

Rapport du Projet de Réseaux : Gestion de comptes bancaires

Étudiants : Ayoub LADJICI

Spécialité : Électronique – Informatique

Enseignants référents : Maria POTOP-BUTUCARU - Francesca FOSSATI

Table des matières

1. Introduction.....	3
2. Architecture Client-Serveur TCP	3
1. Fonctionnement du protocole TCP.....	3
2. Explication du code TCP.....	3
3. Tests et validation du protocole TCP	5
3. Architecture Client/Serveur UDP	7
1. Fonctionnement du protocole UDP.....	7
2. Explication du code UDP.....	7
3. Tests et validation du protocole UDP	8

1. Introduction

Dans le cadre du module Réseaux, nous sommes amenés à réaliser un projet dont le but est d'implémenter une application de gestion de comptes bancaires en utilisant une architecture client-serveur avec les protocoles TCP et UDP.

Le serveur stocke plusieurs comptes et permet aux clients de se connecter pour effectuer diverses opérations telles que l'ajout et le retrait d'une certaine somme d'argent, la consultation du solde et l'affichage des 10 dernières opérations effectuées sur un compte.

Pour mener ce projet, j'ai utilisé plusieurs ressources notamment les codes de ce tutoriel sur les sockets <https://broux.developpez.com/articles/c/sockets/> permettant la création d'une architecture client/serveur de type chat, le cours sur les sockets <https://www.csd.uoc.gr/%7Ehy556/material/tutorials/cs556-3rd-tutorial.pdf> et d'autres sites comme Stack Overflow pour bien comprendre le fonctionnement de l'architecture.

2. Architecture Client-Serveur TCP

1. Fonctionnement du protocole TCP

Le protocole TCP (Transmission Control Protocol) permet une communication fiable entre deux entités d'un réseau. Une connexion entre le client et le serveur est nécessaire avant de démarrer l'échange des données. Ensuite, les données sont découpées en plusieurs segments plus petits et numérotés puis envoyés au destinataire. A l'arrivée, les segments sont réassemblés dans le bon ordre. TCP vérifie également le flux de données pour s'assurer que l'expéditeur n'envoie pas plus de données que le récepteur ne peut en traiter. Donc, TCP garantit que les données échangées entre le serveur et le client sont correctement envoyées, reçues et ordonnées.

2. Explication du code TCP

Tout d'abord, le code client permet d'établir une connexion fiable avec le serveur, d'envoyer des commandes et de recevoir des réponses du serveur. Ce dernier suit les mêmes étapes que sur la Figure 2.2.1. En effet, la première étape pour établir une communication entre 2 programmes est de créer un socket (prise de communication). Cela se fait à l'aide de la fonction `socket()` :

`socket(domain = AF_INET, type = SOCK_STREAM, protocol = 0)`

`AF_INET` indique que le socket utilisera la famille de protocole IPv4. `SOCK_STREAM` garantit une connexion fiable et ordonnée et le dernier paramètre laisse le système d'exploitation choisir automatiquement le protocole approprié, en l'occurrence `IPPROTO_TCP`. Une fois le socket créé, le client utilise une structure `SOCKADDR_IN` pour définir les informations nécessaires à

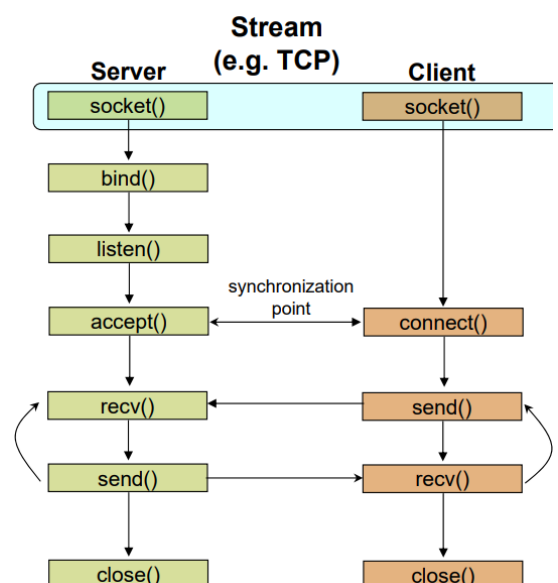


Figure 2.2.1 : Schéma illustrant le processus de connexion TCP entre un serveur et un client

