iMap – Monitorizarea traficului

Opariuc Rares Ioan – 2A4

1 Introducere

Acest proiect este asemanator cu alte aplicatii de navigare (spre exemplu Waze sau Google Maps). Aplicatia va fi capabila sa gestioneze traficul si sa ofere informatii virtuale soferilor. Acestia vor avea posibilitatea de a raporta incidente din trafic sistemului. Fiecare masina va trimite automat catre sistem informatii despre viteza cu care circula. Aceste update-uri vor fi trimise catre toti participantii la trafic, iar în funcție de ele sistemul va notifica fiecare sofer despre anumite restrictii de viteza (eventual din cauza unui blocaj in trafic).

De asemenea, fiecare sofer va avea optiunea de a primi informatii despre vreme, evenimente sportive sau preturi pentru combustibili la statiile peco.

2 Tehnologii utilizate

Pentru comunicarea intre server si client, se va folosi protocolul TCP pentru o transmisie sigura a informatiilor si, de asemenea, pentru intregritatea datelor. Daca datele nu ar fi trimise corespunzator, utilizatorii ar transmite accidentele pe strazi gresite si astfel sistemul ar inregistra informatii eronate (referitor la blocaje, gradul de aglomerare).

Serverul este concurent, pentru ca este construit un server TCP multi-threaded. Acesta va crea un thread pentru fiecare client pentru a gestiona comunicarea. Pentru fiecare client nou conectat, se vor crea 2 thread-uri astfel: un fir de executie ce face managementul datelor server-client si un fir de executie ce transmite viteza cu care merge clientul (daca parcurge o strada). Pe langa thread-urile ce sunt utilizate in comunicatie, se va crea un nou thread in server ce updateaza la un minut informatiile din oras.

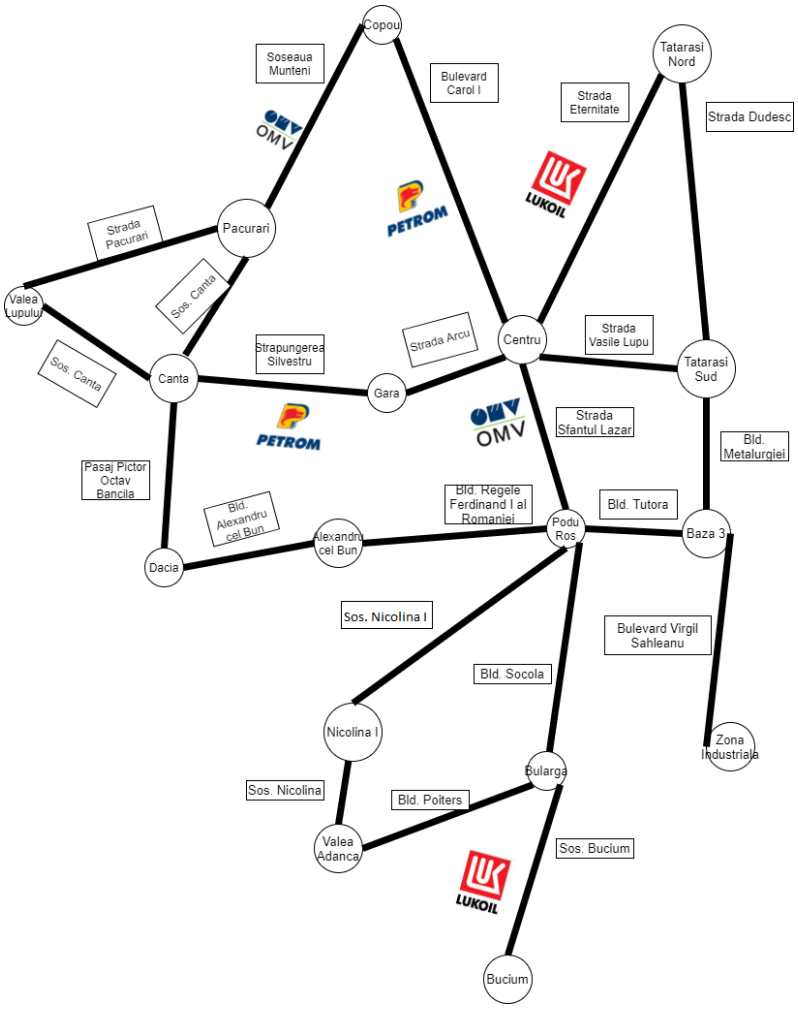
Informatiile despre vreme, eveniment sportive si preturi pentru combustibili vor fi salvate in baze de date. In aceeasi baza de date vor fi salvati si utilizatorii inscrisi la server.

