UNIVERSITÉ DE ROUEN	ANNÉE 2021-2022
FACULTÉ DES SCIENCES	RÉSEAUX
LICENCE INFORMATIQUE 3E ANNÉE	PROJET

Serveur HTTP

1 Le protocole HTTP

L'HyperText Transfer Protocol (HTTP) est le protocole de transfert hypertexte à la base du World Wide Web (WWW). HTTP est un protocole de la couche application qui utilise les services d'un protocole connecté fiable. En pratique il utilise le protocole TCP de la couche de transport. Un serveur HTTP utilise par défaut le port d'écoute 80. Le client est généralement un navigateur web (Mozilla Firefox, Google Chrome, Microsoft Internet Explorer etc.).

1.1 Le protocole HTTP 0.9

HTTP est un protocole de communication qui utilise le modèle client-serveur. À l'origine le principe de fonctionnement du protocole est très simple et le dialogue entre client et serveur se réduit aux étapes suivantes :

1. le client se connecte au serveur (connexion TCP) ;
2. le client envoi une requête `GET` (par exemple `GET /page.html`);
3. le serveur répond en envoyant le contenu de la page demandée ou un message d'erreur ;
4. le serveur ferme la connexion TCP pour signaler la fin de la réponse.

Lorsqu'une page contient des liens (par exemple vers des images), le navigateur client doit refaire une demande de connexion pour chaque image qu'il souhaite afficher.

1.2 Le protocole HTTP 1.0

La version 1.0 du protocole, spécifiée dans la RFC 1945 [1] (voir la version française [2]), permet aux client et serveur d'envoyer plus d'informations entre eux, ainsi que de gérer le cache et les erreurs.

1.2.1 Requête du client

La requête envoyée par le client au serveur se présente sous la forme de plusieurs lignes de texte dans le format qui suit :

```
Ligne de commande (Commande, URL, Version de protocole)
En-têtes de requête
[Ligne vide]
Corps de requête
```

La ligne de commande contient l'URL de la page demandée, par exemple :

```
GET http://www.univ-rouen.fr/ HTTP/1.0
```

1. https://dl.acm.org/doi/10.17487/RFC1945
2. http://abcdrfc.free.fr/rfc-vf/rfc1945.html

L'URL n'est pas nécessairement complète et la commande précédente peut être abrégée en :

```
GET / HTTP/1.0
```

Les en-têtes de requête contiennent une ou plusieurs lignes précisant la requête dans le format MIME. Une requête avec en-têtes peut par exemple se présenter ainsi :

```
GET / HTTP/1.0
Date : Sat, 05 Apr 2014 10:12:05 GMT
Accept : text/html
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
If-Modified-Since : Fri, 04 Apr 2014 14:37:11 GMT
[Ligne vide]
```

L'en-tête `Date` donne la date d'envoi de la requête par le client. `Accept` précise que le client attend une page au format HTML (`text/html` est le type MIME de HTML). `User-Agent` donne des informations sur le navigateur utilisé. `If-Modified-Since` dit au serveur de n'envoyer la page demandée que si elle n'a pas été modifiée depuis la date du 4 avril 2014 à 14h 37mn 11s. Cette dernière en-tête permet au navigateur de charger rapidement les pages déjà présentes dans son cache, s'il n'est pas nécessaire de les recevoir du serveur.

Il arrive que le client ait besoin d'envoyer des informations au serveur, par exemple lorsque l'utilisateur remplit et envoie un formulaire en ligne. Ces informations peuvent être transmises dans l'URL par exemple :

```
GET https://www.google.fr/search?client=ubuntu&q=informatique HTTP/1.0
Date : Sat, 05 Apr 2014 10:12:05 GMT
Accept : text/html
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
[Ligne vide]
```

ou alors en utilisant la commande `POST`, dans ce cas elles sont mises dans la partie corps de la requête comme ceci :

```
POST https://www.google.fr/search HTTP/1.0
Date : Sat, 05 Apr 2014 10:12:05 GMT
Accept : text/html
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
[Ligne vide]
?client=ubuntu&q=informatique
```

