GIANG
Elodie
3678177

MERLIN
Lucas
28705933

Rapport Mini-Projet 1

Exercice 1 - Proxy TCP

Le but de l'exercice 1 est de programmer un proxy connecté avec un client et un serveur en mode TCP. Le rôle du proxy sera de relayer les données du client vers le serveur. Pour cela, nous avons défini notre proxy de la manière suivante. Il stocke le numéro de port et l'adresse IP du serveur (5678), ainsi que son numéro de port (1234).

La première étape consiste à créer un proxy séquentiel TCP. Pour cela, après avoir ouvert la connexion entre le client et le proxy, on ouvre de même une connexion entre le proxy et le serveur. Le proxy étant le seul « client » du serveur, et travaillant celui ci en mode séquentiel, nous n'avons pas besoin de gérer cette connexion pour chaque requête. Une ouverture de connexion en début d'envoi de requêtes et une fermeture en fin d'envoi suffit. Le proxy accepte toute requête venant du client. Tant qu'il recevra des requêtes et pour chaque requête, il l'enverra au serveur, puis réceptionnera directement la réponse du serveur pour l'envoyer au client.
Nous nous sommes confrontés à une difficulté liée à l'utilisation d'un navigateur web pour le rôle de client et d'un serveur web pour le rôle du serveur. En effet, le code ne cessait de tourner, car le proxy attendait indéfiniment des requêtes venant du navigateur qui n'arrivaient jamais. Nous avons donc décider de palier à ce problème en ajoutant un délai qui permettra de lever une exception timeout afin de pouvoir notifier au proxy que le client n'a plus de requêtes à envoyer. Cette exception ajoute une sécurité au code et permet à celui-ci d'assurer une boucle finie quelque soit le type de requête émis.

La deuxième étape consistait quant-à elle d'établir un proxy capable de gérer la connexion de plusieurs clients simultanés. Nous avons décidé d'utiliser les threads afin de diviser la tâche de gérer un client en plusieurs fils d'exécution. Dès l'acceptation d'une connexion, le proxy fait appel à la fonction Thread afin de gérer la tâche indépendamment des autres requêtes. Cette indépendance a soulevé la question de la connexion entre le proxy et le serveur qui devait elle aussi être respective à chaque requête. Pour cela, il a suffit d'ouvrir et fermer la connexion dans la fonction handle_client.
Afin de tester notre code, nous avons écrit 2 codes clients qui effectuent plusieurs affichages pendant une certaine durée. Les affichages simultanées permettent d'illustrer la gestion simultanées des requêtes.

Exercice 2 : Proxy HTTP

La première partie consiste à programmer un proxy cache à partir du proxy de l'exercice 1.
Notre proxy cache bouclera sur la fonction recv() grâce à l'appel de la fonction recv_until() tant qu'il recevra des requêtes venant du client HTTP. La requête retournée par la fonction sera une requête globale de toutes les requêtes émises par le client. Le proxy évaluera ensuite si le fichier demandé par le client (observable dans la requête) se trouve dans son répertoire. Si c'est le cas il renverra immédiatement une entête HTTP ainsi que le fichier demandé. Sinon, il enverra la requête au serveur et stockera la réponse du serveur dans un fichier de son répertoire portant le nom de la requête cliente.
Nous avons eu un souci au niveau de l'écriture des données dans le fichier. En effet, l'entête reçue du serveur était elle aussi stockée dans le fichier, alors que nous ne le voulions pas. Pour parer à ce problème nous avons modifié la fonction recv_until afin qu'il envoie directement l'entête HTTP au client et n'enregistre que les données dans le fichier du proxy.

Le proxy loggeur quant-à lui, a pour but d'enregistrer chaque requête reçue avec l'heure de la requête ainsi que les réponses émises par le serveur avec la taille du fichier. A chaque réception de requête du client, le proxy enregistrera la requête ainsi que son heure avec la fonction datetime.now(). A la réception e la réponse du serveur, le proxy analysera l'entête de la réponse. Le champ « Content-Length » lui permettra de déterminer la taille du fichier. Si ce champ n'est pas disponible, nous ne pouvons pas connaître la taille du fichier transmis.

Le proxy censeur a dans son répertoire un fichier « fichier interdits » qui lui permet de vérifier chaque requête envoyée par le client. Si la requête se trouve dans la liste des fichiers, le proxy enverra une entête HTTP ainsi que le fichier « avertissement.txt » qui est le fichier par défaut, envoyé à chaque requête se situant dans la liste des fichiers interdits. S'il ne s'y trouve pas cependant, la requête sera envoyée au serveur et le proxy enverra au client la réponse du serveur. Le seul problème rencontré est que le fichier retourné par une requête d'un fichier interdit porte le nom du fichier interdit et non pas « avertissement.txt ».

La dernière étape consistait à synchroniser tous les proxy entre eux. Lors d'une manipulation d'un proxy cache lié à un proxy loggeur lié à un serveur, nous avons remarqué des problèmes d'écriture des fichiers dans le cache. Le problème résidait dans le fait que l'entête des réponses n'étaient pas envoyées entre chaque proxy. Nous avons donc modifiés toutes les fonctions recv_until des proxy afin qu'elles puissent renvoyer l'entête HTTP.

Dans notre archive, nous y avons déposé nos codes proxy, mais aussi les fichiers annexes qui nous servent dans l'exécution des fichiers, que vous pouvez, éventuellement, modifier.