Pentru comunicarea dintre server si client, se vor folosi primitivele read() si write().

In acest proiect orasul Iasi este împărțit în diferite zone (de exemplu: Podu Ros, Copou, etc.) , iar intre aceste zone sunt legate strazile. In baza de date vor fi retinute informatiile astfel: nume_drum, zona1, zona2, distanta, nume_peco, ora1, ora2, ora3, ora4 (sunt unele strazi care nu au un Peco, asa ca aceasta valoare poate sa aiba valoarea NULL). Cand se va porni serverul, se va deschide baza de date si vor fi preluate informatiile corespunzator.

3 Arhitecura aplicatiei

Fig. 1. Harta orasului

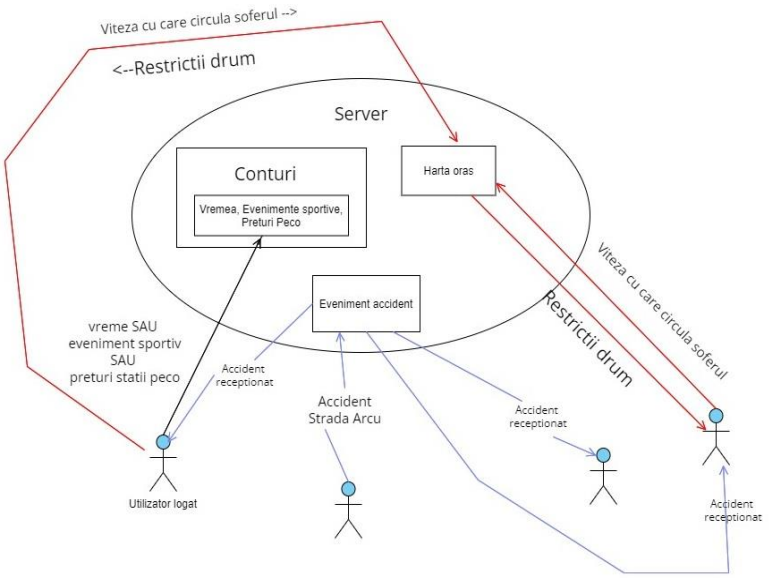


Cand serverul va fi pornit, se va deschide baza de date: strazile si cele 4 intervale orare, vreme, sport, preturi la combustibili si lista cu utilizatori. Dupa ce s-au copiat informatiile din baza de date, se va crea harta orasului.

In momentul in care un client se conecteaza, el va putea sa scrie comenzile care vor fi prezentate la conectare. Pentru ca un client sa parcurga harta, el trebuie sa scrie zona de unde porneste. Daca zona exista, serverul va alege o strada la intamplare adiacenta cu zona de inceput si il va parcurge in functie de gradul de aglomerare a strazii. In baza de date, pentru fiecare strada, vor fi 4 intervale orare (i_1, i_2, i_3, i_4). In intervalul

i_1 si i_2 sau i_3 si i_4, strazile devin aglomerate; in intervalul i_2 si i_3, strazile devin semi-libere; altfel, strazile sunt libere.