la connexion, notamment l'adresse IP du serveur et le numéro de port. Ensuite, le client tente de se connecter au serveur avec la fonction `connect()`. Si cette étape est réussie, le client pourra envoyer son nom au serveur et là on entre dans `while(1)` qui constitue la boucle principale de communication. On a une structure `fd_set` qui se nomme `rfds` et qui permet de stocker le descripteur de notre socket et le descripteur du clavier `STDIN_FILENO` afin que la fonction `select()` puisse surveiller les deux sources d'entrée/sortie et arrêter le programme lorsqu'un événement se produit comme écrire dans le terminal et taper sur Entrée pour envoyer au serveur avec `write_server()` (où `send()` est utilisé pour envoyer le buffer) ou bien lire les réponses du serveur avec `read_server()` (où `recv()` est utilisé pour lire le buffer en provenance du serveur) et les afficher avec `puts()` sur le terminal dans notre cas. Enfin, lorsque le client veut se déconnecter, il doit appuyer sur Ctrl+C, la connexion est fermée grâce à `end_connection()`.

Ensuite, le code serveur est responsable de la gestion des connexions clients, des comptes bancaires et du traitement des requêtes. Il suit les étapes comme décrit dans la Figure 2.2.1. Comme pour le client, la première étape du serveur consiste à créer un socket avec les mêmes paramètres et à le lier via la fonction `bind()` avec une structure `STOCKADDR` où sont renseignés des informations comme le numéro de port du serveur et est défini avec `INADDR_ANY` pour accepter les connexions de n'importe quelle adresse IP. Le socket est ensuite mis en mode écoute pour gérer les connexions entrantes avec `listen()` et définir également la taille maximale de la file d'attente des connexions entrantes. Puis on entre dans la boucle `while(1)` où le serveur va gérer les nouvelles connexions et traiter en permanence les requêtes en provenance des clients connectés. On va initialiser le descripteur du clavier, le socket du serveur pour accepter les nouveaux clients et les sockets des clients connectés. La fonction `select()` bloque l'exécution jusqu'à ce qu'un événement se produise sur l'un des descripteurs surveillés. Si on écrit sur le serveur, il va immédiatement s'arrêter. Lorsque le socket du serveur est prêt, une connexion client est acceptée avec `accept()` et sera ajouté à la liste des clients connectés. Ensuite, un message de bienvenue et une liste des commandes disponibles sont envoyés au client via `write_client()`. Chaque client est surveillé individuellement, ainsi si un client se déconnecte, le socket correspondant est fermé avec `closesocket()` et retiré de la liste des clients connectés. Si le client écrit et envoie une requête, elle sera lue par le serveur à l'aide de `read_client()` puis analysée et traitée par une fonction que j'ai créée `traiter_commande()` qui permet d'interpréter la commande du client (AJOUT, RETRAIT, SOLDE, OPERATIONS) et lui envoyer une réponse à propos de l'état de la requête.

Pour la gestion des comptes bancaires, elle repose sur une structure que j'ai créée dans `server.h` permettant de stocker plusieurs informations comme le nom d'utilisateur (`id_client`), le numéro de compte bancaire (`id_compte`), le mot de passe (`password`), le montant actuel du compte (`solde`), le nombre d'opérations effectuées (`nb_operations`) et l'historique des 10 dernières opérations dans un tableau de structure `Operation` (`Operation operations[10]`). Chaque opération est représentée par la structure `Operation` qui contient les champs comme le type d'opération, la date de l'opération au format "YYYY-MM-DD HH:MM:SS" et le montant de l'opération.

Dans le code, lorsque le client se connecte, le serveur lui rappelle les opérations possibles et les informations nécessaires pour que l'opération puisse être effectué avec succès. En guise de réponse aux commandes du client, le serveur enverra des réponses appropriées comme OK (succès de l'AJOUT ou du RETRAIT), KO (échec de l'AJOUT ou du RETRAIT), RES_SOLDE (succès de l'affichage du solde) et RES_OPERATIONS (succès de l'affichage des 10 dernières opérations du compte).

