MONITORIZAREA TRAFICULUI

PROIECT LA RETELE DE CALCULATOARE

Lupu Silviu-Cristian

Anul II,Grupa B1

UNIVERSITATEA “ALEXANDRU IOAN CUZA”

FACULTATEA DE INFORMATICA

ANUL UNIVERISTAR 2015-2016

1. Introducere

Intr-o lume in care masinile devin din ce in ce mai puternice si mai rapide ,dorim sa punem pe primul loc siguranta tuturor participantilor la trafic.

Ne-am dori un sistem care sa gestioneze traficul intr-o maniera cat mai usoara ,care sa atentioneze soferii atunci cand apar probleme pe traseele pe care le urmeaza,care sa ii atentioneze asupra posibilelor restrictii impuse pe segmentul de drum pe care se deplaseaza ,fie ele permanente sau temporare ,si ,de ce nu ,sa ii tina la curent cu informatii de interes general.

Prezenta aplicatie incearca sa faca aceasta , oferind o interfata intuitiva si atractiva , care permite participantilor la trafic sa se ajute intre ei raportand diverse incidente rutiere ,care ii atentioneaza cand incalca reglementarile rutiere , dar care ii si tine la curent cu informatii precum stiri sportive,vreme sau update-uri cu privire la preturile carburantilor.

1. Tehnologii utilizate

La baza sistemului de comunicare al aplicatiei sta protocolul TCP (“Transmission Control Protocol”). Acest protocol este folosit in aplicatiile care au nevoie de o livrare fara pierdere a informatiilor.

TCP este un protocol orientat-conexiune,adica pentru transmiterea mesajelor intre doua host-uri este nevoie de realizarea unei conexiuni.

Fiecare punct terminal al unei conexiuni TCP este caracterizat prin o pereche de tipul IP-PORT.

Conexiunile TCP sunt full duplex si se realizeaza prin asa numitul “three way handshaking” . Three way handshaking-ul implica 3 pasi :

* SYN :Clientul face o “deschidere activa” ,trimitandu-i serverului SYN si seteaza SEQ=A
* SYN-ACK : Serverul raspunde cu SYN-ACK; el seteaza ACK=A+1 ,adica confirma ca a primit pachetul A si asteapta A+1 ,si ,seteaza SEQ=B
* ACK:clientul trimite ACK spre server;seteaza SEQ=A+1, si ACK=B+1

Astfel se realizeaza conexiunea intre cele doua puncte ,care pot acum comunica pe un canal fiabil.

TCP este un serviciu de incredere,care garanteaza livrarea corecta a datelor la destinatar ,fara pierderi sau duplicari . El ofera diverse mecanisme de transmisie si control al transmisiei ,cum ar fi confirmarea pozitiva cu retransmisie. Aceasta tehnica fundamentala, consta în faptul ca receptorul răspunde cu un mesaj de confirmare (acknowledgement) de fiecare data când primește un pachet de date. Expeditorul păstrează o copie a fiecarui pachet trimis, și așteaptă confirmarea înainte de a trimite pachetul următor. Expeditorul păstrează, de asemenea, și un timer, atunci când pachetul a fost trimis, și va relua retransmiterea pachetului în cazul în care timer-ul expira iar confirmarea recepției întârzie sa apară. Contorul de timp este necesar în cazul în care un pachet se pierde sau este deteriorat.

Totusi , TCP are inconvenientele sale . Desi garanteaza precizia livrarii mesajelor ,el pierde la capitolul rapiditate ,in favoarea protocolului UDP .Spre deosebire de TCP, UDP nu e orientat conexiune , nu ofera nici un fel de garantie a transmiterii corecte a pachetelor la destinatie , dar , este mult mai rapid. Aplicatii care pot beneficia de rapiditatea UDP sunt streamuri online, client de chat sau voice over IP ,dar poate cel mai importanta aplicatie a UDP este in DNS, unde rapiditatea este esentiala . TCP este preferat de aplicatiile sau serviciile in care integritatea mesajului este importanta ,si nu neaparat rapiditatea de transmitere a mesajului. Dintre acestea amintim transferul de fisiere sau posta electronica.

