Monitorizarea Traficului (A) - Documentația proiectului

Filip Tudor-Mihail, Grupa B2, Anul 2

12 Ianuarie 2022

Rezumat

Pentru început vreau să menționez că această lucrare reprezintă documentația referitoare la proiectul final pe care l-am ales pentru materia Rețele de Calculatoare. Proiectul se numeste Monitorizarea Traficului și este proiect de tip A. Pentru început voi prezenta o scurtă introducere și motivare în legătură cu motivul pentru care am ales acest proiect, apoi voi încerca să prezint pe rând detaliile tehnice ce țin de implementarea lui, cum ar fi tehnologiile utilizate, arhitectura aplicației sau detalii de implementare. Modelul de server ales este unul concurent, utilizând un server de tip *TCP/IP*. Pe lângă acest amănunt, proiectul este scris în limbajul *C/C++*, în care voi folosi și baze de date folosind *SQLite*. Concluziile finale reprezintă ce alte îmbunătățiri s-ar putea aduce proiectului, cât și unele lucruri ce țin de actuala stare a proiectului.

1 Introducere

Acest raport prezintă o documentație care face referire la proiectul Monitorizarea Traficului, proiect final ales ca temă pentru materia Rețele de Calculatoare. Proiectul ales este unul complex, drept urmare, în cele ce vor urma, voi prezenta mai multe detalii legate de implementare, tehnologii utilizate sau arhitectura pe care o are la bază aplicația.

Țin să menționez faptul că proiectul chiar dacă se află într-un stadiu final, unele caracterisiti prezentate în această lucrare s-ar putea să fie schimbate sau modificate în viitor.

2 Motivație

Am ales să lucrez și să dezvolt această aplicație din mai multe motive. Unul dintre ele face referire la faptul că proiectul este unul foarte ofertant, având capabilitatea să includ în el și să dezvlot diferite caracteristici. Acest lucru ar putea să mă ajute să îmi dezvolt și mai bine unele cunoștințe pe care le aveam deja, cât și să descopăr tehnologii și metode de programare noi pentru mine. Un al doilea motiv este unul personal. Proiectul, cel puțin așa cum îl vizualizez eu, seamană puțin și cu alte aplicații deja folosite în toată lumea, cum ar fi ***Waze*** sau ***Google Maps***, care sunt utilizate pentru a ajuta șoferii să se orienteze mai bine în trafic, cât și sp găsească traseul cel mai potrivit pentru ei. Am fost mereu curios să știu cum funcționează acest tip de aplicații și cum ar fi să fac și eu la rândul meu una, iar proiectul mi-a oferit ocazia perfectă să încerc acest lucru. Sunt convins că nu va ieși nici pe departe atât de complex sau bine optimizat ca cele două aplicații amintite mai sus, dar voi încerca, câtuși de putin, să iasă o aplicație care să poată fi folosită cu succes și care să ofere o experiență pentru utilizator cât mai decentă.

3 Tehnologii utilizate

Aplicația pe care o dezvolt trebuie să poata permite conectarea la server a oricât de mulți clienți și, totodată, să le poată raspunde acestora independent și în funcție de cerințele fiecăruia. Astfel, protocolul de comunicare între server și clienți va fi unul de tip *TCP/IP* concurent, unde fiecare client va rula în paralel cu celălalt, serverul fiind cel care va face operațiunile specifice. Motivul pentru care aleg să folosesc modelul *TCP/IP*, și nu *UDP*, este acela că protocolul *TCP/IP* este mult mai sigur și mai fiabil, având certitudinea corectitudinii datelor în fiecare moment, lucru care nu îl putem spune cu certitudine pentru protocolul *UDP*. Chiar dacă folosind un server de tip *UDP* transmisiunea datelor ar fi mult mai rapidă, acest lucru nu reprezintă un avantaj pentru aplicația mea, deoarece am în permanență nevoie ca informațiile transmise să nu se piardă. În plus, voi avea o perioadă de reactulizare a datelor la fiecare câteva secunde, special pentru a avea timp ca datele să se încarce corect la fiecare client. O problemă peste care aș putea da dacă aș folosi un server de tip *UDP* ar fi, de exemplu, că nu aș vrea ca din cauza unei erori de transmisie clientul meu să nu mai primească viteza afișată corect sau, și mai rău, să fie localizat greșit pe hartă.

