

Rapport du projet de Chat distribué avec Java RMI

| | |
|--|-----------|
| 1. LE SUJET | 2 |
| 2. VERSION 1 | 4 |
| A. PRESENTATION ET PRINCIPE | 4 |
| B. CARACTERISTIQUES CLIENT / SERVEUR | 6 |
| 3. VERSION 2 | 8 |
| A. PRESENTATION ET PRINCIPE | 8 |
| B. CARACTERISTIQUES CLIENT / SERVEUR | 10 |
| 4. VERSION 3 | 12 |
| A. PRESENTATION ET PRINCIPE | 12 |
| B. CARACTERISTIQUES CLIENT / SERVEUR | 14 |
| 6. SCREENSHOTS | 17 |
| 7. CODE SOURCE | 19 |
| 8. REMARQUES | 20 |
| 9. PROBLEMES | 21 |
| 10. AMELIORATIONS | 22 |

1. Le sujet

On se propose de réaliser un "chat" avec Java RMI. Au cours de ce projet nous réaliserons plusieurs versions de ce "chat". La dernière version aura pour but de limiter la charge (c'est à dire le travail à faire) des différentes machines.

Première version : version client/serveur.

La première version comporte un serveur qui reçoit chaque message et le fait suivre aux autres clients.

On obtient donc une architecture en étoile avec le serveur au centre de l'étoile.

On peut utiliser les interfaces suivantes :

```
public interface ClientDistant extends java.rmi.Remote {
    public void msg(Message m) throws RemoteException;
}

public interface ServeurChat extends ClientDistant {
    public void connect(String url) throws RemoteException;
    public void disconnect(String url) throws RemoteException;
}
```

Notes :

- La classe Message (à écrire) devra implémenter Serializable et transporter au minimum la chaîne de caractère du message.
- Le client Chat est ici à la fois client et serveur au sens RMI.
- Les méthodes connect et disconnect doivent être appelées par un client qui fournira en paramètre l'URL de son objet serveur implémentant l'interface ClientDistant.

Deuxième version : version "peer to peer" complètement connectée.

Dans cette seconde version, le serveur sert juste d'annuaire et permet à tout nouveau client de se connecter directement aux autres clients.

Par la suite, chaque client envoie les messages directement aux autres clients.

On obtient donc une architecture totalement connectée.

On peut utiliser les interfaces suivantes :

```
public interface ClientDistant extends java.rmi.Remote {
    public void msg(Message m) throws RemoteException;
    public void connect(String url) throws RemoteException;
    public void disconnect(String url) throws RemoteException;
}

public interface ServeurChat extends java.rmi.Remote {
    public PeerList register(String url) throws RemoteException;
    public void unregister(String url) throws RemoteException;
}
```

Notes :

- La classe Message (à écrire) devra implémenter Serializable et transporter au minimum la chaîne de caractères du message.

- La classe PeerList (à écrire) devra elle aussi implémenter Serializable et transporter la liste des URL des clients déjà connectés.
- La méthode connect de ClientDistant doit permettre à tout nouveau client d'indiquer son URL aux clients déjà connectés.

Dernière version : version "peer to peer" optimale.

Dans cette dernière version, on veut optimiser le maillage de connexions. Chaque client sera connecté à un nombre restreint d'autres clients (appelé les Peers ou Pairs en Français) et fera suivre en plus de ses propres messages, les messages envoyés par n'importe quel autre Peer à tous ses autres "Peers".

Ainsi tout message envoyé par un client sera reçu par tout autre client après avoir traversé 0 ou plusieurs clients.

On peut utiliser pour cette version les interfaces de la version précédente.

Notes :

- Il faut éviter dans cette version qu'un message ne boucle indéfiniment sur un ensemble de clients.
- Le rôle du serveur, dans cette version est de mettre en place un maillage optimal de façon incrémentale. C'est à dire que lors de chaque ajout de client, il devra s'assurer que tous les clients sont connectés et qu'aucun client n'est surchargé (c'est à dire connecté à un ensemble très important de clients). On pourra par exemple connecter le nouveau client au client qui est le moins connecté.

Extensions de l'application

En plus de ce qui est demandé vous pouvez :

- Rajouter une interface graphique en AWT ou en Swing (bibliothèques graphiques Java) en utilisant le constructeur d'interface de JBuilder ou de Forte for Java (suivant ce qui est disponible dans la salle de TP).
- Rajouter la possibilité à tout client de proposer un fichier qui sera téléchargé sur n'importe quel autre client via une connexion Socket. On pourra par exemple utiliser cette possibilité pour échanger des fichiers musicaux (comme avec le défunt Napster).
- Permettre la définition de canaux "à la IRC". Les canaux permettent de limiter les échanges aux participants intéressés par un sujet donné. Dans ce cas là, les canaux peuvent être gérés soit par le serveur avec un maillage par canal soit par les clients directement (dans ce cas un client pourra être amené à faire suivre un message qui ne l'intéresse pas).

2. Version 1

A. Présentation et principe

Dans cette version, c'est le serveur de chat qui enregistre et désenregistre des clients distants. C'est aussi à lui que revient le rôle de faire suivre les messages envoyés par les clients distants à tous les autres clients ou à un unique client.

Enregistrement d'un client :

Pour enregistrer un client, on récupère une référence (stub) vers l'objet distant associé à l'URL fournit en paramètre pour obtenir des informations sur ce client. Ensuite on ajoute dans une carte de hashage l'URL du client avec le nom de l'utilisateur.

Le serveur de chat envoie ensuite un message à tous les clients pour les prévenir qu'un nouvel utilisateur est entré dans le chat. Le serveur de chat affiche aussi le nom de l'utilisateur si celui-ci est un nom anonyme généré et un message de bienvenue.

Enfin message de log est affiché sur la sortie standard.

