Projet d'applications réseaux

Romain Soumard
Yann Miguel

11 mai 2020

Table des matières

1 Introduction

Le présent document est un rapport écrit à l'issue du projet de fin d'année d'applications réseaux. Il a été rédigé par M.Romain Soumard et M.Yann Miguel à l'université d'Aix-Marseille durant l'année scolaire 2019-2020.
Vous pourrez y trouver une brève description du projet, de son architecture, de ses features, des bugs restants ainsi que des améliorations possibles.

2 Usage

Le programme est composé de divers modules. Il contient notamment un client, un modèle de serveur esclave / autonome, et un serveur maître central.
Afin de vous faciliter la tâche dans votre correction, nous avons créé deux lanceurs :

- ServerLauncher vous servira à lancer les serveurs chatAmuCentral **uniquement**.
- Le FederationLauncher vous servira à lancer l'intégralité des serveurs esclaves et le serveur maître en fonction du contenu du fichier pairs.cfg présent dans le dossier Config.

En plus de cela, un script shell vous ait fourni afin de pouvoir lancer plus facilement les divers modules sans avoir à vous préoccuper du chemin relatif de l'exécutable java. Notez que l'utilisation du script shell conduira au mélange des sorties du serveurs et des clients. La meilleure manière de lancer le projet reste de l'importer dans IntelliJ Idea et de configurer un compound pour lancer le client et les serveurs désirés en même temps.

3 L'architecture du code

Il y a 4 grand types de paquets dans le projet :

- Le paquet Client, qui contient un client classique pouvant se connecter aux serveurs.
- Le paquet Config, donnant la configuration de base(adresse ip et port) du serveur fédéré maître.
- Le paquet Serveur, qui contient le code des serveurs.
- Le paquet Tools, qui contient tout les outils utilisé par les serveurs et les clients pour communiquer.

Le fichier ServerLauncher.java lance seulement les serveurs chatAmuCentral présent dans pairs.cfg, alors que le fichier FederationLauncher.java lance le serveur maître et les serveurs esclaves conformément au contenu de config.cfg.

3.1 Serveur

Ce paquet contient tout les serveurs, que ce soit les serveurs de chat(ChatAmu), ou le serveur fédéré(Federation).
Tout les fichiers de ce paquet étendent la classe abstraite TemplateServer, qui correspond à la base d'un serveur, sauf la gestion des clés de connections, qui sont précisé par les sous classes, ChatAmuCentral.java et MasterServer.java.

3.2 Tools

Ce paquet contient tout les outils crées pour assurer la communication entre les serveurs et les clients.
Communication contient la classe IOCommunicator.java, qui est chargée de la lecture et de l'écriture dans les buffers des serveurs.
ConfigParser permet de récupérer les données du fichiers de config, afin de pouvoir connecter les serveurs au serveur maître.
Extended contient diverses classes étendue par délégation (conformément aux bonne pratiques objet.) et permet la gestion des erreurs du projet. L'idée était entre autre d'augmenter la lisibilité du code des serveurs en délocalisant la gestion des erreurs dans des classes et méthodes spécialisées. Ce paquet contient également un enum contenant l'ensemble des codes erreurs du projet.
Protocol contient la classe ProtocolHandler.java, qui se charger de gérer ce qui a trait au protocole. Sa responsabilité est entre autre de reconnaître les headers, et de les manipuler.
UserManagment représente la base de clients connectés au serveur. Il contient les classes Register qui sert simplement de registre où sont enregistrés les clients au fur et à mesure de leurs connexions, et la classe ClientQueueManager qui contient les files de priorités des clients enregistrés

4 Difficultés rencontrés

Lors de l'élaboration de l'architecture du code, de nombreux problèmes ce sont posés, notamment de refactoring, dû aux nombreux éléments similaires entre les itérations successives des serveurs. La gestion des erreurs en particulier fût relativement sportive dans la mesure où le nombre d'exceptions soulevés par les diverses classes du projet est assez nombreux.

5 Vue d'ensemble du projet

5.1 Features

- Configuration de l'intégralité des serveurs avec pairs.cfg.
- Connexion de plusieurs clients.
- Utiliser du MultiThreading côté client afin de gérer les messages asychrones,
- Faire le relai des messages en fonction de leur type par les serveurs esclaves.
- Utilisation du design pattern Template.

NOTE : Les connexions directes au serveur central ne sont pas autorisées. Cela aurait été possible à l'aide du multithreading en créant un thread chatAmuCentral responsable des connexions des clients, mais dans la mesure où cela revient à la même chose que de connecter un autre serveur esclave au serveur maître, l'idée n'a pas été retenue.

5.2 Bugs

- Aucune demande d'authenfication. Plusieurs personnes peuvent donc s'identifier sous le même login.
- Les utilisateurs déconnectés ne sont pas retirés de la liste des clients, ce qui peut entrainer un dédoublement des messages.

5.3 Améliorations possibles

- utiliser le pattern Chains of responsibility, ce qui permettrait d'améliorer la souplesse de l'implémentation du protocole, et la réduction de la taille du code de chatAmuCentral,
- une gestion de mots de passe chiffrés, afin d'améliorer la sécurité,