Compte-rendu TP Réseaux

SOTIR Paul-Emmanuel

CHAPELLE Victoire

B3330

Introduction

**Objectifs du TP**: Notre mission, pour ce TP qui s’est déroulé sur 3 séances de 4h, était de réaliser un chat interactif où des participants peuvent rejoindre, envoyer des messages et quitter celui-ci. Deux variantes du chat étaient demandées : une réalisée avec l’interface de programmation RMI et l’autre basée sur les Sockets.

Suivant l’avancement de notre application nous pouvions aller plus ou moins loin dans notre programmation et rendre notre chat plus élaboré.

Une des améliorations possibles était la création d’une interface graphique du chat, qui nous permettra de rendre le chat plus réaliste et la démonstration plus intéressante.

Conception du système de chat

Nous avons conçu notre chat en choisissant de rajouter une contrainte au cahier des charges : la possibilité de créer des groupes de chat. En effet, nous voulions réaliser un système d’échanges de messages plus réaliste et élaboré afin de répondre au mieux aux attentes que pourraient avoir des utilisateurs d’un chat. Cette amélioration n’a pas été faîte au détriment du cahier des charges initial évidemment.

Les deux variantes de chat réalisées (RMI et Sockets) ont la même interface graphique, si bien qu’aucune différence n’est discernable hors de l’implémentation. Elles ont toutes deux des classes en commun et d’autres séparées.

Pour les deux réalisations, nous avons créé un historique des messages. Cet historique est persistant c’est-à-dire qu’il est sauvegardé même si le serveur est coupé.

Les deux variantes sont asynchrones et permettent

Fonctionnement du chat

A chaque ouverture d’une fenêtre un client peut se connecter en utilisant son nom d’utilisateur et son mot de passe ou, si c’est sa première connexion, en créant son nom d’utilisateur. Le mot de passe est généré par l’application et le client doit le sauvegarder pour pouvoir se connecter. Chaque nouvel utilisateur est enregistré sur le serveur qui conserve la liste de tous les clients.

Le client doit ensuite rejoindre ou créer un groupe de chat pour pouvoir communiquer avec les autres. Une fois dans un groupe, il peut envoyer des messages et quitter le groupe quand il le souhaite.

A tout moment le client peut voir quels utilisateurs sont connectés dans le groupe et tous les groupes créés qu’il peut rejoindre.

Gestion des erreurs

Notre programme gère les erreurs qui pourraient se produire dans l’utilisation du chat.

Dans un premier temps les erreurs dues à certaines actions de l’utilisateur : création de groupe sans nom, choix d’un nom d’utilisateur ou de groupe déjà utilisé ou connexion à un compte avec un mauvais mot de passe sont impossibles.

De plus, si le serveur se coupe de manière inopinée, l’historique des messages, groupes et utilisateurs est conservé.

Améliorations possibles

Notre programme répond en parti aux attentes du cahier des charges mais peut cependant être perfectionné.

Nous n’avons pas fini la partie RMI par manque de temps. Toute l’implémentation est faîte mais il manque cependant l’ajout des méthodes dans le DAL. Cela ressemble beaucoup à ce que nous avons fait dans la partie Sockets.

Nous pourrions aussi améliorer l’interface graphique du chat en la rendant visuellement plus attractive et afficher la gestion des erreurs aux clients quand ils utilisent un nom d’utilisateur, de groupe qui existe déjà ou quand ils créent un groupe sans nom. Nous pourrions également afficher le créateur du groupe. Un autre élément du chat qui pourrait être amélioré est la connexion d’un client une fois qu’il a créé son nom d’utilisateur. En effet, à la place de demander le mot de passe de l’utilisateur, nous pourrions le reconnaître tant qu’il ne s’est pas déconnecté de la machine.