Online Info Olympiad
Tema 2

Andrei Dascălu 2A3

Facultatea de Informatică, Universitatea "Al. I. Cuza" Iași

1 Introducere

Acest raport urmăreşte realizarea unui model client-server pentru o olimpiadă de informatică online (OnlineInfoOlympad) [1]. Astfel, scopul acestui proiect este de a oferi posibilitatea desfășurării unei astfel de activități în care un număr prestabilit de participanți concurează în rezolvarea unei probleme, după evaluarea surselor fiecăruia fiind transmise rezultatele.

Structura raportului este reprezentată, pe lângă **introducere**, de **tehnologiile utilizate** unde sunt detaliate caracteristici ale proiectului, **arhitectura aplicației** ce descrie conceptele impicate, conținând și o diagramă detaliată a aplicației, **detaliile de implemantare** ce presupun o detaliere a codului și sceneraiile de utilizare ale aplicației și **concluziile**.

2 Tehnologii utilizate

Principala tehnologie utilizată este TCP-ul, un **P**rotocol de **C**ontrol al **T**ransmisunii care stabilește o conexiune duplex, în cazul de față, între server și clienți (participanții la olimpiada online). Acest protocol oferă un flux de date de încredere prin mecanisme de detecare și corectare a erorilor, precum și de confirmare a erorilor. Ambele părți trebuie să participe la realizarea conexiunii, identificată prin perechile reprezentate de adrese *IP:PORT*. Conexiunea este stabilită prin intermediul soclurilor (*eng.* sockets).

În cazul aplicației prezentate, a fost implementat un **server TCP concurent cu pre-thread**. Partea de pre-thread, adică pre-inițializarea unui număr determinat de fire de execuție (**eng.** threads) este un aspect convenabil deoarece se cunoaște numărul de participanți care ar trebui să participe. Fiecare thread deservește un client, identificându-se astfel aspectul concurent al implementării. De asemenea, este utilizat un lacat **mutex** pentru protecția primitivei *accept()*. În acest fel, numai un singur fir de execuție poate deservi un client.

Enunțurile problemelor, precum și testele cu datele de intrare și ieșire (input/output) sunt stocate într-un fișier **XML**. Acest lucru permite accesarea relativ facilă a datelor necesare prin intermediul parserului specific bibliotecii *libxml2*.

3 Arhitectura aplicației

Aplicația este reprezentată de două programe, unul pentru server-ul TCP concurent și unul pentru client. Programul desemnat server-ului dispune de anumite funcții care oferă funcționalitățile specifice unei olimpiade de informatică online. Astfel, funcția *chooseProblem()* alege aleatoriu o problemă cu datele sale aferente din fișierul XML pe care o trimite participanților prin funcția *sendProblem()*. Prin intermediul funcției *receiveSource()* sunt primite rezolvările (codurile sursă) ce sunt notate cu ajutorul funcției *markSource*. Rezultatele sunt transmise prin funcția *sendResults*.

În cazul programului desemnat clientului, acesta dispune de funcția *readProblem()* care primește enunțul problemei și funcțiile *sendSource()* și *getResults()*.

