Retele de calculatoare - Connect4 (B)
Raport tehnic

Ghimp Sergiu

December 10, 2019

Universitatea "Alexandru Ioan Cuza" din Iasi,
Facultatea de Informatica,
Cursant, Grupa X4,
sergiu.ghimp@info.uaic.ro

Abstract

In aceasta lucrare este prezentat modul de functionare a aplicatiei Connect4, de la felul in care serverul gestioneaza clientii pana la felul in care este afisata interfata textuala de catre client. Se regasesc si detalii despre modul de tratare a situatiilor speciale, precum deconectarea unui client in timpul jocului sau netrimiterea in timp util a unui raspuns catre server de unul din cei doi jucatori.

Keywords: joc, connect4, jucator, online

1 Introducere

Proiectul reprezinta o implementare a clasicului joc Connect4 si are ca scop oferirea unei pauze relaxante pentru una sau mai multe partide de joc. Clientii vor avea acces la o aplicatie, disponibila din terminalul sistemelor de operare din familia UNIX/Linux, care ii va plasa intr-o partida unul contra celalalt intr-un mediu online.

2 Tehnologiile Utilizate

2.1 TCP

Interactiunea dintre client si server se bazeaza pe o o conexiune TCP deschisa de la momentul initializarii conexiunii pana la finalul acesteia. Deoarece proiectul este de tip joc, avem nevoie de un nivel de incredere ridicat a faptului ca miscarile jucatorilor ajung cu succes la server si ca serverul transmite cu succes starea jocului catre jucatori iar TCP ne ofera acest nivel de siguranta.

3 Arhitectura aplicatiei

3.1 Server

3.1.1

Serverul asteapta pana la conectarea unei perechi de clienti iar apoi verifica daca numarul de jucatori deja conectati nu depaseste o anumita limita. In cazul in care numarul de utilizatori conectati depaseste deja aceasta limita, serverul trimite un mesaj de refuzare a conexiunii catre cei doi jucatori iar apoi trece in starea de asteptare a unei noi perechi de clienti. In cazul contrar, cand numarul de utilizatori conectati nu depaseste aceasta limita, un nou thread este creat al carui scop este interactionarea cu cei doi clienti.

3.1.2

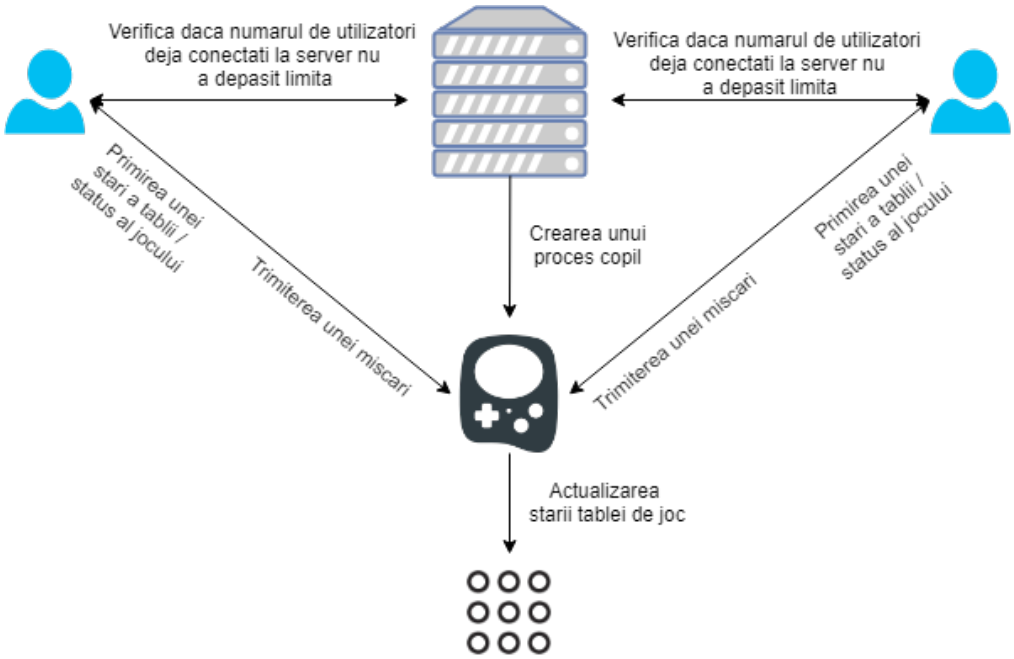
Thread-ul nou creat va alege aleator jucatorul care incepe jocul si va transmite acest lucru. Acesta va astepta apoi miscarile celor doi utilizatori, pe rand, verificand daca aceastea sunt valide si daca unul din cei doi jucatori a castigat transmitand acestora noua stare a tablii de joc. In momentul in care unul dintre cei doi jucatori castiga, procesul copil va anunta acest lucru si va lasa posibilitatea unei rejucari a partidei in cazul in care ambii utilizatori sunt de acord, tinand de asemenea si scorul.

