Barbaste Clément
Schummer Bastien
Master 1 IMAGINA

Projet
réseaux et communication

Mode d'emploi du programme :

Tout d'abord lancez le server sur une machine . Puis lancer le client sur une ou plusieurs autres machines et donnant l'adresse IP du server en paramètre.

Le client dispose de trois commandes de base :

export <nomdufichier> pour exporter un fichier

import<nomdufichier> pour télécharger un fichier

exit pour quitter l'application

Architecture de l'application :

L'application se divise en 2 parties : une partie server et une partie client. Le server ne stocke aucun fichiers mais il est nécessaire pour donner pour que les clients connectés puissent se partager leur adresse IP après chaque déconnexion (vu que celle-ci est dynamique).

Partie server :

L'architecture de la partie server est simple. Il ouvre un port d'écoute qui accepte toutes les connexions dans une boucle. Lorsqu'un client se connecte, il le gère dans un thread, le tread principal se remet à attendre des connexions, tandis que les threads secondaires (pour gérer les clients) retournent les données (souvent des adresses IP de pair) aux clients.

L'intérêt des threads et de pouvoir gérer plusieurs pair connectés en même temps, ils nécessitent néanmoins une gestion des ressource partagées plus complexe (mutex).

Il y a une liste doublement chaînées de structures qui contient la liste des pair connectés avec une correspondances entre leur adresse IP et une clé unique à chaque pair. Cette liste est une variable globale (pas besoin de mémoire persistante vu qu'elle doit être nettoyée couramment car les clients n'ont pas une adresse IP statique). Chaque thread doit être synchronisé dans la modification de cette liste.

Le server héberge également des fichiers méta-data qui sont des données persistantes. Il y a un fichier méta-data pour chaque fichiers partagés entre pairs. Ce fichier méta-data contient le nom du fichier partagé, sont identifiant unique pour le server, la liste des pair qui en ont une partie. L'accès à un de ces fichiers par un thread doit être géré et ce pour tous les fichiers.

Partie client :

L'application client est particulière car elle dispose premièrement d'un thread qui joue un rôle de serveur qui écoute les requêtes du serveur ou des pairs en boucle. Ce thread est lancé dès le démarrage de l'application. Le serveur et des pairs peuvent alors s'y connecté. A chaque connexion entrante, il créer un nouveau thread pour traiter plusieurs requêtes simultanément.

Ensuite, lorsque l'utilisateur saisie une commande d'export ou d'import, un thread sera également lancé pour chaque commande. Ainsi, l'utilisateur peut très bien demander un export, suivit d'un téléchargement, même si son export n'est pas terminé.

Lors de la première connexion, le serveur donne une clé de pair unique qui sera stocké dans un fichier local du pair contenant sa clé de pair. Lors des prochaines connexions,

elle sera envoyée au serveur, qui pourra ainsi le reconnaître. Les pairs possèdent aussi des fichiers .part qui sont des partitions de fichiers stockées.

Protocoles d'échanges

Partie server :

La partie server utilise les sockets en mode connectés pour communiquer avec les pair. Le mode connecté est utilisé pour pouvoir dialoguer avec les pairs, envoyer plusieurs messages facilement, depuis le pair ou le server ainsi que des fichier.

Comme le server doit pouvoir répondre à plusieurs requêtes, un protocole de communication entre le server et un pair a été mis en place. Un simple code à 3 chiffres définit la nature d'une requête. Il peut être suivi d'un ou plusieurs arguments.

Voici la liste des messages possible entre le server et le client.

Connexion initiale au server:

100 monAddrIP/Port	"Je suis un pair et je suis connecté à cette addrIp."
101 cléPair	"Voilà ma clé unique en tant que pair".
102	"Je suis un nouveau Pair, je n'ai pas de clé client."
103 cléPair	"Voila votre cléPair unique à conserver."

Envoi de données:

200 des	"Donne moi la liste des pair qui peuvent recevoir données."
201 cléFichier addrServer/port\n...	"Je t'envoi la liste des pairs après le code '201 '."
202	"Je t'envoie le fichier metat-data après le code '202 ' ."
203	"Peux-tu recevoir un fichier."

Télécharger un fichier:

300 cléFichier une	"Donne moi la liste des addrIP des pairs ayant partition de ce fichier."
301 fichierMeta-data ce	"Je t'envoie la liste des pairs ayant un bout de fichier. "
302 cléFichier	"As-tu une partition de ce fichier ?"
304 cléFichier cléPair fichier	"Le pair cléPair n'as plus la partition du cléFichier."

Mise à jour meta-data :

400 cleFichier/Part\n... tes	"Je t'envoie la liste des partitions que j'ai pour mettre données à jour."

Message généraux:

901	"Ok"
902	"Refus"
903 addrServer/port	"Refus je t'envoie les explications dans un fichier sur ce socket"
904	"Reçu"
905	"Envoi terminé"
906	"Toujours connecté?"
907	"Oui"
908	"Non"
909	"Fermeture de la connexion."