3.1 Diagrama

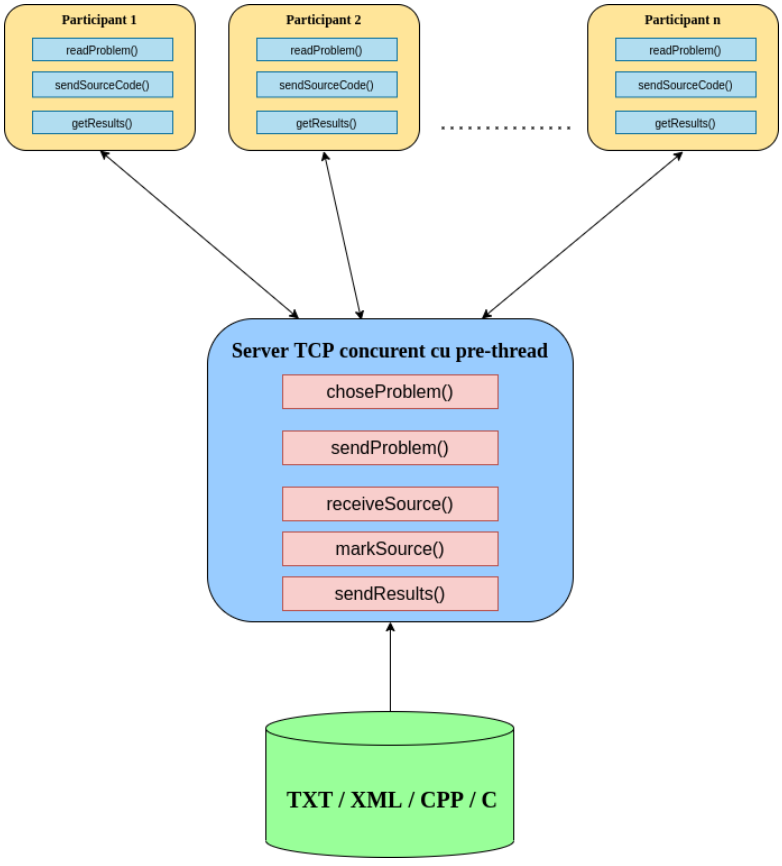

Fig. 1. Diagrama detaliată a aplicației

4 Detalii de implementare

(client) Conectarea clientului la server:

```
if(argc != 3)
   {
       printf("Lipsesc argumentele: %s <adresa_server> <port>\n",
            argv[0]);
       return -1;
   }
   port = atoi (argv[2]);

   if((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
   {
       perror ("Eroare la socket().\n");
       return errno;
   }

   server.sin_family = AF_INET;
   server.sin_addr.s_addr = inet_addr(argv[1]);
   server.sin_port = htons (port);

   if(connect (sd, (struct sockaddr *) &server,sizeof (struct
        sockaddr)) == -1)
   {
       perror ("[client] Eroare la connect().\n");
       return errno;
   }
```

(server) Funcția de tratare a clientului:

```
void *treat(void * arg)
{
   int client;

   struct sockaddr_in from;
   bzero (&from, sizeof (from));
   printf ("[thread]- %d - pornit...\n", (int) arg);
   fflush(stdout);

   while(1) {
       int length = sizeof (from);
       pthread_mutex_lock(&mlock);

       if ((client = accept(sd, (struct sockaddr *) &from, &length)) < 0)
           handle_error("[thread] Eroare la accept().\n", -1);
       pthread_mutex_unlock(&mlock);
       threadsPool[(int)arg].thCount++;
```

```
        login(client);
        sendProblem(client);
        receiveSource(client);
        evaluateSource(client);

        close (client);
    }
}
```

(client) Funcția de primire a fișierului sursă al clientului:

```
void receiveSource(int clientSock)
{
   printf("[server] Se accepta surse cu rezolvarea problemei.\n");
   int source_fd, readcode;
   char sourceName[DIMBUF+1], buffer[DIMBUF];
   sourceName[0] = 'S';
   if(read(clientSock, sourceName + 1, DIMBUF) < 0)
       handle_error("[server] Eroare la citirea numelui fisierului sursa
           trimis.\n", errno);

   if((source_fd = open(sourceName, O_RDWR | O_CREAT, 0644)) == -1)
       handle_error("[server] Eroare la crearea fisierului sursa de
           catre server.\n", errno);

   while(1)
   {
       bzero(buffer, DIMBUF);
       if((readcode = read(clientSock, buffer, DIMBUF)) < 0)
           handle_error("[server] Eroare la citirea din socket.\n",
               errno);
       printf("%s\n", buffer); fflush(stdout);
       //printf("%d", readcode); fflush(stdout);
       if(readcode == 0)
           break;
       if(write(source_fd, buffer, DIMBUF) < 0)
           handle_error("[server] Eroare la scrierea in fisierul
               sursa.\n", errno);
   }
   if(write(clientSock, "Am primit fisierul sursa.\n", DIMBUF) < 0)
       handle_error("[server] Eroare la scrierea in socket.\n", errno);
   close(source_fd);
}
```

4.1 Scenarii de utilizare

În început, participantul trebuie sa își introducă username-ul său în aplicație cu ajutorul comenzii **login**. Dupa ce a introdus un username valid, acesta primește enunțul problemei alese aleatoriu cu comanda **getproblem**. Dupa ce termină rezolvarea, acesta poate trimite fișierul sursă cu ajutorul comenzii **sendsource**, urmând să introducă numele fișierului de trimis. Acesta va primi rezultatele rezolvării sale, urmând ca acesta să iasă din aplicație cu ajutorul comenzii **exit**.

Serverul poate configura la inceput, înaintea competiției, un anumit numar de parametri (numărul de participanți, timpul de rezolvare etc.) prin comanda **modify** și ulterior comenzile **participanți** și **timp**. Competiția este inițiată de server prin comanda **start**.

