TP9 : messagerie instantanée en UDP

Matthieu Boutier *(boutier@pps.univ-paris-diderot.fr)*

Le but de TP est d'écrire deux logiciels : un client de mini-messagerie, et un serveur. Le serveur communique avec un ou plusieurs clients par UDP. Le port utilisé pour ce protocole (par le serveur) est 4200.

Communication par datagrammes

Une socket UDP IPv4 se crée par `socket(PF_INET, SOCK_DGRAM, 0)`. Elle peut être liée à un numéro de port ou une adresse (avec `bind()`, comme en TCP). Nous utilisons `sendto()` et `recvfrom()` respectivement pour envoyer et recevoir les messages.

1 Le client

Le client est un processus interactif dans lequel l'utilisateur écrit ses messages et lit ceux des autres. Nous utiliseront deux processus pour pouvoir à la fois attendre les messages de l'utilisateur et ceux du serveur.

1. Écrivez une fonction `tty_to_server()` qui lise ce que tape l'utilisateur, et l'envoie au serveur.
2. Écrivez une fonction `server_to_tty()` qui attende un message du serveur, et l'affiche pour l'utilisateur.
3. Écrivez le client complet.

2 Le serveur

2.1 Version simplifiée : le serveur echo

Cette première version du serveur, simplifiée, sert à tester le client. C'est un processus non-interactif. Il commence par créer une socket et faire un `bind()` sur le port 4200 de la machine distante. Puis, en boucle, il attend de recevoir un message par `recvfrom()` et immédiatement renvoie *le même* message, à *la même* adresse par `sendto()`.

2.2 Le serveur de mini-messagerie

Ce serveur peut executer des **commandes**. Une commande est un message commençant par #. La principale fonction du serveur est de maintenir une *liste des utilisateurs* du service. Un utilisateur doit se *connecter* au serveur. Cette connection est au niveau du protocole application, puisque la socket UDP, elle, est sans connection.

La liste des utilisateurs connectés est une liste de triplets (nom, IP, port, nom d'utilisateur). Un nom d'utilisateur est formé de 1 à 8 caractères, sans espace. On peut supposer qu'il y a un nombre fixé d'utilisateurs.

Un message qui n'est pas une commande est renvoyé à tous les utilisateurs, précédé du nom de celui qui l'envoie. Par exemple si l'utilisateur *Fabien* envoie le message "Bonjour !" tous les utilisateurs connectés recevront du serveur le message "<Fabien> Bonjour !". Tous les messages et les commandes se terminent par `\n\0`. La taille d'un message est bornée à 160 caractères.

Le serveur qui reçoit un message autre qu'une commande de la part d'une IP ne correspondant pas à un utilisateur connecté lui renvoie un message d'erreur.

Pour vous aider, vous trouverez dans didel un fichier `serveur.c` où certaines fonctions sont implémentées.

Liste des commandes

- **`#login <nom>`** : demande de connection. Exemple : `#login Fabien` pour se connecter avec le nom Fabien ;
- **`#who`** : le serveur renvoie la liste des utilisateurs connectés ;
- **`#quit`** : demande de déconnection ;
- **`#to <nom> <message>`** : message privé, qui n'est envoyé qu'à un utilisateur. Exemple : "#to Fabien Bonjour, Monsieur" ;
- **`#kick <nom>`** : l'utilisateur *nom* est deconnecté du service ;
- **`#help`** : le serveur renvoie la liste des commandes supportées.

Extensions

Vous êtes libres de rajouter des commandes au protocole, il faut alors les documenter clairement dans le code du serveur ou dans un fichier à part. Vous pouvez implémenter les extensions suivantes :

Commande #dcc

#dcc <nom> <fichier> déclenche un transfert, par une liaison TCP, du fichier local *fichier* vers le disque dur de l'utilisateur *nom*.

Gestion des canaux

Un canal (*channel*) est un ensemble d'utilisateurs connectés.

Au lieu de la commande **#login <nom>**, un utilisateur peut choisir de se connecter avec **#connect <nom>**. Le serveur l'inscrit dans ses listes mais ne lui envoie pas les messages des autres utilisateurs.

Ensuite, grâce à la commande **#join <canal>** l'utilisateur (connecté par connect) déclare qu'il est membre de ce canal. Il recevra alors les messages des utilisateurs du même canal, mais pas ceux des autres utilisateurs. Il se désinscrit avec la commande **#leave <canal>**.

On peut être inscrit à plusieurs canaux simultanément. Un message d'un autre utilisateur ne sera reçu qu'une fois, même si lui et nous sommes dans trois canaux en même temps. La liste des canaux n'est pas connue à l'avance : quand un utilisateur fait un **#join** le serveur regarde s'il a déjà ce canal en mémoire. Si oui il y ajoute l'utilisateur. Si non il le crée. Un canal dont le dernier utilisateur a fait **#leave** est supprimé.