Documentație ChesServ

Begu Maria-Florina (2E2)

Universitatea "Alexandru Ioan Cuza", Facultatea de Informatică, Iași

1 Introducere

Șahul reprezintă un joc complex, de strategie care se desfășoară între două persoane. Fiecare participant are la început un set, alb sau negru, format din piese cu diferite atribute, fiecare având un tipar al mișcărilor. Dintre acestea, cel mai important este regele, deoarece el poate decide învingătorul : jucătorul care reușește să captureze regele oponentului câștigă.

Proiectul pe care l-am ales constă în realizarea unei aplicații server care pune la dispoziție o tablă de joc și supervizează desfășurarea fiecărei partide de șah, acționând ca un punct central la care clienții din rețea se conectează. Serverul determină momentul în care jocul s-a terminat și anunță câștigătorul.[1]

Am ales acest proiect deoarece șahul reprezintă una dintre pasiunile mele, iar implementarea aplicației mă poate ajuta să îmi consolidez cunoștințele dobândite până în prezent într-o manieră plăcută. De asmenenea, implementarea unui server este imposibilă fără înțelegerea conceptelor de bază prezentate la această materie, precum și asimilarea unor noi informații, ca urmare a rezolvării problemelor apărute pe parcurs. Pe lângă aceste motive, se adaugă și faptul că jocul de șah este o activitate de care multe persoane se bucură în timpul liber, având puterea să antreneze creierul, dar și să consolideze legăturile interumane.

În următoarele secțiuni ale raportului, vor fi prezentate tehnologiile utilizate, arhitectura aplicației, câteva detalii de implementare, precum concluzii care vizează modul în care realizarea acestui proiect poate fi îmbunătățită.

2 Tehnologii utilizate

Jocul de șah constituie o aplicație la nivelul căreia trebuie respectate anumite reguli de bază. Printre ele, se numără respectarea ordinii mutării pieselor și păstrarea tuturor acțiunilor jucătorilor. Pentru ca aceste condiții să fie îndeplinite, trebuie să ne asigurăm că tehnologia aleasă transmite informațiile dorite în ordinea introducerii lor și că nu există șanse ca acestea să se piardă sau să fie comunicate parțial. Astfel, se poate observa că cea mai bună alegere este protocolul TCP (Transmission Control Protocol).Acesta este recunoscut pesntru faptul că este un protocol orientat conexiune, fără pierdere de informație, care controlează fluxul de date.[2]

În primul rând, este important de menționat faptul că acest protocol realizează conexiunea dintre client și server.[2] Deoarece fiecare mișcare trebuie

trimisă de la client la server, iar după ce este evaluată, transmisă înapoi, aplicația nu ar putea fi realizată fără existența acestei legături.

În al doilea rând, controlul fluxului constituie un aspect de importanță majoră în crearea aplicației. Spre deosebire de UDP, unde ordinea informațiilor nu e respectată, TCP oferă servicii de calități maxime care să se asigure că șirul de mutări este transmis exact, în ordinea în care au fost înregistrate.[2]

Nu în ultimul rând, capacitatea de a detecta erori și de a retransmite datele are un rol important în buna desfășurare a jocului.[2] În această manieră, dacă o acțiune a unui jucător nu este transmisă serverului în totalitate, se știe că părțile pierdute sunt detectate, iar informațiile retransmise.