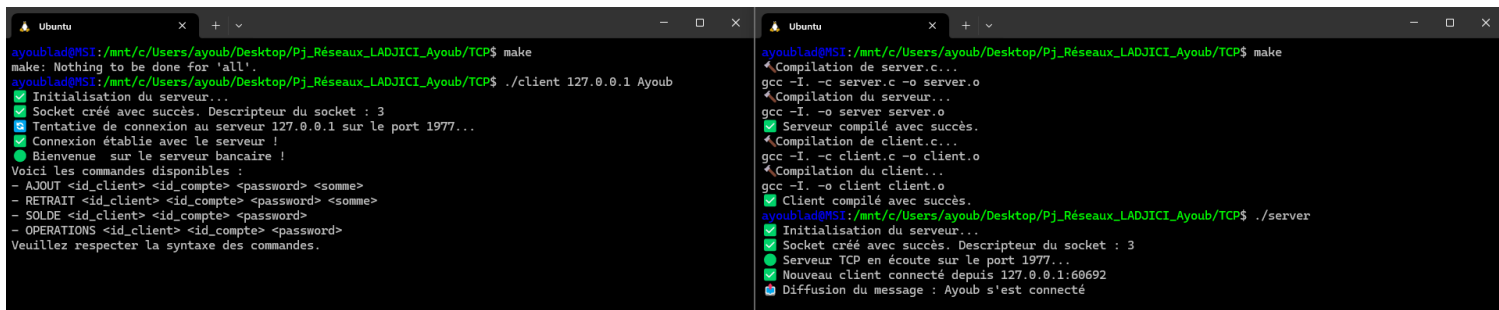
3. Tests et validation du protocole TCP

Pour tous mes tests, le client se trouve dans la fenêtre de gauche et le serveur dans la fenêtre de droite.

On lance d'abord le serveur en premier pour que le client puisse se connecter.

1^{er} test : TENTATIVE DE CONNEXION

On voit que le serveur affiche un message indiquant une nouvelle connexion avec le nom du client. Quant à ce dernier, il reçoit bien un message de bienvenue et une liste des commandes qu'il peut exécuter pour gérer son compte bancaire.

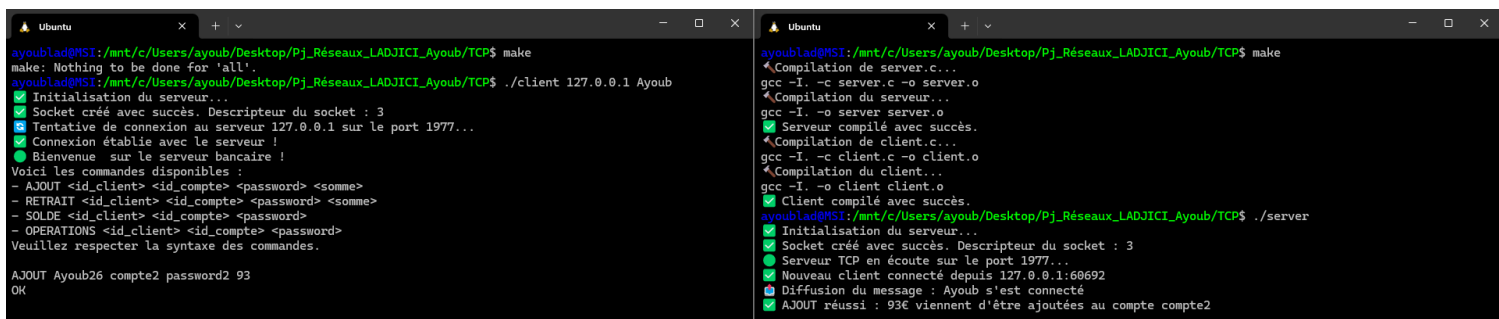


```
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
✗ Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
✓ Connexion établie avec le serveur !
● Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
✗ Compilation de server.c...
gcc -I. -c server.c -o server.o
✗ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✗ Compilation de client.c...
gcc -I. -c client.c -o client.o
✗ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Serveur TCP en écoute sur le port 1977...
✓ Nouveau client connecté depuis 127.0.0.1:60692
✗ Diffusion du message : Ayoub s'est connecté
```

2^{ème} test : AJOUT D'UN MONTANT

Le client envoie une commande en respectant bien la syntaxe imposée par le serveur pour réussir l'opération. Si c'est le cas le client reçoit le message OK et quant au serveur, il indique bien que le montant a été ajouté dans le compte.



