PROGRES – Mini-Projet 1

GIANG Elodie – MERLIN Lucas
phuong.giang@etu.sorbonne-universite.fr - lucas.merlin@etu.sorbonne-universite.fr

Pour réaliser ce premier mini-projet, j'ai décidé de travailler de travailler en groupe avec GIANG Elodie.
Ce mini-projet avait pour but de programmer en python des proxys qui ont pour rôle d'être vu comme un serveur pour le client et comme un client pour le serveur. Cela entraîne donc que le proxy réceptionne l'ensemble des requêtes envoyées par le client et les transmets au serveur, puis réceptionne les réponses du serveur pour les renvoyer au client.

Dans l'exercice 1 nous avions donc pour but de programmer un proxy TCP capable de servir à plusieurs clients.
Pour cela nous avons d'abord écrit un proxy TCP fonctionnant pour un client, puis avons essayé avec plusieurs clients et avons constaté qu'il été nécessaire de modifier celui-ci afin que plusieurs clients puissent effectuer des requêtes en simultanés. Nous avons donc mis en place un proxy TCP multi-thread. Le multi-thread permet en effet à plusieurs clients de réaliser des requêtes en simultanés car lorsque un client créé une connexion avec le proxy, le proxy va lui créer un thread qui lui sera dédié et lui attribuer une socket afin de pouvoir communiquer avec lui. Cela permet au proxy de créer plusieurs threads/sockets et donc pouvoir traiter les requêtes des différents clients facilement.
Pour s'assurer de l'efficacité du proxy nous l'avons donc par la suite testé en créant des clients demandant un très grand nombre de requêtes successives afin d'exécuter simultanément les clients et constater que les requêtes de chaque clients sont bels et bien traitées en mêmes temps. Par la suite nous avons aussi ajouté un délai afin d'éviter le fait qu'un client qui ne communique plus avec le proxy pendant un certains temps (que nous pouvons moduler) continue d'utiliser les ressources du proxy et nous fermons donc sa connexion pour libérer le proxy de son threads et sockets. Nous sommes donc arrivé à un proxy TCP multi-threads capable de gérer les multiples requêtes de plusieurs clients en simultanés, tout en fermant les connections des clients ne communiquant plus avec celui-ci.

Dans l'exercice 2, le but était différent car nous devions programmer différents proxys qui cette fois-ci géré des échanges entre un client et un serveur utilisant le protocole HTTP. Pour réaliser ceci, nous sommes partit sur les base du proxy TCP réalisé dans l'exercice précédent.
Premièrement, nous avons réalisé un proxy servant de cache HTTP. Cela à pour objectif de garder en « mémoire » un fichier ayant fait l'objet d'une requête GET pour que si il est redemandé , soit accessible directement. Pour que notre proxy prenne ce rôle, lorsqu'il reçoit une requête GET, il va vérifier si le fichier demandé est dans son répertoire et le fournir au client si tel est le cas sans allé demander le fichier au serveur. Sinon, le proxy va donc demander au serveur le fichier, réceptionner sa réponse et créer une copie de ce fichier dans son répertoire afin de pouvoir le fournir directement la fois suivante, avant de transmettre la réponse au client. Cela nous donne donc un proxy HTTP fonctionnant comme un cache.

Dans un second temps, nous avions a faire un proxy servant de loggeur HTTP. Un loggeur à pour objectif de garde en mémoire toutes requêtes effectuées par le client et toutes les réponses envoyées par le serveur. Pour cela, notre proxy, ouvre (ou créé si nécessaire) le fichier des logs. Lorsque notre

proxy reçoit une requête du client, il ouvre donc les logs et y met la requête HTTP qu'il à reçu ainsi que la date et l'heure à laquelle la requête a été faite. Ensuite, il transmet la requête au serveur. Lorsque le serveur lui répond, le proxy ouvre de nouveau les logs afin d'y mettre la réponse du serveur et donc le contenu du fichier demandé tout en y ajoutant la taille du fichier transmit (récupéré dans l'en-tête HTTP de la réponse) . Cela nous donne donc un porxy HTTP fonctionnant comme un loggeur.

Pour notre dernier proxy, nous avons travailler sur un proxy servant de censeur HTTP. Le loggeur permet au proxy de limiter l'accès à certains fichier au client. Pour que le proxy endosse ce rôle, lorsqu'il reçoit une requête du client, il ouvre le fichier contenant la liste des fichiers interdits et y compare le fichier demandé dans la requête GET du client. Si le fichier fait partit de cette liste, le proxy renvoie donc au client un fichier par défaut lui indiquant qu'il ne peut accéder à ce fichier. Sinon, il transmet la requête au serveur et retransmet la réponse du serveur avec le fichier au client. Cela permet donc à notre proxy HTTP de fonctionner comme un censeur.

Après avoir réalisé ces différents proxy, nous avons donc testé leur fonctionnement indépendamment les uns des autres puis nous avons put tester de les utilisés les uns avec les autres et avons donc réalisé les compositions suivantes :

- client – loggeur – censeur – serveur
- client – loggeur – cahce – serveur
- client – censeur – cache – serveur

Cela nous à permit de voir qu'il fonctionnaient bel et bien lorsque qu'ils etaient ajoutés les uns aux autres et que chacun effectuait toujours efficacement son rôle.

Ce mini-projet nous a permit de mieux comprendre et donc réaliser les proxys ainsi que les différentes fonctionnalités qu'ils pouvaient endosser.