Retele de calculatoare -TEMA 2

2A1 Cobaschi Emanuel-Aser

December 6, 2022

1 Introducere

Proiect ales: ChessS.Acesta este un joc pentru 2 persoane, a carui durata se poate intinde de la cateva minute la cateva zeci de minute sau mai mult.Fiecare jucator are cate 16 piese(un rege, o regina, 2 turnuri, 2 nebuni, 2 cai, 8 pioni) de o anumita culoare (alb sau negru).Numarul acestor piese poate scadea, pe masura ce sunt atacate si capturate de openent.Piesele se pot deplasa dupa anumite reguli, specifice fiecarui tip in parte.Castigator este declarat acel jucator care reuseste sa-l puna pe oponent in imposibilitatea de a-si situa regele pe o pozitie neatacata.In cazul in care un concurent ramane fara posibilitati de a efectua o mutare, sau in cazul in care piesele ramase nu permit finalizarea jocului prin desemnarea unui castigator, se declara egalitate si jocul se sfarseste.

Enunt: Sa se conceapa o aplicatie server care pune la dispozitie o tabla de joc si supervizeaza desfasurarea fiecarei partide de sah, actionand ca un punct central la care clientii din retea se conecteaza. Serverul determina momentul in care jocul s-a terminat si anunta castigatorul. Regulile pot fi alese (simplificate) de catre proiectant, cu conditia ca jocul sa fie interactiv.

Motivatie: Am ales sa realizez acest proiect deoarece Chess este unul dintre cele mai cunoscute si apreciate jocuri de strategie, eu insumi fiind un fan al acestuia.

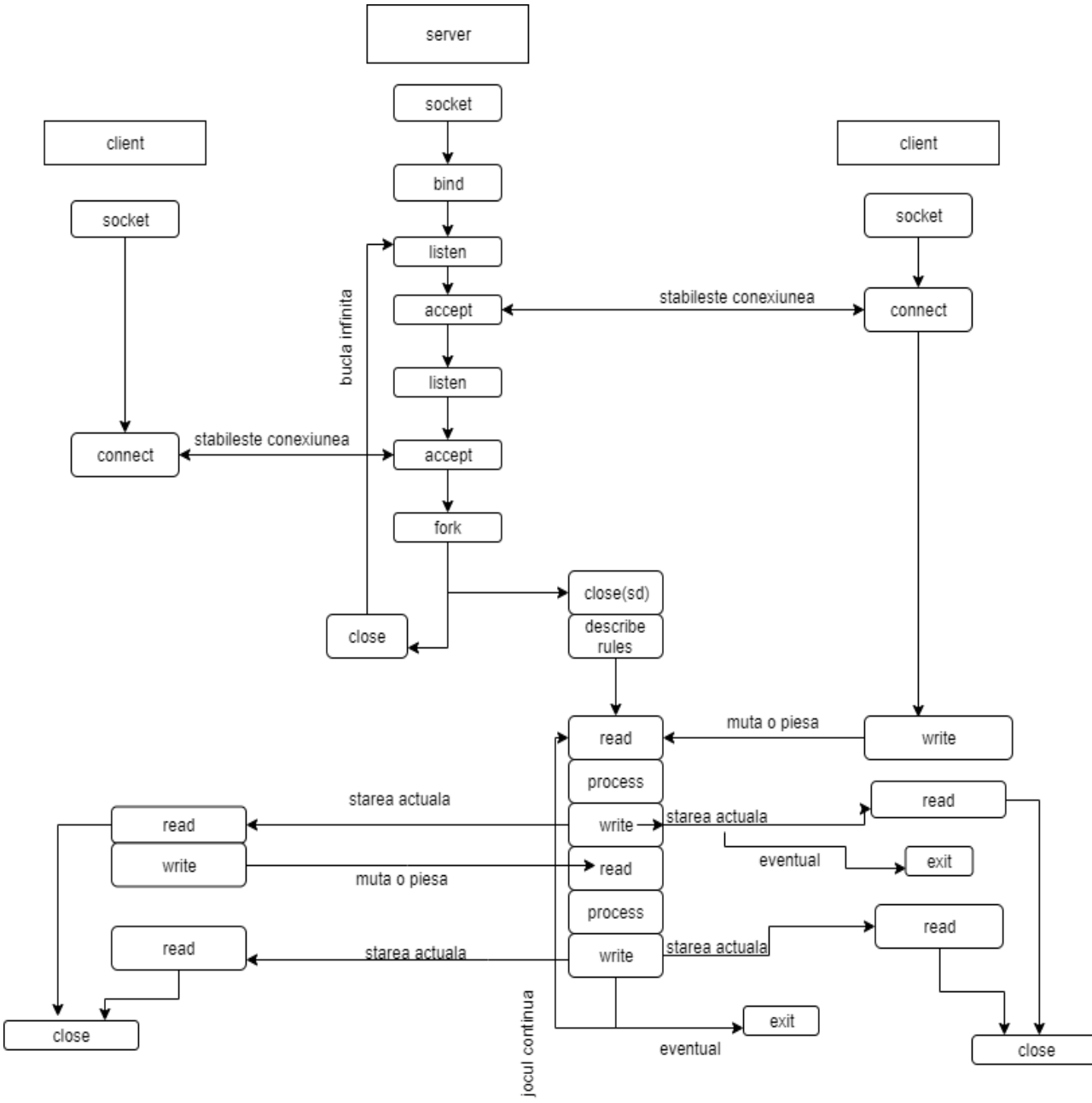
Detalii despre joc: Serverul va seta marimea tablei dupa dimensiunile consacrate: 8x8.Tot de catre server si random se va face alegerea culorilor si implicit a primului jucator(albul incepe mereu).Configuratia tablei de sah va fi stocata intr-o matrice patratica cu 8 linii si 8 coloane.