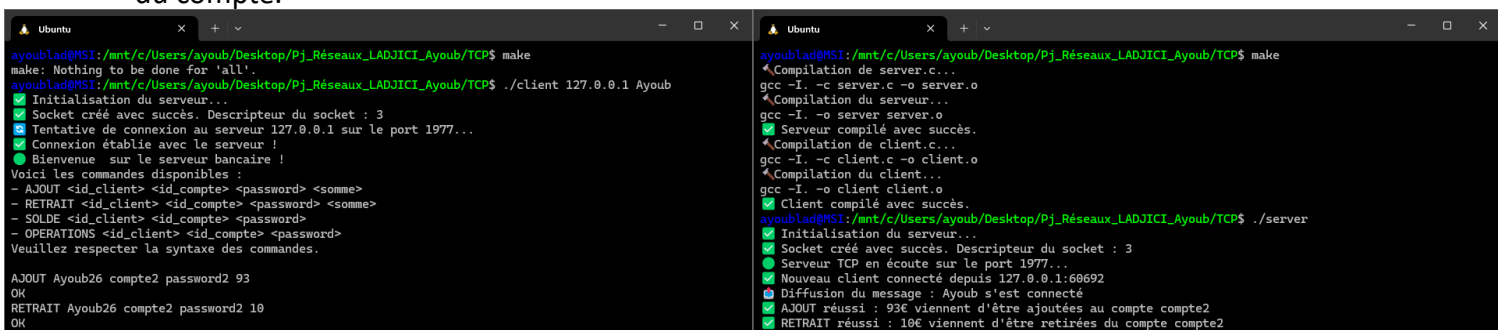
```
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
✗ Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
✓ Connexion établie avec le serveur !
● Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 93
OK

ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
✗ Compilation de server.c...
gcc -I. -c server.c -o server.o
✗ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✗ Compilation de client.c...
gcc -I. -c client.c -o client.o
✗ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Serveur TCP en écoute sur le port 1977...
✓ Nouveau client connecté depuis 127.0.0.1:60692
✗ Diffusion du message : Ayoub s'est connecté
✓ AJOUT réussi : 93€ viennent d'être ajoutées au compte compte2
```

3^{ème} test : RETRAIT D'UN MONTANT

Même chose que le 2^{ème} test sauf qu'ici, le serveur confirme bien que de l'argent a été retiré du compte.



```
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
✗ Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
✓ Connexion établie avec le serveur !
● Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 93
OK
RETRAIT Ayoub26 compte2 password2 10
OK

ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
✗ Compilation de server.c...
gcc -I. -c server.c -o server.o
✗ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✗ Compilation de client.c...
gcc -I. -c client.c -o client.o
✗ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Serveur TCP en écoute sur le port 1977...
✓ Nouveau client connecté depuis 127.0.0.1:60692
✗ Diffusion du message : Ayoub s'est connecté
✓ AJOUT réussi : 93€ viennent d'être ajoutées au compte compte2
✓ RETRAIT réussi : 10€ viennent d'être retirées du compte compte2
```

4^{ème} test : AFFICHAGE DU SOLDE

Lorsque le client demande le solde actuel de son compte, il reçoit un message RES_SOLDE de la part du serveur avec bien évidemment le montant du solde et la date de la dernière opération, en l'occurrence l'opération de retrait de 10€.

```
Ubuntu x + v
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
[✓] Initialisation du serveur...
[✓] Socket créé avec succès. Descripteur du socket : 3
[✓] Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
[✓] Connexion établie avec le serveur !
[✓] Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 93
OK
RETRAIT Ayoub26 compte2 password2 10
OK

SOLDE Ayoub26 compte2 password2
RES_SOLDE Votre solde est de 100083€. Dernière opération : 31/12/2024 17:05:50

Ubuntu x + v
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
[✓] Compilation de server.c...
gcc -I. -c server.c -o server.o
[✓] Compilation du serveur...
gcc -I. -o server server.o
[✓] Serveur compilé avec succès.
[✓] Compilation de client.c...
gcc -I. -c client.c -o client.o
[✓] Compilation du client...
gcc -I. -o client client.o
[✓] Client compilé avec succès.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
[✓] Initialisation du serveur...
[✓] Socket créé avec succès. Descripteur du socket : 3
[✓] Serveur TCP en écoute sur le port 1977...
[✓] Nouveau client connecté depuis 127.0.0.1:60692
[✓] Diffusion du message : Ayoub s'est connecté
[✓] AJOUT réussi : 93€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 10€ viennent d'être retirées du compte compte2
[✓] SOLDE : Ayoub26, votre solde est de 100083€
```

