Rapport - **Programmation Réseaux**
Projet ChatFusion

MARIÉ Brayan
RUFO Corentin

Master 1 - Informatique
Université **Gustave EIFFEL**

2021 - 2022

Contents

1 Introduction

1.1 Objectif

Le but de projet ChatFusion est de réaliser un service de discussions qui permet de mettre en place une fusion entre plusieurs serveurs qui communiquent par liaison TCP.
L'ensemble du projet est a réaliser en TCP.
Dans un premier temps, les clients se connectent à un serveur. Chaque client connecté est identifié par son pseudonyme. Le serveur doit alors garantir que deux clients sur un même serveur, ne peuvent avoir le même nom.
Une fois connectés et identifiés sur le serveurs par différents échangent de paquets, les clients pourront :

- envoyer des messages publiques visibles par tous les utilisateurs du serveur.
- envoyer des messages privés ou des fichiers qui seront visibles uniquement pas l'utilisateur destinataire.

De plus, l'ensemble de la structure du projet sera en mode non-bloquant.

2 Mise en place des clients

Dans ce projet, l'implantation du client, est mineur par rapport aux serveurs car la fusion représente la plus grosse partie du projet, cependant le client mérite quand même une explication dans le projet.

Dans un premier temps, le client va se connecter à un serveur. Il va donc envoyer un paquet à son serveur pour savoir si la connexion est possible. Dans tous les cas, le serveur va lui envoyer une réponse pour savoir si la connexion est possible avec lui. Une fois connecté, le client va pouvoir recevoir des messages des autres utilisateurs du serveurs ou du méta-serveurs(nous reviendront dessus plus tard au moment de la fusion). Il aura alors la possibilité d'écrire dans son terminal des messages qui seront envoyés au serveur pour qu'il puisse procéder à la redistribution des messages aux clients. Ces messages seront publiques et donc soumis à tout le monde.

Il aura aussi la possibilité d'envoyer des messages privés au un autre utilisateur du serveur, pour ce faire il doit fournir la commande :

```
@[login]:[server] [message]
```

Il pourra aussi envoyer un fichier avec la commande :

```
/[login]:[server] [file]
```

Lorsque le client va recevoir un message publique, il aura sur son terminal l'affichage suivant :

```
[nameServer]:[pseudo]
    ↪ [message]
```

Dans le cas d'un message privé :

```
Private message from:[nameServer]:[pseudo]
    ↪ [message]
```

3 Mise en place des serveurs

Pour le serveur, nous allons dans un premier temps présenter comment un serveur communique avec ses clients et dans un second temps, avec ses sous-serveurs.

3.1 Relation Serveur-Client

Pour chaque tentative de connexions d'un client sur le serveur, le serveur alors regarder dans les clients qu'il connaît déjà, si un client possède le même nom, si c'est le cas, le serveur envoie au client un réponse négative pour sa connexion sinon un paquet de validation et la connections est maintenue.
Un fois qu'un client est connecté, le client va alors envoyer des message publique ou privé sur le serveur, dans ce cas le serveur aura deux comportements différents, si c'est un message publique, alors il va envoyer le même paquet qui contient le message et l'envoyeur a tous les clients connectés sur le serveurs.
Si c'est un message privée, alors il va simplement chercher dans tous les clients qui sont connectés si un corresponds au client destinataire du message. Si il trouve, alors il lui envoie si il ne le trouve pas, il notifie l'émetteur que le client n'a pas été trouvé sur le serveur.

3.2 Relation Serveur-Serveur

Dans un méta-serveur, plusieurs serveurs sont connectés entres eux, cependant dans un méta-serveur, il ne peut y avoir qu'un seul et unique leader. C'est par ce leader que toutes les interactions vont passer. En effet, si un serveur qui n'est pas le leader reçoit un message d'un client alors son rôle va être de transmettre son message à tous ses clients puis transmettre le message au leader du méta-serveur pour qu'il puisse le transmettre à tous les serveurs du méta-serveur. C'est le même principe pour les messages privés.

