Monitorizarea traficului

Florea Robert-Andrei

Universitatea "Alexandru Ioan Cuza" Iași, Facultatea de Informatică

9 January 2025

1 Introducere

Tema proiectului constă în implementarea unui program(client-care poate reprezenta o persoană ce utilizează mașina și vrea să afle anumite informații referitoare la străzile pe care acesta urmează să meargă), care va trimite diferite comenzi către server(aplicația în sine) acesta urmând să le execute și să trimită un răspuns clienților(în funcție de comanda utilizată). Pe lângă aceste comenzi mai sunt introduse câteva modalități de trimitere a unor mesaje de tip notificare astfel încât fiecare client să poată fie anunțat de o anumită situație din trafic, eventual viteza sa.

2 Tehnologii Aplicate

Proiectul folosește **TCP**(Transmission Control Protocol) pentru a crea conexiuni între client și server și pentru a se realiza comunicarea între acestea. Socket-urile sunt o metodă standard pentru a permite proceselor să comunice prin rețea, în cazul nostru între mașinile (clienții) care vor să se conecteze la server pentru a trimite și a primi informații. Atât serverul, cât și clientul sunt implementate cu socket-uri. Clientul creează un socket folosind socket() și se conectează la server folosind connect(). Clientul și serverul comunică prin mesaje trimise și primite pe aceste socket-uri, iar transferul de date se face prin funcțiile write() și read(). Atât serverul cât și clientul aplicației au fost implementate în limbajul de programare C. Pentru tratarea concurentă a clienților am utilizat metoda de multiplexare a intrărilor de date prin funcția select(), care permite serverului să gestioneze mai multe conexiuni de la clienți într-un mod eficient.
Pe lângă acestea am utilizat biblioteca pthread care permite crearea, gestionarea și sincronizarea thread-urilor. Acestea au fost folosite pentru a trimite notificări de la server la client și în anumite cazuri de la client la server.
Am introdus în program și o bază de date prin intermediul unui sistem de baze date numit SQLite3. Cu ajutorul acesteia s-a făcut stocarea informațiilor despre fiecare client în parte.

3 Structura Aplicației

Aplicația urmează un model client-server. Serverul are rolul de a centraliza informațiile și de a le transmite tuturor clienților conectați. Serverul gestionează

conexiunile, procesează comenzile primite de la clienți și trimite răspunsuri. Clienții sunt șoferii (prin intermediul aplicației client) care trimit informații despre starea lor, incidentele de trafic, și care primesc informații despre trafic, vreme, oferte la peco de combustibil. Acest model permite comunicarea bidirecțională între server și clienți prin socketuri TCP/IP, iar fiecare client poate trimite și primi mesaje în mod asincron. Clienții sunt obligati de a se conecta mai întâi în conturile lor pentru a folosi aplicația. După conectare aceștia au disponibilitatea de a folosi anumite comenzi care le permit de a informa ceilalți participanți la trafic că pe o anumită stradă a avut loc un accident, acest lucru condamnându-i să folosească altă rută pentru a fluidiza traficul, iar pe lângă această comandă, aceștia se pot abona pentru a vedea vremea, de a vedea oferte la peco și eventual informații legate de anumite evenimente sportive.

Acest lucru se poate observa și din următoarea diagramă: **Fig.1**

4 Aspecte de Implementare

Cea mai importantă secțiune de cod este cea în care facem conectarea între clienți și server pentru că fără aceasta nu am putea realiza o conexiune.

```c
/* cream socketul */
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
  {
    perror ("[client] Eroare la socket().\n");
    return errno;
  }


/* umplem structura folosita pentru realizarea conexiunii cu serverul */
/* familia socket-ului */
server.sin_family = AF_INET;
/* adresa IP a serverului */
server.sin_addr.s_addr = inet_addr(argv[1]);
/* portul de conectare */
server.sin_port = htons (port);

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

Fig. 2. Conectarea clientului

În secțiunea de cod de mai sus, clientul creează un socket și încearcă să se conecteze la server folosind IP-ul și portul serverului.

O a doua secțiune importantă a codului este cea în care serverul procesează comenzile de la fiecare client în parte. Fiecare client în parte trebuie să se conecteze la server pentru a folosi celelalte comenzi. În caz că un client nu e conectat are obțiunea de a se conecta pe un cont deja existent sau de a-și crea un cont nou. Dacă acesta încearcă să utilizeze comenzile aplicației fără a fi conectat, serverul îi va specifica că trebuie mai întâi să fie conectat ca după să poată folosi celelalte comenzi.

