Documentație - Aplicație "Nu te supăra frate!"

Manea Matei-Constantin

14 martie 2025

1 Introducere

Proiectul constă în realizarea unei aplicații client-server care permite mai multor jucători (maximum 4) să joace clasicul joc "Nu te supăra frate!" utilizând o arhitectură bazată pe socket-uri TCP. Aplicația este capabilă să gestioneze mai mulți clienți simultan.

Obiective

- Permite jucătorilor conectarea și participarea la joc în timp real.
- Implementarea tuturor funcționalităților din jocul "Nu te supăra frate".
- Asigurarea comunicării stabile într-un mediu concurent folosind multiplexarea cu `select()`.

2 Tehnologii Aplicate

2.1 Limbajul C++

Limbajul C++ este utilizat datorită eficienței sale și controlului oferit asupra resurselor. Este folosit pentru implementarea serverului și clientului, precum și pentru gestionarea comunicării.

2.2 Protocolul TCP

TCP asigură o transmisie sigură și fiabilă a datelor între server și client. Este utilizat pentru a trimite și primi mesaje fără pierderi.

2.3 Multiplexarea cu `select`

`select` permite gestionarea mai multor conexiuni simultane într-un mod eficient. Serverul poate monitoriza socket-uri multiple pentru activități de citire.

3 Structura Aplicației

3.1 Diagrama

Diagrama de mai jos prezintă structura generală a aplicației client-server.

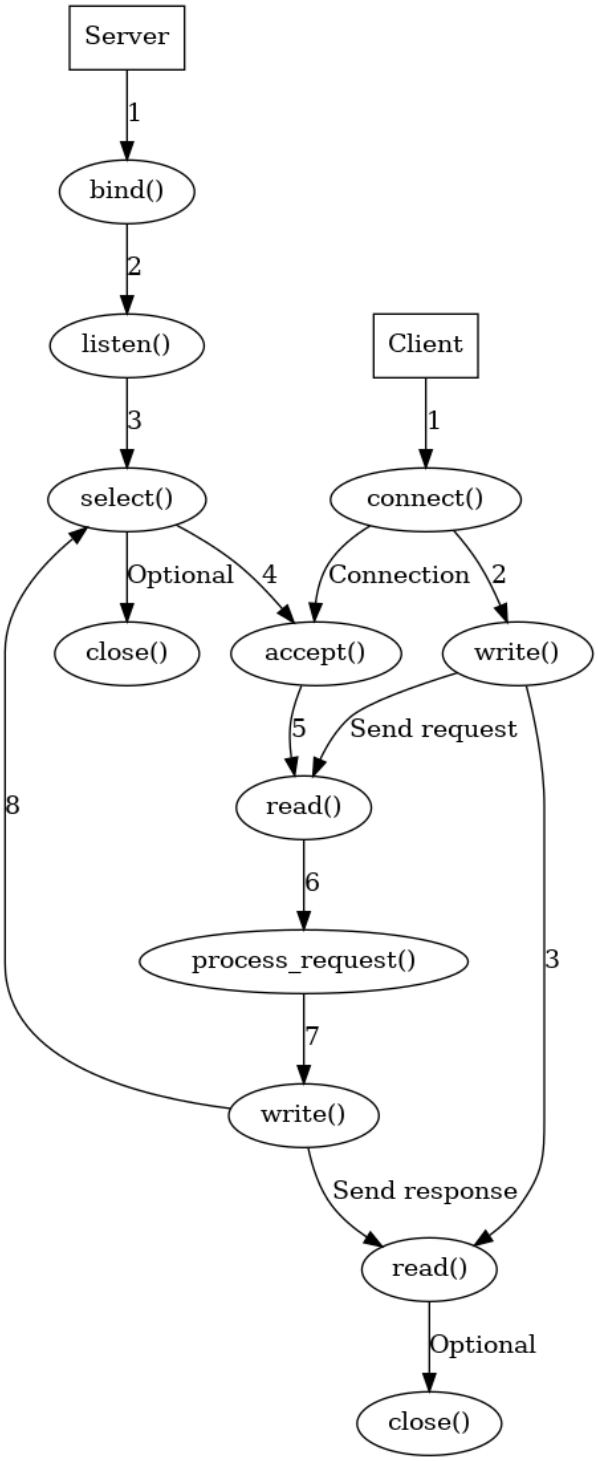

Figura 1: Diagramă ce reprezintă structura generală a aplicației.

