18 janvier 2025

2ème année en SN

FLOISSAC Léo - LAGRANGE Angel
Rapport du Projet : Hagimule

Élèves :
Angel LAGRANGE
Léo FLOISSAC

Enseignant
D. HAGIMONT

Département de Science du Numérique

Table des matières

1 Introduction

Ce projet vise à concevoir un système distribué permettant la gestion et le partage de fichiers entre plusieurs clients. Les fonctionnalités principales incluent :

- La gestion d'annuaires centralisés des fichiers disponibles.
- Le téléchargement parallèle avec compression des fichiers.
- La tolérance aux déconnexions des clients pendant les transferts.

Le projet utilise Java avec RMI (Remote Method Invocation) pour la communication réseau.

2 Architecture du système

Le système est composé de plusieurs modules, chacun ayant un rôle bien défini :

2.1 Diary

Le **Diary** est le composant central. Il maintient un annuaire des fichiers disponibles sur le réseau et les clients qui les possèdent. Les principales fonctionnalités du Diary incluent :

- Enregistrement des fichiers partagés par les clients, avec leurs tailles.
- Fourniture d'une liste des clients possédant un fichier donné.
- Suppression des clients déconnectés, ce qui garantit la mise à jour de l'annuaire.

Le Diary expose ces fonctionnalités via RMI, permettant aux clients de communiquer efficacement avec lui.

2.2 Client

Le **Client** représente les utilisateurs du système. Chaque client peut :

- Ajouter des fichiers locaux au Diary, signalant qu'il les partage.
- Télécharger des fichiers depuis d'autres clients en parallèle.
- Gérer ses fichiers via une interface en ligne de commande.

Le client interagit avec le Diary pour récupérer des informations sur les fichiers disponibles et utilise le module Downloader pour le téléchargement.

2.3 Deamon

Le **Deamon** agit comme un serveur local pour chaque client. Il écoute sur un port dédié et permet à d'autres clients de récupérer des morceaux de fichiers. Les fonctionnalités du Deamon incluent :

- La partition des fichiers en segments compressés avec GZIP.
- La gestion des requêtes de téléchargement, envoyant les morceaux de fichiers demandés.
- La simulation de délais pour imiter des connexions réseau réelles.

La compression des fichiers est effectuée avant leur envoi, réduisant significativement la taille des données transférées.

2.4 Downloader

Le **Downloader** est responsable du téléchargement parallèle. Il :

- Partitionne le fichier à télécharger en segments, chacun assigné à un client.
- Communique avec les Deamons pour récupérer les segments compressés.
- Décompresse les segments reçus et les assemble pour reconstituer le fichier original.

En cas de déconnexion d'un client, le Downloader détecte l'erreur et sélectionne un autre client disponible pour poursuivre le téléchargement, garantissant ainsi la robustesse du système.

3 Fonctionnalitées

3.1 Parallélisation

La parallélisation est une fonctionnalité clé de notre système, permettant d'accélérer le téléchargement des fichiers en exploitant plusieurs clients simultanément. Elle repose sur les étapes suivantes :

1. **Partitionnement des fichiers :**
   - Lorsqu'un client demande un fichier, le système récupère sa taille totale à partir du Diary.
   - La taille est divisée en plusieurs segments, chacun attribué à un client possédant le fichier. La division est effectuée en parts égales, tout en gérant les éventuels restes pour garantir que tous les segments sont couverts.
2. **Téléchargement parallèle :**
   - Chaque segment est téléchargé indépendamment depuis différents clients via le module *Downloader*.
   - Le téléchargement utilise des threads pour gérer chaque segment simultanément, optimisant ainsi l'utilisation de la bande passante.

3.2 Compression des fichiers

La compression est réalisée en utilisant l'algorithme GZIP. Chaque segment de fichier est compressé avant son envoi via le réseau, ce qui réduit la quantité de données transmises et améliore l'efficacité. Lors de la réception, le Downloader décompresse les segments pour reconstituer le fichier. Cette approche optimise l'utilisation de la bande passante tout en maintenant une bonne vitesse de transfert.

