DraughtsS(Dame)

Andrei Iulian Luca 2E3

December 2020

1 Introducere

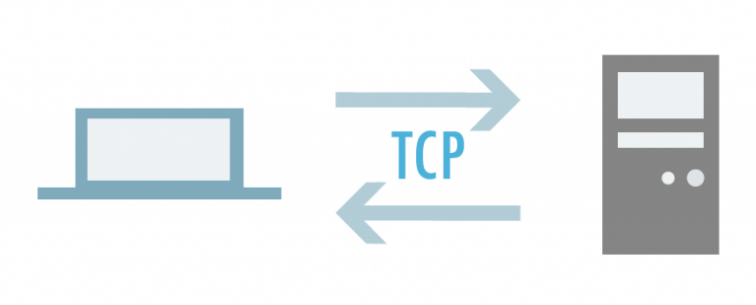
Pentru proiectul final am ales proiectul de tipul B cu numele DraughtsS unde trebuie implementat un server ce poate oferi posibilitatea unor perechi de clienti sa joace diverse variatii de "draughts" (dame) si sa mentina clasamente diverse pentru fiecare tip de joc.

O rețea de calculatoare (engleză: computer network) leagă între ele o mulțime mai mică sau mai mare de calculatoare, astfel încât un calculator poate accesa datele, programele și facilitățile sau resursele unui alt calculator conectat la aceeași rețea. De obicei este nevoie de măsuri de restricție/siguranță a accesului.

2 Tehnologii utilizate

Pentru acest proiect trebuie sa implementam si folosi tehnologia de tip TCP si apelul de sistem fork pentru a implementa server-ul concurent.

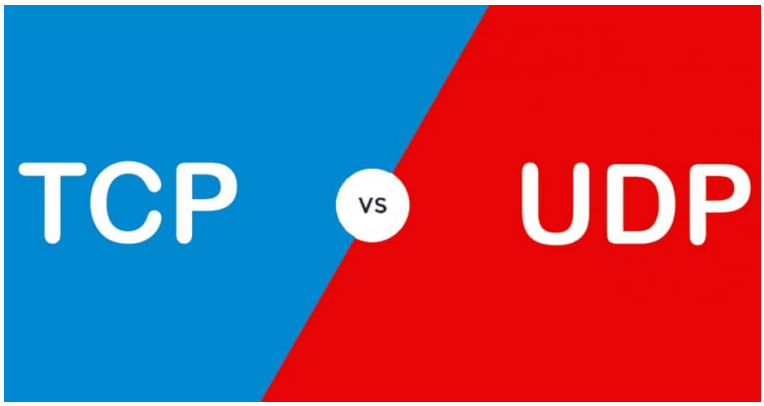
2.1 TCP



Transmission Control Protocol (sau TCP, în traducere liberă din engleză Protocolul de Control al Transmisiei) este un protocol folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o adresă IP și de către un port TCP.

Transmission Control Protocol (TCP) este unul dintre protocoalele de bază ale suitei de protocoale Internet. TCP este unul dintre cele două componente originale ale suitei (celalalt fiind Protocolul Internet, sau IP), astfel încât întreaga suita este frecvent menționată ca stiva TCP/IP. În special, TCP oferă încredere, asigura livrarea ordonata a unui flux de octeți de la un program de pe un computer la alt program de pe un alt computer aflat în rețea. Pe lângă sarcinile sale de gestionare a traficului, TCP controlează mărimea segmentului de date, debitul de informație, rata la care se face schimbul de date, precum și evitarea congestionării traficului de rețea. Printre aplicațiile cele mai uzuale ce utilizează TCP putem enumera World Wide Web (WWW), posta electronica (e-mail) și transferul de fișiere (FTP).

