Connect 4

Duca Alexandru 2E1

Universitatea “Alexandru Ioan Cuza” Iasi
Facultatea de Informatica Iasi

1 Introducere

1.1 Connect 4

Simulati jocul Connect Four intr-o aplicatie client/server. Fiecare client se conecteaza la server unde acesta va stabili jucatorul care face prima mutare si culoare cu care va juca fiecare. Serverul are rolul de a tine scorul, jocul putand fi jucat pe mai multe reprise, precum si rolul de a afisa fiecarui jucator gridul fiecarui jucator, dupa ce se efectueaza o mutare.

1.2 Motivatia alegerii

Am ales acest proiect intrucat am auzit de el in trecut, dar nu am avut curiozitatea sa ma interesez cum se joaca. Dupa ce am consultat resursele suplimentare, jocul in sine mi s-a parut interactiv si distractiv de jucat impreuna cu un prieten si am decis sa incerc sa il realizez.

2 Tehnologii utilizate

Pentru a realiza jocul, am folosit un server concurent TCP si o aplicatie client, folosind fork(). Prin intermediul acestora, ne asiguram ca jocul poate fi jucat de catre doi jucatori, iar apoi asteapta alta pereche de jucatori daca mai doresc si altii sa joace. Fara acest fork(), jocul ar fi fost jucat doar de doi jucatori unici, astfel nu ar putea fi jucat de mai multe perechi in mod individual. Dupa ce se conecteaza doi jucatori, acestia sunt grupati si isi desfasoara jocul lor, fara a putea vedea si jocul altei perechi de juatori. Gruparea are loc in procesul copil din fork().

Utilizand metoda TCP, computerul care trimite datele se conecteaza direct la computerul catre care le trimite si ramane conectat pe durata transferului. Cu aceasta metoda, cele doua computere pot garanta ca datele au ajuns in siguranta si corect, apoi inchid conectiunea. Aceasta metoda tinde sa fie mai rapida si fiabila, dar pune o presiune mai mare pe computer, deoarece trebuie sa monitorizeze conexiunea si datele care trec.

Utilizand metoda UDP, calculatorul trimite pachetele de date informatiile in bucati mai mici si le elibereaza cu speranta ca va ajunge la locul potrivit. Cee ace inseamna

ca UDP nu se conecteaza direct la calculatorul expeditor si computerul destinatar pentru a obtine datele acolo unde se presupune ca merge in mod corespunzator. Aceasta metodade transmitere nu ofera nicio garantie ca datele pe care le trimiteti vor ajunge vreodata la destinatie.

Am ales TCP intrucat vrem ca ambii jucatori sa se conecteze la server, iar informatiile sa ajunga in mod intreg la fiecare.

Chiar daca TCP este relative mai incet decat UDP, preferam siguranta datelor peste viteza de transmitere.

