

Réalisation d'un réseau Peer-to-Peer centralisé

TABLE DES MATIÈRES

Table des matières.....	1
Présentation de l'application.....	2
I. Introduction.....	2
II. Captures d'écrans.....	2
Généralités.....	3
I. Le client.....	3
II. Le serveur.....	4
Conception.....	5
I. Diagramme de classes.....	5
II. Diagrammes de séquences.....	6
Conclusion.....	7

PRÉSENTATION DE L'APPLICATION

I. Introduction

Nous avons choisi de mettre en place un réseau Peer-to-Peer centralisé. Ce réseau est donc composé d'un serveur et de clients devant s'y connecter lors de la recherche de fichiers. Ce réseau est implémenté en RMI afin de faciliter les communications entre les différents éléments. Les principes de transfert de données basiques tels que l'utilisation de sockets sont ainsi masqués.

Notre implémentation permet :

- La recherche de fichier selon un mot clé,
- Le téléchargement d'un fichier depuis plusieurs sources simultanément,
- Le téléchargement de plusieurs fichiers simultanément,
- La reprise du téléchargement d'un fichier partiellement téléchargé,
- La déconnexion « sauvage » des clients.

Néanmoins, il n'est pas possible :

- De partager des parties de fichier non complet (un fichier en cours de téléchargement ne peut être téléchargé par d'autres peers simultanément),
- De contrôler les vitesses d'émission et de réception.

Nous pouvons néanmoins ajouter que l'architecture de notre application ne poserait aucun problème à qui voudrait implémenter ces fonctionnalités.

Étant donné la technologie utilisée (RMI), notre programme doit nécessairement être écrit en Java.

II. Captures d'écrans

Avant de passer à des considérations plus techniques, voici des captures d'écrans de l'application :