Figura 2. Diagrama generala de utilizare



In functie de aglomerarea strazii, serverul va transmite clientului limita de viteza (Daca strada este libera, atunci limita de viteza este 50km/h; daca strada este semi-libera, atunci limita de viteza este de 30km/h; daca strada este aglomerata, atunci limita de viteza este de 10km/h). Clientul va transmite serverului o data la un minut viteza cu care circula. Acel numar se va genera aleator in functie de aglomerare (libera: 30 - 50km/h; semi-libera: 11 - 29km/h; aglomerata: 1-10km/h). Daca clientul depaseste limita de viteza, serverul il va atentiona. Insa, daca clientul va respecta limita de viteza, serverul il va felicita ca respecta regulile de circulatie. Cand clientul a ajuns la urmatoarea zona, se va alege o strada la intamplare adiacenta cu noua zona si va parcurge strada in functie de pasii anteriori. Daca un client a executat comanda “-zona: zonaInceput”, se va plimba pe tot orasul pana cand se va deconecta.

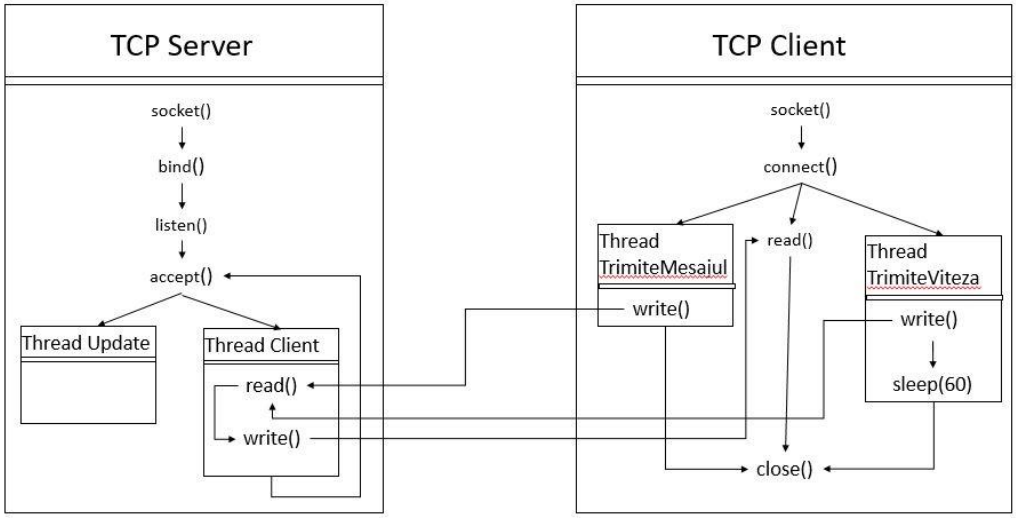
De asemenea, daca clientul executa comanda “-login: <Nume Prenume>” si exista acest utilizator in aceasta lista, atunci clientul va putea accesa celelalte setari. Daca scrie comanda "-vreme", serverul va receptiona mesajul clientului si ii va transmite informatiile despre vreme. La fel se va intampla daca va scrie comanda "-peco" sau "-sport".

Daca un client observa un accident, acesta poate sa scrie "-accident <strada>" si sa notifice restul clientilor (daca strada este scrisa corect si exista in harta orasului). Clientul respectiv transmite informatia la server, iar serverul va transmite restul clientilor evenimentul.

Comunicarea dintre client si server se va face prin thread-ul de scriere si in main().