3 Arhitectura aplicației

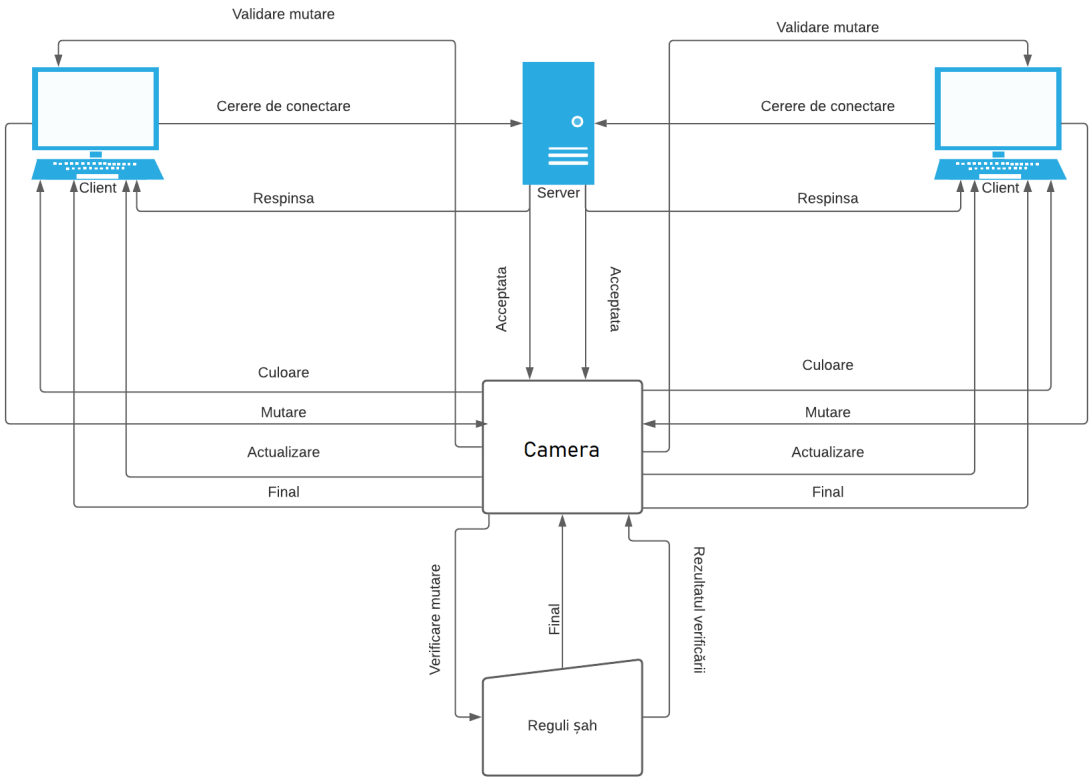

Fig. 1. Diagrama aplicației
[3]

Conceptele implicate la nivelul acestei aplicații sunt:

- **Protocol** : regulile și convențiile prin care se realizează comunicarea; [4]
- **TCP** : protocol de transport orientat conexiune; [2]
- **Modelul client/server** [5]
  - **Proces server**
    - Oferă servicii în rețea;
    - Acceptă cereri de la proces client;
    - Realizează un anumit serviciu și returnează rezultatul;
  - **Proces client**
    - Inițializează comunicarea cu serverul;
    - Solicită un serviciu, apoi așteaptă răspunsul serverului;
- **Server concurent**: cererile sunt procesate concurent; [5]
- **Socket**: permite comunicarea între procese aflate pe mașini diferite, în aceeași rețea; [5]
- **Server TCP concurent, prethreaded, cu blocare pentru protecția *accept()*** : creează un număr de *thread*-uri care sunt gata să servească clienții. Se folosește mecanismul de blocare (*mutex lock*) a apelului primitivei *accept()* și doar un singur thread, la un moment dat, va apela *accept()*; [6]

4 Detalii de implementare

4.1 Rules

În cadrul fișierului *"rules.c"* au fost scrise diferite funcții de verificare care asigură corectitudinea jocului. Acesta conține:

- **possible_road** : se verifică dacă piesa are voie să fie mutată în direcția indicată (dacă drumul îi este caracteristic); [7]
- **free_road** : se verifică dacă drumul până la destinație e liber; [7]
- **free_dest** : se verifică dacă la destinație nu se află o piesă de aceeași culoare; [7]
- **check** : se verifică dacă regele este in pericol;
- **move_king** : se verifică dacă regele se poate muta pentru a ieși din șah; [7]
- **block** : se verifică dacă piesa care atacă regele poate fi blocată; [7]
- **eliminate** : se verifică dacă piesa care atacă regele poate fi eliminată; [7]
- **checkmate** : se verifica dacă este șah mat; [7]

4.2 Board

În cadrul fișierului *"board.c"* au fost scrise diferite funcții care implică tabla de joc, precum :

- **begin_board** : este realizată tabla inițiala de joc; [7]
- **update_board** : este actualizată tabla de șah după o mutare validă;
- **print_board** : este afișată tabla de șah;

4.3 Server

Vom folosi un server de tipul TCP concurent, prethreaded, cu blocare pentru protecția accept care funcționează în modul următor: se așteaptă conectarea a câte doi jucători, jucători care sunt puși în aceeași cameră de joc. Primului client i se atribuie piesele albe, iar celui de-al doilea, piesele negre, urmând ca mai apoi, jocul să înceapă. Fiecare jucător va face o mutare când îi este rândul, mutare primită de server pentru a fi validată.Serverul va valida sau invalida mutarea, acțiune care determină trecerea la tura următoare sau reintroducerea unei mutări valide.La final, serverul trimite câte un cod, in funcție de situația în care se află, iar aceștia se vor deconecta. Dacă un jucător, cand îi este rândul, dorește să se deconecteze, serverul va trimite un mesaj celuilalt jucător, iar ambii vor fi deconectați. Deoarece lucrăm cu un server concurent, vor exista multiple astfel de camere de joc, independente una de cealaltă.

