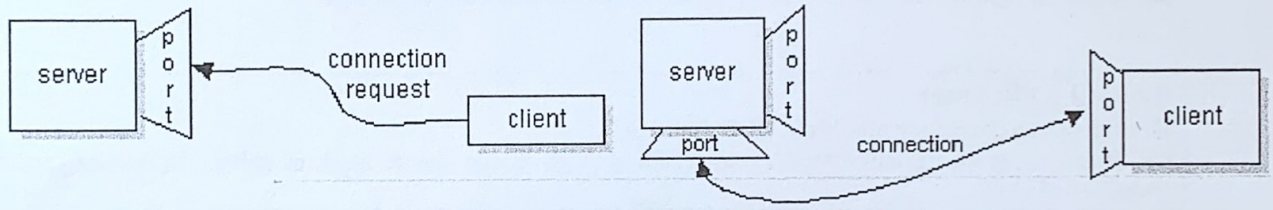
RSA Réseau
TP n° 2
Programmation Socket en langage C

Objectif du TP:

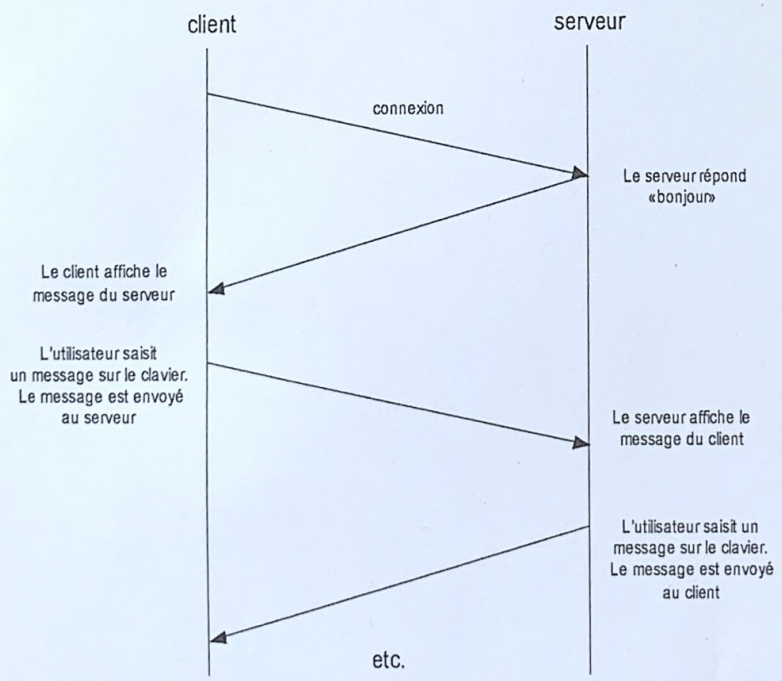
L'objectif de ce TP est de découvrir la programmation socket en langage C. Pour cela, vous développerez des couples "client-serveur" pour les protocoles de transport TCP et UDP.

Protocole TCP (communication en mode connecté):

Dans le mode connecté, les données sont transmises via un canal de communication entre deux entités (l'une client, l'autre serveur). Ce canal est appelé, dans le cas de TCP, une connexion.
La connexion nécessite d'être établie et maintenue entre ces deux entités, ce qui génère plus de traitement que dans le cas de la communication par datagramme (UDP).
TCP s'inscrit dans le cadre d'une communication de type client serveur. Concrètement, cela sous-entend que le client fait appel à des services du serveur qui traite les demandes et retourne le résultat des requêtes. La communication client serveur repose sur différentes étapes : le serveur **ouvre une socket**, **lie** la socket à l'adresse locale sur **un de ces ports**, se place **en écoute** de demande de connexion, le client **ouvre une socket** et se **connecte**, le serveur **accepte** et le **dialogue** peut commencer avec le client. La notion de numéro de port permet à TCP de retrouver l'application en question.



Le scénario de la connexion est donc le suivant:



Travail à effectuer:
Développer un serveur (serverTCP.c) et un client (clientTCP.c) qui permettent à deux utilisateurs de discuter à distance (cf. figure).

- **Vous commencerez par développer le serveur. Vous le testerez en utilisant telnet en tant que client.**
- **Une fois le serveur opérationnel, vous pourrez développer le client en C. Pour obtenir l'adresse IP du serveur, vous utiliserez successivement les fonctions inet_addr(), gethostbyname()/gethostbyname2().**
- **Modifier le serveur pour qu'il puisse gérer plusieurs clients (utilisation de fork()).**

Protocole UDP (communication en mode datagramme):

Certaines applications ne requièrent pas le canal de communication fourni par TCP. De plus, elles peuvent spécifier un mode de communication qui ne garantisse pas l'ordre d'arrivée des messages. Le protocole UDP fournit ce mode de communication réseau où les applications envoient des paquets de données appelés datagramme. Un datagramme est un message indépendant pour lequel ni son arrivée, ni son délai de transmission, ni la protection de ses données sont garanties.

Le principe de communication par datagramme est plus simple : un client doit récupérer l'adresse d'un serveur (adresse IP + numéro de port pour UDP), puis lui envoyer un paquet (datagramme) de données.
Dans ce cadre, le programme serveur **ouvre une socket** et se met en **attente de lecture** sur un port UDP. Le client **ouvre une socket**. Il **construit ensuite un datagramme** en précisant cette fois outre les données, l'adresse de destination ainsi que le numéro de port visé. Le serveur recevant le message détient alors la possibilité de répondre au client en récupérant l'adresse du client du paquet reçu.

Travail à effectuer:
Développer un couple client-serveur UDP client en C.
Le client envoie l'heure au serveur et quitte. Le serveur affiche l'heure reçue et quitte. Utiliser les fonctions ftime() et ctime()