2.2 TCP vs UDP



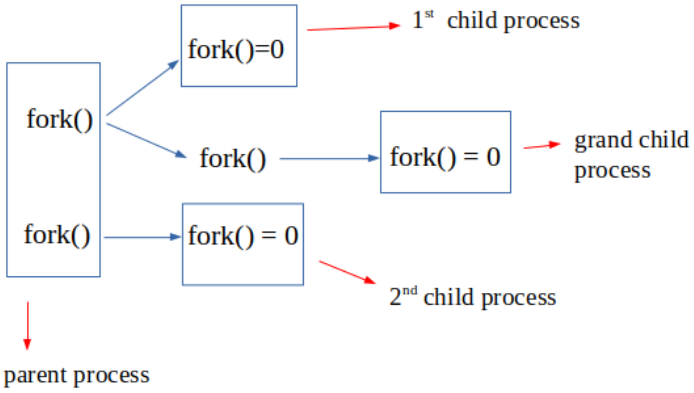
Desi este mai lent decat optiunea UDP-ului, TCP/IP foloseste protocoale de tip handshake, oferindu-ne siguranta ca toate datele au ajuns in siguranta la destinatie si daca apar eventuale erori server-ul le poate rezolva fara probleme, asta fiind cel mai important in acest tip de proiect.

In tehnologia TCP suntem siguri ca destinatarul va primi pachetul trimis pe cand in UDP nu putem fi siguri de asta, prin TCP pachetele pot fi rearanjate in modul specificat de noi insa UDP-ul nu are o ordine fixa, pachetele fiind independente.Nu in ultimul rand, tehnologia TCP verifica daca exista erori si le corecteaza, UDP-ul verifica la randul lui si el daca sunt erori numai ca atunci cand le gaseste le elimina si nu le corecteza.

2.3 FORK

Fork-ul este un apel de sistem folosit in a produce noi procese numite de obicei procese copil ruland in mod concurent cu procesul parinte (cel ce-l produce in prima instanta).Dupa ce un nou copil este creat, ambele procese vor executa urmatoarele instructiune ce urmeaza dupa apelul de sistem fork().Procesul copil va avea acelasi registru CPU si aceleasi fisiere deschise de catre procesul parinte.

In acest proiect vom folosi apelul de sistem fork() in detrimentul thread-urilor deoarece ne dorim securitatea proceselor separate, plus ca nu avem nevoie de informatiile din ceilalti clienti.



