Raport Tehnic "Quizz Game"

Platon Andreea

Facultatea de Informatica Iasi, UAIC
andreea.platon@info.uaic.ro

1 Introducere

In cadrul proiectului Quizz Game imi propun sa implementez un sistem de tip server-client pentru a crea un joc interactiv de tip quizz. Sistemul utilizeaza un server multithreading, care permite lansarea mai multor runde simultan, unde se pot conecta mai multi clienti in acelasi timp - 3 mai exact, adica serverul este unul concurent. Scopul aplicatiei este de a coordona acesti clienti intr-un mod sincron, odata ce se conecteaza, acestia sunt asignati unei runde, urmand sa primeasca cate o intrebare si avand un numar n de secunde in care pot oferi un raspuns (nu se trece la urmatoarea intrebare pana cand fiecare client nu raspunde sau expira timpul alocat), sa gestioneze formarea rundelor si inregistrarea fiecarui participant (odata inceputa o runda, niciun client nu se mai poate conecta la acea runda, fiind asignati unei alte runde disponibile la acel moment), sa retina scorul fiecaruia si sa trimita la final clasamentul total al rundei respective.

2 Tehnologii aplicate

2.1 Protocolul TCP

Protocolul TCP l-am ales datorita principiului sau de functionare, acesta asigurand o trimitere corecta si mai eficienta a datelor. TCP controlează mărimea segmentului de date, debitul de informație, rata la care se face schimbul de date, precum și evitarea congestionării traficului de rețea, ceea ce imi ofera mie siguranta ca nu se pierd date in transmisiune si sunt livrate in ordinea corecta in care au fost trimise. Acest lucru este necesar, intrucat, fiecare client trebuie sa-si primeasca intrebarea de la server si, la randul sau, trebuie sa trimita inapoi raspunsul, pentru a se mentine o functionare corecta si deplina a jocului.

2.2 Threading si Mutex

Serverul este implementat folosind un sistem multithreading pentru a putea fi deserviti clientii in numar nelimitat, intr-un mod simultan. Am ales thread-uri in loc de procese, deoarece acestea nu lucreaza in mod independent unele de altele, acestea impartind cu alte thread-uri sectiunea lor de cod, zona de memorie si resurse ale sistemului de operare cum ar fi fisierele deschise si semnalele. Mutex-urile sunt utilizate pentru a sincroniza accesul la datele partajate (scorurile si lista clientilor conectati). De asemenea, mutex-urile asigura ca anumite date sunt accesate la un moment de timp doar de un singur thread (cel care blocheaza, este si cel care deblocheaza mutexul).

2.3 Stocarea intrebarilor in fisier XML

Am ales sa stochez intrebarile si raspunsurile intr-un fisier XML, datorita facilitatii de a gestiona datele intr-un mod structurat si accesibil de modificat (extinderea sau reducerea numarului de intrebari).

Intrucat aplicatia la etapa actuala este doar un prototip, in implementare nu se regaseste fisierul XML mentionat mai sus. De asemenea, pentru a facilita utilizarea aplicatiei, proiectul va fi insotit si de o interfata grafica corespunzatoare.

3 Structura aplicatiei

Aplicatia este formata din 2 componente principale:

3.1 Serverul

Acesta gestioneaza conectarea clientilor, incarcarea intrebarilor din fisierul XML si coordonarea intregii logici a jocului. In functia main a serverului, acesta initiaza intr-o bucla while cate un thread pentru fiecare client conectat, in cadrul caruia va retine progresul clientului in runda la care a fost asignat, intrucat serverul suporta mai multe runde ruland in acelasi timp.

3.2 Clientul

In client este creata interfata grafica a aplicatiei, iar la nivel de functionalitate in retea, acesta se conecteaza la server, trimitandu-i acestuia un username care ii va permite asignarea la o runda. Ulterior, odata runda inceputa, acesta receptioneaza intrebarile una cate una de la server si trimite raspunsul introdus de catre jucator, la final, jucatorul ramas conectat pana la urma primind clasamentul final total al rundei.

Diagrama se regaseste pe ultima pagina a documentului.