5^{ème} test : AFFICHAGE DES 10 DERNIERES OPERATIONS DU COMPTE

Depuis que je me suis connecté au serveur, j'ai réalisé exactement 11 opérations (AJOUT et RETRAIT combinée), et on constate qu'elle ne prend pas en compte l'ajout de 93€ dans le compte, qui constitue l'opération la plus ancienne, respectivement la 11^{ème} dernière opération donc le client reçoit bien les 10 dernières opérations du compte.

```
Ubuntu x + v
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
[✓] Initialisation du serveur...
[✓] Socket créé avec succès. Descripteur du socket : 3
[✓] Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
[✓] Connexion établie avec le serveur !
[✓] Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 93
OK
RETRAIT Ayoub26 compte2 password2 10
OK

SOLDE Ayoub26 compte2 password2
RES_SOLDE Votre solde est de 100083€. Dernière opération : 31/12/2024 17:05:50

AJOUT Ayoub26 compte2 password2 1000
OK
RETRAIT Ayoub26 compte2 password2 7
OK

AJOUT Ayoub26 compte2 password2 213
OK
RETRAIT Ayoub26 compte2 password2 26
OK

AJOUT Ayoub26 compte2 password2 10
OK
RETRAIT Ayoub26 compte2 password2 9999
OK

AJOUT Ayoub26 compte2 password2 35
OK
RETRAIT Ayoub26 compte2 password2 99
OK

AJOUT Ayoub26 compte2 password2 17
OK
OPERATIONS Ayoub26 compte2 password2
RES_OPERATIONS
RETRAIT 31/12/2024 17:05:50 -10€
AJOUT 31/12/2024 17:08:55 1000€
RETRAIT 31/12/2024 17:09:03 -7€
AJOUT 31/12/2024 17:09:17 213€
RETRAIT 31/12/2024 17:09:26 -36€
AJOUT 31/12/2024 17:09:34 10€
RETRAIT 31/12/2024 17:09:48 -9999€
AJOUT 31/12/2024 17:10:02 35€
RETRAIT 31/12/2024 17:10:09 -99€
AJOUT 31/12/2024 17:10:30 17€

Ubuntu x + v
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
[✓] Compilation de server.c...
gcc -I. -c server.c -o server.o
[✓] Compilation du serveur...
gcc -I. -o server server.o
[✓] Serveur compilé avec succès.
[✓] Compilation de client.c...
gcc -I. -c client.c -o client.o
[✓] Compilation du client...
gcc -I. -o client client.o
[✓] Client compilé avec succès.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
[✓] Initialisation du serveur...
[✓] Socket créé avec succès. Descripteur du socket : 3
[✓] Serveur TCP en écoute sur le port 1977...
[✓] Nouveau client connecté depuis 127.0.0.1:60692
[✓] Diffusion du message : Ayoub s'est connecté
[✓] AJOUT réussi : 93€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 10€ viennent d'être retirées du compte compte2
[✓] SOLDE : Ayoub26, votre solde est de 100083€
[✓] AJOUT réussi : 1000€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 7€ viennent d'être retirées du compte compte2
[✓] AJOUT réussi : 213€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 26€ viennent d'être retirées du compte compte2
[✓] AJOUT réussi : 10€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 9999€ viennent d'être retirées du compte compte2
[✓] AJOUT réussi : 35€ viennent d'être ajoutées au compte compte2
[✓] RETRAIT réussi : 99€ viennent d'être retirées du compte compte2
[✓] AJOUT réussi : 17€ viennent d'être ajoutées au compte compte2
[✓] OPERATIONS envoyées pour le compte compte2
```

