TELECOM Nancy

RSA

Projet Réseaux - MyP2P

Membres projet :
Marie-Astrid Chanteloup
Milan Kania

Responsables de module :
Isabelle Chrisment
Remi Badonnel

Table des matières

1 Introduction

1.1 Fiche Projet

1.1.1 Contexte du projet

Ce projet a été réalisé dans le cadre de RSA de la deuxième année du cycle ingénieur sous statut étudiant de TELECOM Nancy pour les approfondissement IL, ISS et LE. Il est à rendre pour le 9 mai 2021.

1.1.2 Rédacteurs, commanditaires

Ce document a été rédigé par CHANTELOUP Marie-Astrid et KANIA Milan, les deux membres de l'équipe projet. Le projet quant à lui a été commandité par CHRISMENT Isabelle, responsable de la partie réseaux du module de RSA.

1.1.3 Présentation générale du projet

Le but du projet est de créer une plateforme peer-to-peer. Cela consiste en deux éléments, un server et un (des) client(s) permettant trois types d'échanges :
- la recherche de fichiers dans la base de données (client - serveur)
- la publication de fichiers dans la base de données (client - serveur)
- la récupération de fichiers depuis un autre client (client - client)

1.1.4 Livrables

Ce projet est évalué sur différents livrables :
- le code réalisé
- les fonctionnalités développées
- ce rapport de projet

1.2 Organisation du document

Dans la partie 2, nous présentons les fonctionnements généraux du serveur et du client.

Dans la partie 3, nous présentons la structure de notre base de données (BDD).

Dans la partie 4, nous expliquons nos choix de développement pour la recherche de fichier sur la base de données.

Dans la partie 5, nous présentons la conception de la publications d'un fichier sur le serveur.

Dans la partie 6, nous présentons la mise de la récupération d'un fichier depuis un autre client.

Dans la partie 7, nous effectuons un bilan sur le projet et notre approche de celui-ci.

1.3 Précisions légales

Ce projet n'est pas destiné à un usage commercial, ainsi, les images présentes, notamment les images de test, ne sont pas destinées à la publication et ne sont pas toutes libre de droit.

Cependant, le caractère strictement scolaire de ce projet nous autorise à les inclure en accord avec :
- Code civil : articles 7 à 15, article 9 : respect de la vie privée
- Code pénal : articles 226-1 à 226-7 : atteinte à la vie privée
- Code de procédure civil : articles 484 à 492-1 : procédure de référé
- Loi n78-17 du 6 janvier 1978 : Informatique et libertés, Article 38

2 Programme Principal

2.1 Côté serveur

Le serveur est lancé par le programme `central_server`. Lors de son lancement, il crée sa BDD (expliquée en partie 3). Il crée ensuite une socket DGRAM et attend d'être sollicité par un client.

Lorsqu'il est sollicité, il regarde le header du message qu'il a reçu. Si celui-ci est `"SEARCH"`, il lance une recherche dans la BDD (voir partie 4). Si le header est `"PUBLISH"`, il commence l'ajout d'une nouvelle entrée dans la BDD (voir partie 5).

2.2 Côté Client

Le client est lancé par le programme `peer`. Lors de son lancement, le processus se sépare en deux par un fork. Dans le fils, il est "actif" et dans le père, il est "passif".

Dans son fonctionnement "actif", il demande à l'utilisateur ce qu'il compte faire : lancer une recherche, publier ou s'arrêter là. En fonction du choix de l'utilisateur, il lance la fonction appropriée.

Dans son fonctionnement passif, il lance la fonction `TCP_server` pour attendre une demande de connexion vers son port 2000 dans le cadre de la récupération de fichiers (voir partie 6).

3 Format de la BDD

3.1 Entrée dans la BDD

Le serveur connaît une BDD sous le format d'un tableau d'entrée. Ces entrées sont représentées par une structure qui connaît les attributs suivants :

- `ip` : addresse IP à l'origine de l'entrée sous format d'une chaîne de caractères
- `name` : nom de l'entrée sous format d'une chaîne de caractères
- `type` : le type de fichier de l'entrée, sous format d'une chaîne de carctères
- `keyWords` : l'ensemble des mots-clés associés à l'entrée, sous format d'un ensemble de chaînes de caractères
- `keyWordNbr` : le nombre de mots-clés sous le format d'un entier
- `hash` : le code de hachage assurant l'intégrité du fichier, sous format d'une chaîne de caractères

3.2 Création de la BDD du serveur

Cette BDD est créée à partir du document `content.csv` à chaque lancée du serveur grâce aux fonctions `createDB`, `createDB_entries` et `fillDB_entries`.