Désenregistrement d'un client :

Pour désenregistrer un client, on commence par récupérer le nom de l'utilisateur via la carte de hashage et la clef : l'URL fournit en paramètre. Cette clef (URL) et sa valeur (nom de l'utilisateur) sont ensuite supprimés de la carte de hashage.

Le serveur de chat envoie ensuite un message à tous les clients pour prévenir qu'un utilisateur est sorti du chat.

Enfin message de log est affiché sur la sortie standard.

Envoi d'un message :

Pour envoyer un message à tous les clients, le serveur de chat parcourt la liste de toutes les URL. Pour chaque URL, on récupère une référence (stub) vers l'objet distant associé à l'URL et on affiche le message chez le client distant. Si le client n'existe plus (il a quitté brutalement) alors on le désenregistre.

Envoi d'un message privé :

Pour envoyer un message à un unique client, le serveur de chat parcourt la liste de toutes les URL. Pour chaque URL, on vérifie via la carte de hashage si le nom de l'utilisateur correspondant à l'URL est celui qui doit recevoir le message. Si l'utilisateur n'a pas été trouvé alors on envoie un message à l'émetteur. Si l'utilisateur a été trouvé alors on récupère une référence (stub) vers l'objet distant associé à l'URL et on affiche le message chez le client distant.

Les commandes :

/help : affiche la liste des différentes commandes disponibles et leur description.

/exit : permet de stopper le serveur de chat.

/users : permet d'afficher le nombre d'utilisateurs connectés.

/lang fr | en | de | es : change la langue du serveur.

En ce qui concerne le client, il peut se connecter et se déconnecter du serveur de chat. Un client peut aussi afficher un message et demander au serveur de chat d'envoyer un message à tout le monde ou à un unique client.

Connexion au serveur :

Pour se connecter au serveur de chat, le client distant lie son adresse à lui-même auprès du RMIRegistry. Il récupère ensuite une référence (stub) vers le serveur de chat. Enfin, il demande au serveur de chat de l'enregistrer.

Déconnexion du serveur :

Pour se déconnecter du serveur de chat, le client distant demande au serveur de chat de le désenregistrer. Ensuite le client distant délie son adresse à lui-même auprès du RMIRegistry.

Envoi d'un message :

Lorsque l'utilisateur envoie un message, on vérifie s'il a envoyé une commande ou non. S'il a envoyé une commande, on effectue l'action appropriée (quitter le chat, renommer son pseudo, afficher l'aide, afficher des informations sur le serveur, afficher la liste des utilisateurs, activer ou désactiver l'affichage de l'heure, envoyer un message en privé). Dans le cas contraire, le client distant demande au serveur de chat d'envoyer un message à tous les utilisateurs.

Envoi d'un message privé :

Pour envoyer un message en privé, on affiche le message chez l'émetteur puis on demande au serveur de chat d'envoyer le message à un unique utilisateur.

Les commandes :

/help : affiche la liste des différentes commandes disponibles et leur description.

/quit : quitte le chat.

/rename nouveau : permet de renommer le nom de son pseudo.

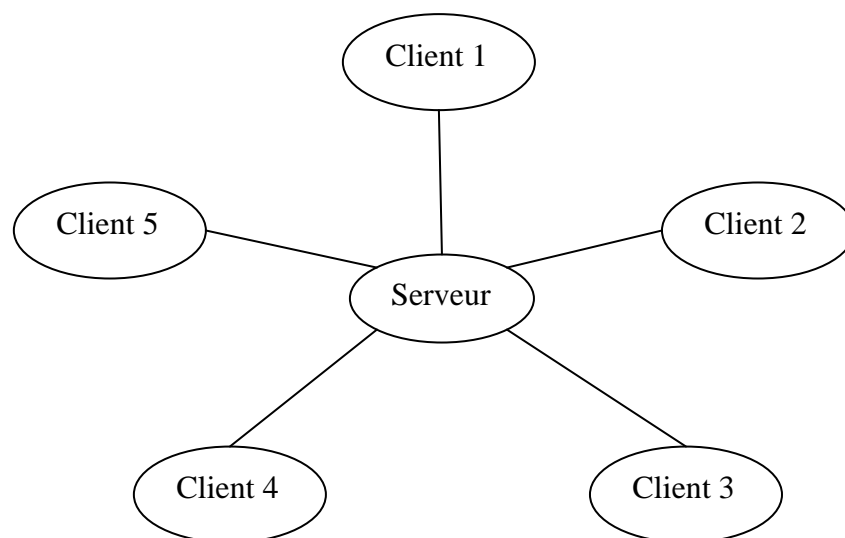
/pm pseudo message : envoi un message à un unique utilisateur.

/info : affiche des informations sur le serveur.

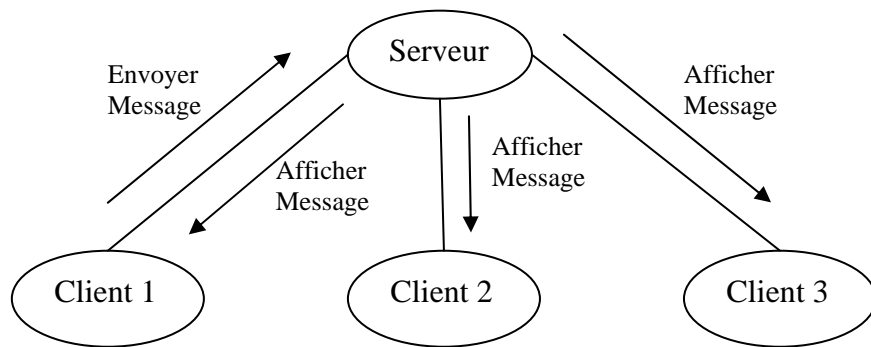
/list : affiche la liste des utilisateurs connectés.

/date on | off : active ou désactive l'affichage de l'heure.

