TP de Réseaux – RICM 4
Programmation d'une application d'échange de messages courts en utilisant les sockets

12 avril 2019

1 Introduction

L'objectif est de développer une application d'échange de message court un peu comme le permet l'application bien connue Twitter. Vous utiliserez le langage C et la librairie des Sockets. Une documentation synthétique des Sockets et des fonctions nécessaires au développement de ce programme vous est fournie.

2 Cahier des charges

2.1 Utilisations des programmes

L'utilisateur lance le client en fournissant l'adresse IP (ou le nom associé), le numéro de port du serveur.

Exemple : *flash-client mandelbrot 9999*

Le serveur sera lancé en fournissant un numéro de port d'écoute :

Exemple : *flash-serveur 9999*

2.2 Fonctionnalités

- Le client se connecte au serveur. A la première connexion le client fournit son pseudo sur 6 caractères.
- Le client peut ensuite au choix :
  - s'abonner à un compte Flash en donnant le nom d'un pseudo au serveur. Une fois son abonnement fait le serveur lui donnera par la suite tous les messages envoyés par ce pseudo. Attention ces messages peuvent être ceux envoyés en temps réel depuis un autre client, mais ils peuvent être aussi des messages sauvegardés par le serveur lors d'une ancienne connexion d'un client.
  - se désabonner d'un compte,
  - lister tous les pseudos auxquels il est abonné.
  - publier un message courts de moins de 20 caractères à ses abonnés,
  - quitter l'application.

- Le client au moment de la connexion au serveur reçoit tous les messages publiés par les comptes auxquels il est abonné. Pour chaque message le pseudo de l'émetteur est affiché.
- Le serveur mémorise les messages envoyés par un utilisateur si celui ci fait partie de la liste des abonnés d'au moins un compte. Les messages sont mémorisés jusqu'au que tous les abonnés l'ai reçu.

2.3 Contraintes

- Pour des raisons de fiabilité, les échanges se feront à l'aide du protocole TCP.
- Le serveur doit rester à l'écoute des demandes des différents clients.

3 Conseils de réalisation

Il est assez courant dans les applications de type client/serveur TCP de procéder côté serveur à la mise en place d'une socket d'écoute avec une création de processus (*fork*) lors de chaque connexion d'un nouveau client.

Dans notre cas de figure suivre une telle approche pose le problème de l'échange de donnée (les messages) entre les clients connectés En effet, si chaque nouvelle connexion de client est traitée par un processus différent qui possèdent son propre espace mémoire, il est nécessaire de mettre en place un mécanisme de communication inter-processus (IPC pour Inter Processus Communication).

Une deuxième approche possible est l'utilisation de processus légers (*threads*) pour traiter chaque nouvelle connexion. Les processus légers possèdent la particularité de partager le même espace mémoire. Il s'agit alors de mettre en place une structure de données d'échange gardée par des verrous, des sémaphores ou des moniteurs qui assureront un fonctionnement correct sans interblocage (*deadlock*). L'approche par processus légers est conseillée pour les applications relativement complexes (exemple les serveurs webs).

Finalement la troisième approche consiste à utiliser un seul processus qui va surveiller l'état de l'ensemble des sockets du système et va effectuer les différentes actions nécessaires. Cette dernière voie est connue pour être la la plus réactive et efficace en terme de performance et de charge du système, mais est aussi la plus complexe quand l'application est complexe (grand nombre de cas de figure à traiter, automate d'état, programmation événementielle).

Dans le cas présent, Nous vous conseillons de choisir la troisième approche avec un seul processus coté serveur pour gérer l'ensemble des clients. L'appel système *select* sera utilisé pour gérer l'écoute de l'état des connexions. Cet appel système sera le coeur de votre application (coté serveur et coté client).

Il est particulièrement recommandé de lire attentivement les documentations ainsi que de s'inspirer des exemples qui y sont présentées (par exemple dans les pages de documentation sur un système Linux).

4 Compte-rendu

Vous devrez fournir comme compte rendu :

- Le code source du programme client et serveur fortement commenté.
- Une documentation annexe sur le code si cela vous semble nécessaire.
- Éventuellement les restrictions de vos programmes par rapport au cahier des charges initial.
- Des jeux de tests.
- Éventuellement des commentaires sur les problèmes rencontrés.

5 Références Bibliographiques

1. Les différentes pages de documentation sur un système Linux ou FreeBDS par exemple sur *select*
2. Introduction au C :
   `http://www-clips.imag.fr/commun/bernard.cassagne/Introduction_ANSI_C.html`