heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

ASI - Laboratoire 1

Serveur Web

12.10.2011

IL-2012

Brahim Lahlou

Numa Trezzini

1. Résumé

Au cours de ce laboratoire, nous avons spécifié les fonctionnalités d'un serveur HTTP simplifié. Il a ainsi fallu spécifier la façon de recevoir les requêtes et d'y répondre. Les requêtes et réponses ont aussi été spécifiées. Une forme fixe permet une communication uniformisée entre client et serveur. C'est pourquoi les tests associés aux spécifications ont également été produits. Le contenu de ce rapport servira à tester les serveurs produits par les équipes de développement.

2. Introduction

Lorsque nous allons sur internet, entrer un nom de site dans la barre d'adresse afin d'afficher du contenu nous semble normal. Quels sont en réalité les mécanismes sous-jacents ? Comment les pages web sont-elles appelées et chargées ? Comment fonctionne HTTP ? Pour répondre à ces questions, l'une des manières les plus simples reste de produire soi-même un serveur web HTTP.

Pour qu'un serveur soit totalement fonctionnel, il faut non seulement le coder, mais aussi le spécifier et le tester. C'est l'objectif de ce rapport. Nous allons commencer par spécifier les besoins du serveur HTTP par l'intermédiaire des requêtes et réponses, puis nous allons présenter les tests à effectuer avant de terminer par une conclusion sur les points les plus importants.

3. But

L'objectif de ce laboratoire est de produire les spécifications et tests permettant de vérifier le bon fonctionnement d'un serveur HTTP. Il s'agit de vérifier les réactions du serveur face à des situations normales et anormales : si la requête suit le cas d'utilisation principal, reçoit-on la bonne ressource? En cas d'erreur dans la requête, le serveur répond-il avec le bon message d'erreur ? Le serveur testé ne correspond pas exactement à la RFC 1945. Nous n'allons donc spécifier que les fonctionnalités précisées dans la donnée.

4. Spécifications

Cette section à pour but de présenter les spécifications que doit remplir le serveur HTTP, les services qu'il doit fournir et les réactions qu'il doit avoir face aux requêtes des clients. Lorsqu'une grammaire est présentée, elle est reprise de la RFC 1945, en particulier la notation, décrite au chapitre 2 de la RFC.

4.0 Spécifications générales

4.0.1 Port de communication

Nous devons pouvoir choisir le port sur lequel le serveur attend les demandes de connexion client. Le client doit pouvoir s'accorder au port choisi par le serveur [LAB].

4.1 Requêtes

Une requête est une demande de ressource par un client. Elle doit respecter une syntaxe pour pouvoir être comprise. En fonction de la version du serveur HTTP, la forme peut changer. Avec HTTP 0.9, une requête est composée d'une unique ligne. HTTP 1.0 permet d'inclure des informations supplémentaires. Le client doit envoyer une requête TCP pour chaque objet de la page web demandée.

4.1.1 Syntaxe d'une requête

La RFC 1945 propose pour une requête la grammaire suivante [1945] :

```
Requête         = Simple-Request | Full-Request
Simple-Request  = "GET" Request-URI CRLF
Full-Request    = Request-Line
                  *( General-Header
                   | Entity-Header )
                  CRLF
                  [ Entity-Body ]

Request-Line    = Method Request-URI HTTP-Version CRLF
Method          = "GET"
                 |"POST"
Request-URI     = relative_path
```

4.1.2 Précisions sur la syntaxe d'une requête

Comme nous n'avons pas besoin de donner toute la syntaxe de la RFC 1945, nous allons expliciter certaines des règles ci-dessus :