/lang fr | en | de | es : change la langue courante.



Réseau.



Envoi de messages.

B. Caractéristiques Client / Serveur

Les caractéristiques du serveur sont :

- Démarrage du serveur.
- Arrêt du serveur.
- Enregistrement d'un nouveau client distant.
- Désenregistrement d'un client distant existant.
- Envoi de messages à tous les clients distants.
- Envoi de messages en privé.
- Renommer un utilisateur.
- Afficher l'aide. (Les différentes commandes)
- Afficher le nombre d'utilisateurs connectés.
- Changer la langue courante.

Les caractéristiques du client sont :

- Connexion au serveur de chat.
- Déconnexion du serveur de chat.
- Afficher un message.
- Envoyer des messages à tous les clients distants.
- Envoyer des messages en privé.
- Changer le nom de son pseudonyme.
- Afficher l'aide. (Les différentes commandes)
- Afficher des informations sur le serveur et les utilisateurs.
- Afficher la liste de tous les utilisateurs.
- Activer ou désactiver l'affichage de l'heure.
- Changer la langue courante.

Les caractéristiques d'un message sont :

- L'émetteur du message.
- Le contenu du message.
- Le type du message. (Normal, Serveur, Entrée, Sortie, Privé, Propre, Renommé)
- La date du message.

Les caractéristiques du GUI sont :

- Affichage en temps réel de la liste des utilisateurs.
- Gère les utilisateurs anonymes.
- Action connecter
- Action Déconnecter
- Action Envoyer
- Action Quitter
- Action Langue
- Action A Propos
- Action Manuel

3. Version 2

A. Présentation et principe

Dans cette version, le serveur de chat ne fait qu'enregistrer et désenregistrer des clients distants.

Enregistrement d'un client :

Pour enregistrer un client, on récupère une référence (stub) vers l'objet distant associé à l'URL fournit en paramètre pour obtenir des informations sur ce client. Ensuite on ajoute dans une carte de hashage l'URL du client avec le nom de l'utilisateur.
Enfin message de log est affiché sur la sortie standard.

Désenregistrement d'un client :

Pour désenregistrer un client, on supprime de la carte de hashage correspondant à la liste des utilisateurs le couple ayant pour clef l'URL fournie en paramètre.
Enfin message de log est affiché sur la sortie standard.

Les commandes :

- /help : affiche la liste des différentes commandes disponibles et leur description.
- /exit : permet de stopper le serveur de chat.
- /users : permet d'afficher le nombre d'utilisateurs connectés.
- /lang fr |en | de |es : change la langue du serveur.

En ce qui concerne le client, il peut se connecter et se déconnecter du serveur de chat. Un client peut aussi envoyer des messages à tous les utilisateurs, envoyer un message à un unique utilisateur et afficher un message. Les clients sont directement connectés entre eux.

Connexion au serveur :

Pour se connecter au serveur de chat, le client distant lie son adresse à lui-même auprès du RMIRegistry. Il récupère ensuite une référence (stub) vers le serveur de chat et demande au serveur de chat de l'enregistrer.

Puis, on récupère la liste de tous les utilisateurs et on met à jour la liste des utilisateurs connus de ce client. Ensuite, pour chaque utilisateur de la liste, on ajoute ce client à leur liste d'utilisateurs.

Enfin on envoie un message à tous les clients pour les prévenir qu'un nouvel utilisateur est entré dans le chat, on affiche sur ce client le nom de l'utilisateur si celui-ci est un nom anonyme généré et un message de bienvenue.

Déconnexion du serveur :

Pour se déconnecter du serveur de chat, le client distant demande au serveur de chat de le désenregistrer.

Ensuite on supprime le couple correspondant à ce client de la liste des utilisateurs (carte de hashage) et pour chaque client restant de cette liste, on supprime ce client de leur liste des utilisateurs.

Puis on envoie un message à tous les clients pour les prévenir qu'un utilisateur vient de quitter le chat.

Enfin, le client distant délie son adresse à lui-même auprès du RMIRegistry.

Envoi d'un message :

Lorsque l'utilisateur envoie un message, on vérifie s'il a envoyé une commande ou non. S'il a envoyé une commande, on effectue l'action appropriée (quitter le chat, renommer son pseudo, afficher l'aide, afficher des informations sur le serveur, afficher la liste des utilisateurs, activer ou désactiver l'affichage de l'heure, envoyer un message en privé). Dans le cas contraire, on parcourt l'ensemble des utilisateurs connus et pour chacun d'eux, on invoque la méthode permettant d'afficher un message. Si une erreur est rencontrée, on désenregistre le client.

Envoi d'un message privé :

Pour envoyer un message en privé, on parcourt la liste de toutes les URL et pour chacune on vérifie via la carte de hachage si le nom de l'utilisateur correspondant à l'URL est celui qui doit recevoir le message. Si l'utilisateur n'a pas été trouvé alors on envoie un message à l'émetteur. Si l'utilisateur a été trouvé alors on récupère une référence (stub) vers l'objet distant associé à l'URL et on affiche le message chez le client distant.

Enfin on affiche le message sur ce client.

Les commandes :

/help : affiche la liste des différentes commandes disponibles et leur description.

/quit : quitte le chat.

/rename nouveau : permet de renommer le nom de son pseudo.

