2.3 Tester avec le client

Etant donné que nous avons un serveur TCP, il doit accepter chacun des client qui souhaitent se connecter. Donc lorsque l'on essaye avec un deuxieme client, il n'est pas accepté puisque nous réalisons un unique accept(), il faudrait autant d'appel qu'il y a de client, cependant c'est une fonction bloquante.
Il semblerait que l'utilisation de thread pourrait palier au problème des fonctions bloquantes.

3.2 Test

On constate que le langage de programmation n'a pas d'impact sur la communication entre le client et le serveur, ce qui témoigne d'une certaine ergonomie permettant à une application de d'accéder à un serveur sans avoir à ce soucier d'un compatibilité des langages.

3.3 Compatibilité UDP et TCP

On constate que les couples clients/serveurs avec des protocoles (UDP, TCP) différents ne fonctionnent pas. En effet, ce sont 2 manière différentes d'établir un connexion, en ce qui concerne le UDP, il n'y a pas de communication continu et de simple envoi de paquet alors que dans le cas du TCP, il faut établir une connexion sécurisée pour pouvoir avoir un flux continu de données qui sont échangées entre le client et le serveur. En sachant cela on peut comprendre que la communication entre UDP et TCP ne fonctionne pas, au niveau programmation, en Java ou bien en C les fonctions et les objets (pour Java) sont différents entre les deux protocoles.