Projet Socket

Téléchargement FTP P2P

But : le but de ce projet est de créer une application répartie en C de téléchargement de fichier compatible avec le protocole FTP en mode P2P (peer to peer ou poste à poste).
Les étapes suivantes sont obligatoires et vous devrez rendre le projet par étapes.

Étape 1 : Téléchargement FTP simple

La première étape doit permettre de télécharger un fichier d'une machine vers une autre machine en suivant le protocole FTP le plus simplement possible (voir implémentation minimale / minimum implementation dans la RFC ci-dessous).

Plan de travail :

- Lire le sujet jusqu'au bout et la RFC FTP (version anglaise, version française) ;
- Écrire l'application serveur et l'application cliente et les tester avec d'autres clients et serveurs FTP (on pourra, par exemple utiliser Filezilla client et serveur ou encore des sites FTP publics comme ftp.lip6.fr, ftp.free.fr ...

Étape 2 : Reprise sur erreur et mode Bloc

Dans la seconde étape, on permet à un client de reprendre le téléchargement s'il y a eu un problème lors d'un précédent téléchargement.
Pour cela, on implantera le mode Bloc qui permettra donc de télécharger n'importe quel bloc d'un fichier. Ainsi on pourra uniquement télécharger les blocs manquant d'un fichier téléchargé partiellement.

Plan de travail :

- Modifier le serveur pour gérer l'envoi de n'importe quel bloc d'un fichier ;
- Modifier le client pour qu'il puisse gérer la reprise sur erreur.

Étape 3 : Téléchargement en parallèle

Dans cette étape, on permet à un client de télécharger le fichier depuis plusieurs serveurs.
Ainsi les blocs d'un fichier seront téléchargés depuis plusieurs serveurs.

Dans cette étape c'est le client qui choisit (par exemple aléatoirement) quel bloc télécharger depuis quel serveur.

Plan de travail :

- Modifier le client pour qu'il puisse se connecter à plusieurs serveurs, demander le téléchargement de n'importe quel bloc et re-créer le fichier complet.

Étape 4 : Transformation en P2P simple

Dans cette étape, les applications sont à la fois client et serveur.
Chaque application devra noter de quelle partie du fichier elle dispose. Au démarrage certaines applications auront le fichier complet et les autres n'auront aucun bloc.
Les applications demanderont aléatoirement chaque bloc manquant à n'importe quelle autre application qui renverra soit le bloc soit un message d'erreur.

Étape 5 : P2P coordonné

Dans cette étape, on ajoute un serveur dont le rôle est de maintenir la liste des applications gérant le téléchargement d'un fichier et quel bloc chaque application possède.
Ce serveur coordonnera le téléchargement en précisant à chaque application, à qui se connecter et ce qui y est disponible.

- Concevoir l'application répartie avec UML ;
- Si vous avez le temps, proposez une implantation de cette application répartie ;

Options :

- Créer une IHM (interface homme machine) pour les applications en GTK ou QT (attention dans ce cas il y aura une partie en C++).
- Permettre la recherche de fichiers à partir de leur nom ou de toute autre caractéristique. A l'issu de la recherche on devra pouvoir connaître un ensemble d'application possédant le fichier et commencer le téléchargement.
- Permettre un P2P coopératif en s'assurant que les applications envoient et reçoivent globalement les même quantités.
  On essaiera ainsi de désavantager les applications qui ne font que télécharger et n'envoient rien.

Note : toute fonctionnalité supplémentaire ne sera prise en compte dans la notation que si toutes les étapes ont été correctement traitées.