3.3 Gestion des déconnexions

Le système est conçu pour gérer les déconnexions des clients :

- Si un client partageant un segment de fichier se déconnecte, le Downloader retire ce client de la liste active
- Un autre client possédant le fichier est automatiquement sélectionné pour reprendre le téléchargement du segment manquant
- Cette gestion dynamique garantit la continuité du téléchargement sans intervention manuelle

3.4 Simulation des délais réseau

Cette fonctionnalité simule des délais réseau en insérant une pause configurable (en millisecondes) avant l'envoi de chaque paquet. Les données sont transmises par segments fixes (par défaut 1024 octets), permettant de modéliser des conditions de réseau lent ou instable. Le délai est défini individuellement pour chaque client.

4 Décisions importantes

4.1 Choix de Librairie

- **GZIP :** Méthode de compression efficace pour des fichiers de taille variable
- **Swing :** Librairie très utile pour créer des interfaces graphiques facilement

4.2 Passage des stream au bytes

Initialement, nous avions prévu d'utiliser des flux de données (streams) pour la transmission via les sockets. Cependant, cette méthode s'est avérée sous-optimale en raison de sa complexité de gestion et des performances limitées pour les gros fichiers. Nous avons donc opté pour des tableaux de bytes compressés, ce qui a considérablement simplifié le processus et amélioré l'efficacité.

5 Résultats expérimentaux

Nous avons mesuré les performances de notre système avec différents fichiers. Nous avons séparé la durée de téléchargement de la durée totale. La durée de totale inclut en plus de la durée de téléchargement parallèle tous les autres aspects du processus, la compression et la décompression des fichiers, ainsi que le temps nécessaire à l'assemblage final du fichier. En d'autres termes, c'est le temps global nécessaire pour que le fichier soit complètement disponible et prêt à l'emploi sur la machine cible.

5.1 Parallélisation

Le tableau ci-dessous présente les résultats des tests émis sur des fichiers de différentes tailles avec plus ou moins de client possédant le fichier téléchargé :

Nom	Nombre de clients	Octets	Durée du téléchargement (ms)	Total (ms)
image.png	1	71 844	7	15
image.png	2	71 844	5	14
test.txt	1	13 568 311	393	707
test.txt	2	13 568 311	161	491
test.txt	3	13 568 311	113	375
video.mp4	1	411 919 642	24 188	34 044
video.mp4	2	411 919 642	10 259	20 108

Ces résultats montrent que le téléchargement parallèle réduit significativement le temps total, surtout pour les fichiers volumineux.

5.2 Compression

Le tableau ci-dessous présente les résultats des tests émis sur un même fichiers avec et sans compression lors de l'envoi des paquets :

Nom	Compression	Octets	Durée du téléchargement (ms)	Total (ms)
grosFichier.txt	Non	366 944 256	56 564	61 104
grosFichier.txt	Oui	366 944 256	2 471	6 740

On remarque une baisse significative du temps de téléchargement en raison de la diminution du nombre de paquets à envoyer par les démons.

Remarque : Les tests ayant été fait en local ont été fait avec un délai rajouté artificiellement à chaque envoi de paquets. Ceci va simuler une connexion lente et va donc mettre en avant l'utilité de faire un téléchargement par parrallélisation.

6 Contributions des auteurs

Partie	Contributeur principal	Relecture
Gestion des fichiers	Léo Floissac	Angel Lagrange
Compression/Décompression	Angel Lagrange	Léo Floissac
RMI et architecture des sockets	Collaboration équilibrée	Léo Floissac
Gestion des erreurs	Angel Lagrange	Léo Floissac
Interface	Léo Floissac	Angel Lagrange

7 Conclusion et perspectives

Les résultats montrent des performances satisfaisantes grâce à la compression et au téléchargement parallèle. Des améliorations futures pourraient inclure :

- Mise en place d'un système qui déciderait du besoin de compresser ou non.
- La gestion par le logiciel des machines hors réseau.
- La gestion des priorités lors du partage des fichiers.