4 Aspecte de Implementare

4.1 Server Multithreading

Pentru a retine progresul fiecarui client, serverul creeaza cate un thread pentru fiecare, folosind pthread.

```
    pthread_t thread_id;
    if (pthread_create(&thread_id, NULL, client_handler, (void *)client_info)
        != 0)
    {
        perror("[server] Eroare la crearea thread-ului.\n");
        free(client_info);
        close(client);
    }

    pthread_detach(thread_id);
```

4.2 Mecanism de sincronizare Mutex

Mutexul, asa cum am mentionat la sectiunea 2 (Tehnologii aplicate) il folosesc pentru a proteja o sectiune critica de cod, astfel incat sa fie accesata de un singur thread la un moment dat, in exemplul de mai jos se poate vedea cum mutexul protejeaza variabila *nrclienti*, astfel ca doar un singur client poate actualiza aceasta variabila intr-un anumit moment. Acelasi mecanism va mai fi utilizat in proiect pentru a retine scorul fiecarui client si castigatorul final al rundei de quizz.

```
    pthread_mutex_lock(&mutex_nr_clienti);
    nr_clienti++;
    printf("[Server]Client nou conectat. Clienti serviti: %d\n", nr_clienti);
    pthread_mutex_unlock(&mutex_nr_clienti);
```

4.3 Comunicare prin socket

Serverul si clientul comunica printr-un socket, acesta asigurand comunicarea bidirectionala dintre ele.

In Server:

```
    struct sockaddr_in server;
    struct sockaddr_in from;
    int sd;

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

In Client:

```
int sd;
struct sockaddr_in server;
struct response res;

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

4.4 Stocarea intrebarilor in XML

Intrebarile si raspunsurile sunt stocate intr-un fisier xml "intrebari.xml", drept exemplu este intrebarea de mai jos:

```
<quiz>
    <question>
        <text>Care este cel mai lung fluviu din Europa?</text>
        <option>A. Dunarea</option>
        <option>B. Volga</option>
        <option>C. Nil</option>
        <correct>Volga</corect>
    </question>
</quiz>
```

Serverul verifica raspunsurile clientilor si actualizeaza scorurile utilizand un parser XML - libxml2.

4.5 Interfata grafica realizata cu biblioteca GTK 3

Aplicatia vine cu o interfata grafica usoara pentru interactiunea cu userul realizata cu ajutorul bibliotecii GTK versiunea a 3-a. Aceasta este implementata in client.c, pastrand toate functionalitatile necesare. In functia main a programului este initializata fereastra grafica avand un thread principal pe care ruleaza logica principala in care clientul comunica cu serverul, urmand ca celelalte subprograme - aspecte ale interfetei sa fie sincronizate cu threadul principal.

```
window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
gtk_window_set_title(GTK_WINDOW(window), "Quizz Game");
gtk_window_set_default_size(GTK_WINDOW(window), 400, 200);

vbox_main = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
gtk_container_add(GTK_CONTAINER(window), vbox_main);
```

Exemplu despre cum este sincronizata interfata grafica, schimbandu-se aspectul acesteia in subprograme, in acest caz, actualizandu-se la primirea clasamentului pentru a-l afisa:

```
g_idle_add((GSourceFunc)switch_to_clasament_interface, clasament);
```

4.6 Protocolul TCP la nivel de aplicatie

Serverul preia intrebarile si variantele de raspuns din fisierul xml si le transmite rundei care a fost initiata. Clientul, la randul lui, trimite raspunsul inainte sa-i expire timpul de raspuns. La final, serverul trimite tuturor clientilor care au ramas conectati pana la final din runda respectiva scorurile acumalate, acestea fiind aranjate conform locurilor ocupate.

4.7 Scenarii reale de utilizare

1. Un client se conecteaza la server si asteapta sa inceapa jocul pentru a primi intrebarile
2. Daca un client nu raspunde la timp, serverul trece la urmatoarea intrebare.
3. In caz ca un client se deconecteaza in timpul jocului, acesta este setat inactiv de catre server fara a crea vreo problema in derularea rundei si a jocului in general, dar reusind astfel sa pastreze progresul sau pana la acel moment pentru a-l putea adauga la clasamentul final.

5 Concluzii

In cadrul acestui proiect am dezvoltat o aplicatie de tip server-client, ce reprezinta un joc de quizz care poate suporta mai multe runde, formate dintr-un numar finit de clienti. Serverul reuseste sa gestioneze clientii din fiecare runda intr-un mod sincron, utilizarea mutex-urilor fiind un element cheie in acest sens. De asemenea, am oferit jocului un aspect prietenos pentru utilizator prin interfata grafica, astfel oricine poate interactiona usor cu aceasta aplicatie.