La fonction `createDB` traite le fichier CSV : elle vérifie que le fichier ne contient pas plus d'entrée que la BDD ne peut en contenir, soit 100 entrées, et si ce n'est pas le cas, elle crée une BDD qu'elle remplit avec `createDB_entries` avant de la renvoyer.

La fonction `createDB_entries` réserve l'espace nécessaire en mémoire pour l'ensemble des entrées de la BDD et renvoie le tableau d'entrées rempli par `fillDB_entries`.

Enfin la fonction `fillDB_entries` traite le fichier CVS ligne par ligne : à chaque ligne, donc chaque entrée, il récupère les informations séparées par un point-virgule suivi d'une saut de ligne. On fait attention à d'abord séparer les mots-clés selon le caractère `'/'`, avant de les enregistrer.

3.3 Suppression de la BDD

En cas de besoin, l'espace mémoire utilisé par une BDD est libéré à l'aide de la fonction `freeDB`. Pour chacune des entrées de la BDD, la fonction libère les espaces retenus par les attributs de l'entrée, avant de libérer l'espace tenu par le tableau d'entrée et la BDD en général.

4 Recherche de fichiers

La recherche est lancée depuis le client par la fonction `UDP_search`.

4.1 Collecte du mot-clé auprès du client

Lors d'une recherche, il est demandé au client d'entrer le mot-clé de sa recherche dans l'entrée standard. Celui-ci est récupéré et préparé à l'envoi : on lui enlève le marqueur de saut de ligne `\n` que l'on remplace par le marqueur de fin de chaîne de caractère `\0`.

Une fois le mot-clé récupéré, il est envoyé par UDP avec le header de message `SEARCH` pour notifier au serveur que l'on veut effectuer une recherche.

4.2 Mise en place de la connexion UDP

4.2.1 Côté serveur

Le serveur récupère le mot clé sur lequel opérer une recherche. Il renvoi la phrase "Aucune entré de la base de donnée correspondant à ces mots-clé", s'il ne trouve aucune entrée, ou l'ensemble des fichiers pouvant correspondre, accompagné du chemin permettant de les retrouver sur la machine du pair, l'adresse IP et le hash permettant la vérification de l'intégrité du fichier transférer.

4.2.2 Côté client

Le serveur lance une socket DGRAM et se lie à l'adresse 127.0.0.1 (adresse du serveur). Une fois lié au serveur, le client envoie le message par UDP.

4.3 Recherche d'élément dans la BDD

Lorsque le serveur reçoit une demande de recherche, il lance la fonction `searchByKeyWords`. Dans un premier temps, cette fonction copie l'entièreté de la BDD, puis pour chaque entrée de la BDD vérifie si le mot recherché fait partie des mots-clés de l'entrée. Si ce n'est pas le cas, la fonction supprime l'entrée dans la copie de la BDD. Une fois toutes les entrées traitées, elle renvoie la BDD réduite.

Une chaîne de caractères contenant toutes les entrées est préparées à partir de la BDD réduite et est envoyé au client par la connexion UDP. Le serveur supprime ensuite la BDD réduite à l'aide de la fonction `freeDB` (expliquée en 3.3).

4.4 Réception des informations

Lorsque le serveur envoie l'ensemble des entrées trouvées au client par la connexion UDP, le client les présente numérotées à l'utilisateur sur la sortie standard. Si celui-ci veut télécharger une des entrées, il lui est proposé d'entrer le numéro de celle-ci dans l'entrée, ce qui lance la fonction `TCP_client` (expliquée dans la partie 6), sinon d'entrer 0 pour s'arrêter là.

5 Publication de fichier

La publication de fichier est lancée depuis le client par la fonction `UDP_publish`.

5.1 Récupération et mise en forme des données à envoyer

Lors de la publication d'un fichier, il est demandé à l'utilisateur de rentrer l'adresse relative du fichier, puis les mots-clés séparés les uns des autres par le caractère `"/"` dans l'entrée standard. L'adresse relative est transformée en adresse abosule par la fonction `realpath`.

À partir de ces informations, un message contenant le header `"PUBLISH"`, l'adresse du fichier, son extension (obtenue à partir de l'adresse du fichier), ses mots-clés et son hash d'intégrité est envoyé. Ce message sera envoyé une fois la connexion au serveur établie.