Pour effectuer une fusion vers un serveur il faut dans le terminal du serveur taper la commande :

`[FUSION] [ipaddress] [port]`

De plus nous avons ajouté la possibilité de vois qui est le leader du meta-servers pour se faire il faut taper la commande :

`[LEADER]`

4 Première version - Soutenance mi-projet

4.1 Avancement du projet

A ce moment du projet, nous avions implémenté quelques fonctionnalités tels que la connexion entre un client et un serveur. Pour ce faire lors d'une connexion, le client va envoyer un paquet avec ses information au serveur. Lorsque le serveur reçoit ce paquet, il va alors faire un traitement sur ce paquet pour savoir si la connexion est possible entre lui et le client. En fonction de cette réponse le client sera autorisé à se connecter ou pas.

De plus, nous avions aussi mis en place la possibilité pour le client d'envoyer des messages sur son propre serveurs(la fusion n'étant pas encore mise en place). Le client va prendre le message en faire un paquet avec l'opCode correspondant pour l'envoyer sur le serveur. A la réception de ce paquet, le serveur va simplement le retransmettre à tous les autres clients présents sur le serveur.

Même principe pour le message privé, à l'exception près que le serveur va transmettre le message qu'à un seul client.

Nous avions aussi essayé de mettre en place la connexion en deux serveurs. Nous avions à ce stade du projet simplement mis en place la connexion entre deux serveurs et la mise en place du paquet de demande de connexion. Cependant celui-ci n'étais pas complet et le serveurs de réception de la demande ne traitait pas encore ce paquet. Donc la fusion n'était pas initiée dans le projet.

5 Version finale

Pour la version finale, nous avons pris en compte les problématiques soulevées lors de la soutenance. En effet, notre notre lecteur des opCodes n'étais pas vraiment adapté, nous avons changé cela en conséquence. De plus les noms des fichiers n'étais pas très cohérent avec leurs fonction nous avons aussi changé cela.

Nous avons aussi mis en place la Fusion inter-serveur. Maintenant plusieurs serveurs et même plusieurs méta-serveurs peuvent se connecter entres eux. Pour se faire, nous avons implémenté le même protocole que celui fourni dans la RFC de la fusion donné par les professeurs. Donc on gère bien les différents cas où un serveurs qui fait la demande et/ou qui reçoit la demande est le leader ou non de son méta-serveur. De ce fait la propagation des changement de leader ainsi que la propagation des adresse des sous serveurs a aussi été gérer par notre programme.

6 Choix d'implantation et difficultés

Nous avons fait le choix de stocker les différents contexte dans des structures de données pour simplifier leurs accès. En effet les clients et les serveurs doivent être différencier et il sera alors plus simple de savoir ou envoyer les données avec ce système. De plus nous avons mis en place un système de contexte qui fait que les serveurs possède deux comportement, on peut dire un comportement de type serveurs et un comportement de type client lors de la fusion car les serveurs doivent pouvoir communiquer entres eux.

La principale difficulté de ce projet a pour été la mise en place de la fusion ainsi que la propagation des changements de leader lorsqu'un nouveau serveur est ajouté a un méta-serveurs. De plus la gestion des contextes des serveurs(ContextServer et ContextServeurFusion) était aussi un partie délicate du projet.

Nous avons aussi mis en place le transfère de fichier cependant lorsque le fichier est très volumineux, le transfère ne se fait pas.

7 Conclusion

Ce projet était pour nous très intéressant à faire. Nous avons trouver qu'il était assez difficile mais tout de même a notre porter. De plus réaliser un projet comme celui-ci permet était d'autant plus intéressant car les outils de chat sont des outils que nous utilisons au quotidien et donc coder ce genre d'outil est d'autant plus amusant. Enfin cela a permis aussi d'utiliser nos connaissance en programmation réseaux dans un projet plus conséquent tout en améliorant nos capacité en programmation Java.