Rețele de calculatoare
Tema2- Monitorizarea traficului

Cheptanariu Dorin Grupa B1 Anul II
cheptanariudorin@yahoo.com

Universitatea Alexandru Ioan Cuza
Facultatea de Informatica

1 Introducere

Proiectul ales de mine este un proiect de tip A : Monitorizarea traficului . Acest proiect are ca și tema o aplicație de tipul client/server care monitorizează traficul dintr-un anumit oraș.

Aceasta aplicație are rolul de a transmite șoferilor informații despre trafic cum ar fi : viteza de deplasare pe porțiunea de drum pe care se află , diverse aglomerări care apar în trafic și pe care aceștia să încerce să le evite , accidente ,radare , și chiar informații despre vreme , evenimente sportive și prețurile combustibilului în stațiile de alimentare.

Aplicația poate fi foarte folositoare în cazul în care șoferii se grăbesc și doresc să evite aglomerația sau accidentele , sau chiar să alimenteze dintr-o stație de alimentare cu preț mai redus.

2 Tehnologii utilizate

Am ales sa folosesc protocolul TCP/IP pentru a realiza conexiunea intre serverul concurent și client deoarece :

- Transmitem informații din trafic , care sunt importante pentru conducătorii auto , și nu dorim ca acestea să fie greșite , punând în pericol viața utilizatorului (exemplu: trimiterea unei viteze legale incorecte), astfel că avem nevoie de un protocol ce asigură integritatea octeților și ordinea corectă a transmiterii.
- Este mai important ca informațiile ajunse la client să fie corecte și mai greu , decât să ajungă mai repede , dar incorecte.

Pentru a memora harta străzilor orașului , am folosit un fișier .xml în care am stocat numele străzilor , lungimea acestora și punctele extreme , pentru a fi stocate într-o matrice , acest fișier fiind disponibil și în server și în client.

Serverul are la dispoziție și alte fișiere .xml din care citește viteza legală admisă de pe fiecare strada , evenimente sportive , prețurile combustibilului și starea vremii în ziua respectivă.

Am ales ca proiectul sa fie scris în limbajul C ,pentru a fi mai usor de lucrat in sistemul de distributie Linux.

Atât serverul , cât și clientul sunt concurente . Serverul a fost necesar sa fie concurent pentru a putea servi toți clienții în același timp , iar clientul pentru a putea și citi o informație fie importantă sau nu , și pentru a trimite serverului o informație importanta , precum un accident sau radar.

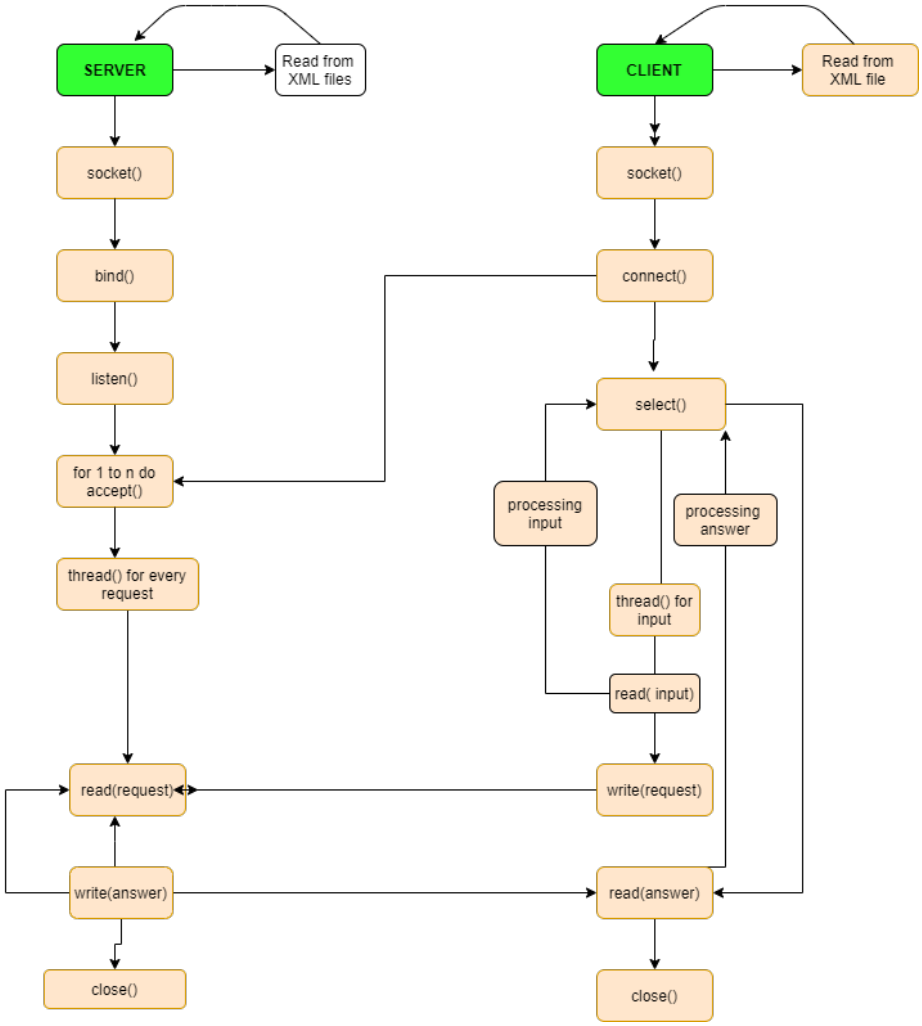
3 Arhitectura aplicației

Proiectul folosește un server concurent , care așteaptă conectări din partea clienților în orice moment.
Concurenta serverului este asigurata prin creearea unui thread pentru fiecare client conectat, acesta ocupandu-se in totalitate de respectivul client.Acesta folosește o matrice pentru a memora străzile și punctele de intersecție ale acestora ,iar pozitia fiecarui sofer va fi stocata in memorie pentru a se putea face verificarea unei posibile agolomerari in trafic.