3 Arhitectura aplicatiei

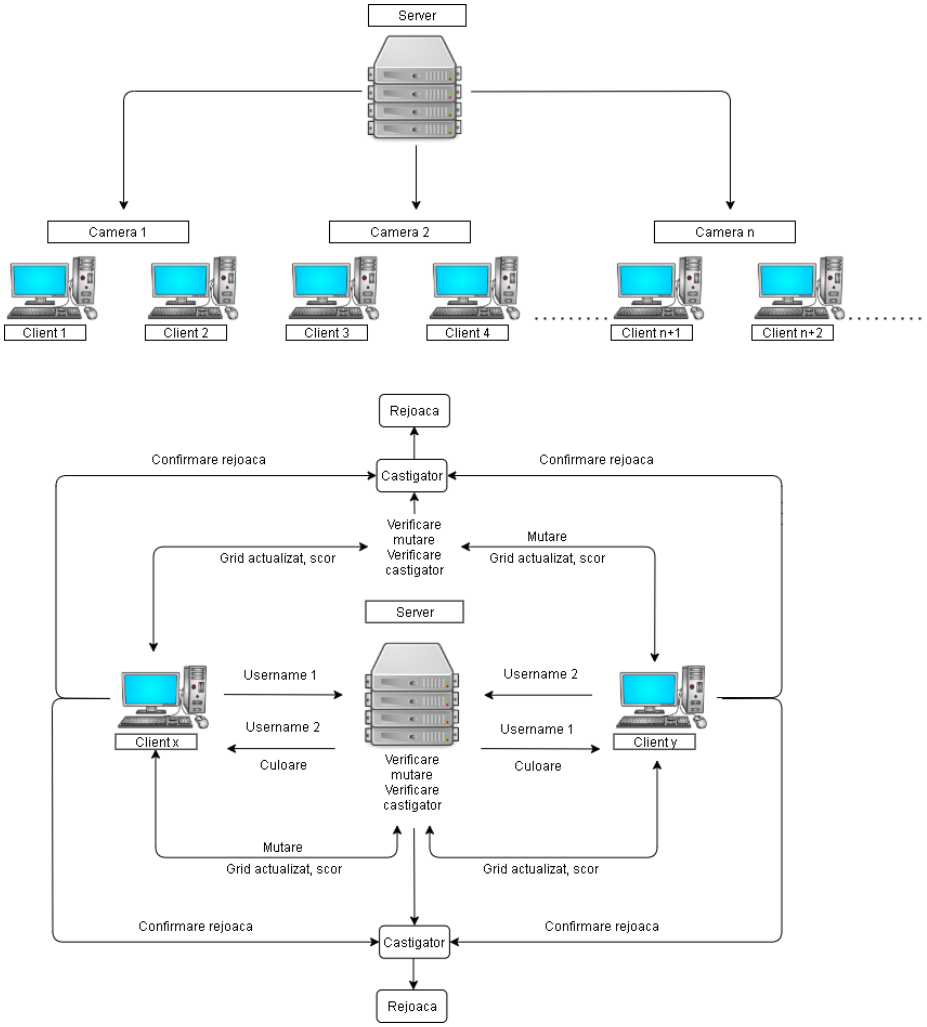

Figure 1 Diagrama

4 Detalii de implementare

Mai intai, in server stabilim faptul ca lucram cu perechi de cate doi client.

```
while (1)
{
    int client1, client2;
    int length = sizeof(from);

    printf("[server]Asteptam la portul %d...\n", PORT);
    fflush(stdout);

    client1 = accept(sd, (struct sockaddr *)&from, &length);
    client2 = accept(sd, (struct sockaddr *)&from, &length);
```

Figure 2 server

Dupa ce se realizeaza fork(), in procesul copil realizam programul. Cand un client se conecteaza la server, el va fi nevoit sa isi introduca un username. Dupa aceasta, doi client sunt cuplati si li se afiseaza contra cui joaca. Totodata, serverul stabileste culorile cu care jucatorii vor juca (conventie: jucatorul cu culoarea rosu incepe primul).

```
else if (pid == 0)
{
    // copil
    close(sd);
    bzero(player1, 100);
    bzero(player2, 100);
    read(client1, player1, 100);
    read(client2, player2, 100);
    tura = RandomCuloare();
    write(client1, player2, 100);
    write(client2, player1, 100);

    write(client1, &tura, sizeof(int));
    turaClient1 = tura;
    tura = SchimbaTura(tura);
    write(client2, &tura, sizeof(int));
    turaClient2 = tura;

    tura = 1;
```

Figure 3 server

```
bzero(msg, 100);
printf("Introduceti un username: ");
fflush(stdout);
read(0, msg, 100);
msg[strcspn(msg, "\n")] = 0;
write(sd, msg, 100);
read(sd, msg, 100);
printf("Adversarul dumneavoastra este %s. ", msg);
fflush(stdout);
read(sd, &tura, sizeof(int));
Culoare(tura);
retinetura = tura;
```

Figure 4 client

Dupa ce se stabileste fiecare jucator cu ce culoare va juca, incepe jocul propriu-zis. Retinem pentru fiecare jucator ce culoare a primit initial pentru a sti de la ce jucator asteptam un input, si care jucator sa astepte.

Programul contine functii pentru afisarea matricei, verificarea in cazul castigului, adaugarea unei noi valori etc.

```
int ConversieInput(char input[20]) ···
void AfisareMatrice(int matrice[LINII][COLOANE]) ···
void AdaugareMutare(int matrice[LINII][COLOANE], int col, int tura) ···
int SchimbaTura(int tura) ···
int Verificare(int matrice[LINII][COLOANE]) ···
int VerificareColoana(int matrice[LINII][COLOANE], int col) ···
int RandomCuloare() ···
```

Figure 5 server

La final, cand un jucator castiga, serverul asteapta confirmarea din partea ambilor jucatori pentru a juca o alta repriza. Dupa ce este primita confirmarea, se actualizeaza scorul si reincepe jocul, oferindu-se din nou, culori random.

5 Concluzii

Solutia propusa ar putea fi imbunatatita astfel:

- Implementarea unei interfete grafice;
- Implementarea unui chat pentru jucatori;
- Implementarea unui cronometru, respectiv un timp asociat pentru o mutare;
- Posibilitatea ca un jucator sa aleaga cu cine vrea sa joace; mai precis, jucatorii sa se conecteze intr-un lobby, iar apoi sa poata comunica intre ei si sa decida cine cu cine joaca.

6 Bibliografie

1. https://profs.info.uaic.ro/~gcalancea/lab7/servTcpConc.c
2. https://profs.info.uaic.ro/~gcalancea/lab7/cliTcpConc.c
3. http://en.wikipedia.org/wiki/Connect_Four
4. https://stackoverflow.com/questions/2693776/removing-trailing-newline-character-from-fgets-input
5. https://stackoverflow.com/questions/822323/how-to-generate-a-random-int-in-c
6. https://stackoverflow.com/questions/5106674/error-address-already-in-use-while-binding-socket-with-address-but-the-port-num
7. https://www.onlaptop.ro/blog/post/ce-sunt-porturile-tcp-si-udp
8. https://app.diagrams.net/