Sylvain VISSIÈRE-GUÉRINET
Soline BLANC

2ème année
Ingénierie du Logiciel

Rapport de Projet

Réseaux et Systèmes Avancés

TELECOM Nancy
2013-2014

Plan

Présentation du projet

Sujet

Le but de ce projet est de mettre en place un système d'échange de fichiers en mode P2P (pair à pair). Ce système sera composé de plusieurs entités : un serveur, qui pourra répondre aux demandes de publication (requête publish) et de recherche (requête search) qui permettra à des clients de publier ou de rechercher des fichiers retrouvés grâce à des mots clefs.

Chaque client aura également un côté purement "client", qui questionnera des serveurs et récupèrera leurs réponses, mais également un côté "serveur", qui sera capable d'envoyer des fichiers qu'il aurait publié et que d'autres clients lui auraient demandé.

Utilisation de Git

Pour faciliter la mise en commun des différents travaux effectués par les deux membres du groupe, et être capable d'avoir toujours la même version chacun, nous avons décidé d'utiliser un dépôt Git pour notre projet.

En voici le lien : https://github.com/SylvainVISSIERE-GUERINET/reseauCtorrent.git.

Stockage des données

Stockage au niveau du serveur

Où sont stockées les données ?

Au niveau du serveur, les données des différents fichiers sont stockées dans des fichiers textes contenus dans un dossier "data". Chaque nom de fichier correspond à un mot clef de la manière suivante : si un mot clef est "motClef1", les données de chaque fichier dont la liste des mots clefs contient "motClef1" sont contenues dans une ligne du fichier "motClef1.txt" présent dans le dossier "data".

Ainsi, les données d'un même document sont présentes dans autant de fichiers de data que le document a de mots clefs.

Sous quel format sont-elles stockées ?

Nous avons dit précédemment que les différentes données sont stockées dans des lignes de fichiers. En effet, pour chaque dossier, les données sont stockées sous forme d'une chaîne de caractères au format suivant :

nom|type|motClef1;motClef2;motClef3|hash|IPduClient

Stockage au niveau des clients

Où sont stockées les données ?

Dans chaque client, les différents fichiers sont stockés dans le dossier "Partage".

Hashage du contenu des fichiers

Dans les échanges entre clients comme dans les interactions client-serveur, il y a intervention du hashage du contenu du fichier en SHA1. Tout d'abord, nous l'avions fait en stockant d'abord tout le contenu du fichier dans un pointeur de caractères. Mais en considérant des fichiers plus gros, nous devons hasher le contenu du fichier au fur et à mesure de la lecture, d'où l'utilisation d'un buffer à dimension finie, souvent plus petite que le contenu du fichier. A partir de là, la fonction "update" de openssl/sha a été utilisée.

Comme le hash du contenu va être stocké dans les fichiers sous forme de chaîne de caractères et affiché dans le terminal, il a fallu transformer chaque caractère de ce que retournait le hash en deux caractères, ce qui retournera des caractères plus "communs" (lettres, chiffres...), car imprimés sur moins de bytes. De base, SHA_DIGEST_LENGTH est égal à 20. Ainsi, comme de chaque caractère hashé, nous en extrayons deux, la longueur du hash final sera de 40 caractères, auquel on ajoute un caractère de fin de ligne, ce qui donne en tout 41 caractères.

Interactions Client – Serveur (UDP)

Le client

Le client est codé dans le fichier Client.c. Quand on lance le client, il faut entrer en paramètre l'adresse IP du serveur qu'il interrogera. Il aura plusieurs choix d'action affichés via une interface console :

- Publier un fichier
- Rechercher un fichier
- Télécharger un fichier
- Quitter.

Les deux choix qui nous intéresse dans les interactions client – serveur sont les deux premiers : publier un fichier et rechercher un fichier.
Lorsque le client fait son choix, cela appellera la fonction publish s'il a choisi de publier un fichier, ou la fonction search s'il a choisi de rechercher un fichier. Ces deux fonctions sont également comprises dans le fichier Client.c. Celles-ci prennent chacune en arguments :

- **int serverSocket :** socket qui sera initialisée dans la fonction
- **struct sockaddr_in * serv_addr :** structure en partie remplie, dont le port sera défini dans la fonction (dans publish, il sera mis égal à 2223)
- **char * servIP :** qui est l'adresse du serveur, qui a été passée en argument lors de l'appel du client
- **socklen_t len :** qui est égal à la taille de serv_addr

Le début des deux fonctions est le même :
le port du serveur va tout d'abord être stocké dans serv_addr. Il s'agit de 2223 pour la publication, et de 2222 pour la recherche de fichiers. Ensuite, le client ouvre une socket UDP qui pingera le serveur. C'est là que les deux fonctions commencent à différer.