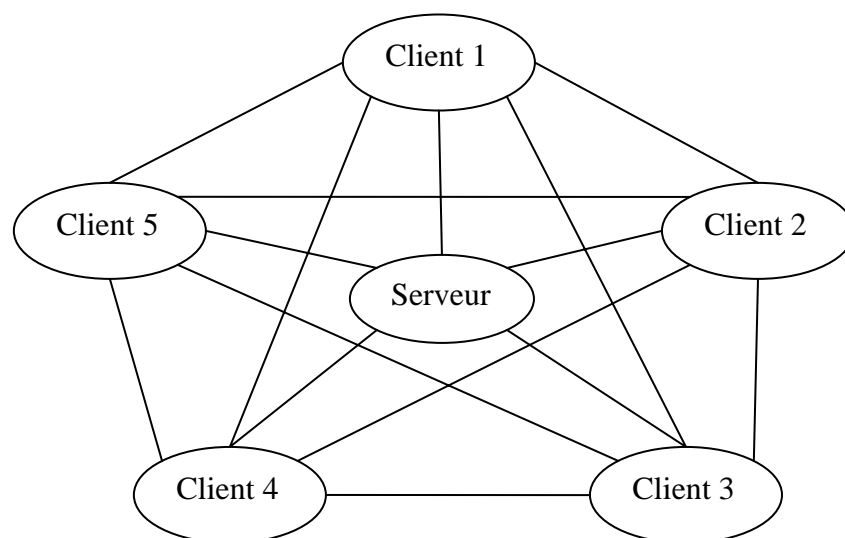
/pm pseudo message : envoie un message à un unique utilisateur.

/info : affiche des informations sur le serveur.

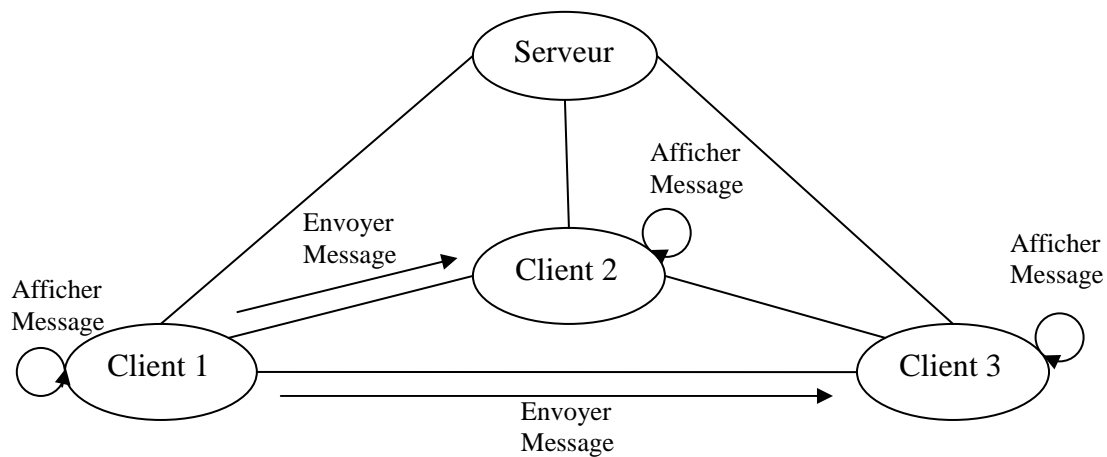
/list : affiche la liste des utilisateurs connectés.

/date on | off : active ou désactive l'affichage de l'heure.

/lang fr | en | de | es : change la langue courante.



Réseau.



Envoi de messages.

B. Caractéristiques Client / Serveur

Les caractéristiques du serveur sont :

- Démarrage du serveur.
- Arrêt du serveur.
- Enregistrement d'un nouveau client distant.
- Désenregistrement d'un client distant existant.
- Renommer un utilisateur.
- Afficher l'aide. (Les différentes commandes)
- Afficher le nombre d'utilisateurs connectés.
- Changer la langue courante.

Les caractéristiques du client sont :

- Connexion au serveur de chat.
- Déconnexion du serveur de chat.
- Afficher un message.
- Envoyer des messages à tous les clients distants.
- Envoyer des messages en privé.
- Changer le nom de son pseudonyme.
- Afficher l'aide. (Les différentes commandes)
- Afficher des informations sur le serveur et les utilisateurs.
- Afficher la liste de tous les utilisateurs.
- Activer ou désactiver l'affichage de l'heure.
- Changer la langue courante.

Les caractéristiques d'un message sont :

- L'émetteur du message.
- Le contenu du message.
- Le type du message. (Normal, Serveur, Entrée, Sortie, Privé, Propre, Renommé)
- La date du message.

Les caractéristiques du GUI sont :

- Affichage en temps réel de la liste des utilisateurs.
- Gère les utilisateurs anonymes.
- Action connecter
- Action Déconnecter
- Action Envoyer
- Action Quitter
- Action Langue
- Action A Propos
- Action Manuel

4. Version 3

A. Présentation et principe

Dans cette version, le serveur de chat ne fait qu'enregistrer et désenregistrer des clients distants. Le but du serveur est ici d'optimiser le maillage de connexions. Chaque client sera ainsi connecté à un nombre restreint d'autres clients.

Enregistrement d'un client :

Pour enregistrer un client, on récupère une référence (stub) vers l'objet distant associé à l'URL fournit en paramètre pour obtenir des informations sur ce client. On récupère ainsi le pseudonyme du client.

Puis, on initialise le niveau (permet de savoir si un client est parent d'un autre, de classer les clients, ...) à l'entier maximum et la taille au nombre de clients maximum auquel un client peut être connecté.

Ensuite on va rechercher le meilleur client avec qui se connecter, pour cela on parcourt la liste des URL des clients enregistrés. Pour chaque URL, on récupère une référence (stub) pour le client distant associé et on teste si ce client est meilleur : si le client a un niveau inférieur au niveau courant et que le nombre de clients connectés à ce client ne dépasse pas le maximum ou alors que le client a un niveau égal au niveau courant et que le nombre de clients connectés à ce client est inférieur à la taille courante alors on a trouvé un meilleur client avec qui se connecter. On met ainsi à jour le client courant, l'adresse courante, le niveau courant et la taille courante.

