TELECOM Nancy

Réseaux Système Avancés

MyP2P

Membres du groupe :
Niels TILCH
Marie-Hélène THOMAS

Responsable du module :
Isabelle CHRISMENT

Table des matières

1 Introduction

Le but de ce projet était de créer un réseau de communication pair à pair. Ce type de réseau est utilisé par exemple lors d'un transfert de fichier entre deux ordinateurs sans passage par un système central. Dans un tel système, chaque pair peut être à la fois client et serveur. Ici, l'un des pairs devait chercher un fichier dans un serveur central à partir de la requête de l'autre pair.

2 Choix d'implémentation

2.1 Client et Serveur

Afin de séparer les différentes fonctions que le peer to peer doit effectuer correctement, nous avons décidé de diviser les fonctionnalités en deux fichiers `client.c` et `serveur.c`. Nous avons aussi créé un fichier `main.c` afin de permettre facilement à l'utilisateur d'être un serveur ou un client.
Les principales fonctions pour le client sont les suivantes :

- `clientfctn` : Cette fonction est la fonction principale du côté client. Elle effectue deux processus : l'une s'occupe de la création d'une interface pour que l'utilisateur puisse effectuer les commandes publish et search (expliquées par la suite), l'autre s'occupe d'attendre une demande de récupération d'un fichier d'un autre client (comme expliqué dans la partie "Transfert des fichiers").
- `publish` : Cette fonction permet d'effectuer la commande publish.
- `search` : Cette fonction permet d'effectuer la commande search.
- `demandeTCPFichier` et `recevoirDemandeTCP` : Ce sont les deux fonctions responsables des transferts de fichiers, leur fonctionnement est expliqué dans la partie "Transfert des fichiers".

Les principales fonctions pour le serveur sont les suivantes :

- `serveurfctn` :
- `publish`$_r$*esponse* :

2.2 Search

Dans le cas de la fonction `search`, plusieurs étapes sont réalisées :

- Réception du ou des mot(s)-clé(s) à rechercher
- Ouverture du fichier servant de base de données
- Recherche dans la base de données des documents contenant le ou les mot(s)-clé(s) recherché(s)
- Pour chaque donnée correspondante trouvée :
  - Récupération du nom de la donnée, de son type et de l'adresse IP correspondante
  - Envoi du nom du fichier sous la forme "nom.type", puis de l'adresse IP
  - Envoi de ces résultats en tant que réponse au client
- Fermeture de la base de données
- Annonce de la fin des résultats de la recherche.

Cette fonction est appelée dans le client lorsque l'utilisateur veut faire une recherche par mots-clés. Le client vérifie que des mots-clés ont bien été saisis, puis il vérifie la validité de l'adresse IP donnée. Cette dernière doit permettre de pouvoir communiquer correctement avec un serveur. Ce n'est qu'après que la fonction search est appelée côté client.

Côté serveur, si l'acquittement reçu est bien de type search, la fonction search est appelée. Elle envoie ses résultats au client, puis le serveur renvoie un acquittement au client pour lui signifier que la recherche est terminée. Le client obtient ainsi une liste de résultats correspondant à sa recherche.

2.3 Publish

Dans le cas de la fonction `publish`, il y a plusieurs étapes principales :

- Demande des données préliminaires

— Envoi des données par le client et réception de celles-ci par le serveur
— Enregistrement des données par le serveur dans la base de données
— Acquittement.

La partie publish s'enclenche dès que l'utilisateur effectue la commande `publish [nom du fichier]` dans le terminal de commande du client. Plusieurs étapes se déroulent alors :

— En premier lieu, le système vérifie si le fichier existe bien dans la machine du client. S'il existe, le programme passe à l'étape suivante, sinon, le programme affiche une erreur de fichier inexistant et la fonction publish ne se fait pas.
— En second lieu, il demande les mots-clés concernant le fichier envoyé. Si l'utilisateur a confirmé les mots-clés sans en avoir mis, le programme lui redemande une liste de mots-clés valables (non vides). Sinon, le programme passe à l'étape suivante.
— Ensuite, le programme demande l'adresse du serveur auquel le client veut envoyer ses données. Le programme vérifie si l'adresse IP donnée contient 4 nombres entre 0 et 255. Si la vérification n'est pas réussie, alors le programme redemande l'adresse du serveur. Sinon, les données sont envoyées.