- General-Header : ce sont des en-têtes applicables aux requêtes comme aux réponses. Un en-tête général peut être une date. Un en-tête général non reconnu sera traité comme un en-tête d'entité [1945]
- Entity-Header : ce sont des méta-informations sur le corps d'une réponse ou sur la ressource demandée par la requête [1945]. On trouve par exemple l'encodage, le type et la longueur du corps, ainsi que la dernière date de modification[1]
- Entity-Body : il s'agit du corps d'une requête ou réponse. La RFC prévoit le corps comme une série d'octets. Leur interprétation est donnée par les Entity-Headers (Content-Type et Content-Encoding)[2]
- Request-URI : Un relative_path est utilisé pour demander une ressource du serveur auquel la requête est faite. Par exemple : `GET /toto HTTP/1.0`.
- Caractères spéciaux : le caractère CRLF (Carriage Return Line Feed), termine une ligne et en commence une nouvelle. La séquence correspond au string « \r\n ».

4.1.3 Spécification d'une requête

4.1.3.1 Requête simple

La forme la plus condensée d'une requête. Il s'agit d'un vestige de HTTP 0.9. Elle est composée d'une « GET », séparé d'un espace avec l'URI demandée. Le serveur du labo doit accepter une telle requête, et répondre soit avec la ressource demandée, soit avec le code d'erreur 400 « Bad Request » [LAB].

4.1.3.2 Requête complète

Il s'agit cette fois d'une requête HTTP 1.0. Celle-ci possède au moins un ligne de requête, zéro à plusieurs en-têtes, et éventuellement un corps (contenu). Un <CRLF> obligatoire sépare le corps des en-têtes/ligne de requête. La requête doit contenir un champ de longueur de contenu.

[1] Les en-têtes d'entité seront détaillés dans la section 4.2.3.4
[2] Le corps d'un message sera détaillé dans la section 4.2.3.5

4.1.3.3 Ligne de requête

La ligne de requête ressemble à la requête simple. Il est cette fois possible de préciser la méthode à utiliser. La version HTTP de la requête est également requise.

4.1.3.4 Méthodes

Les méthodes indiquent au serveur quel type de traitement il doit appliquer à la requête. Nous ne traiterons dans ce labo que les méthodes GET et POST [LAB].

- GET : Cette méthode indique au serveur que le client souhaite récupérer la ressource associée à la requête.
- POST : Cette méthode permet au client d'inclure des données dans le corps de la requête afin que le serveur les traite.

En cas de la présence d'une requête avec la méthode HEAD, le serveur répondra avec une erreur de type 501 « Not Implemented ».

4.2 Réponses

Une réponse est un message envoyé suite à la réception d'une requête cliente. La réponse doit respecter la syntaxe ci-dessous pour pouvoir être comprise. En fonction de la version du serveur HTTP, la forme peut changer. Une réponse simple est un vestige de HTTP 0.9 et ne devrait être utilisée que pour répondre à une demande de la version correspondante.

4.2.0 Serveur multi-threadé

Le serveur doit être capable de gérer plusieurs connexions TCP en parallèle. Il écoute les demandes sur un port fixe

4.2.1 Syntaxe d'une réponse

La RFC 1945 propose pour une réponse la grammaire suivante [1945] :

```
Response        = Simple-Response | Full-Response
Simple-Response = [ Entity-Body ]
Full-Response   = Status-Line
                  *( General-Header
                   | Response-Header
                   | Entity-Header )
                  CRLF
                  [ Entity-Body ]
Status-Line     = HTTP-Version Status-Code Reason-Phrase CRLF
```

4.2.2 Précisions sur la syntaxe d'une réponse

Les en-têtes généraux des réponses sont identiques à ceux des requêtes[3].

4.2.3 Spécification d'une réponse

4.2.3.1 Réponse simple

Une réponse simple ne contient que le corps de la réponse. Si cette fonctionnalité est implémentée, une réponse simple ne devrait être envoyée que lorsqu'une requête simple HTTP 0.9 est effectuée. Une réponse sans ligne de statut doit être considérée comme une réponse simple et être traitée de façon appropriée [1945].

[3] cf. section 4.1.2

4.2.3.2 Ligne de statut

Une ligne de statut commence toujours avec la version du protocole HTTP. La version est suivie du code de statut et de la phrase de raison appropriée[4] [1945].