3.2 Funcționalități

1. **Conectare:** Clienții se conectează la server folosind socket-uri TCP.
2. **Start Joc:** Serverul începe o nouă sesiune de joc, inițializând și alocând fiecărui jucător câte 4 pioni și o poziție de start. Trebuie ca măcar doi clienți să fie conectați pentru a putea începe jocul.
3. **Gestionare concurentă:** Serverul poate deservi până la 4 jucători simultan, gestionând în mod corect conexiunile active.
4. **Roll:** La tura jucătorului, acesta dă cu zarul, primind de la server valoarea dată. În continuare, clientul alege un pion din cei activi, urmând ca acesta să fie mutat. Imediat după, jocul avansează la următorul client. În cazul în care jucătorul nu are nicio mișcare posibilă, serverul transmite un mesaj sugestiv și trece la următorul jucător.
5. **Quit:** Clientul se deconectează de la server și primește un mesaj de confirmare a comenzii. Consecința acestei comenzi este faptul că toți clienții, și serverul, se deconectează.
6. **Victory:** Atunci când un jucător are toți pionii în poziții câștigătoare, serverul va anunța toți jucătorii de victoria acestuia, primul jucător având ca posibilitate începerea unui nou joc.

3.3 Structura codului

- **Server:**
  - Acceptă conexiuni noi și gestionează fiecare client folosind multiplexarea cu `select()`.
  - Monitorizează socket-urile active pentru a detecta mesajele primite de la clienți.
  - Analizează mesajele primite de la client, le procesează și utilizează logica implementată pentru a avansa starea jocului.
  - În funcție de starea jocului, trimite înapoi un mesaj clientului, printr-un "handshake".
- **Client:**
  - Procesează mesajele primite din terminal și le trimite către server.
  - Primește handshake-urile de la server și le procesează, afișând un mesaj corespunzător jucătorului.

4 Aspecte de implementare

4.1 Comunicarea Client-Server

Serverul ascultă pe un port predefinit (8080) și gestionează conexiuni multiple folosind `select()`. Fiecare client este identificat prin descriptorul socket-ului asociat, iar mesajele sunt gestionate pe baza acestuia.

Funcții Cheie

- **Server**
  - `send_client()`: Trimite clientului un mesaj specific, în funcție de starea jocului și posibilitățile sale.
  - `read_client()`: Citește mesajele primite de la client și apelează `handle_command()`, trimițând mesajul procesat.
  - `handle_command()`: Creierul logicii aplicației. Parcurge mesajele primite de la client, aplicând logica specifică mesajului și stării jocului.
  - `check_victory()`: Verifică cazul în care un jucător are toți pionii în poziții câștigătoare, urmând astfel să anunțe restul programului de victoria unui client.
  - `no_valid_moves()`: Verifică dacă clientul curent poate efectua orice mișcare, iar în caz contrar returnează 1.
  - `print_game_state()`: După fiecare rundă, afișează statusul jocului (pozițiile fiecărui pion).
- **Client**
  - Se conectează la server utilizând un socket TCP.
  - Trimite mesaje citite de la tastatură folosind `write()`.
  - Primește și interpretează mesajele de la server.

4.2 Gestionarea Conexiunilor

Serverul utilizează `select()` pentru a monitoriza mai multe conexiuni simultan. Clienții deconectați sunt eliminați din lista de socket-uri active pentru a evita problemele de resurse.

5 Concluzii

Această aplicație implementează arhicunoscutul joc „Nu te supăra frate”, demonstrând utilizarea eficientă a arhitecturii client-server într-un mediu concurent.

Posibile îmbunătățiri

- Adăugarea unei interfețe grafice pentru client.
- Implementarea unui sistem de salvare a stării jocului.
- Implementarea unui sistem de login și de salvare a scorurilor.
- Posibilitatea de a încărca sesiuni salvate.

Bibliografie

- https://edu.info.uaic.ro/computer-networks/cursullaboratorul.php
- https://sites.google.com/view/fii-retele-de-calculatoare/labs?authuser=0
- https://www.ibm.com/docs/en/zos/2.5.0?topic=calls-select
- https://www.ibm.com/docs/en/i/7.3?topic=designs-example-nonblocking-io-select