Compte Rendu du TP1 : Étude du Protocole HTTP

Nom : Lorenzo Olivieri, Yacine Tarzout, Mischa Maisonneuve

Module : B1 Programmation Web

Sujet : Étude du Protocole HTTP:

Introduction:

L'objectif de ce TP était de comprendre comment fonctionne le protocole HTTP (Hypertext Transfer Protocol), qui est la base de tout ce qu'on fait sur Internet. On a vu que tout repose sur le modèle client-serveur et qu'il y a plein de protocoles qui travaillent ensemble, comme TCP/IP et DNS, pour que le navigateur puisse afficher une page web.

Le but était de mettre les mains dans le cambouis en simulant des serveurs et en analysant les échanges pour bien comprendre la théorie.

Partie 1 : Les Fondamentaux et l'Architecture

On a commencé par revoir le modèle client-serveur, avec l'analogie du restaurant :

- Le Client (nous, avec notre navigateur) fait une Requête (la commande).
- Le Serveur (l'ordinateur qui héberge le site) envoie une Réponse (le plat servi).
- Le Réseau (la salle) assure le transport.

On a bien compris que HTTP ne travaille pas seul. Il s'appuie sur :

- DNS (port 53) pour traduire l'adresse du site (comme google.fr) en adresse IP.

- TCP/IP pour établir une connexion fiable (mode connecté) entre le client et le serveur, en utilisant des ports (comme le port 80 pour HTTP).

Partie 2 : Les Manipulations Pratiques

Le TP nous a fait passer par trois modes de fonctionnement pour bien voir la différence entre un simple fichier et un vrai site web.

1. Mode Fichier Local

Manipulation : On a créé un fichier index.html avec un simple "Hello World !" et on la ouvert directement dans Chrome. Observation : L'URL commençait par file:///.... Le navigateur a juste lu le fichier sur le disque. Conclusion : Dans ce mode, il n'y a pas de serveur web et le protocole HTTP n'est pas utilisé. C'est juste une lecture locale.

2. Mode Localhost (Client et Serveur sur la même machine)

Manipulation : Nous avons lancé un serveur web très léger avec la commande “ python -m http.server 8000 “ dans le terminal. Ensuite, On a accédé à la page via “ http://localhost:8000 “. Observation : Ma machine jouait les deux rôles : elle était à la fois le Client (le navigateur) et le Serveur (le script Python). C'est super pratique pour développer sans avoir besoin d'Internet. Conclusion : Ce mode simule un vrai serveur web, ce qui est essentiel pour le développement.

3. Mode Client-Serveur (avec un trinôme)

Manipulation : En Trinôme, on a fait le test en accédant au serveur de l'autre machine en utilisant son adresse IP (par exemple, http://192.168.1.50:8000). On a aussi utilisé la commande netstat -a -n pour voir les ports en écoute (LISTENING) et les connexions établies (ESTABLISHED). Observation : On a vu la connexion se faire clairement entre deux machines distinctes. Le serveur Python affichait les requêtes reçues. Conclusion : C'est le fonctionnement réel du Web, où le client et le serveur sont séparés par le réseau.

Partie 3 : Analyse des Échanges HTTP

Nous avons analysé la structure des requêtes et des réponses, ce qui est la partie la plus technique.

La Requête

Quand le client demande une page, il envoie une requête en trois parties :

1 Ligne de Requête : Elle dit ce qu'on veut (GET), où (/index.html) et la version du protocole (HTTP/1.1).

2 En-têtes : Des infos supplémentaires (ex: quel navigateur j'utilise).

3 Corps de la Requête : Surtout utilisé pour envoyer des données (comme un formulaire) avec la méthode POST.

La Réponse

Le serveur répond aussi en trois parties :

4 Ligne de Statut : Le plus important, c'est le Code de Statut.

- 200 OK : C'est bon, la requête a marché.
- 404 Not Found : Erreur client, la ressource demandée n'existe pas (par exemple, si on tape une mauvaise URL).
- 500 Internal Server Error : Erreur serveur (un problème sur le serveur lui-même).

5 En-têtes de Réponse : Infos sur le contenu (ex: Content-Type: text/html).

6 Corps de la Réponse : Le contenu de la page (le code HTML, CSS, etc.).

Manipulation **curl** : On a utilisé curl -v pour voir exactement ce que le client envoie et ce que le serveur répond, y compris tous les en-têtes. Ça m'a permis de voir le Content-Type pour différents fichiers (HTML, CSS, images).

Conclusion :

Ce TP a été très utile pour passer de la théorie à la pratique. Nous avons compris que le protocole HTTP est un langage de communication très structuré et que le rôle du serveur web est beaucoup plus complexe qu’on ne le pensais (gestion des droits, des redirections, du cache, etc.).

Le fait de lancer mon propre serveur Python et de voir les requêtes et les codes de statut en direct m'a vraiment aidé à visualiser le fonctionnement du Web. Maintenant, quand on tape une URL, nous savons exactement ce qui se passe derrière !