Aplicatia foloseste protocolul TCP intrucat rapiditatea nu e esentiala ,integritatea datelor fiind mai importanta si se pot admite mici intarzieri in receptionarea mesajelor trimise de clienti. Folosirea UDP-ului ar permite unele vulnerabilitati ,cum ar fi trimiterea cu erori sau esuarea trimiterii unei avertizari a unuia dintre clienti , lucru care ar putea duce la congestionarea traficului sau la penalizarea anumitor participanti la trafic.

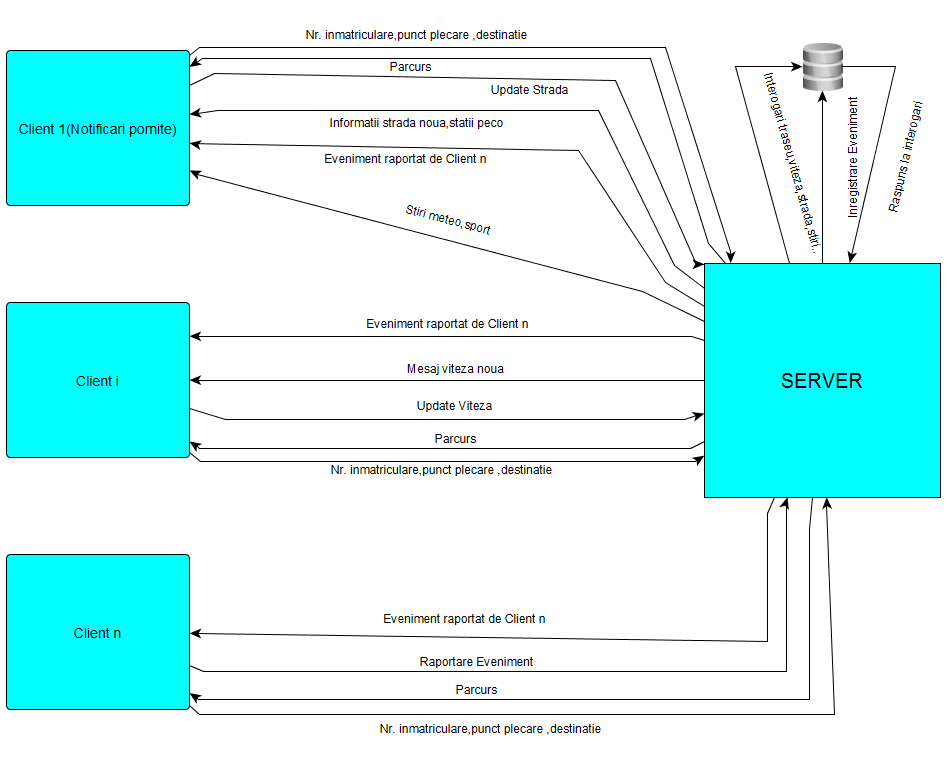
1. Arhitectura aplicatiei

Aplicatia este proiectata pe modelul Client/Server TCP .

Serverul este unul concurent,adica permite deservirea mai multor clienti in paralel . Cand un client initiaza conexiunea la server , dupa stabilirea conexiunii prin “Three way handshaking” , acestuia i se cer primele detalii despre :punctul de plecare, destinatia, drumul pe care circula, viteza si numarul de inmatriculare ,care joaca rol de identificator unic al unui autoturism . Dupa ce serverul preia aceste informatii ,el continua sa deserveasca clientul pana la deconectarea acestuia.

Clientul este cel care initiaza conexiunea si care trimite diferite cereri catre server ,asteptand apoi raspuns .

