Thu, 21 May 1998 05:33:29 -0500<CRLF>
  - C : FROM: <*votre adresse ou une inventée même différente de celle de MAIL FROM*><CRLF>
  - C : TO : <*liste d'adresses séparées par des virgules même différentes de RCPT*><CRLF>
  - C : SUBJECT : test<CRLF>
  - C : <CRLF>
  - C : Ceci est un test…<CRLF>
  - C : etc…<CRLF>
  - C : .<CRLF>

  S :
- C : QUIT<CRLF>

  S :

C'est fini. Consultez votre courrier dans le navigateur, ou bien avec un client courrier, ou encore avec le protocole POP3 (voir son RFC) et si vous avez bien tout rentré, le message devrait apparaître dans votre boite !

Maintenant, mettez-vous d'accord avec un autre apprenant, et échangez-vous des courriels forgés ainsi, en faisant apparaitre qu'ils viennent d'une autre personne, et pourquoi pas éventuellement avec une autre date.

A noter : pour faire ceci à la main facilement, il faut avoir un serveur SMTP qui parle en texte clair, ou bien encodé. En effet, s'il faut crypter l'échange, come avec le SMTP de Google, alors mieux vaut utiliser un logiciel client de courriel comme Thunderbird ! Dans ce type de logiciel, on peut au moins forger l'adresse d'émetteur !