2 Tehnologiile Utilizate

Aleg sa folosesc protocolul TCP/IP pentru comunicarea client-server pentru a ma asigura ca toate mesajele sunt primite in ordinea buna si corecte.TCP fiind un protocol orientat conexiune implica, in prealabil transmiterii de date, conectarea socketului clientului la un socket al serverului. Initierea acestei conexiuni este fireste realizata de client cand necesita un anumit serviciu, serverul

asteptand pasiv cererile de conectare. .Se cunoaste asadar faptul ca acest protocol asigura siguranta transmisiei de date, neexistand astfel posibilitatea ca un jucator sa mute de doua ori consecutiv sau sa se sara peste randul lui. In cazul in care ar aparea o astfel de eroare in comunicare, jocul ar fi compromis iar continuarea lui nu ar mai avea sens. Doi clenti care se joaca intre ei vor putea trimite pe rand cate o mutare, iar perechile de clienti se vor putea juca deodata, concurent.
Aleg sa folosesc API-ul Socket BSD pentru fluxul de date input-output deoarece este unul dintre cele mai utilizate, stabil, usor de utilizat (socketul asemanandu-se mult cu un descriptor de fisier) si poate fi asociat cu mai multe procese.

3 Arhitectura Aplicatiei



Concepte implicate:
Clientul este utilizatorul, cel care joaca. In cadrul unui joc vor 'comunica' doi clienti intre ei prin intermediul serverului. Serverul, care poate fi asemanat cu un arbitru,pune la dispozitie o tabla de joc si supervizeaza desfasurarea fiecarei partide de sah, fiind cel care stabileste si afiseaza regulile jocului, si anunta castigatorul. Dupa anuntarea castigatorului, daca cei doi clienti mai vor sa joace, vor mai putea juca in continuare. Aplicatia va functiona astfel: serverul stabileste domeniul de comunicare si familia de protocoale care va fi folosita si creeaza un socket pentru comunicare. (Domeniul va fi AF-INET iar tipul SOCK-STREAM, specifice protocolului TCP) Apoi, serverul isi pregateste stucturile de date si asigneaza o adresa la socket folosind functia bind. In continuare, serverul va astepta sa primeasca cereri de conexiuni si le va accepta pe rand. Daca are o singura conexiune asteapta sa se mai conecteze inca un client. Daca are doua conexiuni acceptate va creea un proces fiu (folosind primitiva fork) in cadrul caruia vor comunica cei doi clienti prin intermediul serverului, iar procesul parinte va astepta noi utilizatori sa se conecteze. Astfel, aplicatia este construita conform modelului concurent TCP/IP.

4 Delatii De Implementare

Cod relevant proiectului

Pregatim serverul pentru a putea face conexiunea cu clientii.

```
void pregatire_server(){

    sd=socket(AF_INET, SOCK_STREAM, 0); //facem un socket, modalitate de comunicare in
    if(sd==-1){
     perror("[Server] Eroare la creare socket!");
     return;
    }

    bzero(&server, sizeof(server));// pregatim structura pentru server
    bzero(&from1, sizeof(from1));// pregatim structura pentru client_1
    bzero(&from2, sizeof(from2));// pregatim structura pentru client_2

    server.sin_family = AF_INET;// stabilim familia de socketuri
    server.sin_addr.s_addr = htonl (INADDR_ANY);//acceptam orice adresa    ---  host-t
    server.sin_port=htons(PORT); // utilizam un port     ---      host-to-network shor

    if(bind(sd, (struct sockaddr *)  &server, sizeof(struct sockaddr)) ==-1)  // folos
    {
        perror("[Server] Eroare la bind");
        return;
    }

    if(listen(sd,1)==-1)  // folosim listen pentru a astepta clienti
    {
      perror("[Server] Eroare la listen");
      return;
    }

}
```