6^{ème} test : DECONNEXION D'UN CLIENT DU SERVEUR

Lorsque le client tape Ctrl+C pour se déconnecter, on voit bien que le serveur affiche un message de déconnexion en indiquant le nom du client, qui sera immédiatement retiré de la liste des clients connectés.

```
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
make: Nothing to be done for 'all'.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./client 127.0.0.1 Ayoub
Initialisation du serveur...
Socket créé avec succès. Descripteur du socket : 3
Tentative de connexion au serveur 127.0.0.1 sur le port 1977...
Connexion établie avec le serveur !
Bienvenue sur le serveur bancaire !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.
AJOUT Ayoub26 compte2 password2 93
OK
RETRAIT Ayoub26 compte2 password2 18
OK
SOLDE Ayoub26 compte2 password2
RES_SOLDE Votre solde est de 100083€. Dernière opération : 31/12/2024 17:05:50
AJOUT Ayoub26 compte2 password2 1000
OK
RETRAIT Ayoub26 compte2 password2 7
OK
AJOUT Ayoub26 compte2 password2 213
OK
RETRAIT Ayoub26 compte2 password2 26
OK
AJOUT Ayoub26 compte2 password2 10
OK
RETRAIT Ayoub26 compte2 password2 9999
OK
AJOUT Ayoub26 compte2 password2 35
OK
RETRAIT Ayoub26 compte2 password2 99
OK
AJOUT Ayoub26 compte2 password2 17
OK
OPERATIONS Ayoub26 compte2 password2
RES_OPERATIONS
RETRAIT 31/12/2024 17:05:50 -10€
AJOUT 31/12/2024 17:08:55 1000€
RETRAIT 31/12/2024 17:09:03 -7€
AJOUT 31/12/2024 17:09:10 213€
RETRAIT 31/12/2024 17:09:26 -26€
AJOUT 31/12/2024 17:09:34 10€
RETRAIT 31/12/2024 17:09:40 -9999€
AJOUT 31/12/2024 17:10:02 35€
RETRAIT 31/12/2024 17:10:09 -99€
AJOUT 31/12/2024 17:10:30 17€
^C
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$

ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ make
Compilation de serveur.c...
gcc -I. -c serveur.c -o serveur.o
Compilation du serveur...
gcc -I. -o serveur serveur.o
Serveur compilé avec succès.
Compilation de client.c...
gcc -I. -c client.c -o client.o
Compilation du client...
gcc -I. -o client client.o
Client compilé avec succès.
ayoublad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/TCP$ ./server
Initialisation du serveur...
Socket créé avec succès. Descripteur du socket : 3
Serveur TCP en écoute sur le port 1977...
Nouveau client connecté depuis 127.0.0.1:60692
Diffusion du message : Ayoub s'est connecté
AJOUT réussi : 93€ viennent d'être ajoutées au compte compte2
RETRAIT réussi : 10€ viennent d'être retirées du compte compte2
SOLDE : Ayoub26, votre solde est de 100083€
AJOUT réussi : 1000€ viennent d'être ajoutées au compte compte2
RETRAIT réussi : 7€ viennent d'être retirées du compte compte2
AJOUT réussi : 213€ viennent d'être ajoutées au compte compte2
RETRAIT réussi : 26€ viennent d'être retirées du compte compte2
AJOUT réussi : 10€ viennent d'être ajoutées au compte compte2
RETRAIT réussi : 9999€ viennent d'être retirées du compte compte2
AJOUT réussi : 35€ viennent d'être ajoutées au compte compte2
RETRAIT réussi : 99€ viennent d'être retirées du compte compte2
AJOUT réussi : 17€ viennent d'être ajoutées au compte compte2
OPERATIONS envoyées pour le compte compte2
Client déconnecté : Ayoub
Diffusion du message : Ayoub disconnected !
```

3. Architecture Client/Serveur UDP

1. Fonctionnement du protocole UDP

Le protocole UDP (User Datagram Protocol) est un protocole de communication utilisé pour l'envoi de paquets de données sur un réseau IP. Contrairement au TCP, UDP ne garantit pas ni l'ordre des paquets, ni la livraison des paquets et ne nécessite pas l'établissement d'une connexion entre le client et le serveur. Il est souvent utilisé pour des applications comme le streaming vidéo ou les jeux en ligne. L'en-tête UDP est plus petit que celui de TCP. Donc, UDP est idéal pour des transmissions rapides et légères, mais il n'offre pas les garanties de livraison des paquets.

2. Explication du code UDP

Tout d'abord, le fonctionnement du code client en UDP est sensiblement le même que la version TCP. Cependant, une différence réside dans l'absence de phase de connexion au serveur. On précise bien lors de la création de notre socket que le type doit être `SOCK_DGRAM` pour permettre une transmission sans connexion et sans garantie de la livraison du message. Une fois l'initialisation terminée, le client peut envoyer des messages avec la fonction `write_client()` (qui utilise `sendto()` pour envoyer le buffer au serveur) et lire des messages avec la fonction `read_client()` (où `recvfrom()` est utilisé pour lire le buffer en provenance du serveur).

