Rapport Explicatif : Gestion Bancaire Client-Serveur

TOURE SEKOUBA
Département Electronique-Informatique

January 8, 2025

Établissement : Polytech Sorbonne

Encadrant : POTOP-BUTUCARU Maria

Date : January 8, 2025

Ce rapport détaille les aspects techniques du projet de gestion bancaire basé sur une architecture client-serveur en C.

Introduction

Ce rapport présente un projet de gestion de comptes bancaires implémenté en langage C en utilisant une architecture client-serveur. Le projet utilise des sockets pour la communication réseau et supporte deux protocoles : TCP et UDP. L'objectif est de permettre aux clients d'effectuer diverses opérations bancaires telles que l'ajout, le retrait, la consultation du solde, et la visualisation des dernières opérations.

1. Serveur

Le serveur gère plusieurs comptes bancaires et traite les commandes des clients. Il prend en charge deux protocoles :

- **TCP** : Protocol orienté connexion qui garantit la fiabilité.
- **UDP** : Protocol sans connexion qui offre une meilleure performance dans des scénarios où la fiabilité n'est pas critique.

Les principales commandes supportées sont :

- **AJOUT** : Ajouter de l'argent à un compte.
- **RETRAIT** : Retirer de l'argent d'un compte si le solde est suffisant.
- **SOLDE** : Consulter le solde d'un compte.
- **OPERATIONS** : Afficher les 10 dernières opérations effectuées sur un compte.

2. Client

Le client est une interface permettant à un utilisateur d'envoyer des commandes au serveur et de recevoir des réponses. Deux versions du client sont disponibles :

- **Client TCP** : Communique avec le serveur en utilisant une connexion fiable.
- **Client UDP** : Envoie des requêtes sans connexion, ce qui peut être plus rapide mais moins fiable.

Le client permet d'envoyer des commandes telles que `AJOUT`, `RETRAIT`, `SOLDE`, et `OPERATIONS` au serveur pour gérer les comptes bancaires.

3. Comparaison entre TCP et UDP

TCP et UDP présentent des différences fondamentales dans leur fonctionnement :

- **TCP** :
  - Orienté connexion : garantit que les messages arrivent dans l'ordre et sans perte.

  - Plus lent en raison de la surcharge liée à la gestion de connexion et aux acquittements.
  - Idéal pour des applications nécessitant une fiabilité élevée (ex. transferts bancaires).

- **UDP** :
  - Sans connexion : envoie les messages directement sans garantie de livraison.
  - Plus rapide grâce à une surcharge minimale.
  - Convient pour des scénarios où la rapidité est prioritaire (ex. notifications en temps réel).

4. Démonstration

Voici une capture d'écran illustrant les interactions entre un client et le serveur en utilisant les deux protocoles : TCP et UDP.

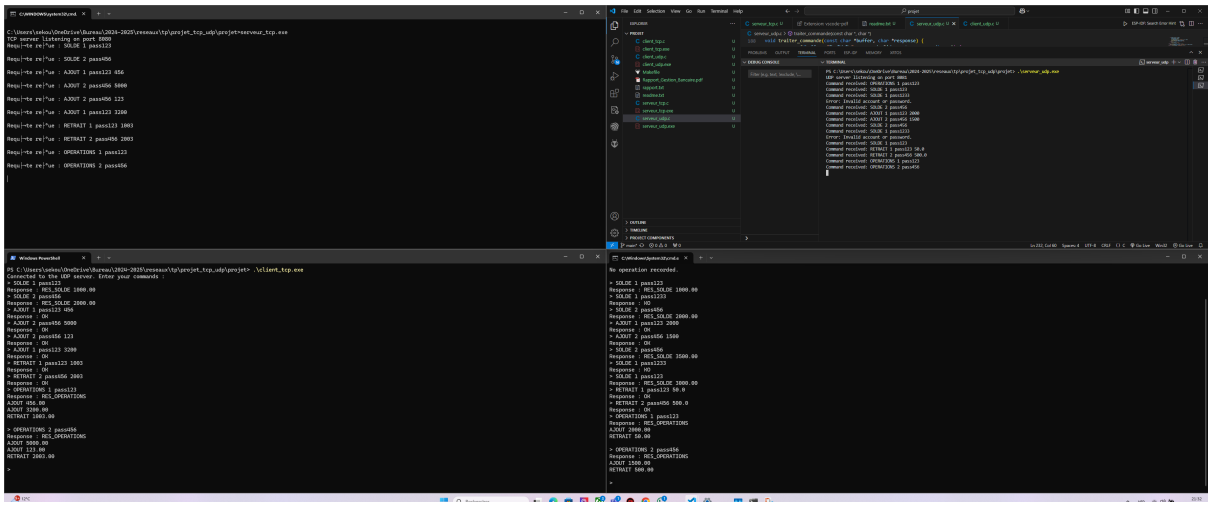

Figure 1: Capture d'écran des interactions TCP et UDP

5. Conclusion

Le projet démontre les principes fondamentaux des architectures client-serveur et l'utilisation des protocoles TCP et UDP. Les tests montrent que :

- TCP garantit une communication fiable et ordonnée.
- UDP est plus rapide, mais peut perdre des messages dans des conditions réseau difficiles.

En conclusion, chaque protocole a ses avantages, et le choix dépend des exigences spécifiques de l'application.