Le serveur

Afin de pouvoir répondre aux demandes de publication et de recherche, le serveur est séparé en deux parties, respectivement codées dans les fichiers ServerPublish.c et ServerSearch.c. Chaque serveur a un port déterminé. Ainsi, ServerPublish sera accessible à partir du port 2223, et ServerSearch sera accessible à partir du port 2222.

Contrairement au client, le serveur ne prend par d'argument à son lancement. Les deux serveurs commencent eux aussi de manières similaires :
tout d'abord, le serveur créera une socket, qu'il associera à une interface et un port au travers d'un bind.
Après cela, il entrera dans une boucle infinie (while(1)) dans laquelle il attend un message "PING" de la part d'un client. Une fois ce PING reçu, il fera un fork. Le père continuera son attente de client sur serverSocket, tandis que le fils fermera cette socket pour en créer une nouvelle (dialogSocket) avant de répondre au client qui avait envoyé le PING. Tout d'abord, il enverra un "PING ACK" au client. A partir de là, les agissements des deux serveurs diffèrent et seront donc traités dans les points suivants.

Remarque : La seule différence dans les débuts des deux serveurs est que le serveur de publication créera un dossier "data" si celui-ci n'existe pas. Le serveur de recherche n'opérant qu'en lecture, il ne lui sera pas nécessaire de faire cela : si le dossier n'existe pas, aucun fichier ne peut de toutes façons être trouvé.

Publier un fichier

Après avoir reçu le ACK de confirmation de réception du PING, le client parcourera tous les fichiers qu'il a dans son dossier "Partage" et les affichera. Là, l'utilisateur va devoir répondre à quelques questions afin de pouvoir envoyer son fichier :

- quel fichier parmi la liste créée veut-il publier ?
- quel est le type de ce fichier ?
- quels sont les mots clefs qui seront à associer à ce fichier ?

Une fois que l'utilisateur a répondu à ces questions, le contenu du fichier concerné est récupéré et hashé en Sha1. Le hash récupéré fera 41 caractères, en comptant le EnfOfFile. Enfin, ces différentes informations seront regroupées en une chaîne de caractères où chaque information sera séparée par des pipes "|", sauf les mots clefs qui seront entre eux séparés par des points-virgules ";". Cette chaîne est ensuite envoyée au serveur.

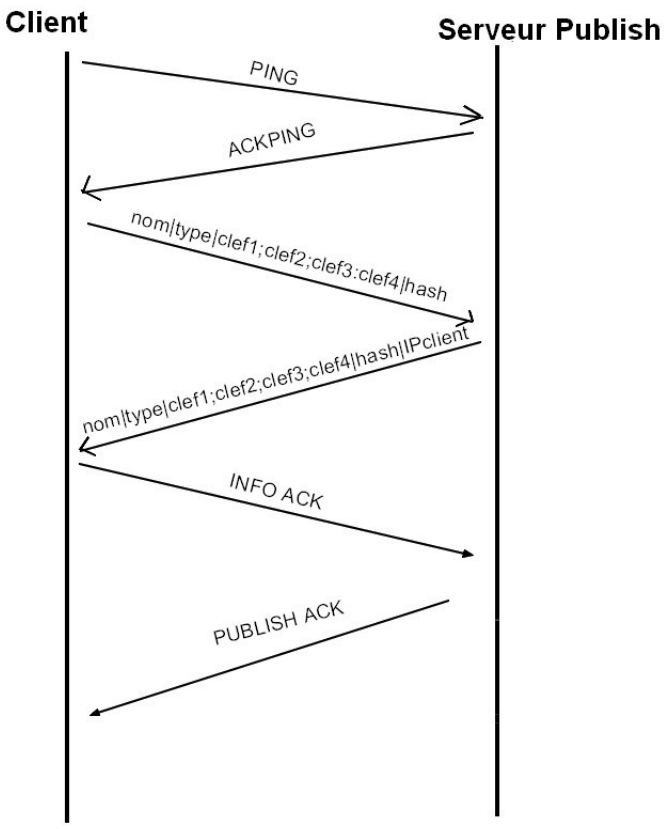
De son côté, le serveur attendait une réponse du client. Une fois que ce dernier a envoyé les différentes informations du fichier, le serveur y ajoute, après un pipe, l'adresse IP du client qu'il aura récupérée. Il renverra ensuite cette chaîne de caractères au client.