3.2 Client

3.2.1

La pornirea clientului se primeste fie un mesaj conform caruia jocul este gata sa inceapa sau un mesaj conform caruia se asteapta un al doilea jucator. Cand jocul este gata sa inceapa, se primeste un mesaj din care utilizatorul afla daca el incepe jocul sau nu. Se trimit apoi miscarile dorite in fiecara tura catre server, starea tablii de joc actualizandu-se in functie de raspunsurile utilizatorului si de raspunsurile adversarului. La finalul unei partide utilizatorul are optiunea de a accepta sau de a refuza o rejucare a partidei contra adversarul curent.

3.3 Diagrama aplicatiei



4 Detalii de implementare

4.1 Cod relevant particular proiectului

4.1.1 Setarea unui timp limita de asteptare la read pe socket

In cazul in care unul dintre jucatori nu trimite miscarea dorita in timp util, putem folosi acest cod pentru a seta un timp limita de asteptare a unei functii de input pe socket.

```
timeout.tv_sec=timeoutSec;
timeout.tv_usec=0;
setsockopt(currentPlayerSd, SOL_SOCKET, SO_RCVTIMEO, (char*) &timeout,
sizeof(move));
```

4.1.2 Folosirea select() pentru asteptarea inputului de la utilizator un numar limitat de secunde

Utilizatorul are la dispozitie 30 de secunde pentru a introduce o miscare valida, acest lucru se realizeaza in cod cu ajutorul functiei select().

```
FD_ZERO(&inputSet);
FD_SET(0,&inputSet);
```

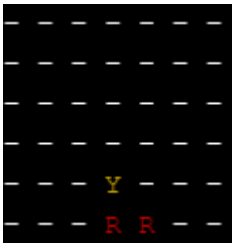
```
timeout.tv_sec=timeRemainingForMove;
timeout.tv_usec=0;

printd(" nter  a move (%d seconds remaining): ",timeRemainingForMove);
fflush(stdout);
r=select(1,&inputSet,NULL,NULL,&timeout);

if(r == -1){
    fprintf(stderr,"Error at select call! [%d:%s]\n",errno,strerror(errno));
    close(serverSd);
    exit(EXIST_FAILURE);
}
else if(r){
    scanf("%d",&move)
    }
else{
    moveStatus = -1;
    break;
}
printf("You chose to add a '%c' disk on coloumn %d! \n",playerColor,move);
```

4.1.3 Starea tablii si modul in care arata in terminal

Codul folosit pentru a printa starea tablii de joc pe parte de client si modul in care va arata starea tablii de joc in terminal




```
...
#define RED "\x1B[31m"
#define YEL "\x1B[33m"
#define RESET "\x1B[0m"
...
for (int i = 0; i < 6; ++i) {
    for (int j = 0; j < 7; ++j) {
        if (gameBoard[i][j] == 'R') {
            printf(RED "R " RESET);
        } else if (gameBoard[i][j] == 'Y') {
```

```
            printf(YEL "Y " RESET);
        } else {
            printf("- ");
        }
    }

    printf("\n");
}
```

