Objectifs du TD :

Exercice 1

1 Un client/serveur UDP

Cette partie permet de mettre en place un client serveur en UDP. La partie serveur vous sera fourni. La partie client sera à programmer, l'exercice qui suit permet de se poser les bonnes questions avant de coder.

Exercice 2

Un serveur jouant au « perroquet » tourne sur un hôte. Son code exécutable se trouve sur le Moodle et on peut le lancer sur tout hôte. Il ne fait qu'attendre des messages dans la boîte réseau numéro 31469, sous le protocole UDP, et renvoie le texte reçu à l'expéditeur.
Dans cet exercice vous devrez écrire un client qui expédie un message et attend la réponse qui sera le même message que celui que vous avez envoyé.

1. Quelles sont les informations nécessaires au client en plus des informations données ci-dessus ?
2. Est-on sûr de recevoir une réponse au message envoyé ?
3. Lors d'une réception, est-on sûr de recevoir tout le message ? (préciser si c'est en une seule réception).
4. Noter que le client ne doit pas se contenter d'expédier un seul message, mais doit envoyer en boucle chaque ligne entrée par l'utilisateur, jusqu'à une demande d'arrêt.

Remarque-A préparer pour le TP : les classes `Sock` et `Sockdist` sont à votre disposition sur le moodle au format source et objet. **Mais** vous devez faire le nécessaire pour ne **pas** inclure les `.cc` dans vos programmes. Il est donc demandé de construire un fichier `makefile` qui permettra de fabriquer les exécutables directement avec les `.o` des classes.

2 Communications UDP entre deux applications

Exercice 3

L'objectif de cet exercice est de mettre en évidence toutes les caractéristiques d'une communication en mode *non connecté* : synchronisations, tampons, spécificités des entrées-sorties (blocage), etc.
Soient `MonServeur` et `Monclient` deux applications (deux processus) sur deux hôtes distincts quelconques. Ces hôtes ne sont pas figés dans les applications (nom passé en paramètre), car on prévoit de lancer ces applications en plusieurs exemplaires sur plusieurs hôtes.

Première étape

On commencera par la mise en place d'une communication échangeant un seul message texte (chaque application envoie un message à l'autre).
On peut supposer le schéma algorithmique de base suivant pour `MonServeur` :

```
entier descLocal= allouerBRlocale (nomHôte, numLocal, proto) ;
initialiser BRdistante (nomDistant, numDistant, proto);
initialiser longueurBRdist;
initialiser tamponExp, longueurExp ;
initialiser tamponRec, longueurRec;
expédier (descLocal, tamponExp, longueurExp, 0, BRdistante, longueurBRdist);
recevoir (descLocal, tamponRec, longueurRec, 0, NULL, NULL);
```

On peut supposer un schéma algorithmique semblable pour `Monclient`, en inversant l'ordre d'expédition et réception.

1. Que se passe-t-il si on lance une seule des deux applications ? Jusqu'où se déroule-t-elle sans encombre ?
2. Que se passe-t-il si une application fait un envoi sans que l'autre soit en réception ? Pour illuster la réponse à cette question, il faut compléter et modifier le schéma algorithmique précédent. En effet, il faut s'assurer que l'application censée recevoir soit lancée, tout en étant sûr qu'elle n'est pas en réception. Proposer une solution répondant à ce besoin.
3. Réciproquement, est-ce qu'une application peut être en attente de réception sans qu'un expéditeur n'ait créé la boîte réseau assurant le service d'expédition [1] ?
4. Que se passe-t-il si on se trompe dans l'affectation des numéros des *boîtes réseaux* ?

1. c'est un peu une question piège.

5. Que se passe-t-il si on lance deux applications identiques, pour le même service sur un même hôte ? sur deux hôtes différents ?
6. Est-il possible d'envisager une communication entre une application qui vous appartient et une autre qui ne vous appartient pas ? On envisagera ici les deux cas :
   — l'autre personne lance l'application écrite par vous,
   — l'autre personne lance sa propre application.
   Quels sont les éléments sur lesquels il faut se mettre d'accord afin que deux applications écrites par deux personnes différentes puissent communiquer [2] ?

Deuxième étape

On veut généraliser ces exemples pour mettre en évidence les pertes de paquets alors que les deux applications sont lancées, c'est-à-dire que si on perd des paquets, ce n'est pas parce que le destinataire ou la boîte de destination n'existe pas.

1. Quels exemples peut-on proposer ? On peut proposer ici des exemples qui pourront facilement être mis en œuvre en TP, et d'autres cas possibles, qu'on ne pourra pas mettre en évidence.
2. Quelles sont les modifications à apporter aux programmes afin de mettre en évidence une telle perte ?
3. Est-il possible de créer un exemple d'interblocage des deux applications (au besoin en modifiant les programmes) ?
4. Est-il possible alors de les débloquer par des moyens autres que l'arrêt pur et simple ?

Troisième étape

On veut qu'une seule application ait une adresse de boîte réseau connue (le numéro de service est connu) par l'autre. Par exemple `MonServeur` est lancée avec un numéro connu par `Monclient`, mais cette dernière se fait attribuer un numéro quelconque par le système d'exploitation local. Que proposez-vous pour la mise en place d'un tel dialogue ?

2. est-ce une question piège ?