Puis, on connecte les deux clients entre eux, pour cela, on ajoute le client courant au nouveau client et on initialise son niveau. On ajoute aussi le nouveau client au client courant.

Enfin, on ajoute le nouveau client à la liste des clients du serveur de chat et on affiche un message de log sur la sortie standard.

Désenregistrement d'un client :

Pour désenregistrer un client, on supprime de la carte de hashage correspondant à la liste des utilisateurs le couple ayant pour clef l'URL fournie en paramètre.

Puis, on va récupérer la liste des clients connaissant ce client. Pour cela, on parcourt la liste des clients enregistrés et on vérifie si leur liste contient le client qui se désenregistre et si c'est le cas on l'ajoute à la liste des clients connaissant ce client.

Ensuite on récupère l'adresse de client qui a le niveau minimum.

Il faut maintenant reconstruire le maillage pour que celui-ci reste totalement connecté.

On parcourt donc la liste des clients connaissant le client qui se désenregistre et si ce n'est pas celui qui a le niveau minimum alors, on va rechercher le meilleur client avec qui se connecter.

La méthode ressemble fortement à ce qui a été décrit dans enregistrement d'un client.

Enfin, on affiche un message de log sur la sortie standard.

Les commandes :

/help : affiche la liste des différentes commandes disponibles et leur description.

/exit : permet de stopper le serveur de chat.

/nwk : affiche la topologie du réseau.

/users : permet d'afficher le nombre d'utilisateurs connectés.

/lang fr | en | de | es : change la langue du serveur.

En ce qui concerne le client, il peut se connecter et se déconnecter du serveur de chat. Un client peut aussi envoyer/faire suivre des messages à tous les utilisateurs qu'il connaît, envoyer un message à un unique utilisateur et afficher un message. Les clients sont connectés à un nombre restreint d'autres clients.

Connexion au serveur :

Pour se connecter au serveur de chat, le client distant lie son adresse à lui-même auprès du RMIRegistry. Il récupère ensuite une référence (stub) vers le serveur de chat et demande au serveur de chat de l'enregistrer.

Enfin on envoie un message à tous les clients pour les prévenir qu'un nouvel utilisateur est entré dans le chat, on affiche sur ce client le nom de l'utilisateur si celui-ci est un nom anonyme généré et un message de bienvenue.

Déconnexion du serveur :

Pour se déconnecter du serveur de chat, le client distant demande au serveur de chat de le désenregistrer.

Puis le client envoie/faire suivre un message à tous les clients auquel il est connecté pour les prévenir qu'un utilisateur vient de quitter le chat.

Enfin, le client distant délie son adresse à lui-même auprès du RMIRegistry.

Envoi d'un message :

Lorsque l'utilisateur envoie un message, on vérifie s'il a envoyé une commande ou non. S'il a envoyé une commande, on effectue l'action appropriée (quitter le chat, renommer son pseudo, afficher l'aide, afficher des informations sur le serveur, afficher la liste des utilisateurs, activer ou désactiver l'affichage de l'heure, envoyer un message en privé). Dans le cas contraire, on affiche le message sur ce client et on fait suivre le message aux clients auquel on est connecté autre que le client parent (celui qui vient de faire suivre le message : c'est ce qui permet d'éviter une boucle infinie). Si une erreur est rencontrée, on désenregistre le client et on fait suivre un message indiquant qu'un client a quitté le chat.

Envoi d'un message privé :

Pour envoyer un message en privé, on commence par vérifier si ce client est dans la liste des clients privés, si c'est le cas on récupère une référence (stub) de ce client et on invoque la méthode afficher message. Dans le cas contraire, on recherche le client et si on le trouve on l'ajoute dans la liste des clients privés et ensuite on récupère une référence (stub) de ce client et on invoque la méthode afficher message.

Enfin on affiche le message sur ce client.

Les commandes :

/help : affiche la liste des différentes commandes disponibles et leur description.

/quit : quitte le chat.

/rename nouveau : permet de renommer le nom de son pseudo.

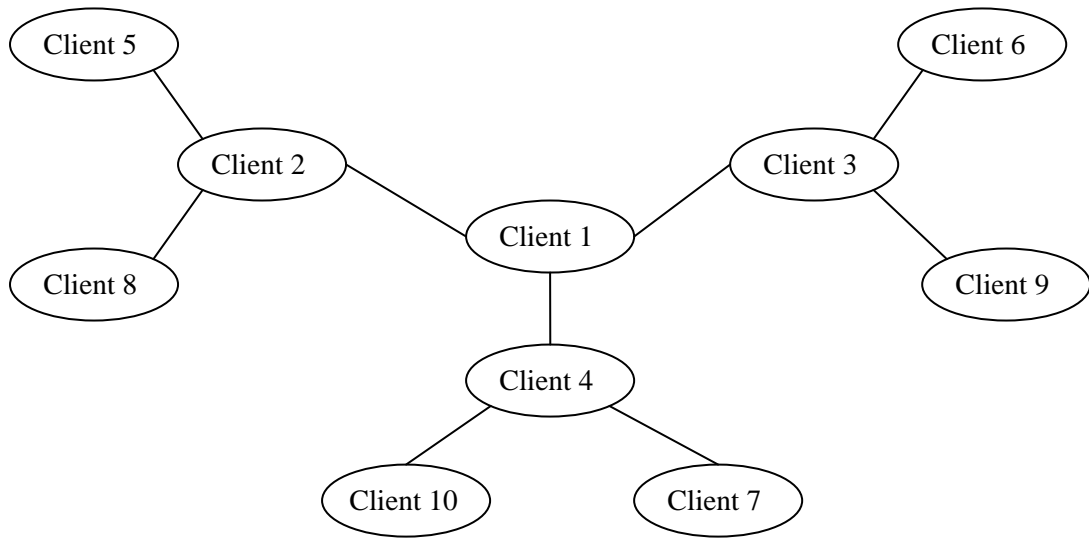
/pm pseudo message : envoi un message à un unique utilisateur.

