Adrien Gennevois - Augustin Lacour - Bertine Lemée

Projet pair à pair

Le projet consistait à mettre en place un réseau en utilisant le protocole pair à pair Chord. Nous allons présenter ici nos choix d'implémentation.

I. Langage

Nous avons choisi de coder en Python car c'est un langage que nous avions déjà utilisé à plusieurs reprises et qui permet une utilisation relativement facile et efficace des sockets et des threads. Le code résultant est assez compact : ~300 lignes.

II. Ajout de pairs dans le réseau

Pour rejoindre le réseau, il faut exécuter *join_network.py*.
Le script récupère l'adresse IP du client et demande au serveur de Hash de lui attribuer un hash (entre 0 et 100 dans le cas du TP). On crée ensuite le pair (classe Peer de *Peer.py*) avec l'IP et le hash associé. Au début le pair ne possède aucune route dans sa table de routage. On contacte ensuite le serveur d'accueil qui va l'ajouter au réseau.
Si le pair est le premier arrivé :

- Le serveur d'accueil envoie "yaf" au pair et ce dernier met à jour sa table de routage. Il est son propre successeur.

S'il y a déjà des pairs dans le réseau :

- Le serveur d'acceuil envoie au nouveau pair l'adresse IP d'un pair du réseau qui servira de point d'entrée. Le point d'entrée regarde si le pair se trouve entre son hash et le hash de son successeur.
    - Si c'est le cas alors il le signale au pair entrant, celui ci met à jour son successeur et notifie son prédecesseur qu'il est son nouveau successeur.
    - Si ce n'est pas le cas il demande à son successeur de faire le même travail et ainsi de suite jusqu'à ce qu'un pair du réseau ait pour successeur le nouveau pair.

Une fois le réseau rejoint, on donne la main au pair qui pourra intéragir avec les autres membres du réseau (classe User_interaction de User_intereaction.py)
La table de routage d'un pair possède seulement la route vers son successeur.

III. Transmission de messages

En ce qui concerne la transmission de messages nous avons utilisé le système cyclique que nous avons mis en place. On peut découper notre processus en plusieurs étapes qui vont être effectuer par chaque pair.
Le pair regarde si le message lui est destiné :

- Si oui, il l'affiche.
- Si non, il vérifie que le hash du destinataire n'est pas situé entre son hash et celui de son successeur :
    - Si oui, le destinataire n'existe pas : on renvoie alors une notification à l'expéditeur lui indiquant que le pair destinataire de son message n'existe pas.
    - Si non, il transmet le message au successeur avec l'expéditeur et le destinataire qui répètera ce test.

IV. Extension : Gestion des données

Les données ont, comme les pairs, un identifiant (hash).
Pour ajouter une donnée, depuis n'importe quel pair :

- On entre la donnée.
- On récupère son hash auprès du serveur de hash.
- On regarde si la donnée doit être prise en charge par notre successeur :
    - Si oui, on lui ordonne de la prendre en charge.
    - Si non, on lui transmet la demande et il répète ce test.

Note : les données ne sont pas connue du serveur d'accueil.

Pour récupérer une donnée, depuis n'importe quel pair :

- On demande son identifiant (hash)
- On regarde si la donnée est censée être prise en charge par notre successeur :
    - Si oui, on lui ordonne de transmettre la donnée au demandeur, ce qu'il fait s'il la possède sinon il en informe le demandeur.
    - Si non on lui transmet la demande et il répète ce test.