Diagrama de mai jos reprezinta o sumarizare a arhitecturii aplicatiei:



1. Detalii de implementare

In implementarea serverului au fost folosite fire de executie(threads) ,care permit concurenta dorita . Modul de functionare al serverului este urmatorul : programul principal asteapta cereri de conectare ale clientilor .Atunci cand un client initiaza o cerere de conectare , dupa realizarea conexiunii , se creeaza un fir de executie care va deservi clientul in cauza. Fiecarui thread i se va da spre executie o functie .

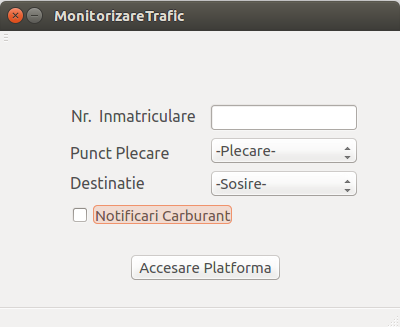
Comunicarea cu clientul incepe prin preluarea a 4 informatii principale :

1)Numarul de inmatriculare al masinii

2)Punctul de plecare

3)Destinatia

4)Daca soferul doreste sa primeasca informatii despre carburant



De asemenea,clientul va trmite si o viteza de deplasare ,care ,la inceput este de 30 Km/h.

Mecanismul de trimitere al stirilor catre clienti este urmatorul : stirile din sport sunt trimise “on demand” ,la apasarea unui buton din interfata grafica , stirile meteo sunt transmise la initierea unei calatorii(daca optiunea de a primi notificari este bifata) si apoi ,periodic ,din 3 in 3 minute . Informatiile despre statiile peco sunt trimise de fiecare data cand se intra pe o noua strada ,atat timp cat optiunea de a primi notificari este bifata. In momentul in care optiunea e debifata, soferul va primi doar notificarile esentiale ,pana cand va reporni notificarile.

Am considerat ca nu e nevoie de un mecanism de logare al clientilor,simpla identificare prin numar de inmatriculare (atunci cand sunt raportate evenimente) fiind suficienta.

Aceste informatii vor fi retinute intr-o structura care defineste clientul; aceasta structura mai retine si descriptorul de socket ,precum si id-ul threadului care deserveste clientul respectiv.

typedef struct thData

{

int idThread; //id-ul thread-ului care va deservi clientul

int cl; //descriptorul intors de accept

int viteza;//viteza de circulatie

char LicensePlate[100];//placuta de inmatriculare

char strada[100]; // strada pe care circula un client

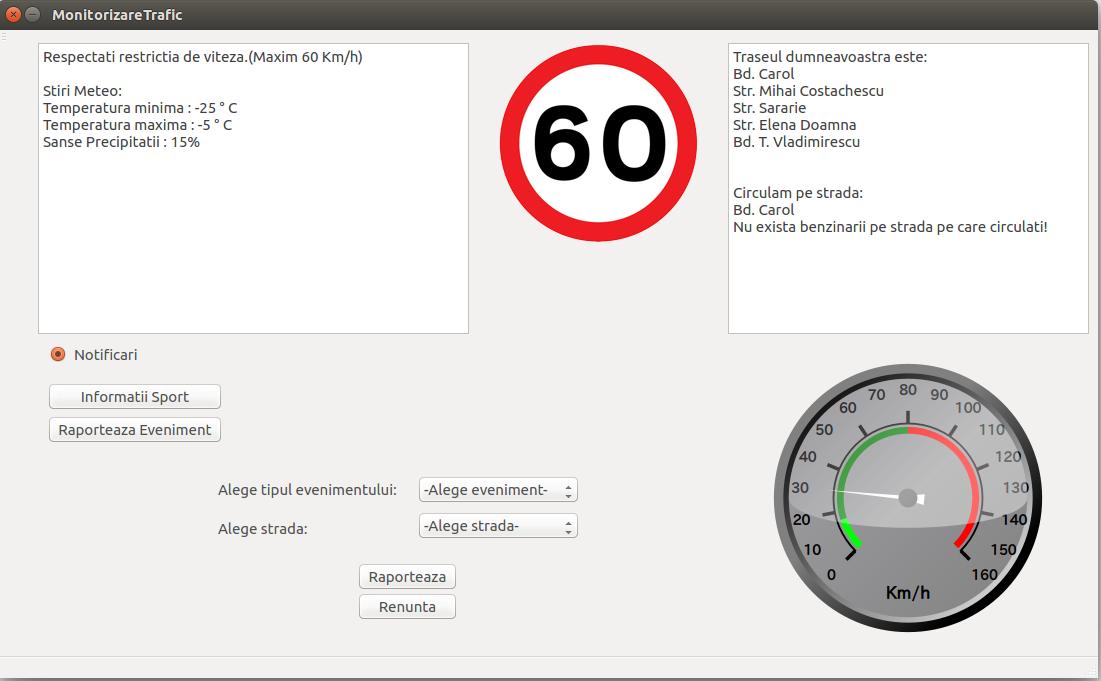
char plecare[100]; // punctul de plecare