```
if(strcmp(p,"/register")==0)
{
p=strtok(NULL," ");
p[strlen(p)-1]='\0';
if (check_user_exists(p)==1) {
    strcpy(msgrasp,"Username-ul pe care l-ai introdus deja exista");
}
else {
    if (add_user(p)) {
        printf("%s\n",p);
        strcpy(msgrasp,"Te-ai inregistrat cu succes!");
        set_user_logged_in(p, 1);
        set_client_id(p, fd);
        login[fd]=1;
    }
}
}
else
  if(strcmp(p,"/login")==0)
{
p=strtok(NULL," ");
 p[strlen(p)-1]='\0';
if (check_user_exists(p)==0)
{
    strcpy(msgrasp,"Username-ul pe care l-ai introdus nu exista");
}
else
{
    if (check_user_logged_in(p))
    {
        strcpy(msgrasp,"Deja este cineva conectat pe acest cont");
    }
    else
    {
        if (set_user_logged_in(p, 1))
        {
            strcpy(msgrasp,"Te-ai conectat cu succes!");
            login[fd]=1;
            set_client_id(p, fd);
        }
        else
        {
            strcpy(msgrasp,"Eroare la conectare");
        }
```

Fig. 3. Procesarea comenzilor de către server

O altă bucată de cod importantă este crearea bazei de date. În aceasta sunt stocate informații despre conturi precum:

1. Numele contului
2. Dacă este deja cineva pe acest cont
3. Id-ul clientului
4. Dacă contul este abonat sau nu
5. Viteza cu care se deplasează clientul
6. Strada pe care se află clientul

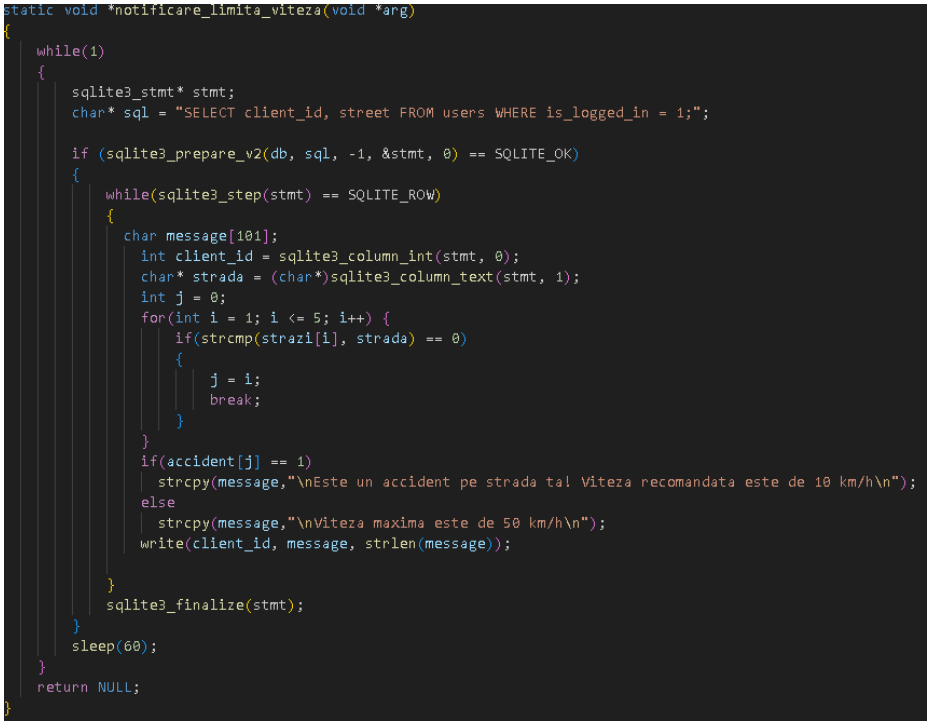
```
int init_database()
{
    char* err_msg = 0;
    if (db_init==1)
        return 1;
    int rc = sqlite3_open("conturi.db", &db);
    if (rc!=SQLITE_OK)
    {
        fprintf(stderr, "Nu poate fi deschisa baza de date: %s\n", sqlite3_errmsg(db));
        return 0;
    }