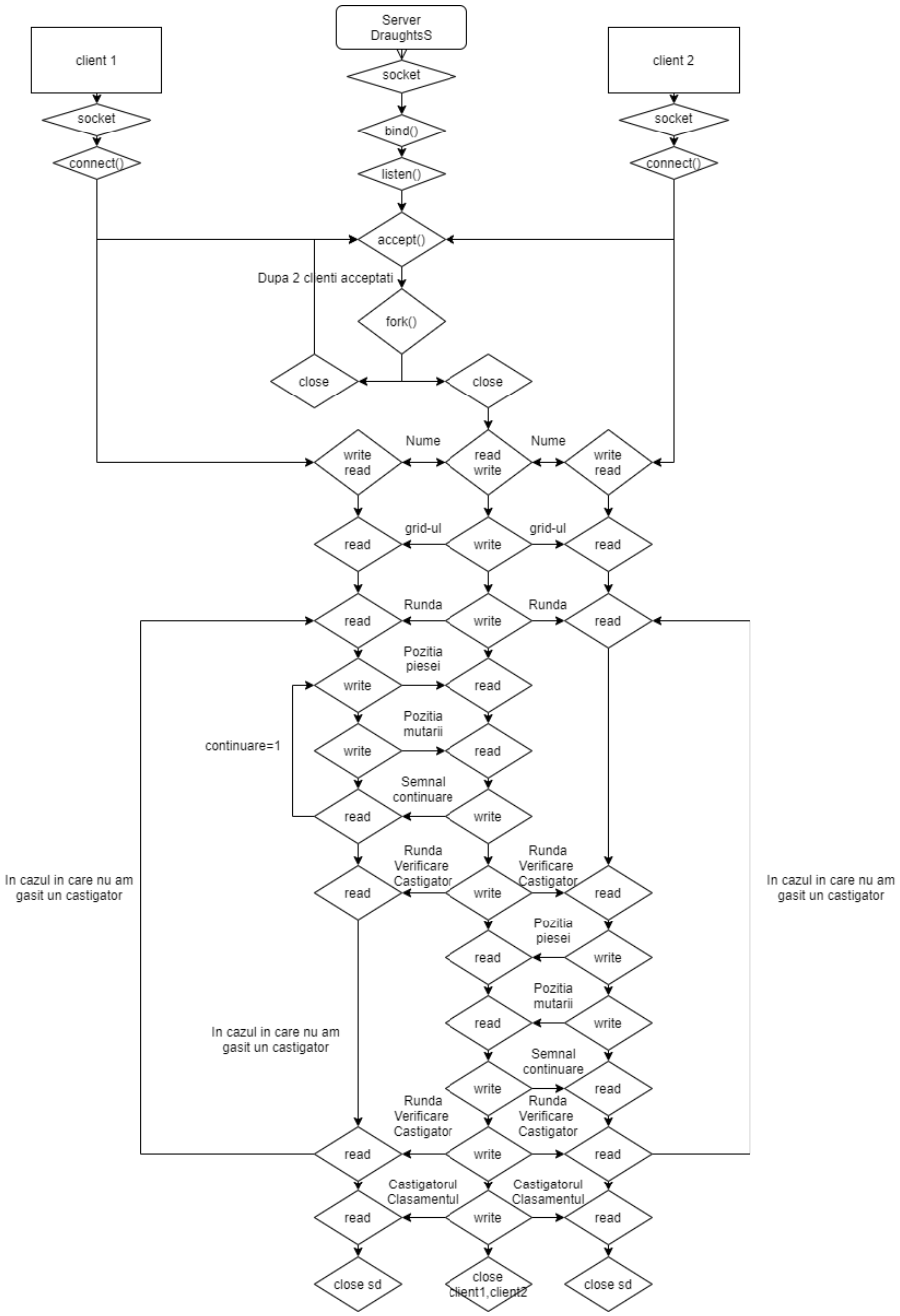
Apel fork

3 Arhitectura aplicatiei

3.1 Conceptele aplicate

- Pentru implementarea acestui proiect vom utiliza tehnologia TCP fiind cea mai sigura metoda de a transmite date pe care o avem la dispozitie
- Pentru a realiza server-ul in mod concurent vom folosi apelul de sistem fork(), realizand un nou proces pentru fiecare nou client ce se va conecta, este mai lent comparativ cu thread-urile insa beneficiem de siguranta proceselor individuale
- Comunicarea intre jucatori o vom realiza utilizand socket-uri si apeland functiile asociate precum send/receive/read/write/listen.

3.2 Diagrama aplicatiei detaliate



4 Detalii de implementare

La inceputul jocului, am implementat o mica functie prin care cei doi jucatori isi pot cunoaste numele unui celuilalt.

```
Server side
  bzero(nickname1,10);
  bzero(nickname2,10);

  read(client1,nickname1,10);
  read(client2,nickname2,10);

  write(client1,nickname2,10);
  write(client2,nickname1,10);

Client side
  printf("Bine ati venit la jocul de Dame!\n");
  fflush (stdout);

  printf ("Introduceti va rog un nickname: ");
  fflush (stdout);
  bzero (nickname, 10);
  scanf("%s",nickname);
  fflush (stdin);
  write(sd, nickname, 10);

  printf("Imediat veti primi si numele adversarului!\n");
  fflush (stdout);
  bzero (nickname, 10);
  read(sd, nickname, 10);
```

Transmiterea matricei va avea loc in multiple locuri pe parcursul programului

```
for(int i=0;i<8;i++)
        for(int j=0;j<8;j++)
        {
        write(client1, &dame[i][j], sizeof(int));
        write(client2, &dame[i][j], sizeof(int));
        }
```

Pentru transmiterea si verificarea coordonatelor am folosit primitivele write() si read(), acelasi mod de gandire a fost aplicat si pentru verificarea coordonatelor pozitiei.

```
Server side
/*aici vom primi coordonatele din partea clientului*/
read(client1,&linia1,sizeof(int));
printf("%d\n",linia1);
fflush(stdout);
```

```
read(client1,&coloana1,sizeof(int));
printf("%d\n",coloana1);
fflush(stdout);
if(VerificarePozitie(linia1,coloana1)==1)
{write(client1,&OK,sizeof(int));
pozitie=1;
printf("Pozitia este OK!\n");
fflush(stdout);}
if(VerificarePozitie(linia1,coloana1)==0)
{write(client1,&NOTOK,sizeof(int));
pozitie=0;
printf("Pozitia NU este OK!\n");
fflush(stdout);}
}while(pozitie==0);
```