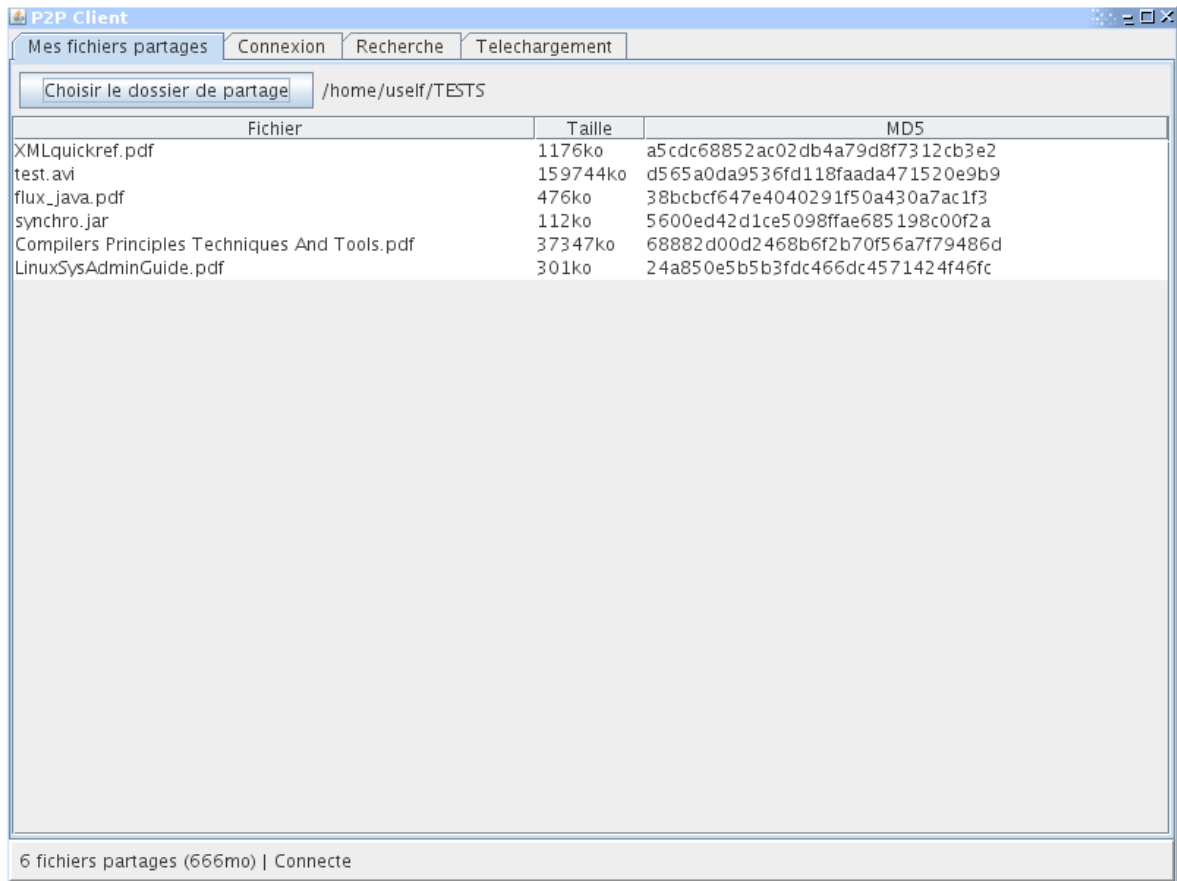


Illustration 1: Mes fichiers partagés

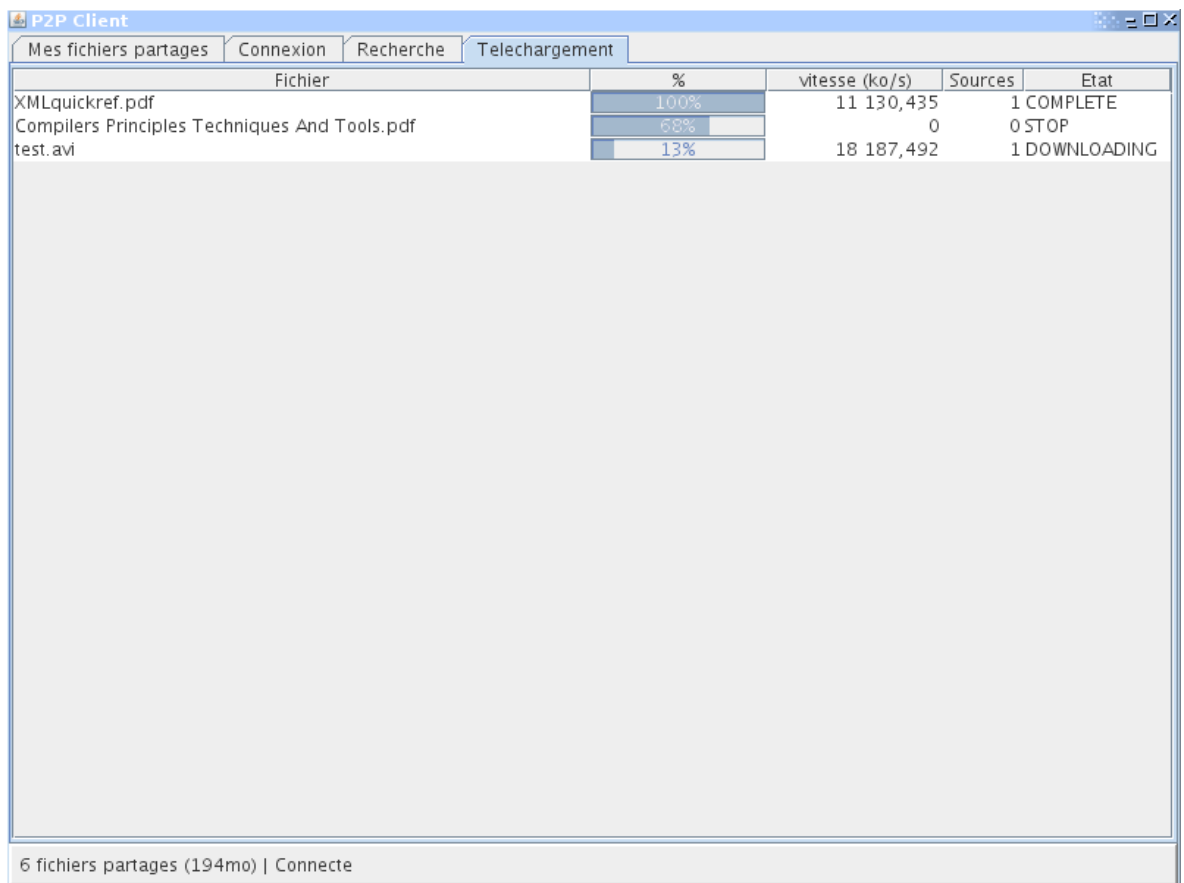


Illustration 2: Téléchargements

GÉNÉRALITÉS

I. Le client

Le client est la partie « visible » et utile de notre système. C'est l'application permettant de partager et de télécharger des fichiers.

Avant de pouvoir télécharger, un client doit remplir certaines contraintes :

- Il doit partager des fichiers. Il sera en effet impossible à un client de se connecter si il ne partage aucun fichier. Ce critère quantitatif n'est pas qualitatif, il est tout à fait possible de partager un seul fichier, n'ayant aucun contenu. La encore, l'architecture adoptée ne serait pas un frein à une amélioration de ceci.

Lors du chargement des fichiers, un hash MD5 de chaque fichier sera calculé. Il servira à différencier de manière unique chacun d'entre eux.