    char* sql = "CREATE TABLE IF NOT EXISTS users (username TEXT PRIMARY KEY,is_logged_in INTEGER DEFAULT 0,client_id INTEGER DEFAULT -1,subscribed INTEGER DEFAULT 0,
    speed INTEGER DEFAULT 0,street TEXT DEFAULT '');";
    rc = sqlite3_exec(db, sql, 0, 0, &err_msg);
    db_init = 1;
    return 1;
}
```

Fig. 4. Inițializarea bazei de date

De asemenea, trimiterea anumitor notificări de la server la client și de la client la server au fost realizate de către thread-uri. Cu ajutor acestora am putut transmite de la client la server viteza cu care se deplasează fiecare client, iar de la server la client am trimis informații legate de limita de viteză, accidente din zonă și informații despre vreme, oferte peco și evenimente sportive doar dacă clientul respectiv are activat abonamentul.

```
/* servim in mod concurent (!?) clientii... */
pthread_t thread_abonat,thread_limitare_viteza,thread_resetare_accidente;
pthread_create(&thread_abonat, NULL, &notificare_abonat, NULL);
pthread_create(&thread_limitare_viteza, NULL, &notificare_limita_viteza, NULL);
pthread_create(&thread_resetare_accidente, NULL, &resetare_accidente, NULL);
```

Fig. 5. Crearea thread-urilor



```
static void *notificare_limita_viteza(void *arg)
{
    while(1)
    {
        sqlite3_stmt* stmt;
        char* sql = "SELECT client_id, street FROM users WHERE is_logged_in = 1;";

        if (sqlite3_prepare_v2(db, sql, -1, &stmt, 0) == SQLITE_OK)
        {
            while(sqlite3_step(stmt) == SQLITE_ROW)
            {
              char message[101];
                int client_id = sqlite3_column_int(stmt, 0);
                char* strada = (char*)sqlite3_column_text(stmt, 1);
                int j = 0;
                for(int i = 1; i <= 5; i++) {
                    if(strcmp(strazi[i], strada) == 0)
                    {
                        j = i;
                        break;
                    }
                }
                if(accident[j] == 1)
                  strcpy(message,"\nEste un accident pe strada ta! Viteza recomandata este de 10 km/h\n");
                else
                  strcpy(message,"\nViteza maxima este de 50 km/h\n");
                write(client_id, message, strlen(message));