De même, pour le code serveur en UDP, le fonctionnement est également similaire à la version TCP. On commence par l'initialisation d'un socket UDP qui permet d'écouter les messages du client. Contrairement au protocole TCP, il n'y a pas d'étape d'acceptation de connexions. Le serveur reste en

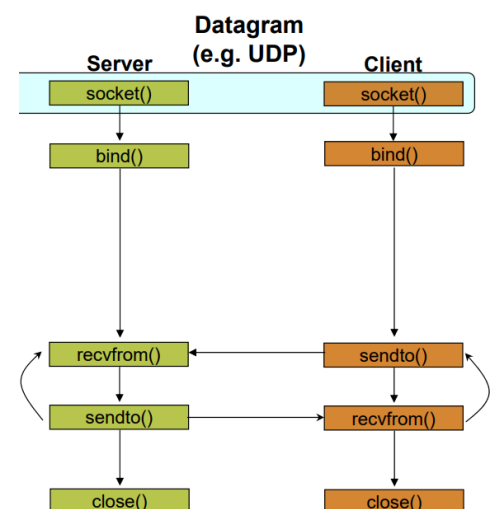


Figure 3.2.1 : Schéma illustrant le processus de connexion UDP entre un serveur et un client

attente de messages grâce à la fonction `recvfrom()` qui permet de récupérer les données ainsi que les informations du client (adresse IP et port). Bien évidemment, les descripteurs du clavier (`STDIN_FILENO`) et du socket du serveur sont surveillés via la structure `rfds`. La fonction `select()` vérifie si quelqu'un écrit sur le serveur ou bien si un client veut nous parler. Lorsqu'on reçoit un message d'un client, on vérifie s'il est enregistré dans notre structure Client avec la fonction `check_if_client_exists()`, si ce n'est pas le cas, on l'ajoute à la liste des clients connectés et il reçoit un message de bienvenue. Sinon, s'il est déjà enregistré, sa requête est analysée par la fonction `traiter_commande()`.

3. Tests et validation du protocole UDP

Sur les captures d'écrans suivantes, l'affichage est le même que pour le protocole TCP.

1^{er} test : TENTATIVE DE CONNEXION

```
Ubuntu x + v - □ ×
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
make: Nothing to be done for 'all'.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Bienvenue Ayoub sur le serveur bancaire UDP !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
✓ Compilation de server.c...
gcc -I. -c server.c -o server.o
✓ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✓ Compilation de client.c...
gcc -I. -c client.c -o client.o
✓ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./server
✓ Initialisation du serveur...
✓ Serveur UDP en écoute sur le port 1977...
✓ Nouveau client ajouté : Ayoub
```

2^{ème} test : AJOUT D'UN MONTANT

```
Ubuntu x + v - □ ×
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
make: Nothing to be done for 'all'.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Bienvenue Ayoub sur le serveur bancaire UDP !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 26000
OK

ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
✓ Compilation de server.c...
gcc -I. -c server.c -o server.o
✓ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✓ Compilation de client.c...
gcc -I. -c client.c -o client.o
✓ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./server
✓ Initialisation du serveur...
✓ Serveur UDP en écoute sur le port 1977...
✓ Nouveau client ajouté : Ayoub
✓ AJOUT réussi : 26000€ viennent d'être ajoutées au compte compte2
```

3^{ème} test : RETRAIT D'UN MONTANT

```
Ubuntu x + v - □ ×
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
make: Nothing to be done for 'all'.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Bienvenue Ayoub sur le serveur bancaire UDP !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 26000
OK
RETRAIT Ayoub26 compte2 password2 13000
OK

ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
✓ Compilation de server.c...
gcc -I. -c server.c -o server.o
✓ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✓ Compilation de client.c...
gcc -I. -c client.c -o client.o
✓ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./server
✓ Initialisation du serveur...
✓ Serveur UDP en écoute sur le port 1977...
✓ Nouveau client ajouté : Ayoub
✓ AJOUT réussi : 26000€ viennent d'être ajoutées au compte compte2
✓ RETRAIT réussi : 13000€ viennent d'être retirées du compte compte2
```

4^{ème} test : AFFICHAGE DU SOLDE

```
Ubuntu x + v - □ ×
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
make: Nothing to be done for 'all'.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./client 127.0.0.1 Ayoub
✓ Initialisation du serveur...
✓ Socket créé avec succès. Descripteur du socket : 3
● Bienvenue Ayoub sur le serveur bancaire UDP !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 26000
OK
RETRAIT Ayoub26 compte2 password2 13000
OK

SOLDE Ayoub26 compte2 password2
RES_SOLDE Votre solde est de 113000€. Dernière opération : 31/12/2024 20:48:59

ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
✓ Compilation de server.c...
gcc -I. -c server.c -o server.o
✓ Compilation du serveur...
gcc -I. -o server server.o
✓ Serveur compilé avec succès.
✓ Compilation de client.c...
gcc -I. -c client.c -o client.o
✓ Compilation du client...
gcc -I. -o client client.o
✓ Client compilé avec succès.
ayoub@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./server
✓ Initialisation du serveur...
✓ Serveur UDP en écoute sur le port 1977...
✓ Nouveau client ajouté : Ayoub
✓ AJOUT réussi : 26000€ viennent d'être ajoutées au compte compte2
✓ RETRAIT réussi : 13000€ viennent d'être retirées du compte compte2
✓ SOLDE : Ayoub26, votre solde est de 113000€
```


