Rapport du Mini-Projet 1 : Proxy

A- Les choix techniques

Exercice 1

Le serveur est capable de traiter deux types de requêtes. Dans le cas où le client est un client web, lorsque le serveur reçoit une requête, il découpe l'entête de la requête afin de récupérer le nom du fichier, puis il ouvre le fichier si celui-ci existe dans sa base de données. Dans le cas ou la requête provient d'un client python, le message doit être tapé par l'utilisateur en minuscule doit être retourné en majuscule. Ces deux cas sont traités par deux parties distinctes dans le code du serveur. Quand une requête est reçue le serveur détermine si la requête est une requête http, auquel cas c'est une requête provenant d'un client web. La requête est alors traitée adéquatement.

Pour pouvoir tester l'envoie de requêtes simultanées d'un même client, j'utilise une boucle 'for' où chaque tour de boucle envoie une requête. Le serveur n'est pas capable de tenir cette charge. Afin de remédier à ce problème, j'utilise des threads. Pour chaque requête reçu, le serveur et le proxy créent un thread spécifique pour la gestion de la requête.

Exercice 2

Le code du proxy est divisé en trois fichiers, représentant ses trois composants, le « cacheHttp », le « loggeur », le « censeurHttp ». Ces trois fichiers sont dans un dossier « proxy », différent du dossier « serveur » où se trouve le code du serveur, et les fichiers textes représentant sa base de données.

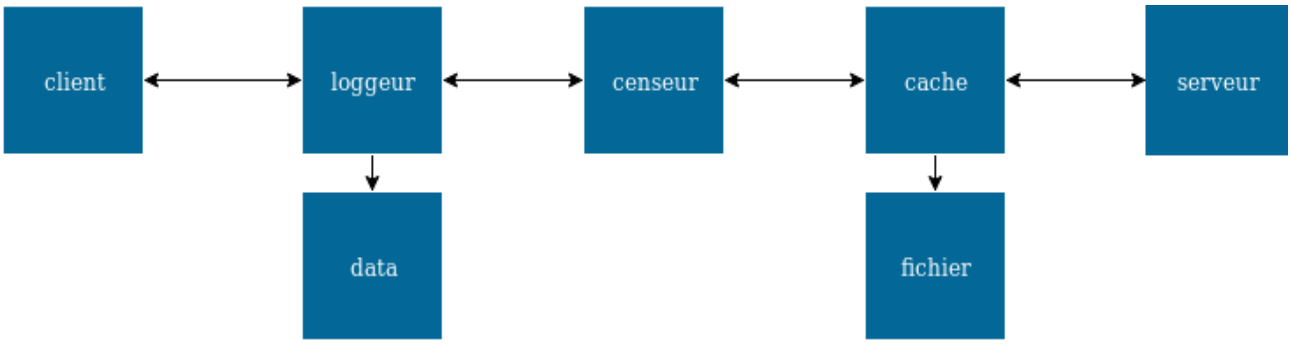

Schéma : architecture du proxy. Chacun des composant est indépendant des autres. Afin de les connecter par le réseau, il suffit de fournir à un composant l'adresse ip et le port de ses deux correspondants.

Dans le fichier cacheHttp, j'utilise une liste contenant les noms des fichiers que le serveur de cache garde en mémoire. Pour une requête reçue, le cacheHttp va faire une découpe de l 'entête de la requête pour récupérer le nom du fichier demandé et ainsi faire une copie dans le dossier « proxy ». Les fois suivantes où le fichier est demandé, ce fichier sera lu non pas dans le dossier « serveur » mais dans le dossier « proxy ». Cette séparation des dossiers permet de bien montrer qu'il n'y aura pas d'envoi de requête vers le serveur et que le proxy conserve bien les fichiers demandés.

Les logs sont enregistrés par le serveur de logs, le « loggeur ». Un log est effectué à deux reprises pour une requête. Lorsque la requête est reçu par le serveur de log, celui-ci l'enregistre dans

un fichier « data.txt ». Enfin, lorsque la réponse est retournée, il l'enregistre dans le ficher « data.txt ».

Dans le fichier censeurHttp, j'utilise une liste contenant la l'ensemble des noms des fichiers interdits. Lorsque le serveur de censure reçoit une requête, il découpe l'entête de la requête, récupère le nom du fichier demandé et le compare aux noms se trouvant dans la liste. Si le ficher demandé est interdit, il ne fais pas de requête auprès du serveur et renvoie une erreur 404. Sinon, il transmet la requête sans modification.

Afin de tester mon code, j'utilise le serveur écrit pour la question 1. Pour une requête reçu, le serveur renvoie une requête commençant par « HTTP/1.1 200 OK » s'il possède le fichier demandé. Sinon, il retourne une requête prédéfinis commençant par « HTTP/1.1 400 NOT FOUND ».

B- Les problèmes rencontrés

Je n'ai pas réussi à tester mon proxy dans un scénario où plusieurs clients différents envois des requêtes de manière simultanée, J'utilise simplement plusieurs instances d'un même client.

Je n'ai pas testé mon code sur des machines distantes. Tous les tests ont été réalisés en local avec l'adresse IP 127.0.0.1.

Le lien pour la vidéo : https://www.youtube.com/watch?v=Dpxxyn-XWio&feature=youtu.be