4 Detalii de implementare

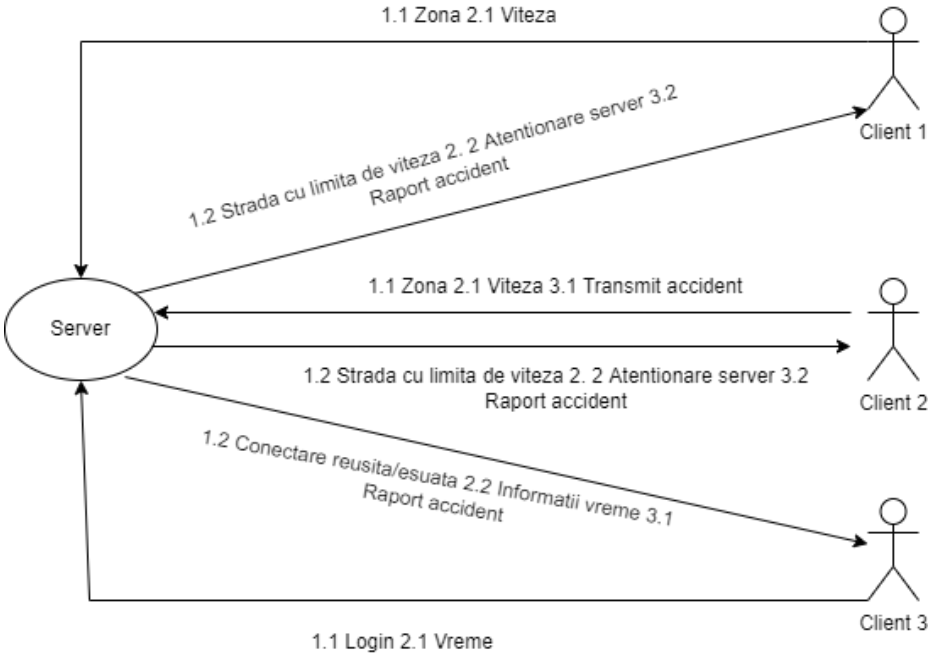
Figura 3. Diagrama detaliata de functionare a aplicatiei



In diagrama este prezentata comunicarea dintre server si client. Serverul se va deschide si va crea un thread pentru fiecare client si un thread pentru updatarea informatiilor din server. Cand un client va trimite un accident la server, se vor folosi descriptorii intorsi de accept si se va transmite mesajul la fiecare client in parte.

Clientul va lua informatia din consola si va scrie in aceasta folosind Thread-ul TrimiteMesaj. De asemenea, daca clientul se plimba prin oras, va scrie viteza o data la 60 de secunde. Serverul va citi mesajul de la client si in functie de mesaj, va intoarce mesajul clientului. Aceasta comunicare se va realiza astfel: se va transmite lungimea mesajului si abia dupa mesajul in sine pentru a se putea face alocarea dinamica a lui.

Figura 4. Diagrama ce descrie un scenariu posibil al aplicatiei



Strazile vor fi retinute intr-o mapa bidimensionala,iar valorile mapei se vor pastra intr-un struct. De exemplu, in harta orasului, "Bld. Alexandru cel Bun" va fi retinut astfel:

```
drum["Canta"]["Dacia"].numeDrum = "Pasaj Pictor Octav
Bancila";
drum["Canta"]["Dacia"].distanta = 5;
drum["Canta"]["Dacia"].numePeco = “NULL”;
drum["Canta"]["Dacia"].ora_1 = “7:00”;
drum["Canta"]["Dacia"].ora_2 = “12:00”;
drum["Canta"]["Dacia"].ora_3 = “16:00”;
drum["Canta"]["Dacia"].ora_4 = “19:00”;
```

Figura 5. Cum o sa fie definite datele orasului

```
struct info{
    std::string numeDrum;
    int distanta = 0; //distanta
    int grad_aglomerare = 0; //daca este libera - 0, semi-libera - 1 sau aglomerata - 2
    int Epeco = 0; //are peco drumul respectiv
    std::string numePeco;
    int ora_1 = 0, minutul_1 = 0, ora_2 = 0, minutul_2 = 0, ora_3 = 0, minutul_3 = 0, ora_4 = 0, minutul_4 = 0;
};
```