4.2.3.3 En-têtes de réponse

Les en-têtes de réponse ajoutent des informations que la ligne de statut ne peut pas contenir [1945]. Les en-têtes de réponse sont les suivants :

- Location : Cet en-tête contient le chemin absolu vers la ressource demandée.
- Server : Cet en-tête contient des informations sur la version des programmes employés par le serveur pour traiter la requête.

4.2.3.4 En-têtes d'entité

Les en-têtes d'entité sont les suivants [1945]:

- Content-Encoding : cet en-tête indique le type de compression du corps de message
- Content-Length : cet en-tête indique le nombre d'octets du corps de message
- Content-Type : cet en-tête indique le type du corps de message. Il peut s'agir notamment de [MIME]:
    - Texte : plain (fichiers .java et valeur par défaut) et html (page web)
    - D'images : JPEG et GIF
    - D'applications : flux d'octets

4.2.3.5 Corps d'entité

Le corps d'un message est une collection d'octets, interprété selon les indications des en-têtes. Toutes les réponses doivent indiquer dans leurs en-têtes la longueur du corps de l'entité [1945].

4.2.3.6 Codes de statuts et phrases de raison

Les codes de statut et leurs phrases de raison définis par la RFC 1945 sont :

- Codes indiquant un succès:
    - 200: OK: envoyé lorsque le serveur a envoyé la ressource demandée
- Codes indiquant une redirection:
    - 301: Moved Permanently : envoyé lorsque la ressource associée à la requête possède une nouvelle URI permanente. Celle-ci doit être indiquée dans l'en-tête « Location : » de la réponse.
- Codes indiquant une erreur survenue du coté client:
    - 400: Bad Request : envoyé lorsque la requête associée n'a pas été comprise par le serveur pour cause de mauvaise syntaxe
    - 404: Not Found : envoyé lorsque le serveur n'a pas trouvé de ressource correspondant à l'URI associée
- Codes indiquant une erreur survenue du coté serveur
    - 500: Internal Server Error : envoyé lorsque le serveur n'a pas pu remplir la requête pour cause d'exception
    - 501: Not Implemented : envoyé lorsque le serveur ne possède pas de méthode appropriée pour traiter la requête

[4] Les codes et les phrases de raison sont détaillés dans la section 4.2.3.6

5 Tests

Voici la table des tests de validation. Pour des questions de place, les colonnes « OK », « KO » et « exécution » ne sont pas représentées ici.

CT	SP	Procédure	Résultat attendu
1	4.0.1	Lancer le serveur sur le port choisi. Ouvrir un navigateur et introduire l'adresse index.html.	La page d'accueil est envoyée au navigateur qui l'affiche.
2	4.0.1 & 4.1	Lancer Wireshark et ouvrir la page index.html du serveur avec un navigateur. Le port doit correspondre à celui choisi par le serveur	Les paquets échangés entre le client et le serveur suivent bien le protocole TCP.
3	4.1	Ouvrir un navigateur et introduire l'adresse index.html. (lancement par défaut sur le port 80)	La page d'accueil est affichée malgré le fait que l'on n'a pas spécifié le port.
4	4.1.3.1	Lancer un Telnet puis faire une requête HTTP 0.9 de type GET sur index.html	Si le serveur implémente HTTP 0.9, il doit répondre avec une ligne de statut « 200 OK ». Sinon, il doit répondre « 400 Bad Request »
5	4.1.3.2	Faire une requête HTTP 1.0 de type GET sur index.html.	La ligne de statut de la réponse indique que le protocole utilisé est HTTP 1.0 et le code d'état affiché est « 200 OK ».
6	4.1.3.3	Faire une requête malformée au serveur	La ligne de statut de la réponse doit être « 400 Bad Request »
7	4.1.3.4	Faire une requête POST valide au serveur	Le serveur renvoie la réponse avec comme code 200
8	4.1.3.4	Faire une requête POST invalide au serveur	Le serveur répond avec une ligne de statut « 400 Bad Request »
9	4.1.3.4	Faire une requête HEAD au serveur	Le serveur répond avec une ligne de statut « 501 Not Implemented »
10	4.2.0	Faire plusieurs requêtes en même temps au serveur	Le serveur doit pouvoir traiter en parallèle les requêtes
11	4.2.3.1	Faire une requête HTTP 0.9 au serveur	Le serveur doit répondre avec une réponse simple ou avec le code de statut « 400 Bad Request »

