Introduction Web et Services Web :
Mon premier serveur Web

Ce TD a pour but de vous faire développer votre propre serveur Web que vous pourrez tester avec un simple telnet localhost 8080 puis via un navigateur Web standard. Vous pourrez vous aider pour cela des informations du cours et/ou celles de : http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

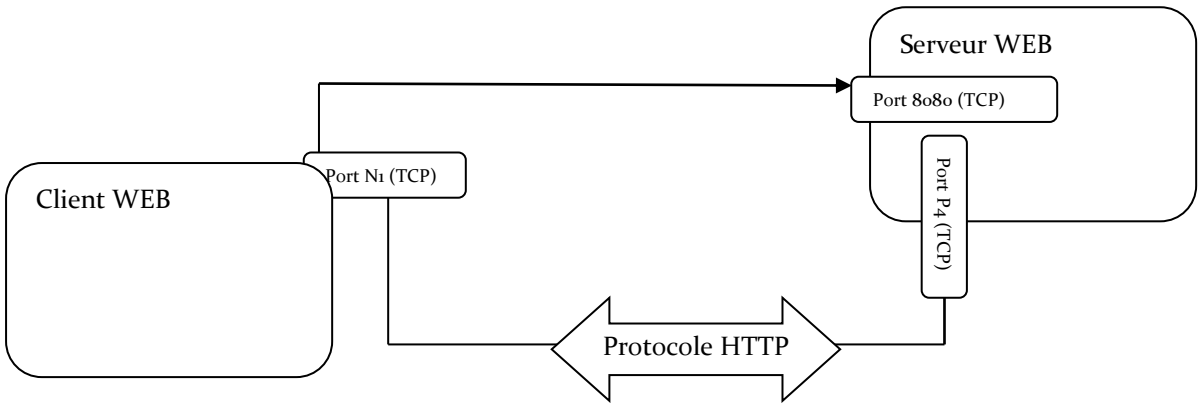
1 Description de « l'architecture » d'un serveur Web

Pour vous aider voici quelques informations et questions :

- Un serveur Web est un serveur socket mode connecté (TCP/IP). Vous pourrez donc réutiliser votre TD TCP/IP python pour vos développements.
- Le port standard pour un serveur WEB est le port 80.

question 1 : Pourquoi devons-nous utiliser ici le port 8080 ?

- Après avoir crée la connexion TCP/IP un serveur dialogue avec un client en utilisant le protocole HTTP (Cf. cours et/ou RFC 1945).



2 Tests avant de commencer

Commencer par tester à l'aide de la commande telnet* (client TCP/IP générique, console connectée à une adresse IP:port quelconque), la mise en place d'un dialogue avec un serveur existant. Vous pourrez par exemple tester la commande suivante (au cas où, installez telnet sur votre machine) :

```
telnet www.tigli.fr 80
GET /
```

Quel est le résultat obtenu ?

(*) pour installer telnet sous windows, utiliser la commande `pkgmgr /iu:"TelnetClient"` ou Assistant Ajout de fonctionnalités du panneau de configuration

(*) pour installer telnet sous Ubuntu, sudo apt-get install telnet

De même installez choisissez votre browser web préféré (Firefox ou Google Chrome), chargez l'outil de développement (outils de développement / onglet Network ou Réseau), et demandez la page :

```
http://www.google.com
```

Combien de requêtes HTTP sont déclenchées pour récupérer l'intégralité de la page ?
Mesurez et comparez les durées de récupération de la page www.google.com avec ou sans utilisation du cache local. Quel est le % de gain de temps ? de bande passante ?

3 Mise en œuvre d'un serveur HTTP basique

Commencer par créer un serveur qui prenne en compte une requête HTTP simple du type :

- *GET <chemin relatif d'un fichier HTML par rapport à la racine du site Web>*
- *Exemple GET /index.html, correspond pour le serveur à l'envoi du fichier /www/pub/index.html*

Voici un extrait d'un exemple de fichier index.html pour vos tests :

```
<TITLE> L'exemple HTML le plus simple</TITLE>
<H1> Ceci est un sous-titre de niveau 1</H1>
Bienvenue dans le monde HTML. Ceci est un paragraphe.
<P> Et ceci en est un second. </P>
<A HREF="index.html">cliquez ici</A> pour réafficher
```

Vous veillerez à déclarer une variable d'environnement HTTP_ROOT qui permette de spécifier le répertoire racine de l'arborescence des documents accessibles depuis le serveur WEB.

N'oubliez de respecter le protocole en renvoyant une ligne d'entête dans la réponse HTTP, comme :

```
http/1.0 200 OK
```

suivi d'un saut de ligne avant le contenu HTML

4 Création de pages Web dynamiques

Pour le moment les pages Web récupérées sont dites statiques. En effet les pages renvoyées au client sont celles présentes et occasionnellement modifiées par leur propriétaire.

Lors de la consultation d'une page Web statique, un serveur HTTP renvoie donc le contenu du fichier où la page est enregistrée. Lors de la consultation d'une page Web dynamique, un serveur HTTP transmet la requête au logiciel correspondant à la requête, et le logiciel se charge de générer et envoyer le contenu de la page.