Alt lucru important care ar trebui menționat este tipul de baze de date pe care îl folosesc. Am decis să folosesc *SQLite*, deoarece am nevoie de o bază de date relațională și ușor de folosit, factori care m-au decis să aleg acest tip de baze de date. Versiunea folosită în proiectul meu este 3.37.0.

Limbajul de programare în care este scrisă aplicația este *C/C++*. Motivul este

acela că una dintre cerintele date în enunțul proiectului este aceea să fie scris în *C/C++*. Pe lânga asta, limbajul este foarte permisibil cu ceea ce am nevoie să implementez și îmi asigură o mare diversitate de librării din care pot alege pentru a dezvolta aplicația așa cum doresc.

4 Arhitectura aplicatiei

Aplicația deservește mai multor clienți care vor să vadă în timp real ce traseu au de parcurs între două puncte de pe o hartă, viteza cu care îl parcurg, cât și alte informații relevante cum ar fi starea vremii, ultimele știri sau prețurile la carburanți din zona în care se află. Știind detaliile acestea, îmi pot împărți aplicația în mai multe părți pentru a fi mai ușor de explicat ce funcționalități are.

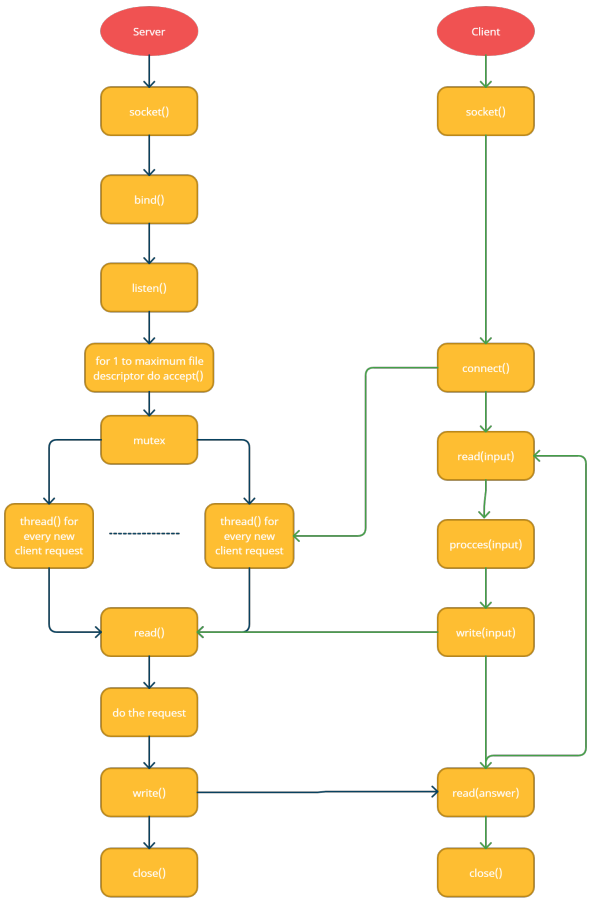
Astfel, așa cum am spus și la punctul anterior, cel legat de tehnologiile utilizate, am utilizat un sistem de baze de date folosind *SQLite*. Tabelele pe care le-am creat și le folosesc sunt următoarele:

- Tabela cu prețurile de la benzinării, ce memorează ID-ul unei benzinării, drumul pe care se află, adică cele două orașe care se află la extremele drumului, cât și prețul motorinei și al benzinei din acea benzinărie.
- Tabela cu știri, ce memorează ID-ul știrii, știrea în sine, adică mesajul transmis, cât și canalul de știri care emite știrea.
- Tabela cu informații despre vreme, ce reține ID-ul unei astfel de informații, împreună cu regiunea geografică specifică, ora pentru care e data prognoza și temperatura respectivă.
- Tabela ce memorează drumurile de pe hartă, ce reține ID-ul străzii, tipul de drum, punctele extreme de la capătul drumului, adică ID-urile celor două orașe aflate la capete, cât și lungimea pe care o are drumul.
- Tabela de memorează toate orașele de pe hartă, ce reține ID-ul unui oraș, numele acestuia, coordonatele la care se află pe hartă, cât și regiunea geografică în care se află.

Pentru a putea deservi toti clienții în funcție de cerințele fiecăruia, așa cum am mai zis la punctul anterior, serverul folosit va fi unul concurent, care stă și așteaptă ca clienții să trimită cereri. Concurența se realizează creând pentru fiecare nou client adăugat câte un thread, care se va ocupa cu gestionarea necesităților clientului respectiv. În cadrul fiecarui thread se trimit informații către client legate de ultimele știri, vreme, prețurile de la stațiile de carburanți. Toate aceste informații se trimit doar dacă clienții au acceptat la început să le fie trimise.

Clientul va avea și el acces la tabela orașele, pentru a putea calcula traseul cel mai potrivit pe care se poate deplasa între două orașe. Totodată, clientul așteaptă de la server date, în mod constant, pentru viitoarele actualizări.

Diagrama care prezintă arhitectura aplicației din punctul de vedere al legăturii dintre server și client este următoarea:

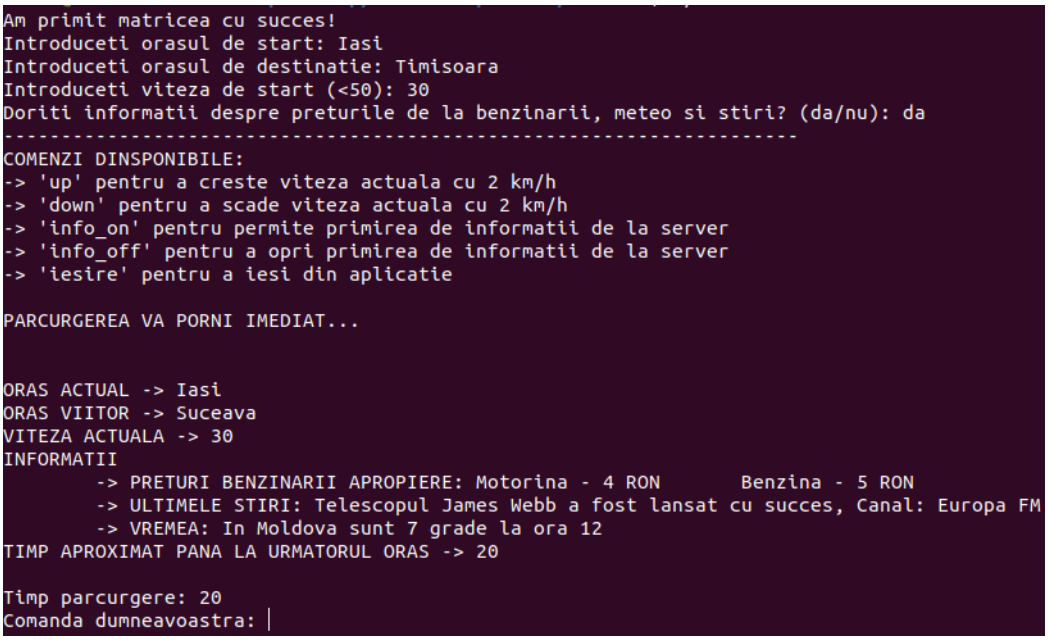


5 Detalii de implementare

5.1 Client

Când un client încearcă să se conecteze la server, acesta îi va oferi în consola poziția de unde vrea să înceapă traseul (adica orașul), și destinația unde vrea să ajungă, plus viteza cu care dorește să circule. Dacă viteza inițială este mai mare de 50 km/h, se va returna un mesaj de eroare și se va cere introducerea din nou a datelor. În momentul