/info : affiche des informations sur le serveur.

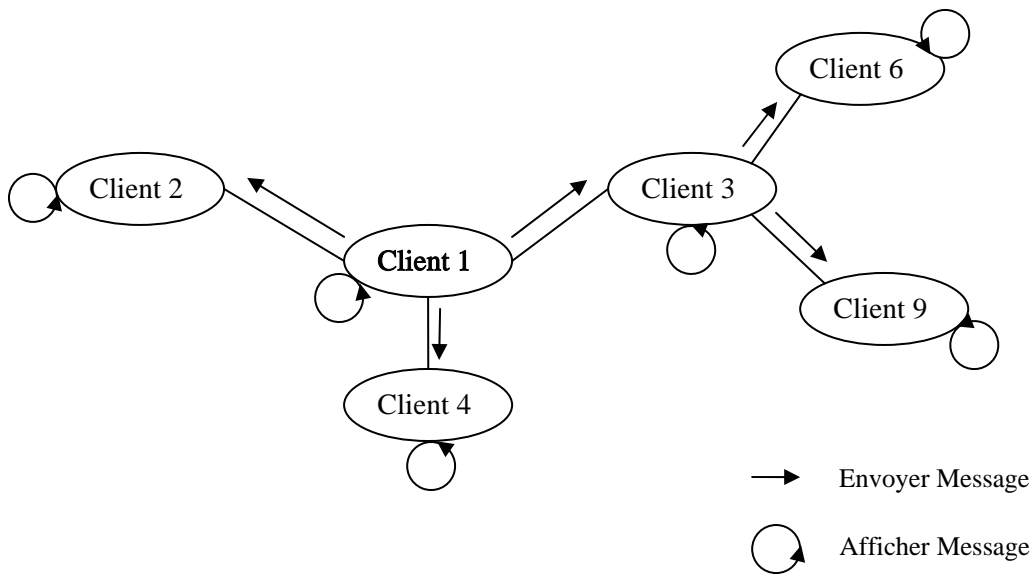
/list : affiche la liste des utilisateurs connectés.

/date on | off : active ou désactive l'affichage de l'heure.

/lang fr | en | de | es : change la langue courante.



Réseau.



→ Envoyer Message

⦿ Afficher Message

Envoi de messages.

B. Caractéristiques Client / Serveur

Les caractéristiques du serveur sont :

- Démarrage du serveur.

- Arrêt du serveur.
- Enregistrement d'un nouveau client distant.
- Désenregistrement d'un client distant existant.
- Renommer un utilisateur.
- Afficher l'aide. (Les différentes commandes)
- Afficher le nombre d'utilisateurs connectés.
- Afficher la topologie du réseau.
- Changer la langue courante.

Les caractéristiques du client sont :

- Connexion au serveur de chat.
- Déconnexion du serveur de chat.
- Afficher un message.
- Envoyer des messages à tous les clients distants.
- Envoyer des messages en privé.
- Changer le nom de son pseudonyme.
- Afficher l'aide. (Les différentes commandes)
- Afficher des informations sur le serveur et les utilisateurs.
- Afficher la liste de tous les utilisateurs.
- Activer ou désactiver l'affichage de l'heure.
- Changer la langue courante.

Les caractéristiques d'un message sont :

- L'émetteur du message.
- Le contenu du message.
- Le type du message. (Normal, Serveur, Entrée, Sortie, Privé, Propre, Renommé)
- La date du message.

Les caractéristiques du GUI sont :

- Affichage en temps réel de la liste des utilisateurs.
- Gère les utilisateurs anonymes.
- Action connecter
- Action Déconnecter
- Action Envoyer
- Action Quitter
- Action Langue
- Action A Propos
- Action Manuel

5. Utilisation

Nécessite la version 1.5 de Java.

Sous Windows :

- Pour lancer le serveur, il faut exécuter serveur.bat
- Pour lancer le client, il faut exécuter client.bat
- Pour lancer le client graphique, il faut exécuter clientGUI.bat

Sous Linux :

- Pour lancer le serveur, il faut exécuter serveur.sh
- Pour lancer le client, il faut exécuter client.sh
- Pour lancer le client graphique, il faut exécuter clientGUI.sh