De nombreuses solutions logicielles sont développées pour faciliter et améliorer la génération de pages Web dynamiques. Citons par exemple les langages PHP, JavaServer Pages (JSP) ou Active Server Pages (ASP)…

Pour bien comprendre que ce principe ne dépend pas d'une technologie donnée, nous allons mettre en œuvre une des toutes premières approches pour la création de page web dynamique : les cgi-bin.

Nous allons maintenant ajouter l'appel à un cgi-bin (soit l'exécution autorisée d'un binaire qui pourra générer une page Web dynamique). Les paramètres sont alors passés après le "?", séparés par un "&", dans l'URL.

Introduction Web et Services Web : *Mon premier serveur Web*

Exemple : pour l'exécution du programme « bonjour jean pascal» dont le fichier bonjour est situé dans $HTTP_ROOT/cgi-bin, l'appel sera :

```
http://localhost:8080/cgi-bin/bonjour?nom=jean&prenom=pascal
```

La page Web retournée sera alors :

```
<HTML>
<HEAD>
<TITLE>Doc. Produit par un CGI</TITLE>
</HEAD>
<BODY>
<H1>Coucou jean pascal !<H1>
</BODY>
</HTML>
```

Attention, n'oubliez pas de rajouter la ligne d'introduction d la réponse l'entête http.

N'oubliez pas non plus de mentionner en tête de votre fichier cgi-bin si ce dernier n'est pas exécutable mais nécessite un interprète le programme qu'il faut lancer (ex # ! /bin/sh, #!/usr/bin/python, #!/usr/bin/perl …).

Dans le cas d'un script shell cgi-bin, nombre de variable d'environnement du processus lancé sont positionnées. Exemple : `HTTP_USER_AGENT, SERVER_ADMIN, SERVER_SOFTWARE,…`

La variable d'environnement qui permet la récupération des paramètres de l'URL dans n'importe quel cgi-bin est conventionnellement `QUERY_STRING.`

Exemple : `QUERY_STRING=namex=valuex&namey=valuey&namez=valuez`

Notons qu'un grand nombre de variables d'environnement sont positionnées par le serveur WEB lors de l'appel à un cgi-bin :

Key	Value
DOCUMENT_ROOT	The root directory of your server
HTTP_COOKIE	The visitor's cookie, if one is set
HTTP_HOST	The hostname of the page being attempted
HTTP_REFERER	The URL of the page that called your program
HTTP_USER_AGENT	The browser type of the visitor
HTTPS	"on" if the program is being called through a secure server
PATH	The system path your server is running under
QUERY_STRING	The query string (see GET, below)
REMOTE_ADDR	The IP address of the visitor
REMOTE_HOST	The hostname of the visitor (if your server has reverse-name-lookups on; otherwise this is the IP address again)
REMOTE_PORT	The port the visitor is connected to on the web server
REMOTE_USER	The visitor's username (for .htaccess-protected pages)
REQUEST_METHOD	GET or POST
REQUEST_URI	The interpreted pathname of the requested document or CGI (relative to the document root)
SCRIPT_FILENAME	The full pathname of the current CGI
SCRIPT_NAME	The interpreted pathname of the current CGI (relative to the document root)
SERVER_ADMIN	The email address for your server's webmaster
SERVER_NAME	Your server's fully qualified domain name (e.g. www.cgi101.com)
SERVER_PORT	The port number your server is listening on

SERVER_SOFTWARE	The server software you're using (e.g. Apache 1.3)

5 Votre premier serveur M2M over Web

Les échanges jusque là concernent un browser, donc une interface utilisateur et une application cgi-bin hébergée sur un serveur Web. Ce type d'utilisation entre dans la catégorie des applications H2M (Human to Machine) où la vocation du Web est de fournir un grand nombre de sources d'informations dynamiques ou non à un utilisateur. Or aujourd'hui le Web est aussi devenu une technologie de communication entre programmes soient des applications dites Web M2M (Machine to Machine).

C'est ce principe qui sera utilisé pour les Services Web dans la mesure où le programme "client" n'est pas un browser Web pour visualiser les données retournées mais une application logicielle quelconque.

A partir des questions précédentes, vous pouvez vous convaincre de la simplicité de ce concept **en mettant en place un programme client qui non seulement enverra des paramètres dans une URL (comme paramètres d'appel d'une fonction) mais récupérera les données de retours dans les données renvoyées par le serveur**.

Vous pouvez par exemple invoquer une méthode incr <val> (qui incrémente la valeur de val) depuis un client TCP/IP qui n'est plus un navigateur WEB. La méthode incr correspondra à l'invocation d'un cgi-bin de même nom (exemple : http://localhost:8080/cgi-bin/incr?val=5). Il vous appartiendra alors de définir le format du contenu du message de réponse du cgi-bin qui ne soit plus de l'HTML mais un format lisible par le client (ex. incr OK val=6).

Nous sommes ainsi devant une technique qui permet d'implémenter de multiples patterns de communication entre applications réparties comme le pattern RPC (Remote Procedure Call) pour l'invocation à distance.