4.1.4 Structurile folosite in schimbul de informatii dintre client si server

Structurile care descriu miscarea pe care utilizatorul a ales sa o faca, starea tablii.

```
struct Move {
  int chosenColumn;
}

struct GameStatus {
  char gameBoard[6][7];
  int gameResultFlag; // -1 daca clientul care primeste
  // structura a pierdut, 0 daca jocul nu s-a decis inca,
  // 1 daca clientul care primeste structura a castigat
}
```

4.2 Schimbul de mesaje dintre client si server

4.2.1 Server

Codul pe partea de server

```
printf("[%ld] Announcing players the new game board state...\n",thId);
if(write(playerOneSd,gameBoard,sizeof(GameBoard)) == -1 ){
    fprintf(strerr,"[%ld] Error writing player one the game board state
    with random generated move![%d:%s]\n",thId,errno,strerror(errno));
    return -1;
}
if(white(playerTwoSd,gameBoard,sizeof(GameBoard) == -1)){
    fprintf(strerr,"[%ld] Error writing player two the game board state
    with random generated move![%d:%s]\n",thId,errno,strerror(errno));
    return -1;")
}
```

4.2.2 Client

Codul pe partea de client

```
if ((r==read(serverSd,gameBoard,sizeof(GameBoard))) == -1){
    fprintf(strerr,"Error reading the game board state after local player
    move! [%d:%s ] \n",errno,strerror(erno));
    close(serverSd);
    exit(EXIT_FAILURE);
}
else if (r==0){
    printf("Your opponent disconnect from the game! \n");
    close(serverSd);
    return 0;
}
    printf("Board state successfully read! New board state:\n");
    printBoardState(gameBoard);
```

4.3 Scenarii de utilizare

4.3.1 Unul dintre cei doi jucatori se deconecteaza in timpul jocului

In cazul in care unul dintre cei doi jucatori se deconecteaza, serverul il anunta pe jucatorul inca conectat, acesta primind un mesaj de instiintare conform caruia oponentul s-a deconectat.

4.3.2 Unul dintre cei doi jucatori nu trimite o miscare in timp rezonabil

Daca unul dintre cei doi jucatori nu trimite o miscare in 30 de secunde de la primerea starii noii table de joc, serverul il va anunta ca i-a expirat timpul si va alege o miscare valida random in locul lui.

5 Concluzii

Cateva din ideile prin care solutia propusa ar putea fi imbunatatita sunt regasite in subsectiunile de mai jos.

5.1 Implementarea unor moduri de joc diferite de cel clasic

Se poate implementa un meniu din care utilizatorul isi poate alege modul de joc dorit: clasic, pop out (jucatorii isi pot elimina discurile puse de catre ei de pe ultimul rand), pop 10, 5-in-a-Row (2 coloane adaugate tablii de joc iar conditia pentru victorie este de a conecta 5 piese) et. al.

5.2 Punerea la dispozitie a unei interfete grafice

In solutia propusa, clientul se foloseste de o interfata textuala. O imbunatatire a clientului ar fi inlocuirea interfetei textuale cu o interfata grafica.

5.3 Oferirea unei posibilitati de a iti selecta adversarul

Solutia propusa creaza o partida de joc cu doi utilizatori conectati unul dupa altul in succesiune la server. In cazul in care un utilizator si-ar dori sa joace o partida cu un prieten de al sau, acest lucru ar putea fi problematic in cazul in care sunt multi clienti incercand sa se conecteze la server simultan. O solutie pentru aceasta problema ar fi implementarea unei interfete de unde utilizatorul isi poate alege adversarul, sau daca nu are o preferinta atunci un adversar aleator.

5.4 Implementarea unui sistem de conturi

Pentru oferirea unor statistici interesante precum numarul de victorii, numarul de infrangeri, rata de castig et. al. s-ar putea implementa un sistem de conturi si salvat aceste date intr-o baza de date.

6 Bibliografie

- `https://en.wikipedia.org/wiki/Connect_Four`
- `https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php`
- `https://linux.die.net/man`