Cu ajutorul functiilor VerificareDubla() care verifica daca mutarea initiala a fost un salt peste o piesa inamica si functia ContinuareMutare() putem decide daca, conform regulilor jocului, mai putem sari peste o piesa daca dorim.

```
if(VerificareDubla(linia1,coloana1,linia2,coloana2,tura)==1)
                                        {
        if(ContinuareMutare(linia2,coloana2,tura)==1)
        {
        continuare=1;
        write(client1,&OK,sizeof(int));
        printf("Se mai poate efectua o mutare!\n");
        }
        if(ContinuareMutare(linia2,coloana2,tura)==0)
        {
        continuare=0;
        write(client1,&NOTOK,sizeof(int));
        printf("NU se mai poate efectua o mutare!\n");
        }
                                        }
```

Cu ajutorul functiei VerificareCastigator() care este aplicata o data la fiecare tura vom sti cand programul trebuie sa se opreasca, mai exact, atunci cand unul dintre jucatori a capturat toate piesele adverse.

```
if(VerificareCastigator()==0){
        write(client1,&NOTOK,sizeof(int));
        write(client2,&NOTOK,sizeof(int));
}
if(VerificareCastigator()==1){
        write(client1,&OK,sizeof(int));
        write(client2,&OK,sizeof(int));
```

}

Dupa ce am verificat pozitia piesei pe care dorim sa o mutam, trebuie verificate coordonatele pozitiei pe care dorim sa o efectuam si sa verificam daca este in conformitate cu regulile jocului de dame(pozitia pe care putem muta este in diagonala sau "diagonala dubla pas" daca sarim peste o piesa adversa si astfel o stergem de pe tabela.Mai exista un caz special, acela in care o piesa ajunge la capatul opus si atunci acestea se denumesc "dame", spre exemplu daca jucatorul 1 cu piesele BLACK ajunge pe linia 7 atunci se transforma in dame si pot merge si in sens opus (in spate).

```
int StergerePiesa(int i,int j){
    int puncte1,puncte2;
    if(tura=1){
    if(dame[i+2][j-2]==0)
          if(dame[i+1][j-1]==2)
               {
                   dame[i+1][j-1]=0;
                   puncte1++;
               }
    if(dame[i+2][j+2]==0)
           if(dame[i+1][j+1]==2)
           {
               dame[i+1][j+1]=0;
               puncte1++;
           }
           }
    if(tura==2){
        if(dame[i-2][j-2]==0)
           if(dame[i-1][j-1]==1)
               {
                   dame[i-1][j-1]=0;
                   puncte2++;
               }
    if(dame[i-2][j+2]==0)
           if(dame[i-1][j+1]==1)
           {
               dame[i-1][j+1]=0;
               puncte2++;
           }
    }
    if(puncte1==12)
```

```
    printf("Utilizatorul cu piese negre a castigat!Felicitari!\n");
if(puncte2==12)
    printf("Utilizatorul cu piese albe a castigat!Felicitari!\n");

            }
```

Prin aceasta functie verficam prima data daca player-ul a ales sa faca un salt peste piesa adversa si verificam daca piesa sarita este chiar cea a adversarului.Pentru eficienta aceasta funtie are dubla functie, sterge piesa adversarului in cazul in care aceasta a fost sarita de catre player, incrementam variabila ce are rolul de a retine punctele unui player iar la finalul functiei testam daca unul dintre jucatori a castigat(conform regulilor jocului de dame, pentru ca un player sa castige trebuie sa "manance" toate piesele adversarului).

5 Concluzii

5.1 Cum poate fi imbunatatita solutia oferita

- Eficienta unor algoritmi pot fi imbunatatiti cu privire la gradul de complexitate si timpul de rulare.
- Grid-ul ar putea fi implementat printr-un GUI in loc de format text.
- Se poate implementa un temporizator pentru fiecare jucator pentru a sti cat timp de gandire mai au jucatorii.
- Se pot implementa mai multe tipuri de dame.
- Se poate implementa un server SQL ce va retine clasamentul intr-un mod mai eficient.

References

[1] Regulile Jocului de Dame https://en.wikipedia.org/wiki/Draughts

[2] Definitia retelei de calculatoare https://ro.wikipedia.org/wiki/Retea$_d e_c alculatoare$

[3] TCP https://ro.wikipedia.org/wiki/Transmission$_C ontrol_P rotocol$

[4] UDP https://ro.wikipedia.org/wiki/User$_D atagram_P rotocol$

[5] Fork https://www.geeksforgeeks.org/fork-system-call/

[6] TvU https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html

[7] Template server TCP concurent https://profs.info.uaic.ro/ gcalancea/lab7/servTcpConc.c