QuizzGame (B)

Diac P. Gabriel

Universitatea Alexandru Ioan Cuza, Iasi

Abstract. Raportului tehnic al unei aplicatii bazate pe modelul client-server in c.

1 Introducere

Folosind un server multithreading am realizat un quizz game conform cerintele proiectului propus: un server care onoreaza fiecare client in functie de ordinea in care s-au inregistrat, alocand fiecaruia un numar n de secunde ca sa raspunda la o intrebare (stacata intr-un fisier XML) si posibilitatea de a trimite un raspuns inapoi, comunicare implementata prin socket-uri. Serverul va fi de asa natura incat sa primeasca clienti noi la orice stadiu al jocului si sa-si deplaseze focusul de la un client la altul in functie de timpul alocat si raspunsurile prompte ale clientilor.

Aplicatia dispune de o interfata grafica minimalista.

2 Tehnologii utilizate

2.1 TCP

TCP (Transmission Control Protocol) este un protocol de comunicare intre server si client pe baza unui IP prin intermediul caruia se transmite date odata ce serverul accepta si dupa care recunoaste un client, caruia ii aloca un socket deschis. In cadrul acestui protocol, serverul este cel care asteapta ca un client sa initieze conexiunea si totodata el retine acel specific client, lucra care il face potrivit pentru acest gen de aplicatie, unde avem nevoie sa oscilam intre jucatori si sa le retinem rezultatele. De asemenea, TCP ofera optiune de timeout care este importanta in procesul de monitorizare a clientilor, intrucat fiecare jucator are un timp limitat pentru a raspunde la o intrebare.

2.2 Ncurses

NCurses este o biblioteca grafica simpla, care se lanseaza in terminal. Permite monitorizarea de click-uri si orice actiune de la tastatura. Am ales-o pentru simplitate si usurinta de folosire si instalare. Nu este supraincarcata cu metode si optiuni, suporta structuri repetitive si alte operatii.