În fișierul *"server.c"* sunt prezente următoarele funcții care contribuie la o bună desfășurare a jocului:

- **culoare** : transmite fiecărui jucător culoarea pieselor;
- **ran** : transmite fiecărui jucător dacă este rândul său sau nu (100-da; 101-nu);
- **read_move** : citește mutarea jucătorului curent; în cazul în care acesta dorește să se deconecteze, această funcție trimite un mesaj adversarului și returnează o valoare care ne va informa daca este cazul sa deconectăm jucătorii;
- **validare_mutare** : verifică dacă mutarea este validă și transmite ambilor jucători un cod (102-mutare validă ; 103-mutare invalidă);
- **send_move** : este transmisă mutarea validă ambilor jucători;
- **finish_game** : verifică dacă jocul s-a terminat și trimite clienților câte un cod în funcție de situație (w-câștigat ; l-pierdut) ;
- **treat** : se acceptă câte doi jucători care sunt asociați câte unui thread și este supervizat jocul de șah;
- **create_thread** : se creează câte un thread;
- **connection** : se creează mulțimea de thread-uri pe care le putem folosi pentru desfășurarea jocului, se atașează un socket, se așteaptă venirea clienților și se servesc în mod concurent clienții;
- **main** : este apelată funcția connection;

Notă: În realizarea acestui server, a fost utilizat ca baza modelul : *servTcpPreTh.c* din Laboratorul 12.[8]

4.4 Client

În cadrul acestui proiect, este utilizat un client simplu, fără interfață grafică, care comunica prin mesaje afișate în terminal cu jucătorii.

În fișierul *"client.c"* sunt prezente următoarele funcții :

- **play** : sunt trimise și primite coduri de la server, în funcție de care sunt afișate mesaje jucătorului, în timpul jocului;

– **main** : este creat socket-ul, se realizează conexiunea la server, este aflată culoarea pieselor, e apelată funcția play și se închide conexiunea;

Notă: În realizarea acestui client, a fost utilizat ca baza modelul : *cliTcpNr.c* din Laboratorul 12.[9]

5 Concluzii

Pe lângă aspectele menționate pot fi aduse o serie de modificări care să îmbunătățească experiența jucătorilor. În primul rând ar trebui realizată deconectarea unui jucător, după un anumit timp de inactivitate, precum și deconectarea unui client, dacă dorește să părăsească jocul, chiar dacă acesta se află în așteptare.

De asemenea, am putea utiliza baze de date și să fie creată o opțiune login care să permită unui jucător să își acceseze propria pagină. Pe aceasta să poate fi găsită o evidență a rezultatelor obținute la meciurile anterioare, o listă de prieteni pe care îi poate provoca și cu care poate comunica prin intermediul unui text chat, dacă aceștia sunt conectați, o listă de concursuri la care poate participa. Totodată, o idee interesantă ar fi existența unei opțiuni care să găsească pentru utilizator un oponent cu rezultate asemănătoare cu ale sale, pe care să îl poată provoca la o partidă de șah.

Bibliografie

1. Rețele de calculatoare - Proiecte propuse
   https://profs.info.uaic.ro/ computernetworks/ProiecteNet2020.php
2. Rețele de calculatoare - Nivelul Transport
   https://profs.info.uaic.ro/ computernetworks/files/4rc_NivelulTransport_Ro.pdf
3. Lucid
   https://lucid.app
4. Rețele de calculatoare - Arhitecturi de rețea
   https://profs.info.uaic.ro/ computernetworks/files/2rc_ArhitecturiDeRetea_Ro.pdf
5. Rețele de calculatoare - Programarea în rețea
   https://profs.info.uaic.ro/ computernetworks/files/5rc_ProgramareaInReteaI_ro.pdf
6. Rețele de calculatoare - Programarea în rețea
   https://profs.info.uaic.ro/ computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
7. Chess Console Game in C++
   https://www.codeproject.com/Articles/1214018/Chess-Console-Game-in-Cplusplus
8. Rețele de calculatoare - Curs, Laborator
   https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerPreThread/servTcpPreTh.c
9. Rețele de calculatoare - Curs, Laborator
   https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerPreThread/cliTcpNr.c