De asemenea, cand vor fi retinute datele orasului, vom folosi o mapa auxiliara in care cheia va fi strada, iar valorile vor fi un struct cu cele 2 zone. Vom folosi aceasta

structura ca clientul sa ajunga in urmatoarea zona adiacenta cu strada respective. De exemplu, pentru strada asta, informatia va fi retinuta astfel:

```
drumZone["Pasaj Pictor Octav Bancila"].zona1Aux =
"Canta";
drumZone["Pasaj Pictor Octav Bancila"].zona2Aux = "Da-
cia";
```

Figura 5. Informatii retinute de fiecare thread

```
typedef struct thData{
    int idThread; //id-ul thread-ului tinut in evidenta de acest program
    int cl; //descriptorul intors de accept
    int esteLogat = 0; //daca este logat sau nu, va accesa restul de informatii
    int isWalking;
    int distantaParcursa;
    int distantaActuala;
    char* numeLogatClient; //numele utilizatorului
    std::string stradaActiva;
    std::string viteza;
}thData;
```

Cand se va executa comanda "-zona: <zonaInput>", se vor updata informatiile din thread-ul respective. Vom stii ca serverul a inceput sa se plimbe prin oras. Strada va fi aleasa aleator si se vor updata informatiile din thread astfel:

```
thData.isWalking = 1;
thData.distantaParcursa = 0;
thData.distantaActuala = drum[drumZone[input].zona1Aux][
drumZone[input].zona2Aux].distanta
thData.stradaActiva = input;
if (grad_aglomerare == 0) //libera
    tdLData.viteza = "50";
else if (grad_aglomerare == 1) //semi-libera
    tdLData.viteza = "30";
else
    tdLData.viteza = "10";//aglomerata
```

Intre orele 7:00 si 12:00 si 16:00 si 19:00, acest pasaj este aglomerat (grad_aglomerare = 2). Intre orele 9:00 si 17:00, acest pasaj este semi-liber (grad_aglomerare = 1; altfel grad_aglomerare = 0).

Serverul va transmite limita de viteza, iar clientul va raspunde serverului in functie de viteza. Daca respecta viteza, serverul il va felicita. Insa, daca depaseste viteza, il va atentiona ca depaseste viteza:

```
    viteza_client = rand() % 60 + 1;
```

5 Concluzie

Serverul va crea pentru fiecare client un nou thread. Insa, daca un client se va deconecta, thread-urile create moarte nu vor fi reutilizate si se va crea un nou thread pentru noul client. Aceasta metoda este costisitoare din punct de vedere a memoriei. O metoda mai eficienta ar fi reutilizarea threadurilor moarte sau crearea unui numar de thread-uri cand se porneste server-ul.

La acest server se vor putea conecta foarte multi clieni si se vor crea multe thread-uri care folosesc aceeasi zona de memorie. Unele informatii s-ar putea sa nu se trimita sau se vor citi gresit. O solutie ar fi crearea unui mecanism de blocare (exemplu mutex lock) in client astfel incat un singur thread sa scrie in server.

Putem imbunatati aplicatia prin a folosi algoritmul lui Dijkstra pentru a gasi cel mai scurt drum dintre 2 zone in functie de gradul de aglomerare si numarul de accidente.

O alta metoda de imbunatatire a acestei aplicatii este de a adauga mai multe functionalitati pentru client: putem adauga pe harta diverse restaurante sau supermarket-uri si sa le recomandam cel mai bun restaurant din orasul respective.

6 Bibliografie

https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
https://app.diagrams.net
https://online.visual-paradigm.com/login.jsp?t=diagrams
https://man7.org/linux/man-pages/man3/pthread_create.3.html
https://www.cplusplus.com/reference/cstdlib/rand/
https://www.cplusplus.com/reference/string/string/
https://www.cplusplus.com/reference/map/map/
https://linux.die.net/man/3/pthread_create
https://www.geeksforgeeks.org/implementing-multidimensional-map-in-c/
https://www.geeksforgeeks.org/sql-using-c-c-and-sqlite/
https://www.geeksforgeeks.org/introduction-to-sqlite/