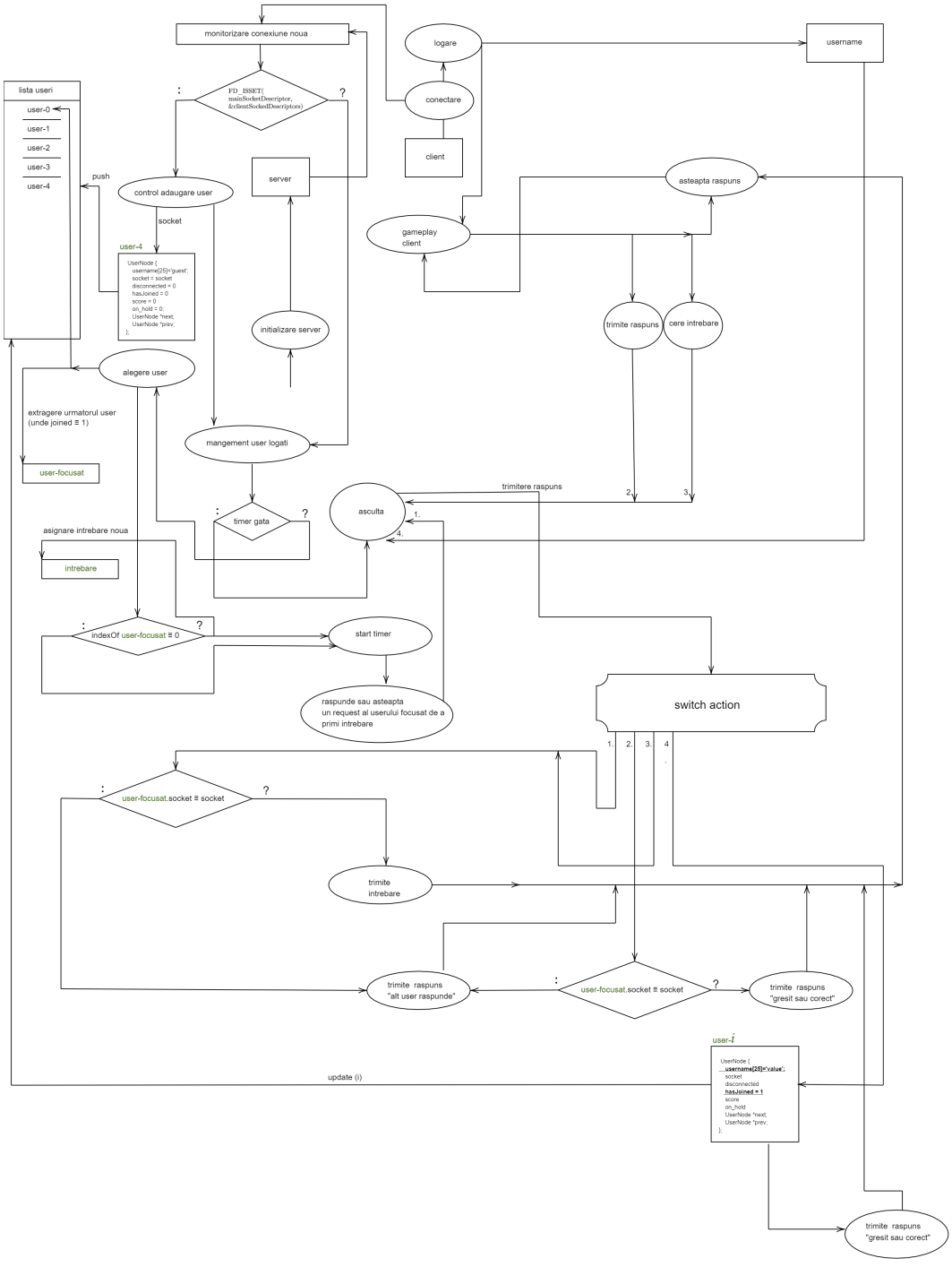
3 Arhitectura

3.1 Concepte

Conceptul central al aplicatiei este acela de "THREE-WAY HANDSHAKE", care impune o sincronizare intre client si server si recunoastere reciproca inainte de orice comunicare. Pe deasupra acestui proces exista inca o preconditie pentru participarea jucatorului: un username primit de la tastatura (care mai mult serveste vizual, intrucat un client este identificat dupa socket-ul asociat). Un client primeste automat la conectare un username unic de "guest" care poate fi utilizat in locul unuia personalizat prin trimitea unui username gol (enter fara a scrie nimic).

Un alt concept utilizat este acela de selectie a clientilor. Serverul ignora activitatea de la socket-urile clientilor in asteptare, cand nicio schimbare nu s-a inregistrat (user-ul care este focusat nu s-a schimbat - lucru ce ar fi necesitat ca ceilalti clienti sa stie - sau jocul inca nu s-a incheiat - lucru ce ar fi necesitat trimiterea tuturor clientilor rezultatele: castigatorul si scorul sau).

3.2 Diagrama detaliata



4 Detalii de implementare

4.1 Sumar

Detaliile care tin de interactiune server-client se pot reduce la urmatoarele: dupa ce conexiunea s-a stabilit, se initializeaza o noua instanta al unei structuri de date cu valoarea socket descriptor-ului, username unic autogenerat, un indicator ca userul inca nu e inclus in joc pana nu confirma un nou username de la tastatura; dupa ce si acest user este primit de catre server si instanta este actualizata, clientul va trimite automat o cerere de intrebare care dupa caz va fi refuzata si drept urmare clientul va primi datele userului care raspunde la momentul respectiv la intrebare pentru a le afisa pe ecranul sau; daca clientul este cel focusat atunci va primi intrebarea si timpul ramas sa raspunda; daca timpul expira clientul va fi tratat ca in cazul anterior si un altul va primi focus-ul.

4.2 Structuri de date

```
struct UserNode {
    char username[25];
    int socket;
    int disconnected;
    int hasJoined; //user chose his username or the guest
   username
    int score; //number of correct answears
    int on_hold; //user will be ignored if he is not the
   focused one untill a change occurs
    struct UserNode *next;
    struct UserNode *prev;
};
struct UserNode *head = NULL;
struct UserNode *last = NULL;
struct UserNode* userInQuestion = NULL;
```

In server.c se itereaza printr-o lista inlantuita a unei astfel de structuri de date si se verifica statusul lor: conectat dar inca nepartipant (isJoined == 0 && socket != 0 && disconnected != 1), deconectat (disconnected == 1), in asteptare (userul este participant si on_hold == 1; clientul a trimis un request pentru a primi intrebarea, serverul ia trimis datele user-ului care este in fata lui drept raspuns, dupa care, in urmatoarele bucle, va ignora alte request-uri din partea clientului respectiv pana cand el sau alt user va prelua randul sau intrebarile expira dupa ce toti au raspuns), "in focus" (este participant, nu este in asteptare si socket-ul sau se potriveste cu o variabila globala de acelasi tip de structura - "userInQuestion").