Une fois les différentes informations reçues, le client les affichera. L'utilisateur pourra alors les valider ou non. S'il les refuse, les questions lui seront posées à nouveau et la procédure recommencera. S'il les valide, un message "INFO ACK" sera envoyé au serveur.

Quand le serveur a reçu confirmation, il affiche la chaîne de caractères contenant les différentes informations et splite cette chaîne pour en extraire les différents mots clefs liés au fichier. La liste des données sera enregistrée dans un ou plusieurs fichiers, en fonction du nombre de mot clef : pour chaque mot clef "motClef", une ligne dans le fichier "data/motClef.txt" indiquera l'existence du fichier.

Une fois qu'il a enregistré les fichiers de la manière expliquée précédemment, le serveur enverra un dernier message au client avant de fermer la socket de dialogue : "PUBLISH ACK".

De son côté, le client, en recevant ce message, va afficher sur la console de l'utilisateur "Publication faite".



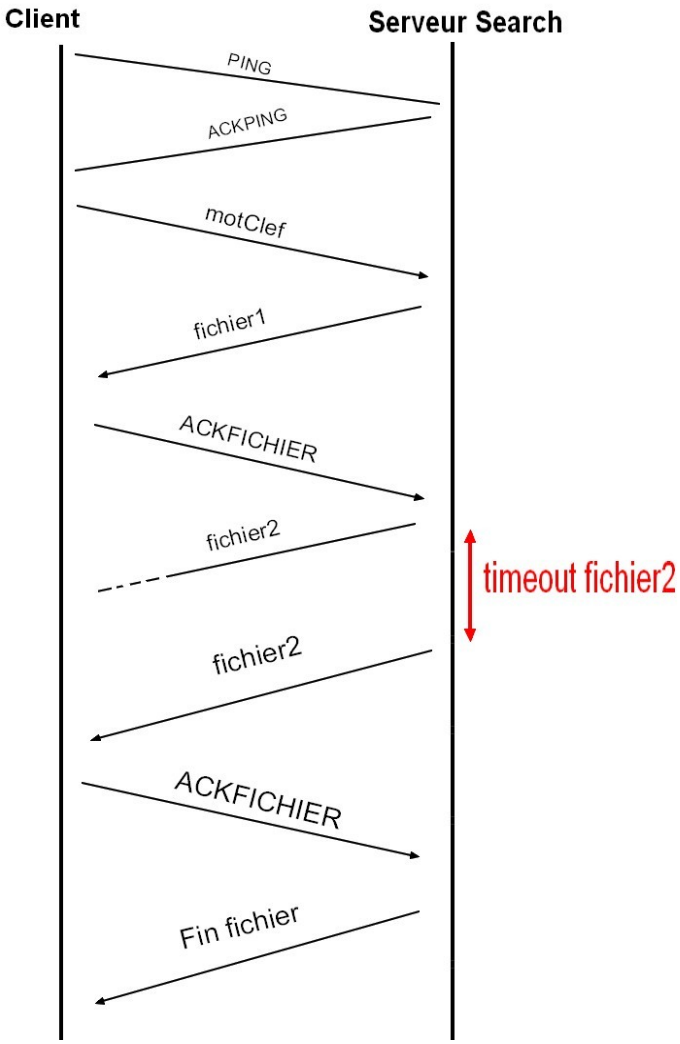
Rechercher un fichier

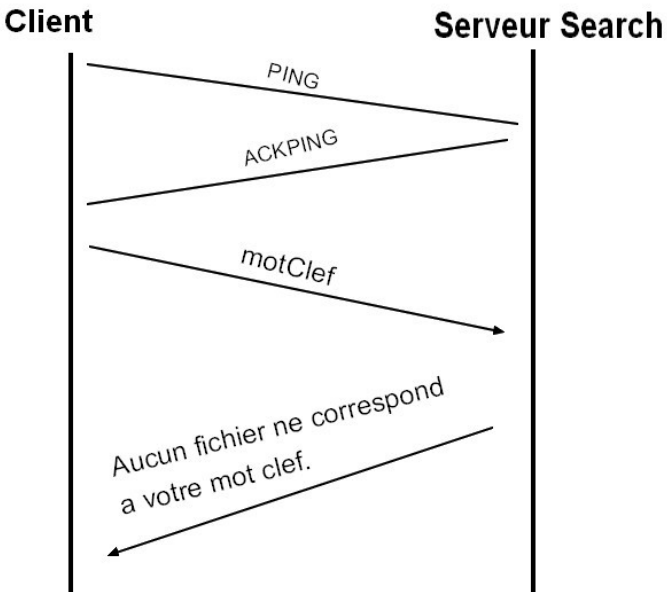
Après avoir reçu le ACK de confirmation de réception du PING, il sera demandé à l'utilisateur d'entrer le mot clef qui sera utilisé pour la recherche de fichiers. Une fois ce mot clef saisi, il est envoyé au serveur.

