TP N°8 : RMI using Callback

Dans ce TP on se propose de réaliser un "**serveur de chat**" avec Java RMI. Ce serveur doit permettre à chaque client distant d'envoyer un message à tous les clients distants connectés ou un client distant spécifique.

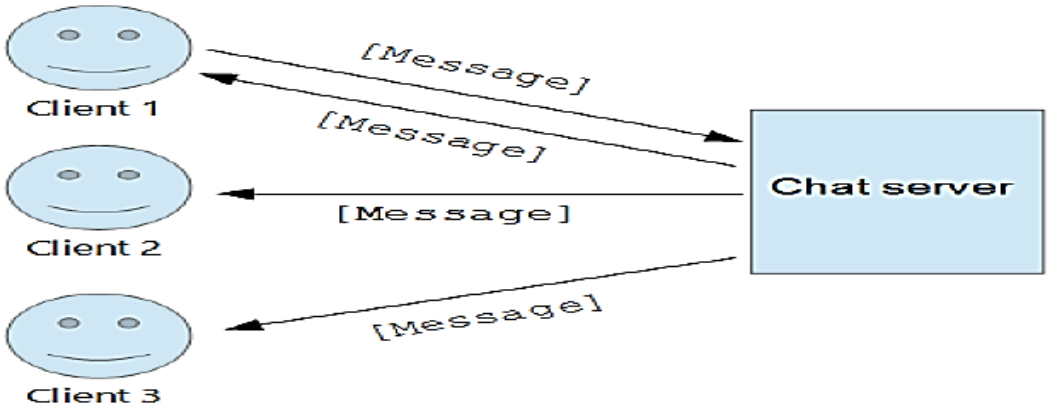

Fig.1 : Chat Server

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

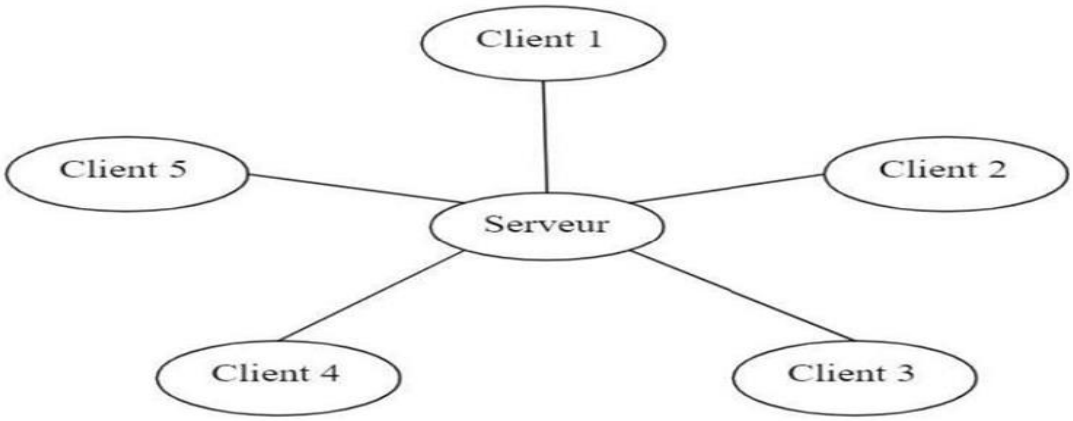

Fig.2 : Topologie réseau

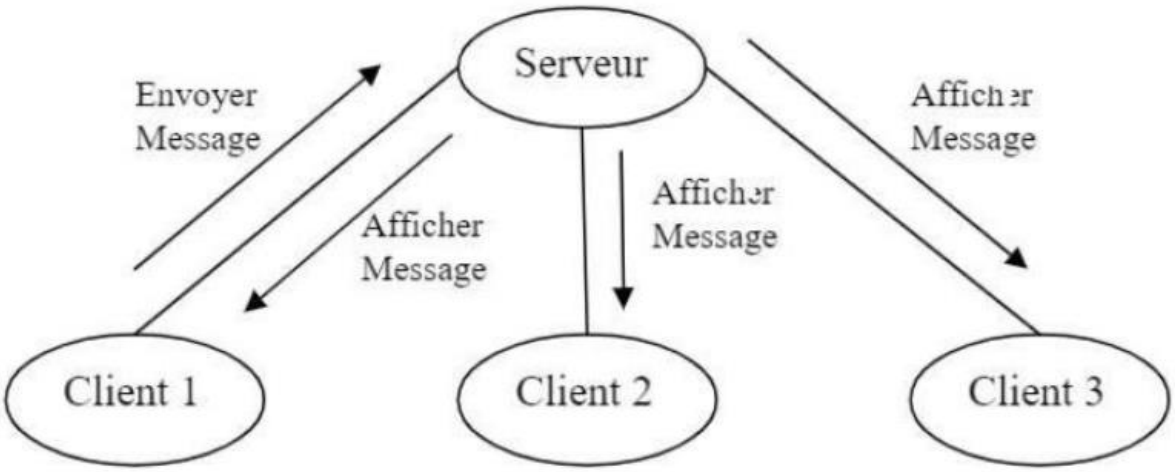

Fig.3 : Envoi de messages

On peut utiliser les interfaces suivantes :

```
public interface ClientDistantInterface extends Remote {
public void receiveMessage(String message) throws RemoteException; // recevoir
et afficher un message
public String getNameDistant() throws RemoteException;
public String getIdDistant() throws RemoteException;
}
```

```
public interface ChatServerInterface extends Remote {
String LOOKUP_NAME="ChatServer";
public void sendToAll(String SenderName,String message) throws
RemoteException;
public void sendToOne(String SenderName,String receiverName, String message)
throws RemoteException;
public void signIn(ClientDistantInterface c) throws RemoteException;
public void signOut(ClientDistantInterface c, String message) throws
RemoteException;
}
```

- La classe client distant hérite de la classe client local qui possède les attributs suivants : Id (Identificateur) et nom(pseudonyme) de type String et implémente l'interface ClientDistantInterface. Ce client distant peut se connecter (signIn) et se déconnecter (signOut) du serveur de chat. Un client distant peut aussi afficher un message et demander au serveur de chat d'envoyer un message à tout le monde ou à un client distant spécifique.

- Pour se connecter au serveur de chat (signIn), 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.
- Déconnexion du serveur (signOut) : Pour se déconnecter du serveur de chat, le client distant demande au serveur de chat de le désenregistrer. Puis le client envoie/fait 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.

- Le client distant est ici à la fois client et serveur au sens RMI. Les méthodes signIn et signout doivent être appelées par un client distant qui fournira en paramètre le stub de son objet serveur implémentant l'interface ClientDistant.

- 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 en paramètre pour obtenir des informations sur ce client. Ensuite on ajoute dans une liste de clients distants.
- Désenregistrement d'un client : Pour désenregistrer un client, on commence par supprimer le stub du client distant de la liste. Le serveur de chat envoie ensuite un message à tous les clients pour prévenir qu'un utilisateur est sorti du chat.
- Envoi d'un message à tout le monde : Pour envoyer un message à tous les clients, le serveur de chat parcours la liste de tous les stubs des clients distants. Pour chaque stub vers l'objet distant associé au nom et on affiche le message chez le client distant.
- Envoi d'un message privé : Pour envoyer un message à un unique client distant, le serveur de chat parcours la liste de tous les stubs des clients distants qui sont connectés. Pour chaque stub, on vérifie via la liste si le nom de l'utilisateur correspondant au nom de celui qui doit recevoir le message.