4.3 Operatii relevante pentru lista de clienti

```
void pushUserNode(char username[25], int socket) {
    if (!head) {
        head = (struct UserNode *)malloc(sizeof(struct
    UserNode));
        head->next = NULL;
        head->prev = NULL;
        head->disconnected = 0;
        head->on_hold = 0;
        strcpy(head->username, username);
        head->socket = socket;
        head->hasJoined = 0;
        head->score = 0;
        return;
    }
    struct UserNode *newNode = (struct UserNode *)malloc(
    sizeof(struct UserNode));
    newNode->next = NULL;
    head->disconnected = 0;
    head->on_hold = 0;
    strcpy(newNode->username, username);
    newNode->socket = socket;
    newNode->hasJoined = 0;
    newNode->score = 0;

    if (last == NULL) {
        head->next = newNode;
        newNode->prev = head;
        last = newNode;
    } else {
        newNode->prev = last;
        last->next = newNode;
        last = last->next;
    }
}
//"data_structures.h"
```

Functia de mai sus are loc la conectarea clientului. Se marcheaza faptul ca user-ul este conectat dar nu participant, si se actualizeaza lista.

4.4 Comunicare

Datele transmise intre server si client sunt siruri de caractere care respecta un anumit tipar simplu prin care acestea sunt decodificare dupa primirea lor. Un astfel de sir respecta urmatorul pattern: '${#cod-comand}~${#continut-relevant-separat}'

Codul de comanda se afla mereu pe pozitia 0 a textului si este un numar ASCII in intervalul [48, 57] (numere de 1 cifra). Desi este mai ieftin ca spatiu

si viteza, este de asemenea greu de citit in cod, asa ca am definit o serie de constante intr-un fisier auxiliar, destul de expresive pentru a fi expuse aici:

```
//prefixele raspunsuri client
#define USER_REQUEST_LOGIN 1
#define USER_REQUEST_GET_QUESTION 2
#define USER_REQUEST_SUBMIT_RESPONSE 3

//prefixele raspunsuri server
#define SERVER_RESPONSE_FAIL 0
#define SERVER_RESPONSE_LOGIN_SUCCESS 1
#define SERVER_RESPONSE_SEND_QUESTION 2
#define SERVER_RESPONSE_SEND_RESULT 3
#define SERVER_RESPONSE_ANOTHER_USER_QUESTIONED 4
#define SERVER_RESPONSE_FINISH 5
#define SERVER_RESPONSE_WAIT 6
#define SERVER_RESPONSE_SESSION_NOT_STARTED 7
//utils.h
```

Continutul relevent care urmeaza separat intre caractere tilda poate varia in functie de ce se doreste a se trimite. Un exemplu ar cand serverul trimite o intrebare:

"2 ∼ text_intrebare∼varianta1∼varianta2∼varianta3∼varianta4∼timp_ramas"

5 Concluzii

5.1 Loc de inbunatatiri

O imbunatatire de mentionat ar fi o varianta mai asincrona a implementarii. In mod curent, atat serverul cat si clientul sunt angajati in structuri de tip bucla infinita si mentin foarte apropiat contactul, care cred eu ca ar fi putut fi minimizate. Uneori clientul are nevoie sa termine o actiune (scriere la tastatura) ca sa observe stadiul jocului.

Interfata grafica este foarte minimalista si nu atat de user-friedly cat ar fi cerut un astfel de joc.

5.2 Concluzii finale

Integrarea TCP-ul ca protocol de comunicare este potrivit pentru o aplicatie care doreste sa identifice clienti si sa mentitina conexiunea recunoscuta, ceea ce se pliaza cu scopurile acestei tip de proiect.

References

1. What Is TCP Three-Way HandShake?, https://www.guru99.com/tcp-3-way-handshake.html
2. Protocol operation, https://en.wikipedia.org/wiki/Transmission_Control_Protocol ,
3. Computer Networks – http://www.info.uaic.ro/ computernetworks