char destinatie[100]; //destinatia

}thData;

O data introduse aceste informatii ,ele sufera un set de verificari : utilizatorul trebuie sa fi completat toate campurile , iar campurile “Punct plecare” si “Destinatie” sa fie diferite.

Daca informatiile au fost introduse corect,ele vor fi trimise la server . Serverul va cauta in baza de date ruta dintre cele 2 puncte si o va trimite clientului. Clientul primeste ruta, si trimite catre server prima strada din parcurs.De asemenea,daca optiunea de a primi notificari e bifata,clientul va trimite si un request prin care cere informatii despre statiile peco de pe acea strada. Serverul preia aceasta strada ,verifica daca clientul respecta reglementarea de viteza impusa pe acea strada si trimite informatiile despre statiile peco ,in functie de situatia curenta a optiunii.



In acest moment , clientul va urma parcursul furnizat de server, trimitand cu o anumita periodicitate diverse informatii despre viteza cu care se deplaseaza. Totodata,cand acesta intra pe o strada noua ,serverul este notificat. Vom vedea in continuare fiecare interactiune posibila cu serverul.

* 1. Update viteza

Serverul primeste aceasta comanda periodic, primeste de la client noua viteza de deplasare si in functie de reglementarile existente pe strada pe care se deplaseaza clientul (reglementarile de viteza sunt retinute intr-o baza de date ) va emite mesajul corespunzator : soferul respecta sau nu limita de viteza. Clientul va primi aceasta viteza si va fi notificat vizual in cazul in care depaseste limita legala de viteza. Pe orice segment de drum s-ar deplasa, clientul va sti mereu care e limita de viteza, fiind ajutat de display-ul aplicatiei care ii prezinta aceasta limitare atat sub forma textuala cat si sub forma semnelor de circulatie deja cunoscute de acesta.

* 1. Update strada

Aceasta comanda este primita periodic de server,pana cand clientul ajunge la destinatie. Serverul preia de la client noua strada,verifica daca viteza de deplasare a clientului este in conformitate cu cea impusa pe acea strada si emite raspunsul corespunzator. De asemenea,daca notificarile sunt pornite, serverul va furniza si informatii despre statiile peco de pe acea strada. El va scrie numele statiei,distanta pana la aceasta si pretul principalilor combustibili, pentru strazile pe care exista astfel de statii, respectiv un mesaj negativ pentru acele strazi pe care nu exista statii peco. Daca notificarile sunt oprite,serverul nu va mai interoga baza de date pentru a verifica existenta statiilor .

Cand clientul ajunge la destinatie,serverul este notificat si se realizeaza inchiderea mutuala a conexiunii.