1.2.2 Réponse du serveur

La réponse d'un serveur HTTP/1.x se présente dans le format suivant :

```
Ligne de statut (Version, Code de la réponse, Texte de la réponse)
En-tête de réponse
[Ligne vide]
Corps de réponse
```

Un exemple de réponse (si tout se passe bien) :

```
HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 08:36:49 GMT
Server: Apache
Content-Type: text/html
[Ligne vide]
[Page HTML]
```

Un exemple de réponse en cas d'erreur :

```
HTTP/1.1 404 Not Found
Date: Fri, 04 Apr 2014 09:28:27 GMT
Server: Apache
Content-Length: 202
Connection: close
Content-Type: text/html
[Ligne vide]
```

2 Projet de TP

Le but de ce projet est de créer un serveur HTTP minimal capable de répondre à une requête provenant d'un navigateur web. Vous utiliserez les bibliothèques `AdresseInternet` et `SocketTCP`. Une archive contenant

1. les *fichiers sources commentés* de vos bibliothèques et de votre serveur `http`,
2. un makefile global,
3. des tests de vos bibliothèques et de votre serveur,
4. un rapport [3] explicitant les choix d'implantation effectués,
5. un manuel d'utilisation[3]

devra être déposé sur universitice. Les soutenances auront lieu la semaine suivante. Vous pouvez présenter le projet en binômes (mais pas plus de 2 étudiants).

Les questions suivantes sont là pour guider l'implantation de votre serveur, vous rendrez comme projet la version la plus aboutie de votre serveur.

Exercice 1 Écrire un serveur HTTP 1.0 qui répond au navigateur client par le message `Bonjour`. Le serveur ignorera la requête du client et ses en-têtes, mais devra dans sa réponse spécifier la date et le type du contenu envoyé (`text/plain`). Le serveur devra gérer plusieurs clients à la fois en lançant pour chaque client un thread qui se charge de lui répondre.

Exercice 2 Écrire un serveur HTTP 1.0 multithreads qui répond au navigateur client par une page html contenant le message `Bonjour`. Le serveur ignorera la requête du client et ses en-têtes, mais devra dans sa réponse spécifier la date et le type du contenu envoyé (`text/html`).

Exercice 3 Écrire un serveur HTTP 1.0 multithreads qui envoie au navigateur client un fichier `index.html` qui se trouve dans le répertoire d'exécution du serveur. Ce serveur ignorera aussi la requête du client et ses en-têtes.

3. format texte ou pdf

Exercice 4 Écrire un serveur HTTP 1.0 multithreads qui envoie le fichier **HTML** demandé par le client lorsque celui-ci fait une requête GET. Ce fichier doit se trouver dans le répertoire d'exécution du serveur. Le serveur devra lire la ligne de commande, la découper, extraire le chemin du fichier à partir de l'URL. On ignorera les en-têtes de la requête. Il faudra gérer le cas d'erreur 404 si le nom du fichier ne correspond à aucun fichier présent sur le serveur, l'erreur 400 si la syntaxe de la requête est mal formulée ainsi que l'erreur 501 pour les commandes qui ne sont pas implémentées (POST etc.).

Exercice 5 Modifier le serveur pour qu'il gère aussi les fichiers images et textes. En vous basant sur l'extension du fichier demandé par le client et en répondant avec le type MIME qui correspond (en-tête `Content-Type`).

Exercice 6 Modifier le serveur pour qu'il envoie ou pas le fichier demandé par le client en prenant en compte l'en-tête de la requête `If-Modified-Since` si elle est présente. Si le fichier n'a pas besoin d'être renvoyé, le serveur renverra le code `304 Not Modified`.

Exercice 7 Ajouter à votre serveur une fonctionnalité supplémentaire parmi celles qui vous semblent intéressantes à mettre en oeuvre.