4.2 Diagrama

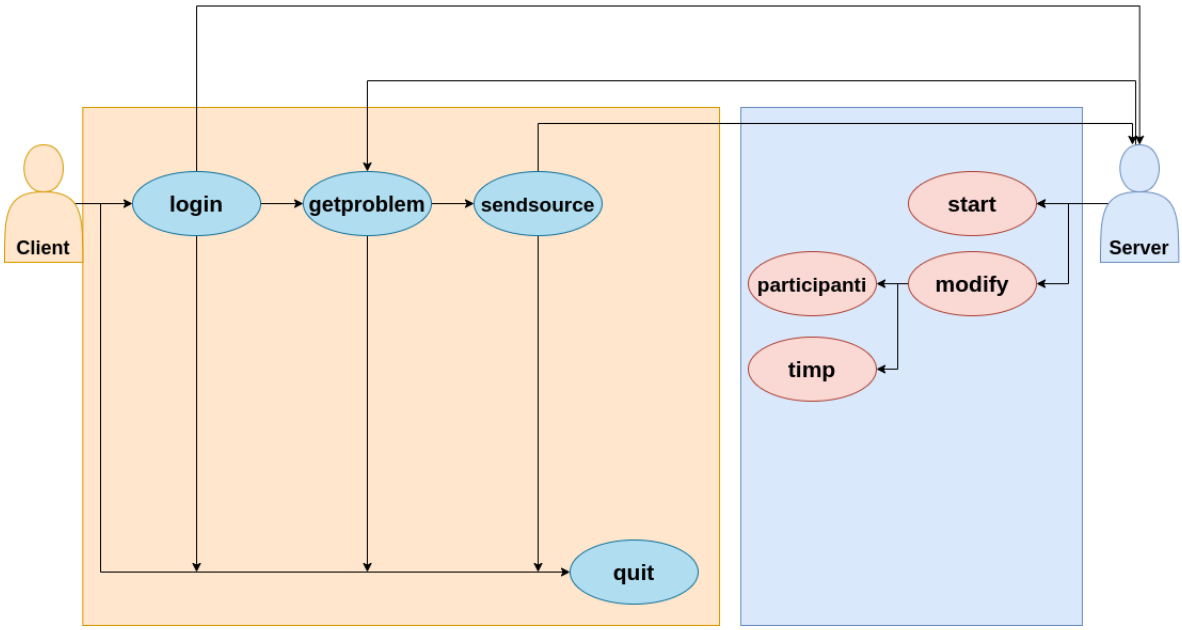

Fig. 2. Diagramă scenarii de utilizare

5 Concluzii

Aplicația prezentată în acest raport este o rezolvare minimală a problemei impuse de o astfel de situație, o olimpiadă de informatică online. Caracterul concurent al serverului permite conectarea mai multor participanți și trimiterea în timpul destinat rezolvari a surselor realizate de către aceștia. O primă idee de îmbunătățire a proiectului este dezvoltarea unei interfețe grafice cu ajutorul librăriei **Qt**. Acest lucru va permite o interacțiune mai ușoară a clientului cu aplicația.

De asemenea, un alt aspect care ar putea fi implementat este rezolvarea problemei de informatică, mai precis dezvoltarea codului, direct în aplicație. Astfel, aplicația ar putea să ofere opțiunea de editare text. În acest fel, participantul ar putea să vizualizeze codul său direct în aplicație.

Un al treilea aspect care ar putea fi implementat este opțiunea de notare treptată a unei surse dezvoltate. Astfel, participantul are posibilitatea de a trimite sursa de mai multe ori în perioada de timp de desfășurare a competiției (un număr limitat sau nu). În cazul în care participantul primește un punctaj parțial, acesta primește de la server indicații cu privire la rezolvarea problemei, participantul având opțiunea îmbunătățirii sursei sale și retrimiterii. Opțiunea aceasta poate fi setată doar de către server.

Bibliografie

1. Retele de Calculatoare — Proiecte propuse https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GIOD_TCP
2. Bibliography management with bibtex - Overleaf, Online LaTeX Editor https://www.overleaf.com/learn/latex/bibliography_management_with_bibtex
3. Conference Proceedings guidelines — Springer https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines
4. Istoric. Concepte si notiuni de baza. Partea 1 https://profs.info.uaic.ro/~computernetworks/files/4rc_NivelulTransport_Ro.pdf
5. Istoric. Concepte si notiuni de baza. Partea 2 https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
6. Laboratoare https://sites.google.com/view/fii-rc/laboratoare
7. Qt for Beginners - Qt Wiki https://wiki.qt.io/Qt_for_Beginners
8. Springer Lecture Notes in Computer Science - Overleaf, Online LaTeX Editor https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj?fbclid=IwAR0gq-3vEypCwcP3zmmSelBdAAyDLD0602m8d4sLNQUGlWmo1mO7O-sUjm4#.WtR5Hy5ua71
9. TCP Client/Server Communication - I/O Device Guide - InterSystems IRIS Data Platform 2020.3 https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GIOD_TCP
10. Transmission Control Protocol - Wikipedia https://ro.wikipedia.org/wiki/Transmission_Control_Protocol