De asemenea, serverul trebuie sa accepte clientii.

```c
int acceptare_client1(){
    int client_1;
    unsigned int length1 = sizeof(from1);


    client_1 = accept(sd,(struct sockaddr *) &from1, &length1 );
    if(client_1 ==- 1){
    perror ("[Server] Eroare la acceptarea clientului 1");
    fflush(stdout);
    return 404;
    }

    printf ("[Server] Am acceptat la portul %d...clientul 1\n",PORT);
    fflush (stdout);


   return client_1;
}
```

De asemenea, serverul trebuie sa stabileasca clientul care urmeaza la rand.

```
39
40  int alege_clientul_care_e_la_rand(int client1, int client2, int jucator )
41  {
42      int client;
43      if(jucator==1) client=client1;
44      else if(jucator==2) client=client2;
45      else {
46          perror("[Eroare] Cod jucator invalid\n");
47          exit(EXIT_FAILURE);
48      }
49      return client;
50  }
```

Serverul poate transmite clientului un anumit mesaj.

```c
void raspunde_clientului(int client1, int client2, int jucator, char* mesaj)
{
    int client=alege_clientul_care_e_la_rand(client1, client2, jucator);
    printf("Trimitem mesajul ..%s..catre jucatorul %d\n",mesaj,jucator);
    fflush(stdout);

    if(write(client,mesaj,MAX)<0)
    {
        perror("[Server] Eroare la transmiterea mesajului catre client\n");
        close(client);
        exit(EXIT_FAILURE);
    }

    printf("Mesajul..%s.. a fost transmis cu succes catre jucatorul %d\n",mesaj ,jucator);
    fflush(stdout);
}
```

Serverul poate incepe un joc nou.

```
69  void joc_nou(int client1, int client2)
70  {
71      for(int i=0;i<8;i++)
72          for(int j=0;j<8;j++)
73              tabla[i][j]='0';
74      char *m1="alb";
75      char *m2="negru";
76
77      raspunde_clientului(client1, client2, 1, m1);// jucatorul 1 va juca cu alb
78      raspunde_clientului(client1, client2, 2, m2);// jucatorul 2 va juca cu negr
79  }
80
```

Serverul este concurent, putand trata simultan mai multe meciuri de chess.

```
while(1)
{
 int client_1=acceptare_client1();
 int client_2=acceptare_client2();
 if(client_1==404 || client_2==404)// nu am gasit pereche
 {
     close(client_1);
     close(client_2);
     continue;
 }
 int pid=fork(); //cream un proces fiu
 if(pid==-1) //Eroare la fork
 {
     close(client_1);
     close(client_2);
     continue;
 }
 else if(pid>0) //parinte ->el continua sa accepte jucatori
 {
     close(client_1);
     close(client_2);
     while(waitpid(-1,NULL,WNOHANG)); //?
     continue;
 }
 else if(pid==0) //copil
 {
     close(sd);
     printf("[Server] Avem 2 jucatori si putem incepe partida\n");
     joc_nou(client_1, client_2);
     close(client_1);
     close(client_2);
     exit(0);
 }
}
```

Clientul verifica daca are parametrii corecti si apoi se conecteaza la server.

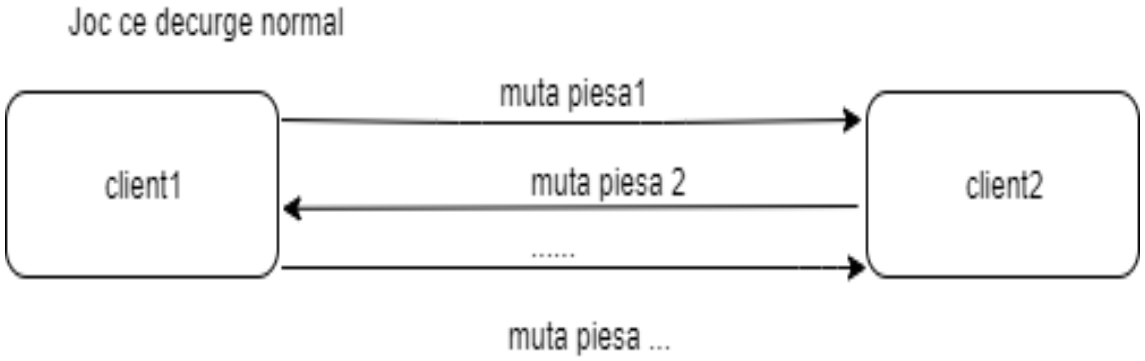
```c
int main(int argc, char *argv[])
{
    if(argc!=3) //verificam existenta tuturor parametrilor in linia de comanda
  {
    printf("Sintaxa corecta: %s, <adresa_server>, <port>\n",argv[0]);
    return -1;
  }

    port=atoi(argv[2]);//stabilim portul

    sd=socket(AF_INET, SOCK_STREAM, 0); //cream socketul

    if(sd==-1)
    {
        perror("[Client] Eroare la creare socket");
        return;
    }
    server.sin_family=AF_INET;//stabilim familia socketului
    server.sin_addr.s_addr=inet_addr(argv[1]);//adresa IP a serverului
    server.sin_port=htons(port); //portul de conectare

    if(connect(sd,(struct sockaddr*) &server, sizeof(struct sockaddr))==-1) //ne co
    {
        perror("[Client] Eroare la conectarea la server.Reincercati...");
        return;
    }
    printf("M-am conectat la server...\n");

    stabileste_regulile_jocului();

   return 0;
}
```

