Cyril Cadoret

Système de Messagerie Instantanée(SMI)

Cette application est un système de messagerie instantanée développé en J2ME dont le modèle est basé sur une architecture simplifiée de type client/serveur. Le serveur peut accepter plusieurs connexions simultanées. Son rôle est de propager les messages émis par un client X à tous les clients connectés au serveur via le protocole TCP. La communication se fait via le modèle « socket ».

Chaque message échangé entre le client et le serveur possède la structure suivante :

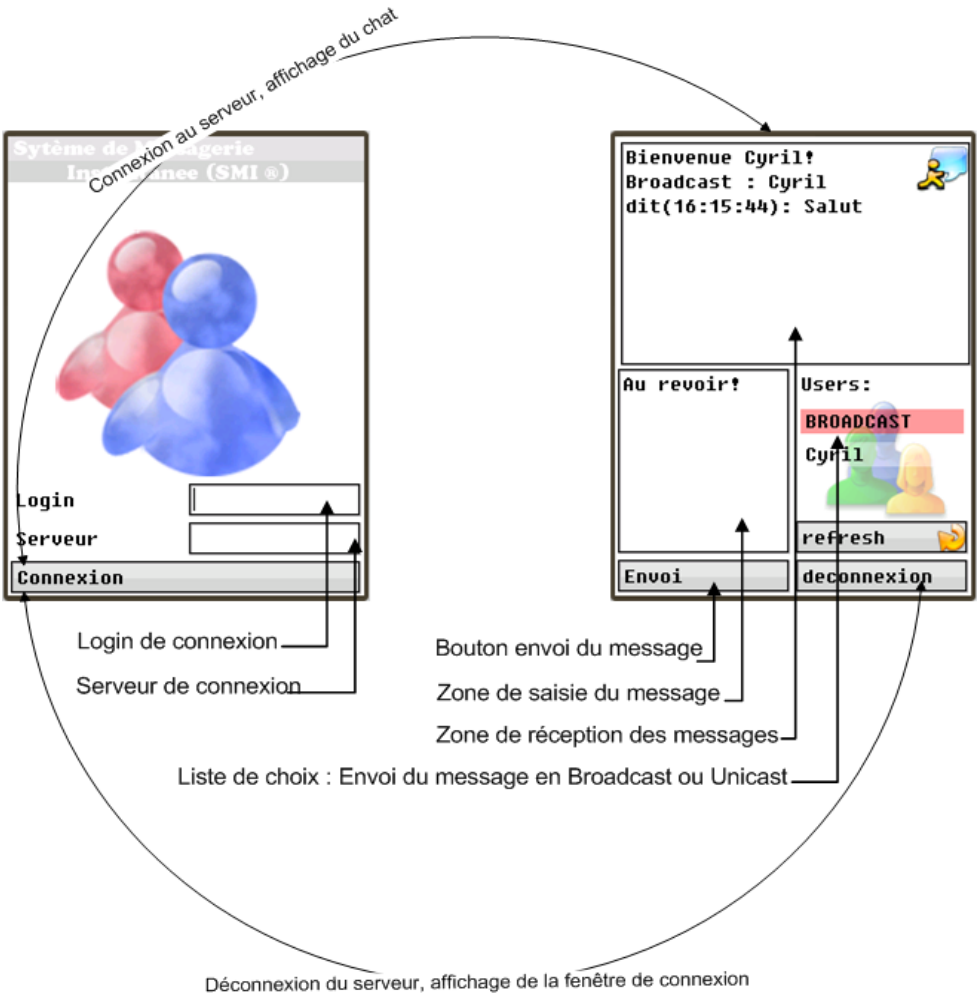
<command> [argument optionnel] <message>

<command> ::= <LOG> | <BYE> | <SAY> | <WHISPER> | <USERS>

<argument> ::= <NICKNAME> | <FROM>

<message> ::= string

Fonctionalités:

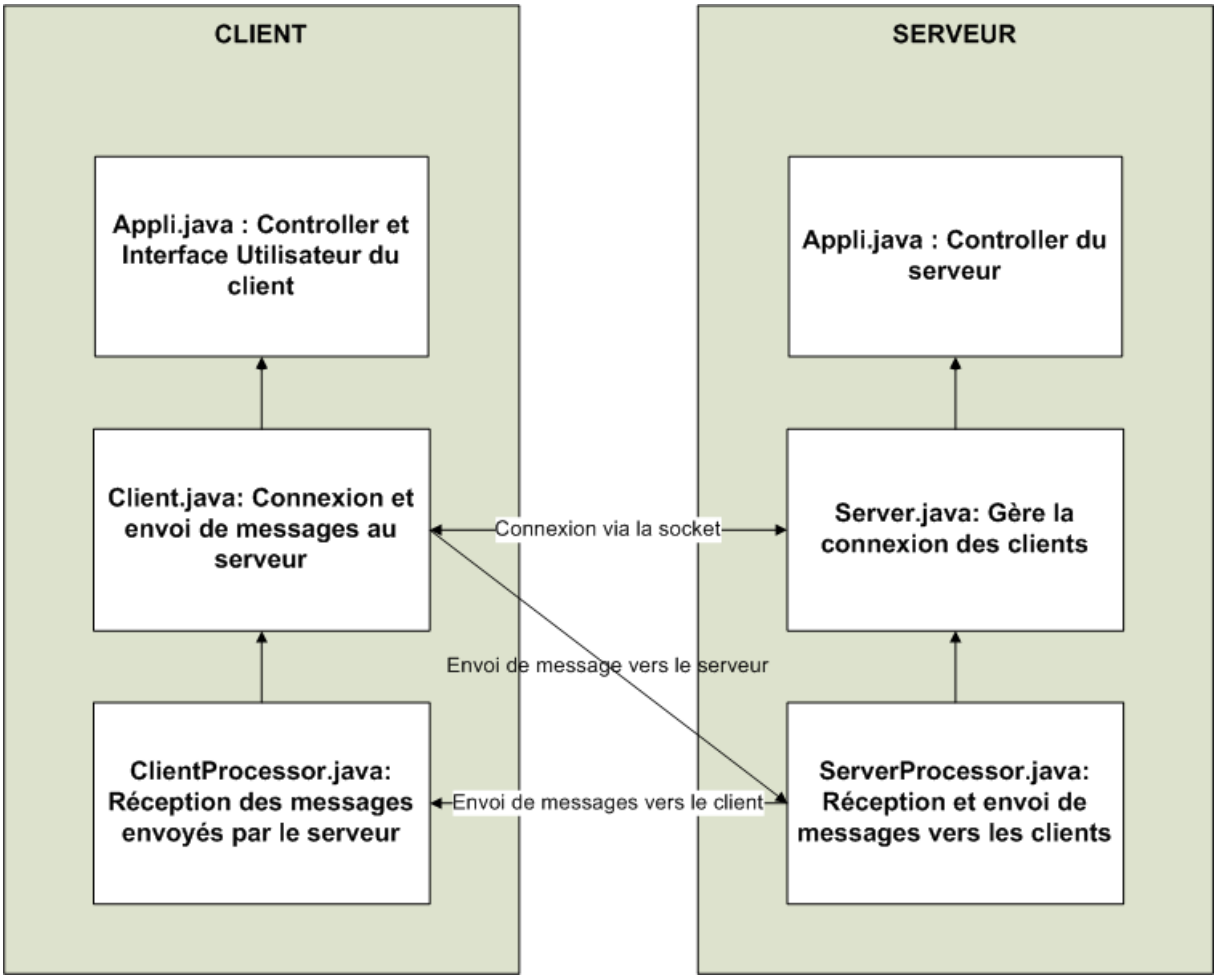


Dans cette application c'est le serveur qui averti le client de rafraichir sa liste de connectés. Afin de permettre au client de fonctionner avec d'autres serveurs, un bouton de rafraichissement « refresh »à été ajouté.

Afin d'avoir une interface graphique « user friendly », l'application utilise la librairie LWUIT (LightWeight UI Toolkit) plutôt que la librairie standard de J2ME.

Cyril Cadoret

Architecture simplifiée de l'application:



Le ClientProcessor inclut le protocole du client, il joue aussi un rôle de « controller » dans le sens ou il effectue des actions sur le client (sur l'interface graphique) en fonction des messages qu'il reçoit. Il est responsable du rafraichissement de l'interface graphique.

Le ServerProcessor inclut le protocole du serveur, il effectue des actions sur le serveur en fonction des messages qu'il reçoit.

Traitement des cas ésotériques :

Côté client :

- Dans le cas ou le serveur venait brutalement à s'éteindre, le client « catch » une exception. Cette exception ferme la socket et bloque tout envoi de messages.
- Le client ne peut pas envoyer de messages aux clients déconnectés puisqu'il ne figure pas sur la liste des connectés.

Côté serveur : Dans le cas ou un client venait à se déconnecter brutalement, le serveur « catch> une exception. Cette exception ferme alors la socket de connexion du client et averti les autres clients que ce client est déconnecté. Le protocole est le même lorsque le client se déconnecte via le bouton de déconnexion.

Nous avons essayé de suivre scrupuleusement le sujet, afin que notre application soit compatible avec d'autres clients et serveurs. L'implémentation du serveur à été assez périlleux pour qu'il puisse gérer simultanément plusieurs clients.