5^{ème} test : AFFICHAGE DES 10 DERNIERES OPERATIONS DU COMPTE

```
Ubuntu
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
make: Nothing to be done for 'all'.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./client 127.0.0.1 Ayoub
[+] Initialisation du serveur...
[+] Socket créé avec succès. Descripteur du socket : 3
[+] Bienvenue Ayoub sur le serveur bancaire UDP !
Voici les commandes disponibles :
- AJOUT <id_client> <id_compte> <password> <somme>
- RETRAIT <id_client> <id_compte> <password> <somme>
- SOLDE <id_client> <id_compte> <password>
- OPERATIONS <id_client> <id_compte> <password>
Veuillez respecter la syntaxe des commandes.

AJOUT Ayoub26 compte2 password2 26000
OK
RETRAIT Ayoub26 compte2 password2 13000
OK

SOLDE Ayoub26 compte2 password2
RES_SOLDE Votre solde est de 113000€. Dernière opération : 31/12/2024 21:12:22

AJOUT Ayoub26 compte2 password2 1000
OK
RETRAIT Ayoub26 compte2 password2 14000
OK

AJOUT Ayoub26 compte2 password2 293
OK
RETRAIT Ayoub26 compte2 password2 5000
OK

AJOUT Ayoub26 compte2 password2 20000
OK
RETRAIT Ayoub26 compte2 password2 10000
OK

AJOUT Ayoub26 compte2 password2 3000
OK
RETRAIT Ayoub26 compte2 password2 100
OK

AJOUT Ayoub26 compte2 password2 800
OK
OPERATIONS Ayoub26 compte2 password2
RES_OPERATIONS
RETRAIT 31/12/2024 21:12:22 -13000€
AJOUT 31/12/2024 21:13:28 1000€
RETRAIT 31/12/2024 21:13:41 -14000€
AJOUT 31/12/2024 21:13:59 293€
RETRAIT 31/12/2024 21:14:40 -5000€
AJOUT 31/12/2024 21:14:54 20000€
RETRAIT 31/12/2024 21:15:04 -10000€
AJOUT 31/12/2024 21:15:11 3000€
RETRAIT 31/12/2024 21:15:19 -100€
AJOUT 31/12/2024 21:15:31 800€

Ubuntu
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ make
*Compilation de server.c...
gcc -I. -c server.c -o server.o
*Compilation du serveur...
gcc -I. -o server server.o
[+] Serveur compilé avec succès.
*Compilation de client.c...
gcc -I. -c client.c -o client.o
*Compilation du client...
gcc -I. -o client client.o
[+] Client compilé avec succès.
ayoubblad@MSI:/mnt/c/Users/ayoub/Desktop/Pj_Réseaux_LADJICI_Ayoub/UDP$ ./server
[+] Initialisation du serveur...
[+] Serveur UDP en écoute sur le port 1977...
[+] Nouveau client ajouté : Ayoub
[+] AJOUT réussi : 26000€ viennent d'être ajoutées au compte compte2
[+] RETRAIT réussi : 13000€ viennent d'être retirées du compte compte2
[+] SOLDE : Ayoub26, votre solde est de 113000€
[+] AJOUT réussi : 1000€ viennent d'être ajoutées au compte compte2
[+] RETRAIT réussi : 14000€ viennent d'être retirées du compte compte2
[+] AJOUT réussi : 293€ viennent d'être ajoutées au compte compte2
[+] RETRAIT réussi : 5000€ viennent d'être retirées du compte compte2
[+] AJOUT réussi : 20000€ viennent d'être ajoutées au compte compte2
[+] RETRAIT réussi : 10000€ viennent d'être retirées du compte compte2
[+] AJOUT réussi : 3000€ viennent d'être ajoutées au compte compte2
[+] RETRAIT réussi : 100€ viennent d'être retirées du compte compte2
[+] AJOUT réussi : 800€ viennent d'être ajoutées au compte compte2
[+] OPERATIONS envoyées pour le compte compte2
```