în care un client se conectează cu succes la server îi va apărea în consolă meniul de comenzi pe care acesta îl poate folosi. Clientul va putea crește și scade viteza sa cu 10 km/h, sa oprească sau să pornească primirea de informații, cât și să aleagă să iasă din aplicație.



Așa cum am spus, clientul își alege de la bun început un punct din care începe traseul și un punct de destinație. Pentru a afla traseul cel mai scurt pentru el se va utiliza algoritmul lui Dijkstra pe drumurile din hartă. Ca acest lucru să fie posibil serverul trimite către client o matrice de adiacență ce contine valorile rapoartelor dintre lungimea drumului și viteza maximă admisă pe el.

Pentru a fi capabil să realizez toate aceste calcule și funcționalități, în client voi avea mai multe structuri, cum ar fi:

```
//DECLARARE VARIABILE GLOBALE
float hartaTrafic[20][20];                      //matricea folosita pentru a memora harta
vector<pair<unsigned int, unsigned int>> traseu; //vector ce retine drumurile din traseul unui client
int sd;                                         //descriptorul de socket

//DEFINIREA STRUCTURILOR
struct Client
{
    unsigned int idClient = 0;
    unsigned int orasPlecareClient_ID = 0;
    unsigned int orasDestinatieClient_ID = 0;
    unsigned int coordonataClient_X = 0;
    unsigned int coordonataClient_Y = 0;
    unsigned int timpParcurgere_XY = 0;
    unsigned int lungimeXY = 0;
    unsigned int vitezaClient = 0;
    bool notificareAccident = false;
    bool informatii = false;
    bool iesit = false;
} client;

struct Aglomeratie
{
    unsigned int coordonataAglomeratie_X;
    unsigned int coordonataAglomeratie_Y;
};
vector<Aglomeratie> aglomeratii;

struct Accident
{
    unsigned int coordonataAccident_X;
    unsigned int coordonataAccident_Y;
    unsigned int timpRamasAccident = 50;
};
vector<Accident> accidente;
```

5.2 Server

Serverul trebuie sa permită conectarea mai multor clienți care să poata merge concurent unul față de celălalt. Lucrul acesta îl realizam folosind protocolul *TCP/IP*, împreună cu lacătul pentru thread-uri *mutex*. Pentru a putea gestiona corect fiecare client în parte și pentru a avea o evidență exacta a tuturor evenimentelor, în server voi avea 3 vectori, fiecare vector având ca tip de elemente o structură. Voi avea astfel o structura ce memorează informațiile cheie despre un client, cum ar fi coordonatele X si Y unde se află, drumul pe care se află, știind orașul din care pleacă și orașul în care trebuie sa ajungă, un flag pentru accident care inițial este inițializat cu false, încă unul pentru informații, care este la fel, inițializat cu false, și o variabială în care este salvată viteza lui actuală. O a doua structură va ține minte coordonatele unei aglomerații de mașini. Aglomerarea apare atunci când sunt mai mult de 4 mașini pe aceeași porțiune de drum, iar toate mașinile ce trec pe acolo vor avea viteza încetinită. Cea de a 3-a structură este legată de accidente, memorând coordonatele unde a avut loc, cât și timpul pe care trebuie să îl aștepte mașinile până când accidentul dispare. În momentul în care o mașină marchează ca pe drum este accident se oprește, și toate mașinile care trec prin același loc se opresc și ele.

```
float hartaTrafic[20][20]; //matricea folosita pentru a memora harta
int nrClienti = 0;         //numarul de clienti conectati pe server la un moment dat
int sd;                    //descriptorul de socket de ascultare

//DEFINIREA STRUCTURILOR
struct Client
{
    unsigned int idClient = 0;
    unsigned int orasPlecareClient_ID = 0;
    unsigned int orasDestinatieClient_ID = 0;
    unsigned int coordonataClient_X = 0;
    unsigned int coordonataClient_Y = 0;
    unsigned int timpParcurgere_XY = 0;
    unsigned int lungimeXY = 0;
    unsigned int vitezaClient = 0;
    bool notificareAccident = false;
    bool informatii = false;
    bool iesit = false;
};
vector<Client> clienti(20);

struct Aglomeratie
{
    unsigned int coordonataAglomeratie_X;
    unsigned int coordonataAglomeratie_Y;
};
vector<Aglomeratie> aglomeratii;

struct Accident
{
    unsigned int coordonataAccident_X;
    unsigned int coordonataAccident_Y;
    unsigned int timpRamasAccident = 50;
};
vector<Accident> accidente;

//UN ARRAY DE THREAD-URI PENTRU CREAREA UNUI NUMAR DE THREAD-URI DORIT
pthread_t *threadsPool;
//VARIABILA MUTEX CARE VA FI FOLOSITA DE THREAD-URI
pthread_mutex_t mlock = PTHREAD_MUTEX_INITIALIZER;
```