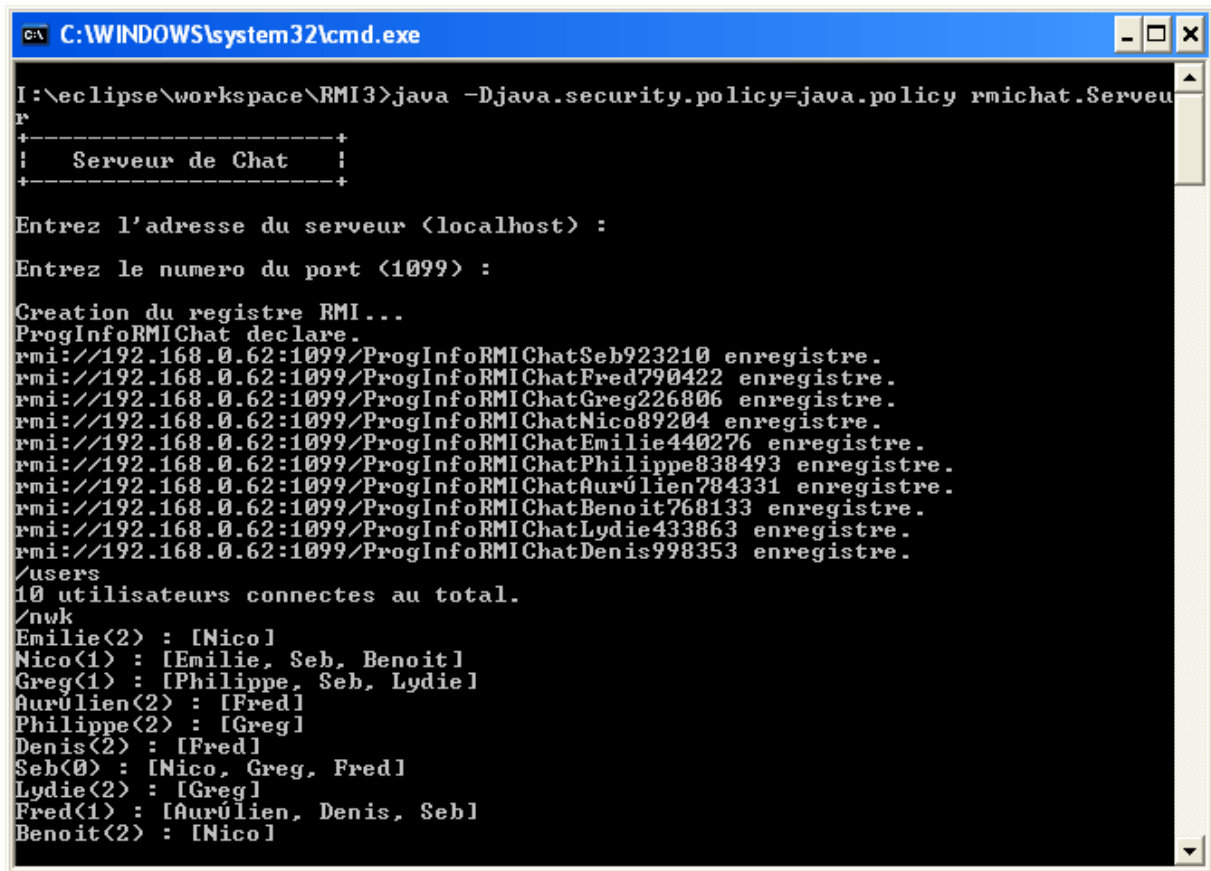
Remarques :

Le client et serveur peuvent être utilisés avec ou sans arguments. Si les arguments ne sont pas donnés, ils seront demandés à l'utilisateur. Si l'utilisateur ne les connaît pas, des valeurs par défaut seront utilisées.

```
java -Djava.security.policy=java.policy rmichat.Serveur
```

```
java -Djava.security.policy=java.policy rmichat.Serveur 192.168.0.1 1099
```

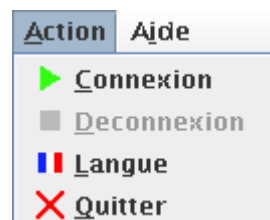
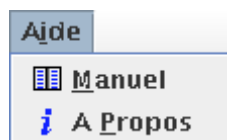
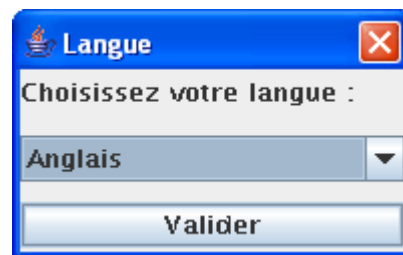
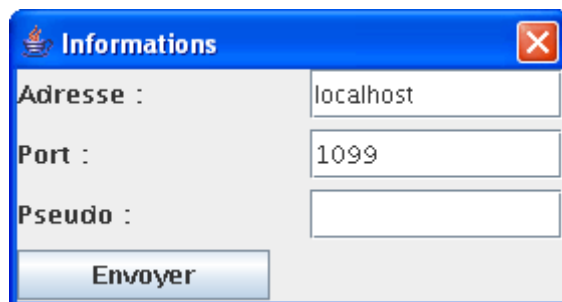
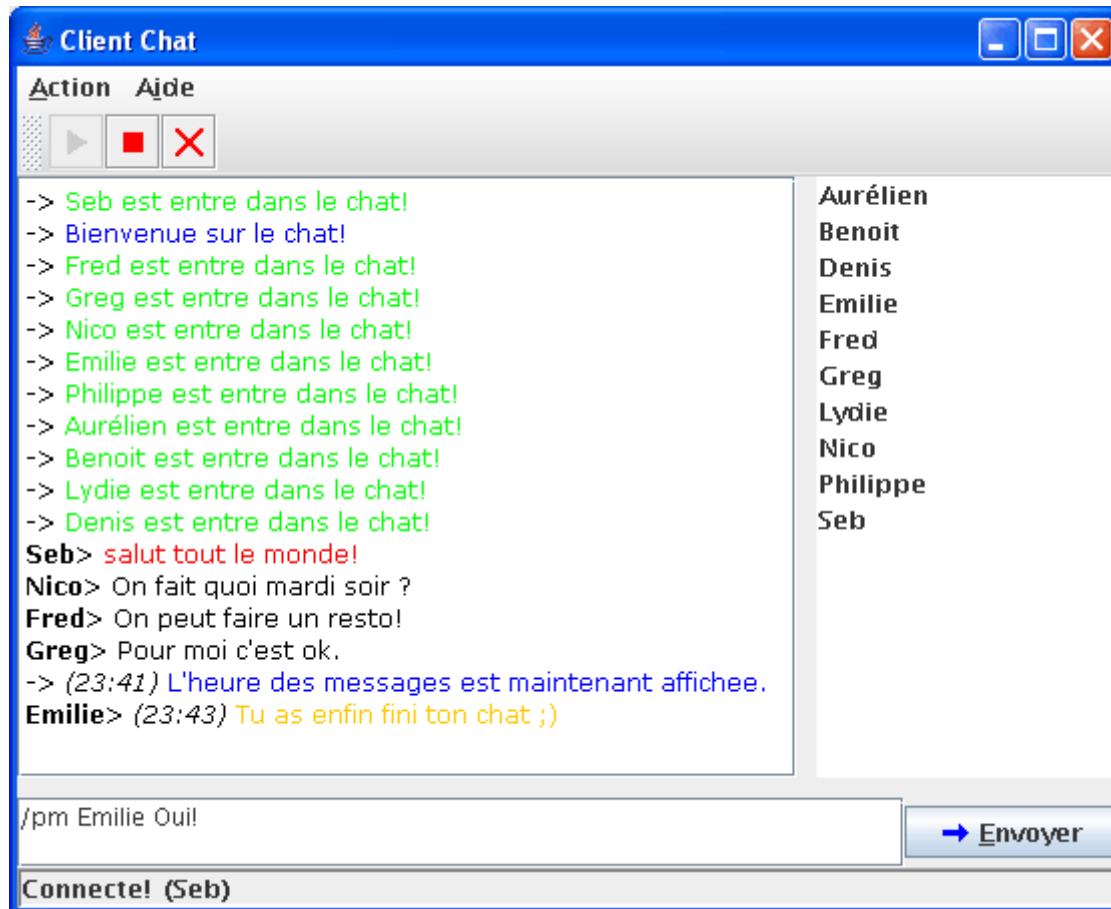
6. Screenshots



```
C:\WINDOWS\system32\cmd.exe
I:\eclipse\workspace\RMI3>java -Djava.security.policy=java.policy rmichat.Serveur
P
+-----+
|  Serveur de Chat  |
+-----+

Entrez l'adresse du serveur (localhost) :
Entrez le numero du port (1099) :

Creation du registre RMI...
ProgInfoRMIChat declare.
rmi://192.168.0.62:1099/ProgInfoRMIChatSeb923210 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatFred790422 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatGreg226806 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatNico89204 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatEmilie440276 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatPhilippe838493 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatAurilien784331 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatBenoit768133 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatLydie433863 enregistre.
rmi://192.168.0.62:1099/ProgInfoRMIChatDenis998353 enregistre.
/users
10 utilisateurs connectes au total.
/nwk
Emilie(2) : [Nico]
Nico(1) : [Emilie, Seb, Benoit]
Greg(1) : [Philippe, Seb, Lydie]
Aurilien(2) : [Fred]
Philippe(2) : [Greg]
Denis(2) : [Fred]
Seb(0) : [Nico, Greg, Fred]
Lydie(2) : [Greg]
Fred(1) : [Aurilien, Denis, Seb]
Benoit(2) : [Nico]
```



