Lors du lancement du serveur, deux threads sont lancés.

Le premier thread attend des demandes de connexion des clients. Pour ce faire, le client va donner son nom. Ce thread créera un nouveau thread avec comme nom, celui du client pour chaque nouveau client. Si le nom existe déjà, la demande de connexion sera jeté. De plus, le nom du client et son identificateur d'échange (correspondant à la socket qui lui est attribué pour les échanges) sera mis dans une liste « csock ». Le serveur identifie donc ces clients et leur messages grâce à un string envoyé par le client.

Ces nouveaux threads créé écoutent le client qui leur a été affecté. Chaque fois qu'un message sera reçu pour ce client, le thread mettra le nouveau message dans une liste d'attente « messages » avec son nom comme identifiant du message. S'il y a déjà un message de se client, le nouveau écrasera l'ancien messages.

Le second thread créé au lancement du serveur, envoi les messages à tout les clients. Dès lors que la liste « messages » n'est pas vide, le message sera envoyé à tout les clients de « csock ». Puis le message sera écrasé de manière sécurisé avec des mutex. Si un client à un problème de réception alors tant pis pour lui, aucun nouveau message ne lui sera renvoyé. Mais comme dans notre contexte, chaque client envoie 60 messages par secondes, cela ne créé pas de latence. Elle l'empêche même!

Lors de la création d'un client, il y a tentative de connexion au serveur puis si tout ce passe bien, notre nom est envoyé au serveur. Ensuite, deux threads sont également lancé. Le premier ecoute, le second envoi.

Tout comme le serveur, ces threads sont associés à des listes. La réception de message insert le message reçu dans une file d'attente « messages_recu » et l'envoi va chercher dans la liste « message_envoi » les messages, les envoi et les supprime de la liste.

Ces threads sont invisibles pour l'utilisateur de la classe « Client ». L'utilisation des méthodes envoyer() et recevoir() permettrons de donner les messages à envoyer et de récupérer les messages reçus. envoyer() va mettre dans la file « message_envoi » et recevoir() va prendre le premier message de « messages_recu » puis le supprimer.