6 Concluzii

Proiectul propus este unul realizabil, dar care necesită mult timp și efort pentru implementare. Din această cauză nu am reusit sa implementez toate caracteristicile sale, iar pe langa asta aplicatia ofera foarte multa libertate in ceea ce te lasa sa faci, lucru care lasa deschisa posibilitatea adaugarii unor noi optiuni si caracteristici in viitor.

Ceva ce s-ar putea face în plus ar fi ca harta utilizată să nu fie una abstractă, ci una reală, folosind harta pusă la dispoziție de Google Maps, folosind unul dintre API-urile lor pentru a o integra. Astfel, experiența utilizatorilor ar fi una mult mai aproape de realitate, oferindu-le posibilitatea de a alege mai multe trasee sau locații existente.

Alt lucru care ar putea să te lase aplicatia să îl faci ar fi acela să te miști cu o mașină într-un mod aleatoriu pe hartă, nu doar atunci când ai un traseu prestabilit, deoarece în viața reală nu folosim localizarea pe hartă doar în momentul în care avem de făcut un anumit traseu.

Ca și îmbunătățire asupra modului în care se alege un anumit itinerariu ar fi modificarea algoritmului lui Dijkstra folosit. Acesta ar putea ține cont nu doar de lungimea pe care o are un drum între două noduri, cât și de viteza medie cu care ar putea face acel drum. Astfel, ne-ar putea oferi la final o combinație dintre cel mai scurt drum și cel mai rapid, soluție mai potrivită pentru tipul nostru de proiect.

Ca și caracteristică viitoare pe care vreau să o implementez ar fi posibilitatea comunicării între clienții aflați în zone apropiate, printr-un serviciu de mesagerie. S-ar putea împărtăși astfel mult mai multe informații și mult mai clar între șoferii participanți la trafic.

7 Bibliografie

[1] Informații despre SQLite:
https://zetcode.com/db/sqlitec/
https://www.sqlite.org/capi3ref.html
[2] Site-ul pe care l-am utimizat să desenez diagrama:
https://creately.com/
[3] Sursa principală:
https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
[4] Informații legate de modelul TCP:
https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
https://www.freecodecamp.org/news/tcp-vs-udp/
https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html
[5] Alte surse:
https://stackoverflow.com/questions/48610207/concurrent-server-tcp-with-select-in-c
https://www.youtube.com/watch?v=io2G2yW1Qk8
https://www.youtube.com/watch?v=LtXEMwSG5-8