References

1. Springer LNCS Guidelines, `https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines`
2. Multithreading in C, `https://www.geeksforgeeks.org/multithreading-in-c/`
3. Documentatie TCP, `https://ro.wikipedia.org/wiki/Transmission_Control_Protocol`
4. Mutex vs semaphore, `https://www.geeksforgeeks.org/mutex-vs-semaphore/`
5. Documentatie XML, `https://www.w3.org/XML/`
6. Documentatie GTK3, `https://docs.gtk.org/gtk3/`

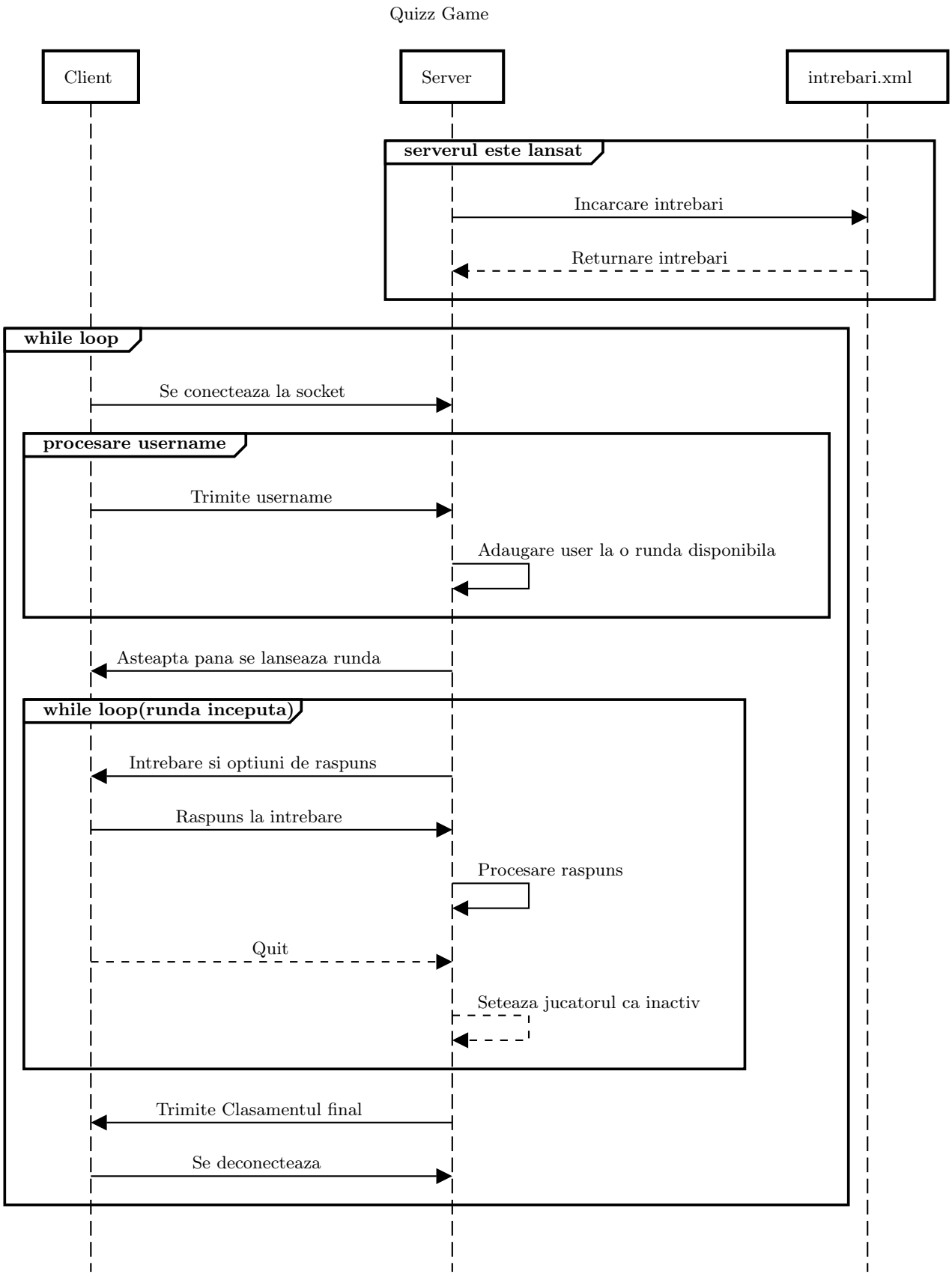

Fig. 1. Diagrama ce reprezinta aspectul de functiune al aplicatiei