- Il doit se connecter au serveur, et l'informer de sa liste de fichiers partagés, ainsi que l'adresse (de la forme « //192.168.1.1:1100/FileServer ») ou il sera possible de télécharger les fichiers partagés. Ceci est transparent pour l'utilisateur, néanmoins, pour se connecter, celui ci devra entrer l'adresse IP du serveur et le port sur lequel il écoute.

Ceci étant fait, le client peut effectuer des recherches sur le serveur et télécharger des fichiers sur d'autres clients.

II. Le serveur

Le serveur ne dispose pas d'interfaces graphiques. Il a pour rôle :

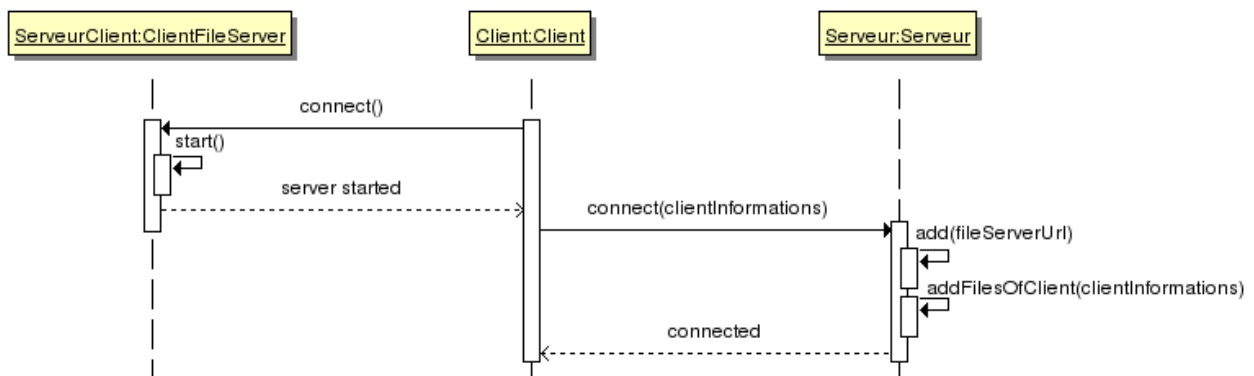
- D'attendre des clients et de référencer les fichiers qu'ils partagent.
- De répondre aux requêtes des différents clients. Autrement dit, il donnera une liste d'adresse de clients possédant le fichier « f » correspondant au critère demandé par le client.
- De vérifier périodiquement que les clients ayant des fichiers référencés soient bien toujours connectés.

Notre application est divisée en trois parties distinctes :

- Le serveur : constitué par la classe « Server » héritant de “Iserver” et possédant une liste de fichiers. Ces fichiers sont représentés chacun par une instance de la classe “ServerFile” qui elle-même possède une liste des clients disposant de ce fichier.
- Le client : Un client est représenté par la classe « Client ». Un client dispose également d'un serveur personnel permettant à quiconque de se connecter à lui pour récupérer des parties de fichiers. C'est le rôle de la classe « ClientFileServer » qui étend « IClientFileServer ». Un serveur client est accessible grâce à une URL et un port. Pour la gestion du téléchargement de fichiers, nous avons mis en place un système de gestion des parties de fichiers téléchargées utilisant des fichiers XML indiquant les octets récupérés. Ceci à pour but de pouvoir reprendre un téléchargement en cas de coupure de l'application ou encore de télécharger des parties sur des sources différentes. La classe « PartFileManager » gère cela. Chaque partie est représentée grâce à la classe « Part » qui spécifie un octet de départ et un octet de fin. La classe « DownloadFileManager » gère le téléchargement d'un fichier et exploite la classe « DownloadRunnable » qui gère le téléchargement d'une partie dans un thread.
- Une partie partagée par les clients et les serveurs : Le client et le serveur côté client disposent tous les deux d'une liste de fichiers partagés. Ces fichiers sont des « ClientFile ». Cette classe recense le hash MD5, le chemin d'accès au fichier ainsi que sa taille. La classe « ClientInformations » regroupe l'adresse du serveur d'un client ainsi que la liste de ses fichiers partagés. C'est une instance de cette classe qui sera envoyée au serveur central lors de la connexion du client afin de référencer ses fichiers. Les interfaces « IClientFileServer » et « IServer » permettent pour la première à des clients de récupérer les fichiers d'un autre et pour la deuxième à des clients de se connecter au serveur central. Il reste ensuite la classe « SearchResult » qui contient une liste d'instances de la classe « Result ». Cette dernière contient les informations relatives à un fichier. Ces deux classes servent lors de la recherche de fichiers par les clients sur le serveur central.

