Progres

Rapport du Mini-projet 1

Réalisé par :

- **Aouam Yasmina : 28718479**
- **BAH Namory 21121885**

Exercice 1 :

Dans cet exercice le but est d'implémenter un client simple, un serveur nominal et un proxy connecté entre eux avec le protocole de transport TCP.

Dans un premier temps nous avons programmé des ports pour chacun des serveurs et du proxy. Quant à l'adresse IP de chacun reste la même comme nous travaillons sur la même machine. (Adresse IP locale : 127.0.0.1).

Ensuite nous avons établie les connections nécessaires grâce au module Socket, entre le client et le servers. A noter donc que le proxy est considéré comme un client pour le serveur, et un serveur pour le client.

Le client envoie des requêtes vers le port du proxy : 1234. Ce dernier redirige cette requête vers le port du serveur nominal : 5678. La réponse est ensuite reçue par le serveur, qui à son tour renvoie une réponse sur le port de l'intermédiaire : 5678, c'est-à-dire le proxy. Ce proxy va à nouveau rediriger cette réponse du serveur, vers le client qui l'a demandé.

Et c'est ainsi qu'à chaque requête reçue, le proxy renvoie la réponse, qui finit par être acheminé par le proxy vers le client qui l'a demandé.

Nous avons testé avec un client qui envoie plusieurs requêtes à la fois.

Quant à la dernière partie, nous avons utilisé le module Threading vue en cours. Pour gérer plusieurs processus à la fois. Dans notre cas il s'agit de recevoir des requêtes simultanément de différents clients et ça a bien marché.

Exercice 2 :

Pour ce dernier exercice, nous avons travaillé sur le proxy TCP, pour lui ajouter une option de cache. nous avons donc associé une fonction qui accepte les requêtes complètes du client web, dans une boucle, dans celle-ci on extrait le nom du fichier demandé par le client web avec la méthode split() vue en cours.

Par la suite, avec la méthode is_file de la bibliothèque pathlib vue en cours. On cherche si un tel fichier existe dans le cache. Il y a une simple condition :

- Si non (le fichier n'est pas dans le même répertoire que le proxy), le proxy cache va envoyer une requête pour demander au serveur de lui envoyer ce fichier si il existe, ensuite nous fermons les connections

-sinon, nous renvoyons la requête avec un accusé de réception 200 OK. Ensuite nous fermons les connections afin que le contenu du fichier s'affiche sur le navigateur.

En second lieu, nous nous sommes occupés du fichier log, celui-ci s'occupe de garder les adresses IP et l'entête http, des machines qui demandent une requête.

De la même manière que le proxy cache, cette fois-ci en plus, il enregistre l'adresse IP, et de la requête http du client qui demande le fichier.

Quant au censeur nous avons eu un souci de requête sans réponse, avec pleins d'erreurs au niveau du proxy censeur. Nous n'avons pas su finir de les résoudre. Le but était de créer un fichier, et de

rajouter des noms de fichiers qui sont dans le répertoire serveur, à interdire l'accès si un client les demande.

Le projet était enrichissant, en travaillant avec la bibliothèque socket avec le protocole TCP, qui gère une communication logique entre systèmes liés au réseau local, dans notre cas. Nous avons appris sans autre bibliothèques orientés réseaux, à créer un proxy, et le modifier pour travailler comme proxy cache.