Objectifs : Vous allez réaliser une simple application communicante et des variantes en utilisant le protocole UDP. L'objectif est d'apprendre à réaliser des échanges entre deux programmes (minimum) et d'observer et commencer à comprendre le fonctionnement d'UDP.

Il va falloir :

1. effectuer une réflexion (sur papier) pour :
   – comprendre la structure de l'application (définir combien de programmes la composent et le rôle de chacun).
   – définir un protocole d'application / d'échange.
2. implémenter les programmes de l'application
3. exécuter les programmes sur différentes machines (impératif)
4. réaliser des tests mettant en évidence quelques propriétés d'UDP.

Notations et rappel :
Le protocole de transport **UDP** permet de réaliser des communications en mode non connecté. Un message envoyé en **UDP** est transféré/acheminé en un seul paquet **UDP**. Enfin, **UDP** ne gère ni la duplication, ni la remise dans l'ordre des paquets à leur réception.

1 Réflexion et développement d'une première application

Ecrire deux programmes :
– un programme émetteur qui envoie un entier N à un programme récepteur dont l'adresse et numéro de port sont à passer en paramètre. N est aussi un paramètre de votre programme.
– un programme récepteur qui reçoit un entier et l'affiche à l'écran.

Exécuter les deux programmes sur deux machines différentes (en utilisant la commande ssh) et assurez vous du bon fonctionnement de votre application avant de passer à la suite.

2 Perte de paquets

Modifier l'application précédente (pensez à faire des sauvegardes avant chaque modification demandée) pour que :
– l'émetteur envoie au programme récepteur des entiers de 1 à N. Après chaque envoi, le programme doit afficher la somme des entiers envoyés depuis le début.
– le récepteur boucle indéfiniment sur la réception d'un entier, l'incrémentation d'une variable *somme* avec cet entier et l'affichage du résultat de cette incrémentation.

Ensuite :

1. Exécuter les deux programmes sur deux machines différentes, en testant avec des petites valeurs de N.
2. Faire des tests en augmentant progressivement N. Qu'observez vous ?
3. Mettez en place deux scénarios permettant de mettre en évidence la perte de paquets.
4. Que feriez vous pour éviter une telle perte ? Ne pas implémenter la réponse.

3 Limites des paquets

Reprendre l'application de la section 1 et modifier les programmes pour que :
– le programme émetteur envoie successivement deux entiers saisis au clavier (il fait donc 2 envois).
– le programme récepteur demande à recevoir une suite d'octets dont la taille est passée en paramètre, et affiche le nombre d'octets reçus.

Ensuite :

1. Exécuter les deux programmes en s'assurant que les deux entiers sont envoyés après que la socket du récepteur soit prête à recevoir des messages et avant de se mettre en réception. Qu'observez vous lorsque le nombre d'octets passé en paramètre du récepteur est 2, 4, 6, 8 et 10 ?

   Pour aller plus loin, modifier le programme émetteur pour qu'il envoie un paquet de grande taille. Cette taille, en nombre d'octets, est à passer en paramètre.

2. Exécuter les deux programmes en faisant évoluer la taille du message à envoyer et à recevoir jusqu'à dépassement de la taille du buffer d'envoi. Que se passe-t-il du coté de l'émetteur ? et du récepteur ?

4 Une autre application : croisement des émissions et échanges entre deux processus et plus

Ecrire un programme qui dans l'ordre :

- envoie une chaine de caractère saisie au clavier vers une socket dont l'adresse est passée en paramètre,
- reçoit une chaîne de caractère de taille maximum 500 octets,
- affiche le message reçu et l'adresse de l'émetteur,
- renvoie le message reçu vers la socket dont l'adresse est passée en paramètre

Votre programme ne doit envoyer que des octets utiles, c'est à dire, les caractères constituant les chaines de caractères saisies, pas plus.

1. Exécuter le programme sur deux machines pour lancer deux processus. Paramétrer ces processus pour faire en sorte qu'ils échangent des messages entre eux et se terminent.
2. Exécuter le programme sur trois machines pour lancer trois processus $P1$, $P2$ et $P3$. Paramétrer ces processus pour faire en sorte que $P1$ envoie les messages à $P2$, $P2$ à $P3$ et $P3$ à $P1$.
3. Que fait cette application ? Que faut-il faire pour que ce comportement puisse se reproduire avec 4 processus et plus ? Ne pas implémenter la réponse.
4. Refaire les deux questions précédentes, mais cette fois, en lançant votre programme sur votre machine et le faire communiquer avec le programme de votre ou vos voisins. Etait-il nécessaire de modifier votre programme ? Quelle est votre conclusion ?