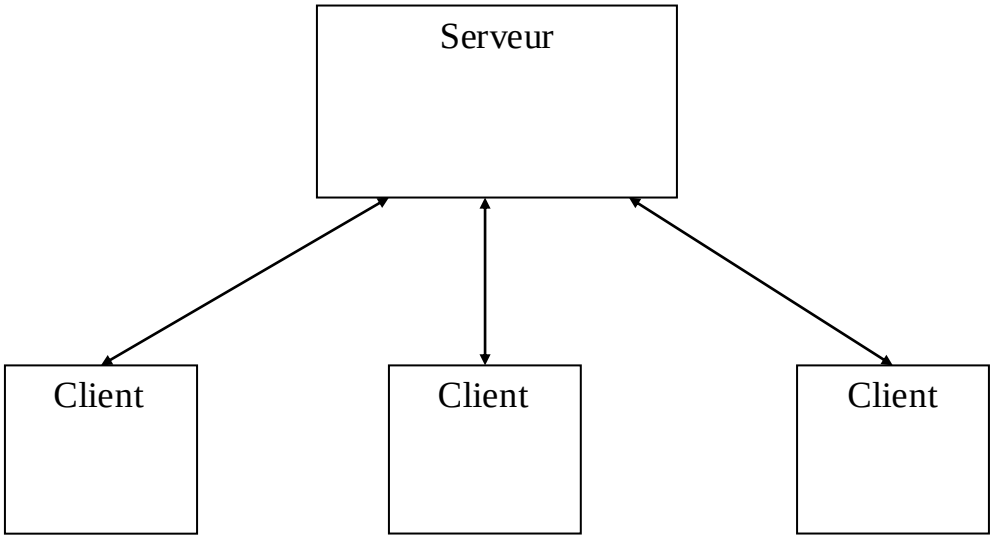
APPLICATION CLIENT/SERVEUR EN MODE TCP POUR UNE COMMUNICATION INTER USAGERS.

***PROBLÈME* :**
L'objectif principal de ce laboratoire est de familiariser l'étudiant avec la communication entre un processus client et un processus serveur. Le laboratoire #1 permettra d'effectuer une communication entre plusieurs ordinateurs. Le programme pourra envoyer et recevoir des chaînes de caractères de longueur variable et ce, de façon asynchrone. Pour ce faire, l'étudiant devra développer une fenêtre d'envoi/réception dans laquelle nous verrons les chaînes de caractères reçues et dans laquelle nous pourrons également écrire des messages à l'aide du clavier. Les caractères seront envoyés lorsque l'utilisateur effectuera un retour de chariot.

Pour faire le laboratoire, l'étudiant intégrera les notions de "socket" TCP/IP.
Les "sockets" seront utilisés de façon asynchrone afin de respecter l'énoncé du laboratoire.

***DESCRIPTION* :**
Le schéma global du laboratoire est le suivant :



Il s'agit d'une application inter-usagés « chat » pour permettre l'échange de messages entre clients d'un même serveur. Chaque client qui s'enregistrera auprès du serveur pourra envoyer des messages aux autres clients, et aussi recevoir les messages envoyés par tous les clients.
Au minimum, 3 clients doivent être supportés en même temps par le Serveur afin de bien démontrer le concept client/serveur.

Processus Serveur :
Le processus Serveur s'occupera de la réception des messages des clients et de la retransmission de ces messages vers les autres clients. Il devra tenir à jour une base de données de tous les clients (nom, adresse) afin de pouvoir compléter correctement la retransmission aux autres clients.
Lorsqu'un paquet sera reçu, le serveur aura à décapsuler ce paquet de façon à récupérer les informations pertinentes (nom du client, commande pour le serveur, message client…).
Quant un client se connecte pour la première fois ou quant un client quitte, le serveur envoie la liste des clients qui sont encore connectés (Push).
Si un client envoie la chaîne de caractères **« list »** (Pull), alors le serveur devra l'interpréter comme une demande d'envoi de la liste des clients enregistrés. Il ne doit pas faire suivre ce message aux autres clients.

Processus Client :
Le processus Client permettra d'effectuer l'envoi des chaînes de caractères entrées par l'utilisateur aux autres clients par l'entremise du serveur. Lorsqu'un paquet sera envoyé, le logiciel aura à encapsuler ce paquet de façon à obtenir le paquet à envoyer.
Lorsqu'un paquet sera reçu, le client aura lui aussi à décapsuler ce paquet de façon à récupérer le message qu'un autre client a envoyé.

PROGRAMMATION :
Il est important dans ce type de programmation d'être le plus clair possible (des points seront attribués pour la clarté et la simplicité du code…). N'hésitez donc pas à commenter votre code.
Comme la communication sera faite éventuellement entre un SUN et un PC, il faudra tenir compte de la notion "little endian, big endian". Les fonctions suivantes vous seront très utiles:
- htonl (**h**ost **to** **n**etwork **l**ong)
- htons (**h**ost **to** **n**etwork **s**hort)

Ces deux fonctions sont définies dans l'implémentation des sockets.

Définitions de Winsock (winsock.h, winsock, lib, winsock.hlp) :
Location : http://www.sockets.com/
***CONTENU DU RAPPORT* :**
- Page titre
- Table des matières (si jugée nécessaire)
- Introduction
- Objectifs et buts
- Méthodologie de conception
- Analyse, description du programme
- Croquis, Organigramme ou algorithmes
- Structures de données
- Discussion
- Conclusion
- Code source et exécutable
- Mode d'emploi