Après réception de ce mot clef, le serveur l'affichera sur la console et ouvre le fichier "data/motClef.txt". Si ce dernier ne s'ouvre pas, c'est qu'il n'a pas été trouvé et donc qu'aucun fichier publié ne correspond à cette requête, le serveur enverra donc un message au client disant qu'il n'existe pas de fichier correspondant à ce mot clef, après quoi il fermera la socket de dialogue. Cependant, si le fichier s'ouvre, chaque ligne qu'il contiendra comportera les données d'un fichier possédant ce mot clef. Le fichier parcourera alors chaque ligne et, pour chacune d'elle, enverra la chaîne de caractères contenant les informations du fichier concerné.Chaque fois que les données d'un fichier seront envoyées, un timer sera attaché à la socket, qui attendra un "ACKFICHIER" de la part du client. Le timer dure 5 secondes. S'il y a timeout, la ligne concernant le fichier sera renvoyée jusqu'à réception d'un ACKFICHIER.

Une fois tous les fichiers envoyés, le serveur envoie un message "Fin fichier" au client avant d'afficher le nombre de fichiers qui a été réellement envoyé au final (donc le nombre de lignes différentes envoyées, un fichier n'est alors pas compté plusieurs fois, même en cas de timeout). Enfin, le serveur fermera la socket de dialogue.

De son côté, le client affiche sur la console de l'utilisateur les données qu'il reçoit dès réception ; il cesse d'être en attente de fichiers quand il reçoit le message "Fin fichier".





Interactions Client – Client (TCP)

Une fois que le fichier a été publié, et recherché par d'autres, il va pouvoir être télécharger par un autre client. Celui-ci ayant récupéré via le ServerSearch l'adresse IP, le nom de fichier et le hash du contenu du fichier en SHA1, il va pouvoir se connecter au client grâce à l'adresse IP, demander le fichier souhaité grâce à son nom et véfifier qu'il s'agit du bon en comparant les hash.

ServerClient

Au lancement de ServerClient, aucun paramètre supplémentaire n'est nécessaire : une socket TCP (sockfd) est créée. Après lui avoir lié l'adresse locale et fait un bind, on fait un listen dans une boucle infinie : on va être continuellement à l'écoute de nouvelles connexions qui arriveraient.

Échanges entre ServerClient et Client

Quand l'utilisateur choisit de télécharger un fichier, le client va tout d'abord lui demander d'entrer trois informations : le nom et l'extension du fichier souhaité, l'adresse IP de la source qui contiendrait ce fichier et le hash du contenu du fichier.

Une fois ces informations entrées, on remplit la structure serv_addr et on ouvre une socket TCP. Le client tentera alors une une connexion au serveur grâce à la fonction connect. Du côté de ServerClient, accept va répondre au connect et créer une nouvelle socket qui se chargera de la liaison avec ce client. On fait alors un fork : le fils fermera sockfd et ne conservera alors que la nouvelle socket : newsockfd. Le père, en revanche, fermera newsockfd et ne conservera que sockfd.

Au niveau du fils, on reçoit du client un message contenant le nom du fichier et son extension. On va alors dans le dossier Partage et on ouvre le fichier concerné s'il existe. On parcourt alors ce fichier, dans une boucle où l'on remplit le buffer autant que possible avant d'en envoyer le contenu au Client. Chaque fois qu'il reçoit des données, le Client réécrit ce qu'il reçoit dans son propre dossier Partage, dans un fichier du même nom.extension. Cela est réitéré jusqu'à ce que le ServerClient arrive au bout du fichier.

Enfin vient la comparaison des deux hashs : si les deux hashs ne sont pas identiques, rien n'est annulé ou supprimer, mais un message d'avertissement est envoyé à l'utilisateur. Enfin, apparaît dans la console Client un message "Fin Telechargement".