Dès que les données sont vérifiées par le programme, leur envoi peut être effectué grâce à la fonction `publish` côté client. Tout d'abord, le programme calcule le hashcode SHA-1 de la concaténation du fichier, de ses mots-clés et de l'adresse IP du serveur avec la fonction `getmyIP()`. Après la configuration de la socket UDP, le programme passe à l'envoi. Afin de signaler au serveur que la requête du client est un publish, le client envoie un "1". Ensuite, il envoie les différentes données au serveur.

Lorsque le serveur réceptionne les données, il vérifie tout d'abord la validité des données grâce au hashcode en le recalculant comme le client l'a fait précédemment :

— Si le calcul est différent de celui reçu, alors le serveur envoie un acquittement demandant au client de lui envoyer les données de nouveau.
— Au contraire, si le calcul est correct, le serveur enregistre les données dans un fichier texte et envoie un acquittement signalant que la donnée a bien été transmise.

2.4 Transfert des fichiers

Cette partie est précédée de la partie search permettant à l'utilisateur de choisir le fichier qu'il veut. Après le choix du fichier, le programme utilise la fonction `demandeTCPFichier` puis une demande pour une connexion TCP est effectuée par le client (client A) (sur un port différent des demandes `publish` et `search`). Dès que le client détenteur du fichier reçoit le signal (client B), il envoie un acquittement confirmant la réception du signal et utilise la fonction `recevoirDemandeTCP`. Tout d'abord, il reçoit le fichier et l'adresse IP de l'émetteur. Le client B vérifie d'abord que le fichier existe dans sa base de données. S'il n'existe pas, le client B envoie un signal et le client A arrête sa demande de transfert de fichier.
Dans le cas contraire, le transfert peut se faire. Après l'initialisation TCP des clients A et B, le client B (le client propriétaire du fichier) donne un acquittement pour signaler le début du transfert. Le client A crée le fichier et reçoit au fur et à mesure les données jusqu'à recevoir un acquittement "-1" signalant la fin du transfert. Ainsi, pour terminer la connexion, le client B donne au client A un acquittement de bonne réception du fichier.

3 Difficultés rencontrées

La principale difficulté durant la conception du programme était de bien connaître l'ensemble des problèmes qui pouvaient être rencontrés lors de chaque étape. Sécuriser l'ensemble du protocole a été un défi que nous avons partiellement réussi. Entre les défauts de connexion pendant les transmissions ou bien la perte de connexion entre le serveur et le client pendant l'attente d'un acquittement, la gestion a été difficile pour sécuriser l'ensemble de la structure. Avec un peu plus de temps, nous pensons que la sécurisation quasi-totale du programme était faisable, bien que celle des transferts de données entre deux clients ou bien entre le client et le serveur n'a pas été faite (possibilité

de transfert de malware par une machine tiers,...)
De même, la visualisation de la structure de chaque problème a pris du temps du fait de deux visions différentes de ce que le programme pouvait faire.

4 Conclusion

Le réseau pair à pair est un moyen de communication souvent utilisé par les machines. Son implémentation nécessite l'utilisation de plusieurs protocoles, notamment TCP et UDP. Nous avons vu au cours de ce projet que différentes solutions étaient possibles pour le mettre en place et que le but était de choisir les plus pertinentes. Le travail rendu fonctionne correctement, néanmoins des améliorations sont toujours possibles.

5 Bilan horaire

Étapes	Marie-Hélène Thomas	Niels Tilch
Conception	4	6
Implémentation	14	22
Tests	6	4
Rédaction de rapport	4	2
Total	28	34