Clientul receptioneaza regulile de la server si se pregateste de un joc nou.

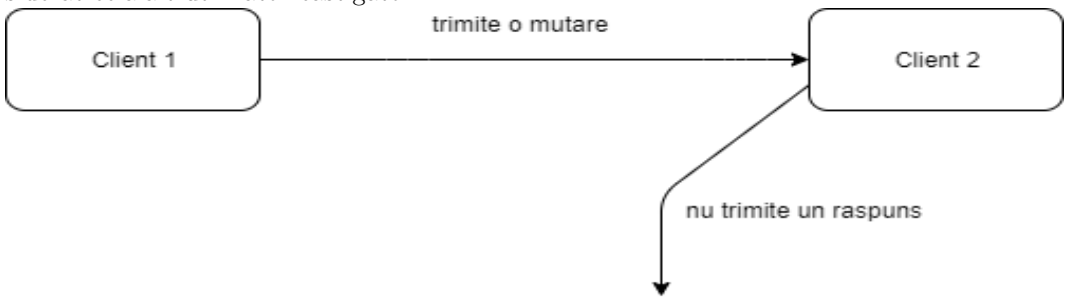
```
void joc_nou(){
    for(int i=0;i<8;i++)
        for(int j=0;j<8;j++)
            tabla[i][j]='0';
}

void citeste_mesajul_serverului()
{
    if (read (sd, msg, MAX) < 0) //(apel blocant pina cind serverul raspunde)
    {
        perror ("[client] Eroare la read() de la server.");
        fflush(stdout);
        return;
    }
    printf("Am citit de la [server]: %s\n", msg);
    fflush(stdout);
}

void stabileste_regulile_jocului()
{
    joc_nou();
    citeste_mesajul_serverului();
}
```

Scenarii de utilizare

Daca totul decurge normal, aplicatia va functiona asemenea unui joc de ping-pong. Fiecare client transmite pe rand cate un numar, iar serverul proceseaza informatiile primite si le trimite inapoi actualizarile.



In cazul in care un utilizator nu muta intr-un anumit interval de timp (sa zicem 2 minute) sau paraseste jocul (se deconecteaza) jocul va fi incheiat si considerat celalalt utilizator castigator.



Inputul clientului va fi verificat inainte de a fi transmis serverului. Daca acesta nu este un o pozitie valida pe care se poate muta o anumita piesa atunci se va afisa un mesaj de eroare si se va cere o noua valoare. Serverul va gestiona datele citite astfel incat in timpul unui joc un utilizator sa nu poata trimite mai multe mutari pe care sa le aplice pieselor sale, ci fiecare va astepta randul sau. Astfel, serverul asteapta sa primeasca datele de la primul client, apoi de la al doilea, apoi iar de la primul, si tot asa, pe rand.

5 Concluzii

Aplicatia ar putea fi imbunatatita adaugandu-se o interfata grafica (GTK+) pentru a fi mai usor de utilizat, fiind intuitiva pentru jucatori.Alta imbunatatire ar putea fi realizarea unui campionat de Chess, intre mai multi jucatori. In prima runda vor fi 2^n jucatori, dintre care in jur de 2^{n-1} vor castiga/ vor obtine remiza. Acestia din urma vor putea juca intre ei in continuare pana cand ramane doar unul ce va fi declarat campion. (In cazul in care se ajunge la un numar impar de jucatori, ori se va alege random o persoana ce va trece direct la etapa

urmatoare ori o persoana dintre cele care nu au castigat pentru a concura cu cine este in plus.)

6 Bibliografie

https://www.diagrameditor.com/
https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
https://profs.info.uaic.ro/ computernetworks/ProiecteNet2022.php
https://en.wikipedia.org/wiki/Chess
https://profs.info.uaic.ro/ matei.madalin/rc//
https://profs.info.uaic.ro/ eonica/rc/