            }
            sqlite3_finalize(stmt);
        }
        sleep(60);
    }
    return NULL;
}
```

Fig. 6. Structura unei notificări

4.1 Scenariu în care putem utiliza aplicația

Aplicația poate fi utilizată de orice persoană de pe întreg pământul care dorește să aibe la cunoștință ceea ce se întâmplă în trafic. Cum ar fi: cu ajutorul acestei aplicații dacă o persoană este pe drum și vrea să meargă pe Str. Lavanda, dar în momentul respectiv este un accident care îi va prelungi timpul în care va ajunge la destinație, acesta va avea la cunoștință că s-a întâmplat acest lucru deoarece aplicația l-a informat și va putea lua altă rută spre destinație, astfel ajungând într-un timp cât mai scurt.

5 Concluzii

În cele din urmă, codul poate face față unui număr rezonabil de utlizatori. Securitatea este slabă deoarece fiecare persoană poate intra pe orice cont, cât timp pe acesta nu este nimeni conectat. Ordinea străzilor este una aleatorie, la fel și viteza cu care se deplasează fiecare mașină. Prima problemă ar putea fi rezolvată prin adăugarea tabele noi numita "Password" în baza de date, astfel încât fiecare client să aibe propriul acces la contul lui. A doua problemă ar putea fi rezolvată prin construcția unui graf neorientat. Străzile ar fi muchiile, iar nodurile ar fi

intersecția dintre străzi. Pentru viteză am putea să o modificăm astfel încât să nu poata obține cineva viteza de +50 km/h pe o stradă unde tocmai s-a produs un accident pentru că în viața de zi cu zi e imposibil ca acest lucru să se poată produce. O altă îmbunătire ar fi adjustarea notificărilor. Serverul mereu o să pornească înaintea clienților, iar din cauza aceasta, unele notificări s-ar putea să se trimită mult mai târziu decât alte notificări.

References

1. Garcia, Felix, and Javier Fernandez. "POSIX thread libraries." Linux Journal 2000.70es (2000): 36-es.https://dl.acm.org/doi/fullHtml/10.5555/348120.348381.
2. Jay Explains. "Multiplexing and demultiplexing in transport layer — port address — socket — transport layer" Oct 9, 2020https://dl.acm.org/doi/fullHtml/10.5555/348120.348381.
3. Jaime Galán Jiménez. "TCP Protocol - Multiplexing and Sockets" Apr 19, 2020https://www.youtube.com/watch?v=XBcmRQXLkXM&ab_channel=JaimeGal%C3%A1nJim%C3%A9nez.
4. Luke Avedon. "Getting started with SQLite in C" Jan 22, 2024https://www.youtube.com/watch?v=EKBFXmjCsRQ&t=1783s&ab_channel=LukeAvedon.
5. GeeksforGeeks. "Multithreading in C" Oct 11, 2024https://www.geeksforgeeks.org/multithreading-in-c/.

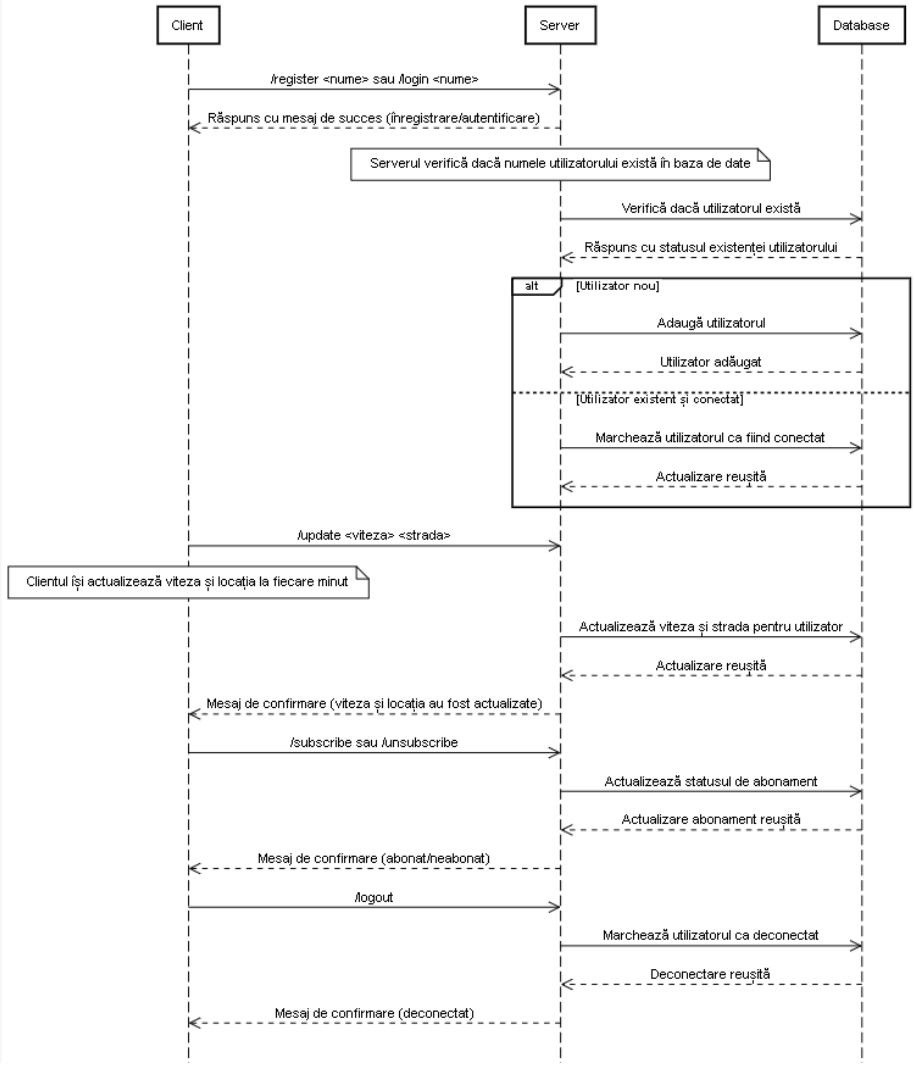


Fig. 1. Diagramă