Il y a 3 requêtes client implémentés sur le server :

- L'initialisation de la connexion. Lorsque le client est lancé, il va dire au server sur quelle IP il est connecté, et quel est son port d'écoute (les pair on aussi une partie 'server' qui leur sert à recevoir des requêtes venant du server ou des autre clients).
La liste possible des messages envoyés (dans l'ordre et on suppose sans erreurs) :

- Pair → 100
ENSUITE
{
- Server → 901
- Pair → 101
- Server → 901
}
OU
{
- Pair → 102
- Server → 103
- Pair → 901
}

- Envoi de données. Un pair peut demander au server la liste des client connectés pour partager un fichier avec eux.
La liste possible des messages envoyés (dans l'ordre et on suppose sans erreurs) :

- Pair → 200
- Server → 201
… Le pair partage son fichier …
… Le pair construit le fichier meta-data avec une correspondance IP/CléFichier …
- Pair → 202
… Le server enregistre le fichier méta-data en remplacent les addresses IP par une cléPair (cette fonctionnalitée n'est pas implémentée) …

- Server → 901

- Telecharger un fichier. Un pair demande la liste des pairs ayant une partie du fichier identifié par cléFichier. Le server lui donne le fichier meta-data correspondant à ce fichier.
La liste possible des messages envoyés (dans l'ordre et on suppose sans erreurs) :

- Pair → 300
- Server → 301
- Pair → 901

Partie Client :

Lors de imports et d'exports, les pairs sont donc aussi amenés à communique entre eux, toujours en respectant la convention d'un code de message à 3 chiffres.

Envoi de données :

600 cléFichier contenu «envoi d'une partition à stocker correspondant au fichier cléFichier

601 « Partition reçu et enregistrée »

602 « Impossible de stocker le fichier »

Télécharger un fichier :

500 cléFichier «demande de téléchargement de la partition du fichier ayant pour clé cléFichier »

501 contenu « message d'envoi de la partition»

Il y a donc 2 dialogues différents possibles entre chaque pair.

Envoi de données : Pair1 souhaite envoyer une partition à stocker chez Pair2.

- Pair1 -> 600

Si l'enregistrement s'est bien passer :

- Pair2 -> 601

Sinon

-Pair2-> 602

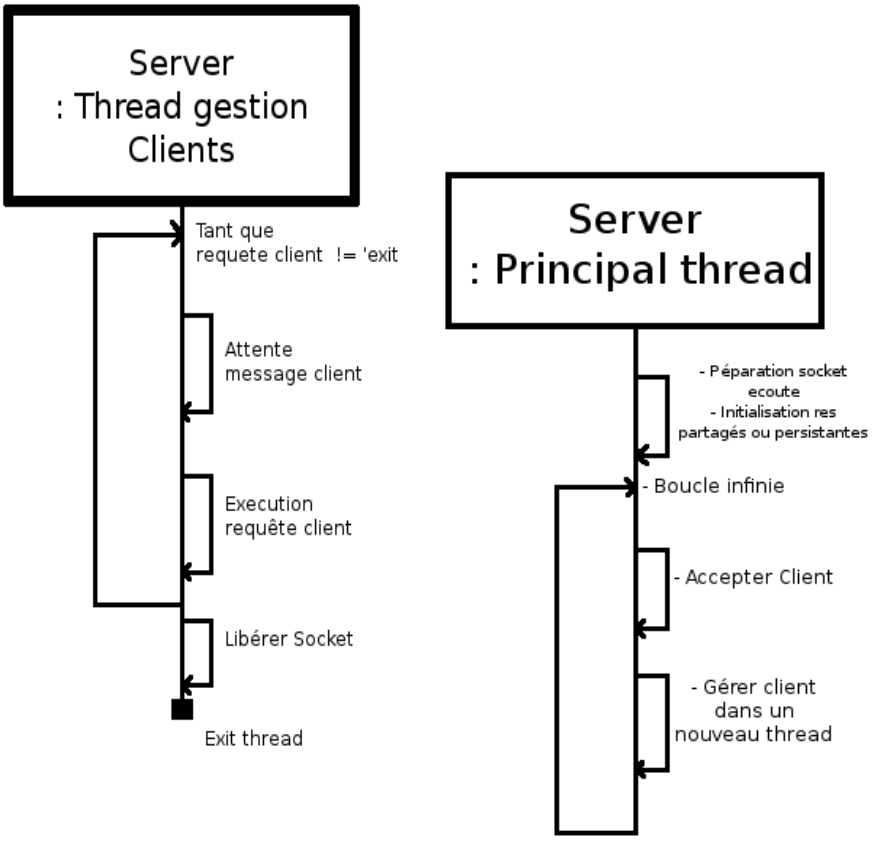
Télécharger un fichier : Pair1 souhaite télécharger une partition stockée chez Pair2. Le dialogue est simple :

- Pair1 -> 500

-Pair2-> 501

Schémas algorithmiques :

Partie server :



Partie Client :