TP/Projet RSA – Programmation Socket
TELECOM Nancy 2A

1. Objectifs du TP

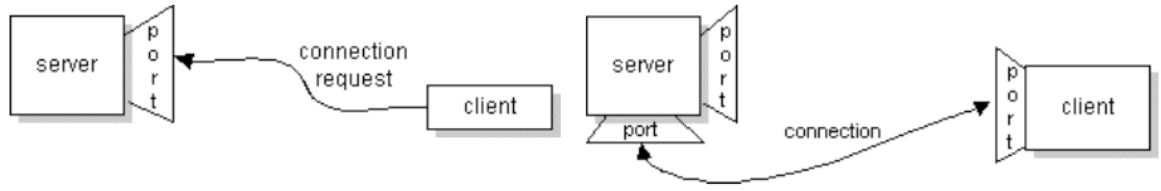
L'objectif de ce TP est de découvrir la programmation socket en langage C, avec le développement de couples client-serveur pour les protocoles de transport TCP et UDP.

Vous déposerez individuellement à la fin de chaque TP les fichiers développés en C dans le dépôt correspondant sous Arche, avec un fichier readme.txt indiquant synthétiquement le travail réalisé et son fonctionnement.

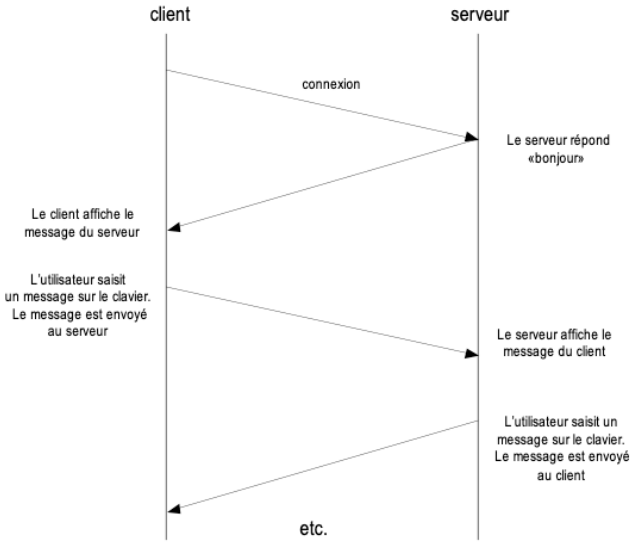
Le projet (section 4) correspond à un projet à réaliser en binôme, et à rendre dans le dépôt correspondant sous Arche, avec un fichier readme.txt indiquant synthétiquement le travail réalisé et son fonctionnement. La date limite de rendu est fixé au **18 avril 2024**.

2. 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 :



Développer un serveur (serveurTCP.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 pourrez utiliser successivement les fonctions inet_addr() et gethostbyname().

3. 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.

Développer un couple client-serveur UDP en C (nommés serveurUDP.c et clientUDP.c). Le client envoie l'heure au serveur et quitte. Le serveur affiche l'heure reçue et quitte. Utiliser les fonctions ftime() et ctime().

4. Projet : conception d'un scanner réseau simple

En partant du couple client-serveur TCP, développer un scanner réseau simple configurable à distance via un gestionnaire :

- ***Contrôle distant : le gestionnaire permet de configurer à distance un scanner réseau simple (similaire à nmap) qui réalise la cartographie d'un réseau local, les résultats d'analyse du scanner sont transmis en retour au gestionnaire (qui les affiche),***
- ***Scan horizontal : le scanner est capable de scanner un réseau/sous-réseau local pour déterminer les machines qui sont opérationnelles (en supposant que le service ping soit bien activé sur les machines),***
- ***Scan vertical : le scanner est capable de déterminer les ports qui sont ouverts sur une machine donnée à partir de son adresse IP.***

Il vous est demandé de développer par vous-même le scanner réseau, et de ne pas simplement réutiliser des commandes systèmes (comme la commande ping). Attention : veillez à bien réaliser les tests dans un environnement où vous y êtes autorisé (salles TP). Ceux qui avancent vite peuvent implanter plusieurs techniques de balayage de ports.

https://nmap.org/docs.html
https://nmap.org/man/fr/man-port-scanning-techniques.html