TD Applications Réparties
Introduction du Web pour les Applications Réparties : du H2M au M2M

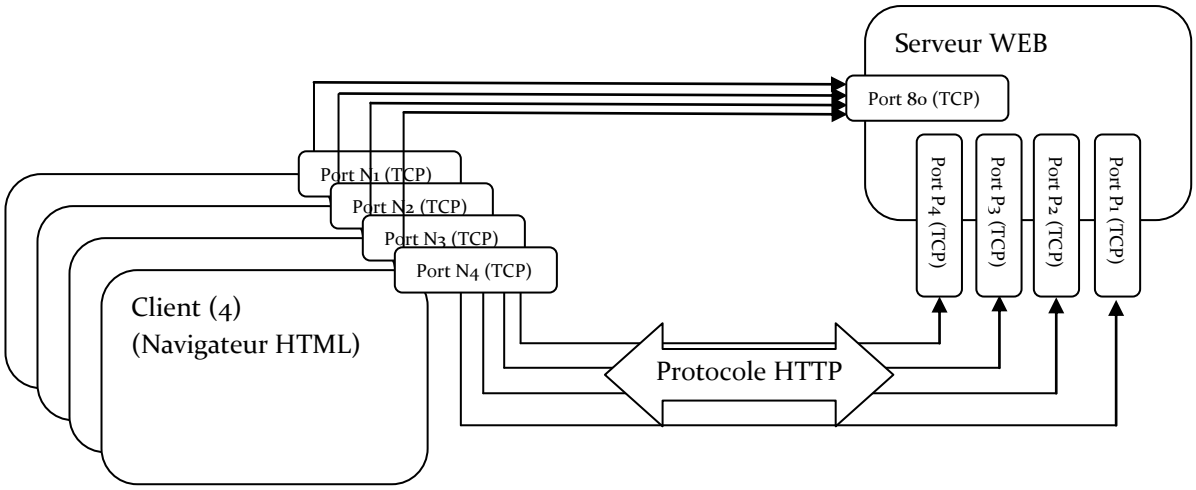
Ce TD a pour but de vous faire développer votre propre serveur Web que vous pourrez tester avec un simple telnet localhost 8080 puis via un navigateur Web standard. Vous pourrez vous aider pour cela des informations du cours et/ou celles de :

http://www.commentcamarche.net/internet/http.php3

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

Pour vous aider voici quelques informations et questions :

1. Un serveur Web est un serveur socket mode connecté (TCP/IP). Vous pourrez donc utiliser au choix Java ou C/C++ pour vos développements.
2. Le port standard pour un serveur WEB est le port 80. Pourquoi devons nous utiliser ici le port 8080 ?
3. 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 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.unice.fr 80
GET / HTTP/1.0
```

Quel est le résultat obtenu ? Quel est le serveur Web utilisé ?

De même installez Google Chrome et utilisé l'outil de développement présenté en cours, 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 ?

TD Applications Réparties
Introduction du Web pour les Applications Réparties : du H2M au M2M

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 http/1.0, 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 HTTP_ROOT qui permette de spécifier le chemin d'accès sur votre machine à l'espace disque qui sera servi par votre serveur Web (répertoire racine des documents accessibles par le serveur).

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.

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

```
http://locahost:8080/cgi-bin/toto?jean&pascal
```

La page Web retournée sera alors :

```
<HTML>
<HEAD>
<TITLE>Doc. Produit par un CGI</TITLE>
</HEAD>
<BODY>
```

TD Applications Réparties
Introduction du Web pour les Applications Réparties : du H2M au M2M

```
<H1>Coucou jean Pascal !<H1>
</BODY>
</HTML>
```

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,…`

Les deux variables qui permettent la récupérations des paramètre de l'URL sont : `REQUEST_URI` et `QUERY_STRING.`

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

5 Les formulaires pour faciliter le passage de paramètres dans l'URL.

Jusque là les paramètre sont fournis à la main, dans le bon ordre, dans la syntaxe même de l'URL demandée.

Un formulaire HTML de positionner dans une page Web des zones interactives qui permettent à la fois la saisies des paramètres mais aussi la création de l'URL demandée.

Exemple :

```
<form action="http://www.example.com/cgi-bin/example.sh" method="get">
 Enter a username: <input type="text" name="username"></input><br>
 <input type="radio" name="whatToDo" value="remove" checked="checked">Value<br>
 <input type="radio" name="whatToDo" value="create">Create<br>
 <input type="submit" name="subbtn" value="Submit">
 <form>
```

Les différents types d'éléments existant pour une balise <input> sont : buttoncheckbox, file, hidden, image, password, radio, reset, submit, text.

Le type submit effectuera à la fois l'affectation de la valeur "Submit"au paramètre submit mais déclenchera aussi la requête sur l'URL créée et paramétrée.

Nous pouvons donc reprendre l'exemple précédent, en passant les paramètres "Jean" et "Pascal" grâce à un formulaire.

6 Les formulaires pour passer les paramètres dans le corps ("body") de la requête sur l'URL.

Si nous reprenons l'exemple précédent, le formulaire permet d'utiliser deux méthodes pour l'émission des paramètres : la méthode GET mais aussi la méthode POST.

La méthode de requête POST passe les données en paramètre dans le corps du message d'une requête.

En revanche, la méthode de demande HTTP GET est conçu pour récupérer des informations à partir du serveur. Dans le cadre d'une requête GET, certaines données peuvent être transmises dans la chaîne de requête de l'URI, en précisant pour un mandat de recherche, par exemple des plages de dates ou d'autres informations qui définit la requête. Dans le cadre d'une requête POST, une quantité arbitraire de données de tout type peut être envoyé au

serveur dans un corps de message de demande. Le champ d'en-tête "Content-Type" dans une requête POST indique généralement le type de media (type MIME) contenu dans le corps du message.

Dans le cas de l'utilisation de la commande POST le formulaire s'écrira :

```
<form action="http://www.example.com/cgi-bin/example.sh" method="post">
 Enter a username: <input type="text" name="username"></input><br>
 <input type="radio" name="whatToDo" value="remove" checked="checked">Value<br>
 <input type="radio" name="whatToDo" value="create">Create<br>
 <input type="submit" name="subbtn" value="Submit">
 <form>
```

Les paramètres ne seront plus récupérables dans l'URL mais dans le corps de la requête. Pour un cgi-bin appelé, ces paramètres lui seront transmis dans le flux de son entrée standard et donc récupérer par une commande "read" dans un shell par exemple.

Si vous voulez tester ce type de formulaire avec votre serveur Web, il faudra bien sûr étendre ses fonctionnalités en rajoutant le traitement adéquat de la commande POST.

7 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.

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.

8 Serveur Web Multithreads :

Les plus avancés pourront modifier le code du serveur développé pour la section 3 afin d'autoriser des requêtes simultanées de plusieurs clients. Pour cela il faudra créer un thread par demande de connexion.

Une fois ce travail réalisé, comparez les performances du serveur de la section 3 et celle de la section 5 dans le cas de multiples requêtes simultanées.