* 1. Raporteaza eveniment

Aceasta optiune este folositoare atunci cand un sofer observa un eveniment in trafic si doreste sa ii anunte si pe ceilalti participanti la trafic. Cand butonul “Raporteaza eveniment” este apasat, clientul va putea alege tipul evenimentului care va fi raportat ( accident , radar, filtru ,liber) si strada pe care s-a petrecut evenimentul . Un client poate raporta un evenimet doar pe o strada care face parte din parcursul lui curent . Daca informatiile au fost introduse corect ,acestea vor fi trimise catre server. Serverul analizeaza tipul evenimentului si actioneaza dupa acesta:

-daca s-a raportat accident , va impune restrictia de viteza 30 Km/h pe strada pe care s-a raportat evenimentul

-daca s-a raportat liber, restrictia de viteza va fi ridicata,revenindu-se la vechea restrictie

Indiferent de natura evenimentului si de strada pe care aceasta a avut loc , fiecare client conectat la server va primi o notificare care il va informa despre tipul evenimentului ,strada pe care a avut loc si cine l-a raportat.

Pentru a tine evidenta clientilor conectati la aplicatie am folosit structura de date fd\_set: de fiecare data cand un client se conecteaza la server ,descriptorul de socket corespunzator acestuia este introdus in fd\_set , iar la deconectare este eliminat.

* 1. Info Vreme

Informatiile despre vreme vor fi furnizate de catre server doar in cazul in care clientul a ales sa primeasca aceste informatii. Un astfel de client va primi o notificare cu temperaturile minime si maxime , precum si sansele de precipitatii , la inceputul calatoriei ,si apoi periodic.

* 1. Info Sport

Informatiile despre sport sunt furnizate la cerere ,indiferent daca notificarile sunt oprite sau pornite in momentul efectuarii cererii .

* 1. Info Peco

Informatiile despre statiile peco sunt furnizate de catre server la inceputul calatoriei si ori de cate ori se intra pe o noua strada, doar clientilor care au activa optiunea de a primi notificari

* 1. Paraseste aplicatia

Daca unul din clienti va inchide prematur conexiunea,serveul va fi notificat.Cand serverul primeste aceasta comanda, starea de asteptare a firului de executie se va rupe , acesta continuand cu inchiderea conexiunii cu clientul. Cand un fir de executie isi va termina executia ,memoria ocupata de el va fi eliberata . Acesta este si unul dintre motivele pentru care am ales un server threaded , firele de executie reprezentand o alternativa mai eleganta si mai eficienta de parelelizare a deservirii clientilor .

Clientul are rolul de a usura interactiunea utilizatorului cu serverul.

Acesta afiseaza mesaje intuitive si lesne de inteles si are o interfata atractiva. Clientul trateaza erorile care pot aparea pe parcurs,asigurandu-se ca serverul va primi mereu informatii corecte .

1. Concluzii

Aplicatia curenta ofera deja functionalitati demne de luat in seama , insa este departe de a fi perfecta.

Aplicatia ar putea beneficia de pe urma adaugarii in baza de date a unei tabele care sa retina orice fel de eveniment raportat . Acest lucru ar putea fi folosit pentru ca atunci cand un sofer va intra pe o strada anume , acesta sa primeasca informatii despre toate evenimentele care nu au fost eliberate pe acea strada.

De asemenea , implementarea unui sistem de logare ar aduce optiuni precum stabilirea unor trasee preferate.

Nu in ultimul rand , s-ar putea mari baza de date cu trasee ,s-ar putea integra un gps, adauga un sistem care ofera stiri in timp real si s-ar putea implementa un meniu de configurare al interfatei grafice.

1. Bibliografie

* Cursurile disciplinei “Retele de Calculatoare”, de Lenuta Alboaie: http://thor.info.uaic.ro/~adria/teach/courses/net/cursullaboratorul.php
* https://en.wikipedia.org/wiki/Transmission\_Control\_Protocol
* <http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm>
* <https://forum.qt.io/category/4/qt-development>
* <http://pytricity.com/qt-circular-gauges/>
* <http://stackoverflow.com/questions/3661285/how-to-iterate-through-a-fd-set>
* http://iasi.harta.orasultau.ro/