5.2 Mise en place de la connexion UDP

La mise en place de la connexion UDP a été faite de manière similaire à celle faite pour la recherche.

Une fois le client connecté au serveur, le message est transmis par UDP.

5.3 Récupération des données et création d'entrée dans la BDD

Lorsque le serveur reçoit une requête de publication, il lance la fonction `addEntry`. Cette fonction découpe le message arrivé en sous-informations pour créer, dans un premier temps, une nouvelle ligne dans le fichier CSV pour permettre de garder l'entrée au long terme, et, dans un second temps, l'entrée dans la BDD du serveur, avec des ré-attributions d'espace mémoire.

Lorsque l'entrée est créée, le serveur envoie un accusé de réception et de publication au client.

6 Récupération de fichier

La récupération de fichier est lancée par la fonction `TCP_client` depuis le demandeur de fichier.

6.1 Création de connexion

Le demandeur de fichier crée une socket SOCK_STREAM qu'il lie avec l'adresse qui est passée en entrée de la fonction. Normalement, le possesseur du fichier est en statut "passif" (voir partie 2.2) et est donc prêt à lancer une connexion aussi de son côté : si une place est disponible dans sa liste de connexions disponibles, il en réserve une pour ce nouveau client et traite sa demande, sinon, la demande de connexion est rejetée et devra être renouvelée par le demandeur.

Cette connexion est limitée par un timeout de 3 secondes qui est passé en option des deux sockets.

6.2 Demande de fichier

6.2.1 Côté possesseur

Une fois la connexion créée, le possesseur de fichier lance la fonction `TCP_server_recv` qui lui permet de recevoir la demande et de la traiter avec la fonction `recv_tcp`.

Cette fonction reçoit les messages du demandeur et les traitent : si le message est vide cela signifie que le demandeur souhaite terminer la connexion, on enregistre donc uniquement le caractère '`\0`' dans le buffer de retour, sinon on y enregistre directement le message.

Si un fichier est ouvrable avec le message reçu, le possesseur l'ouvre en mode lecture et commence l'envoi.

6.2.2 Côté demandeur

Lorsque la connexion est établie, le demandeur envoie le chemin et le nom du fichier qu'il recherche et attends le retour du possesseur.

6.3 Envoi de données

6.3.1 Côté possesseur

Le possesseur traite le fichier qu'il a ouvert : tant qu'il y a des données à envoyer, il envoie le plus grand nombre de données possible par la connexion TCP.

6.3.2 Côté demandeur

Le possesseur essaie de créer un fichier au nom de celui attendu dans le dossier `"received"`. Si celui-ci existe déjà, l'ancien fichier est écrasé.

Puis à l'aide de la fonction `recv_tcp` (expliquée en 6.2.1), tant qu'il reçoit des données, il les écrit dans le nouveau fichier. Une fois les données reçues, il vérifie que le hash d'intégrité calculé sur le nouveau fichier est bien celui attendu : si le hash est incorrect, l'utilisateur est prévenu et doit recommencer sa demande.

Une fois le hash vérifié, l'utilisateur est prévenu du bon téléchargement du fichier et rappelé de son emplacement.

6.4 Fin de la connexion

Lorsque l'envoi est terminé, l'envoyeur ferme la connexion et libère une place dans sa liste de connexion disponible. De même, une fois le téléchargement complété et validé, le demandeur ferme la connexion et la fonction `TCP_client` se termine.

7 Bilan

7.1 Bilans personnels

7.1.1 CHANTELOUP Marie-Astrid

Points positifs	- Duo diversifié en termes de compétences, donnant une bonne répartition du travail
Difficultés rencontrées	- Difficultés de compréhension sur certains aspects de TCP
Expérience personnelle	- Meilleure compréhension du fonctionnement d'un Peer to Peer
Axes d'amélioration	- Meilleure prise en main des concepts abordés en Réseau

7.1.2 KANIA Milan

Points positifs	- Approfondissement de la compréhension vis-à-vis du fonctionnement des échanges sur un réseau
Difficultés rencontrées	- Mise en place du select TCP
Expérience personnelle	- Très enrichissant
Axes d'amélioration	- Amélioration de la recherche sur plusieurs mots-clés

7.2 Bilan d'équipe

Points positifs	Points négatifs	Expérience
Équipe plurivalente et agréable	Contexte de projet difficile vis-à-vis du programme chronophage de 2è année	Meilleure compréhension du fonctionnement des connexions UPD et TCP et d'un système de peer-to-peer. Utilisation de sockets