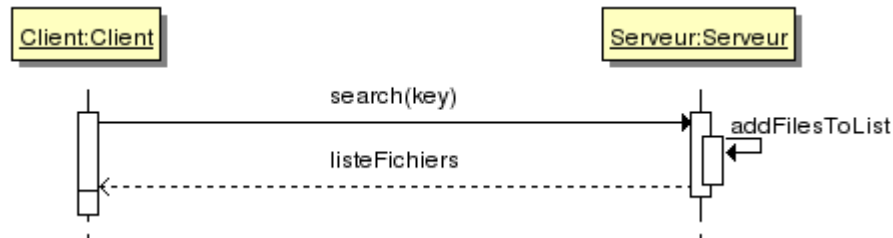
II. Diagrammes de séquences

- Connexion au serveur central



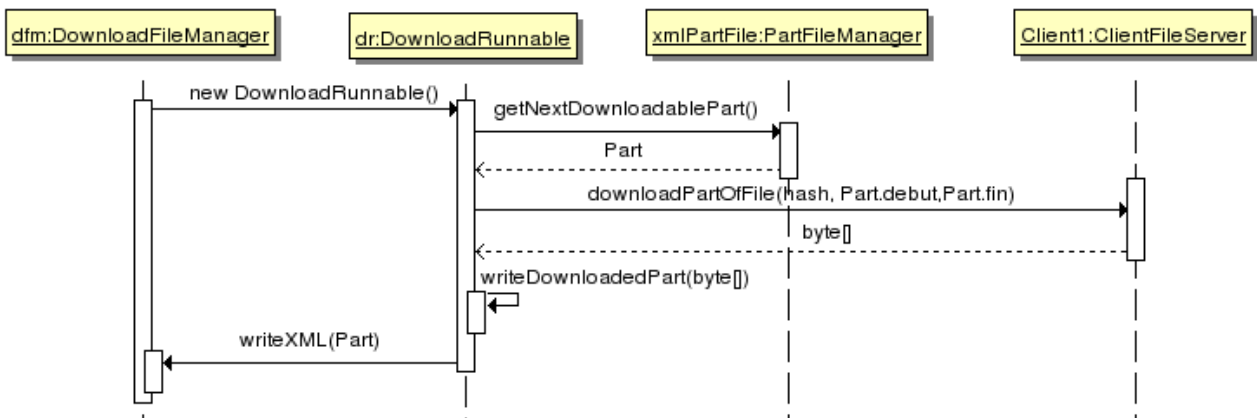
Lorsqu'un client demande à se connecter au serveur central, il met d'abord en marche son serveur de fichier. A ce moment là sont référencés tous les fichiers qu'il partage. Ensuite, il se connecte au serveur central et lui envoie les informations le concernant, c'est-à-dire l'adresse de son serveur de fichier et la liste de ses fichiers. Le serveur central rajoute le serveur client dans sa liste ainsi que la liste des fichiers référencés.

- Recherche de fichier



Nous avons mis en place une fonction de recherche de fichier. Un client soumet une requête de recherche au serveur contenant un mot clé portant sur le nom de fichier. Le serveur recherche dans la liste des fichiers référencés ceux dont le nom comporte le mot clé grâce à une expression régulière et les ajoute dans une liste qu'il retourne au client.

- Téléchargement d'une partie de fichier



Ce diagramme de séquence présente le téléchargement d'une partie de fichier. Ceci est assuré par le downloadFileManager. Un téléchargement sur un client est assuré par la classe DownloadRunnable dans un thread. Une boucle est effectuée dans laquelle sont récupérées les parties manquantes du fichier jusqu'à ce que le fichier soit complet. Pour chaque partie, cette classe récupère l'octet de début et l'octet de fin de la prochaine partie manquante grâce à la méthode « getNextDownloadablePart » associé au fichier à télécharger par l'intermédiaire de la classe PartFileManager. Il récupère ensuite cette partie du fichier par le serveur du client source. Ce morceau de fichier est ensuite écrit en local. Périodiquement, les parties récupérées sont renseignées dans le fichier XML grâce à un thread exécuté à partir de la classe « DownloadFileManager ».

CONCLUSION

Notre programme met à disposition une réelle solution de partage de fichiers. En effet, après de nombreux tests, nous avons pris conscience de la réelle efficacité et stabilité du système. Certes,

quelques petites optimisations seraient certainement nécessaires pour un usage avec des milliers d'utilisateurs, néanmoins, pour une société, une université, où toute autre organisation de taille relativement réduite, cette solution est tout à fait envisageable. De plus, sa conception permet une évolutivité simple.

De plus, du fait du langage utilisé, notre programme est interopérable ; il peut être exécuté sous n'importe quel système d'exploitation disposant d'une jre1.6.