TP Programmation réseaux

Noms et adresses mail

- VO Alexis : alexis.vo@etu-upsaclay.fr
- [redacted] Théo : theo.[redacted]@etu-upsaclay.fr

Répartition du travail

Le travail a été effectué en parallèle par les deux membres du binôme pour une répartition égale.

Liste des fichiers

- client_udp.c
- client_tcp.c
- serveur_udp.c
- serveur_tcp.c
- main.c
- makefile

Exécutables

Le makefile permet de générer les différents exécutables suivant:

- launcher
- serveur_tcp
- client_tcp
- client_udp
- serveur_udp

Fonctionnalités

Les fichiers listés ci-dessus permettent de générer les exécutables gérant la communication entre deux processus basés sur le modèle client/serveur, avec les protocoles de communication TCP et UDP.

Launcher

L'exécution du launcher propose à l'utilisateur les choix suivants:

1. le protocole (TCP ou UDP)
2. le rôle (client ou serveur)

3. le mode développeur pour du debugage (seulement disponible avec TCP).

NB : il reste tout à fait possible de compiler et exécuter les codes séparément, sans utiliser le launcher.

Serveur tcp (serveur_tcp.c)

Le programme du serveur tcp est multi threadé:

- un thread de type console qui attend la saisie clavier de l'utilisateur pour répondre à deux commandes:
  - CLEAR qui efface la console
  - SHUTDOWN qui permet d'arrêter le serveur
- un thread correspondant au programme principal qui assure la communication TCP

Le programme s'arrête si captation du signal envoyé par Ctrl+C

Client tcp (client_tcp.c)

Le programme du client tcp tente d'établir une connexion avec un serveur un certain nombre de fois et s'arrête s'il n'y arrive pas. S'il y arrive, l'utilisateur peut envoyer ses messages et se déconnecter ensuite soit en tapant le message "DISCONNECT", soit grâce à Ctrl+C

Serveur udp (serveur_udp.c)

Le programme du serveur udp écoute sur un port donné (par défaut 9600) et affiche les messages reçus. Il suit le schéma du protocole UDP.

Client udp (client_udp.c)

Le programme du client udp fonctionne selon le principe classique du protocole, en suivant le schéma du protocole UDP. Il lit les messages de l'utilisateur et les envoie au serveur.

Exemple

L'exemple suivant illustre une communication TCP en mode développeur. L'utilisateur peut également tester le protocole UDP et les options comme SHUTDOWN ou CLEAR (seulement sur TCP). Le code fonctionne également pour une communication entre deux machines distinctes du même réseau.

1. Ouvrir un terminal dans le répertoire contenant les fichiers à compiler et exécuter
2. Ecrire `make` ; si tout se passe bien, le message suivant s'affiche dans le terminal :

```
gcc -Wall -O2 client_tcp.c -o client_tcp
gcc -Wall -O2 serveur_tcp.c -o serveur_tcp
gcc -Wall -O2 client_udp.c -o client_udp
gcc -Wall -O2 serveur_udp.c -o serveur_udp
gcc -Wall -O2 main.c -o launcher
```

3. Ecrire `./launcher`.

```
=== Choix du protocole ===
1. TCP
2. UDP
Entrez votre choix : 1
```

4. Pour l'exemple, je choisis TCP. Je saisis `1`.

```
=== Choix du rôle ===
1. Client
2. Serveur
Entrez votre choix : 2
```

5. Je suis le serveur, je saisis `2`.

```
Voulez-vous activer le mode développeur ? (o/n) : o
```

6. Pour l'exemple, j'active le mode développeur.

Terminal serveur

```
--------------------------------------------------
-------------------SERVEUR_TCP--------------------
--------------------------------------------------

Serveur en attente sur le port 9600...
```

7. Dans un nouveau terminal, j'effectue les opérations de 1) à 6), mais je saisis `1` pour le client.

Terminal client

```
Tentative de connexion au serveur (essai 1/3)...
Connexion réussie au serveur !

--------------------------------------------------
--------------------CLIENT_TCP--------------------
--------------------------------------------------

Connecté au serveur.
 Saisir les messages ou utiliser Ctrl+C pour quitter.
Mon message :
```

8. Je saisis mon message, le serveur accuse réception :

Terminal client :

```
Mon message : test
```

Terminal serveur :

```
Connexion acceptée depuis 127.0.0.1:55734
Message reçu du client 127.0.0.1:55734 : test
Taille du message reçu : 4 octets.
```

9. Je décide de fermer la connexion client avec `Ctrl+C`:

Terminal client :

```
Message de déconnexion envoyé au serveur. Déconnexion...
Client déconnecté.
```

Terminal serveur :

```
Message reçu du client 127.0.0.1:55734 : DISCONNECT
Taille du message reçu : 10 octets.
Déconnexion du client 127.0.0.1:55734
```

10. La connexion client est fermée. Je peux arrêter le serveur avec ```Ctrl+C```` :

Terminal serveur :

```
Arrêt du serveur suite à Ctrl+C.
```

Conclusion

Ce projet, codé en langage C, nous a permis d'approfondir le cours du module Réseaux et de comprendre les caractéristiques de ces deux protocoles TCP-UDP.

Nous avons appris a créer et gérer des sockets, manipuler des adresses IP, et mettre en œuvre des fonctions clés comme socket(), bind(), connect(), send(), et recv() comme décrites dans le schéma du TP.

On retiendra que TCP est un protocole fiable qui garantit l'ordre des messages ; tandis que UDP est non fiable et privilégie la rapidité.

Une partie conséquente du temps de travail a été consacrée aux gestions des erreurs et des bugs afin de garantir un code robuste.

Enfin, une des idées de poursuite de ce projet serait la mise en œuvre d'un système de messagerie instantannée, éventuellement en garantissant la confidentialité des données.