Système et réseaux : Serveur FTP

Rapport de projet

BARROIS Florian - DUVERNEY Thomas

Introduction :

Ce projet consiste à implémenter un serveur similaire au serveur FTP possédant une partie serveur multi-processus comprenant un certain nombre d'optimisations.

Réalisations (par ordre chronologique) :

- Téléchargement d'un fichier stocké sur le serveur :
  pour cette fonctionnalité le sujet du TP7 a été repris : lorsqu'un client se connecte, le processus exécutant attend un nom de fichier. Lorsque le fichier a été téléchargé, le client est déconnecté.

- Découpage du fichier téléchargé :
  afin d'optimiser la mémoire lors des téléchargements de fichiers conséquents, le transfert des données est effectué par paquets de 128 octets.

- Gestion des pannes côté client : non fonctionnel
  Idée : lorsque le client se connecte et demande un nom de fichier à télécharger, on vérifie si le fichier en question est présent dans le répertoire courant. Si oui, la taille est envoyée au serveur. Le serveur nous vérifie si la taille envoyée par le client correspond à la taille du fichier demandé sur le serveur. Si la taille est différente, alors le dernier s'est mal effectué. Dans ce cas, le serveur le détecte et envoie le nombre d'octets manquants au client.

- Load balancer :
  le client se connecte à un serveur master. Ce serveur, sur un système de Round Robin, redirige la requête du client sur un serveur esclave. Le serveur esclave fait alors une demande de connexion au client. Le client réalise la fonction *Accept* afin d'établir la connexion. L'envoi de fichier fonctionne de la même manière que dans les parties précédentes. De plus, lorsque la connexion avec le client est terminée, le serveur esclave envoie une information au serveur master pour indiquer la fin de connexion avec le client. Le serveur master conteint deux processus exécutants. Le premier effectue les connexions via *Accept* et les envoie au deuxième processus par le biais d'un *pipe.* Le deuxième processus reçoit les informations et les transmet à son tour au serveur esclave. Ce système permet de réduire le temps d'attente de connexion des clients au serveur.

- Plusieurs demandes de fichier par connexion :
  fonctionne sans renouveler la connexion avec le serveur FTP. La connexion est alors terminée lors de la réception par le serveur de la commande *bye*.

- Commandes ls, pwd, cd : non fonctionnel
  les commandes ls et cd ont été implémentées et fonctionnent, elles sont appelées par la fonction *execcmd* qui effectue un switch traitant les différentes commandes. Cependant, la partie du programme permettant au client d'envoyer les commandes et de recevoir le résultat est incomplète, par conséquent ces commandes peuvent seulement être exécutées depuis le serveur pour le moment. Les commandes ont été implémentées grâce à la bibliothèque *<dirent.h>*.

Description des tests :

- Téléchargement d'un fichier stocké sur le serveur : testé avec l'envoi de fichiers texte, jpg et mp4.

- Découpage du fichier téléchargé : testé en affichant chaque bloc de données envoyé sur la sortie standard suivi d'un temps d'attente avant l'affichage du bloc suivant.

- Gestion des pannes côté client : testé suivant le fonctionnement décrit dans la partie Réalisations.

- Load balancer : testé avec quatre clients et deux serveurs ayant chacun cinq processus exécutants. Vérification que le Round Robin fonctionne bien.

- Plusieurs demandes de fichier par connexion : même système que pour l'envoi de fichier unique, mais il suffit de ne pas fermer la socket après l'envoi d'un fichier.

- Commandes ls et cd : fonctionnent côté serveur.

Conclusion :

Une bonne majorité des améliorations ont été implémentées. Les notions et fonctionnements des entités client et serveur sont bien assimilées et ont permis de comprendre l'échange d'informations ainsi que les vérifications effectués par un serveur FTP classique.