12	4.2.3.2	Faire une requête et analyser la ligne de statut de la réponse	La réponse doit correspondre à la grammaire spécifiée sous 4.2.1
13	4.2.3.3	Faire une requête et analyser les en-têtes de réponse	Les en-têtes de réponse doivent contenir les entrées « Location » et « Server »
14	4.2.3.4	Faire une requête pour une page html	L'en-tête d'entité « Content-Type » doit être « text/html »
15	4.2.3.4	Faire une requête pour un fichier java	L'en-tête d'entité « Content-Type » doit être « text/plain » (valeur par défaut)
16	4.2.3.4	Faire une requête pour une image JPEG	L'en-tête d'entité « Content-Type » doit être « image/jpeg»
17	4.2.3.4	Faire une requête pour une image GIF	L'en-tête d'entité « Content-Type » doit être « image/gif»
18	4.2.3.4	Faire une requête pour une page html	L'en-tête d'entité « Content-Length» doit correspondre au nombre d'octets du fichier source
19	4.2.3.5	Faire une requête pour une page html	Le corps de la réponse doit contenir les octets du fichier demandé. Le navigateur affiche donc la ressource
20	4.2.3.6	Lancer le navigateur et introduire une adresse URI inexistante (par ex : laboASI.html)	Le serveur renvoie une page d'erreur indiquant que la page n'a pas pu être trouvée. Le code de statut de la réponse doit être « 404 Not Found »
21	4.2.3.6	Lancer le navigateur et entrer comme adresse URI la racine du serveur (http://localhost:<port>)	Le serveur renvoie la page par défaut (dans notre cas : index.html).
22	4.2.3.6	Faire une requête sur /heig-vd.html	La ligne de statut de la réponse indique un code d'état « 301». La réponse doit aussi contenir l'adresse de redirection dans l'en-tête « Location ».
23	4.2.3.6	Faire une requête levant une exception interne au serveur	La ligne de statut de la réponse indique un code d'état « 500 Internal Server Error »

6. Conclusion

Afin de bien comprendre le fonctionnement d'un serveur HTTP, l'un des meilleurs moyens est d'en faire un soi-même. Ceci nous a permis de nous familiariser avec les échanges effectués entre le navigateur et les serveurs lors de la navigation sur le web. Ce rapport a également été l'occasion pour nous d'avoir un vrai contact avec les spécifications et les tests, notamment par l'intermédiaire des RFC. L'écriture des tests a d'ailleurs été l'un des points les plus difficiles à traiter, car n'ayant que très peu d'expérience dans ce domaine, il nous a fallu comprendre la structure à donner à ceux-ci.

7. Références

[1]. Berners Lee et al., Requête For Comments n°1945, [en ligne] http://www.ietf.org/rfc/rfc1945.txt
[2]. Collectif Wikipédia, Multipurpose Internet Mail Extensions, [en ligne] http://fr.wikipedia.org/wiki/Multipurpose_Internet_Mail_Extensions
[3]. Buchs Christian, ASI - Laboratoire Serveur Web Multi-Threadé en Java, HEIG-VD, 2011
[4]. Buchs Christian, Applications et Services Internet, HEIG-VD, 2001

8. Liste des symboles de référence

Les symboles de référence utilisés dans ce rapport sont :

- [1945] : RFC 1945
- [MIME] : Multipurpose Internet Mail Extensions
- [LAB] : Données du laboratoire