7. Code source

AfficherAPropos.java
AfficherManuel.java
Client.java
ClientDistant.java
ClientDistantImpl.java
ClientGUI.java
Constantes.java
EcouteurDeconnexion.java
EcouteurMessage.java
Message.java
Serveur.java
ServeurChat.java
ServeurChatImpl.java

APropos.html
Manuel.html

client_de_DE.properties
client_en_US.properties
client_es_ES.properties
client_fr_FR.properties
server_de_DE.properties
server_en_US.properties
server_es_ES.properties
server_fr_FR.properties

8. Remarques

Il est facile de tester son chat sur la même machine mais cela ne permet pas de détecter les nombreux problèmes qu'il peut y avoir.

Le test sur différentes machines est impossible étant donné que les temps de réponses sont pitoyables ce depuis la version 1.5 de Java. Le bug a été répertorié sur le site Sun.

9. Problèmes

Dans la version 1 :

Si un client quitte brutalement c'est-à-dire via un kill, un ctrl+Alt+Suppr ou un redémarrage, il se peut qu'une erreur de ConcurrentModification apparaisse. La synchronisation semblait avoir réglé ce problème mais il semble persister.

Dans la version 2 :

Si la dernière personne quitte le chat brutalement c'est-à-dire via un kill, un ctrl+Alt+Suppr ou un redémarrage, plus personne ne pourra se reconnecter.

Dans la version 3 :

Si un client quitte brutalement c'est-à-dire via un kill, un ctrl+Alt+Suppr ou un redémarrage, il se peut qu'un certain nombre d'utilisateurs ne reçoivent pas juste le prochain message.

Si la dernière personne quitte le chat brutalement c'est-à-dire via un kill, un ctrl+Alt+Suppr ou un redémarrage, plus personne ne pourra se reconnecter.

Il est très difficile de tester les problèmes de synchronisation du chat tout seul. Il est donc possible qu'il y en est.

En ce qui concerne la version 3 du chat, en réfléchissant un peu on peut aisément comprendre que lorsque qu'un client envoie un message et qu'un autre se déconnecte pendant l'envoi du message ce qui implique que le réseau est reconstruit par le serveur, il est fort probable qu'un utilisateur reçoive deux fois le même message ou pas du tout. Ce genre de problèmes est difficile à provoquer et à corriger. Des solutions à ce genre de problème pourraient être d'identifier de manière unique un message, de donner une durée de vie à un message, de demander un accusé de réception, ...

Une autre solution pourrait être aussi de changer de topologie de réseau. Celle qui a été choisie est plus simple à tester : il est difficile d'effectuer des tests quand on a besoin d'avoir un grand nombre de clients pour vérifier que tout fonctionne correctement.

Un exemple d'une topologie peut être plus simple à gérer est une topologie formée de petits groupes totalement connectés avec un chef dans chaque groupe permettant de relier les groupes entre eux. Aucune étude approfondie n'a été faite sur cette topologie car cela dépasse le cadre d'un mini projet !

10. Améliorations

De nombreuses améliorations peuvent être effectuées :

- Corriger les problèmes cités précédemment.
- Gérer les problèmes de synchronisations possibles.
- Limiter la taille d'un message.
- Implémenter la gestion des canaux.
- Ajouter du son.
- Remplacer les smileys par des images.
- Mettre les utilisateurs en privés dans différents onglets.
-