Pe partea de client,pe baza primivitei *select*() pentru ca acesta sa fie concurent , pentru a realiza output-urile de la tastatura , se foloseste un thread care citeste input-ul dat de utilizator , iar main clientul se ocupa de citirea din partea serverului .De asemenea clientul va avea o matrice în care va retine rețeaua stradală , pentru ca acesta sa poată continua pe alta strada când este la capătul uneia.

Aplicația se folosește de 5 fișiere .xml , unul pentru rețeaua stradală , unul pentru vitezele legale , unul pentru prețurile combustibilului, unul pentru evenimentele sportive și unul ce va conține informațiile despre vreme, ultimele 4 fiind accesibile doar serverului.

Diagrama aplicației este următoarea :



4 Detalii de implementare

Utilizarea clientului se realizeaza in consola unde utilizatorul are cateva comenzi la dispozitie. Acesta își va seta locația curenta , și viteza cu care se deplasează , acestea fiind trimise serverului.

Clientul va trebuie sa isi seteze o destinatie , pentru a putea simula drumul pe care il parcurge acesta , aplicandu-se algoritmul lui Dijkstra , pentru a calcula cel mai scurt drum posibil. După aceea clientul va putea să-și modifice viteza cu care se deplasează cu ajutorul comenzilor *up* si *down* , trimițând constant informații serverului despre viteza de deplasare și poziția sa .

În cazul în care un client are în drumul său un accident sau un radar , acesta

va scrie in consola comanda accident , iar clientul se va ocupa sa trimita serverului locatia accidentului .

Dacă dorește , clientul va putea selecta să primească informații sportive folosind comanda *option* : *evenimente* , preturile combustibilului *option* : *preturi* și starea vremii *option* : *meteo*, iar acestea ii vor apărea în cadrul consolei și se vor schimba constant cu primirea unora noi din partea serverului.

Când clientul va ajunge la ajunge la destinatia dorita , va fi întrebat daca doreste sa calatoreasca la alta locatie .

Pentru a putea citi din fișierele .xml am folosit următorul cod :

```
void readstreets(){
size_t read;
char *row=NULL;
int i,j;
int id,ok;
char name[100];
int dist,x,y;
size_t len=0;
 int len2;
 char buff[100];
fp=fopen(streetsfile,"r");
if(fp==NULL){
printf("Error opening file %s",streetsfile);
return;

}else{

while((read=getline(&row,&len,fp))!=-1){

len2=strlen(row);
row[len2-1]='\0';
if(strstr(row,"ROW")){
    ok=0;
    for(i=0;i<5;i++){
        read=getline(&row,&len,fp);
        if(strstr(row,"id")){
strcpy(buff,gettext(row));
id=atoi(buff);
ok++;

        }
        if(strstr(row,"name")){
strcpy(buff,gettext(row));
strcpy(name,buff);

ok++;
        }
        if(strstr(row,"p1")){
strcpy(buff,gettext(row));
x=atoi(buff);
ok++;

        }
        if(strstr(row,"p2")){
strcpy(buff,gettext(row));
y=atoi(buff);
```

în care parcurg fiecare element al fișierului , iar dacă acesta aparține unei coloane din fișier , voi retine valoarea într-o variabilă locală , iar la sfârșitul parcurgerii liniei va fi asignată structurii pe care am creat-o pentru a reține rețeaua stradală.Structurile folosite sunt următoarele:

```
int nr_streets;
struct strazi{
    char name[100];
    int dis;
 }st[50][50];
 struct streets2{
char name[100];
 int dist;
 int x,y;

 }st2[50];
 struct position{
    int x,y,offset,dir;
 }pos,des;
 struct route{
  int dis,next;
 }ruta[100];
```

În care prima structură (*strazi*) este o matrice folosită pentru a face legătura între străzi , și pentru a putea ști pe ce străzi poate continua si a putea calcula drumul minim .

A doua structura o folosesc pentru a calcula mai ușor când un client ajunge la capătul unei străzi.

Structura position o folosesc pentru a retine poziția clientului de-a lungul călătoriei sale , astfel ca x si y vor reprezenta capetele unei străzi (2 intersecții) , iar offset pentru a ști cât a parcurs clientul de-a lungul străzii xy ,iar *dir* este pentru a sti pe ce sens de mers este clientul.

La partea de server , acesta creeaza o structură de tip server și așteaptă clienții să se conecteze , iar pentru fiecare utilizator conectat va crea un thread care sa se ocupe de acel utilizator , va citi mesajele În cazul în care mesajul primit raportează un accident pe o anumita stradă , thread-ul va trimite de indata mesajul tuturor clienti .

Pentru a putea monitoriza aglomerările din trafic , serverul se va folosi de o structură în cadrul căreia va memora pentru fiecare client poziția sa , iar dacă mai mult de 3 clienți se vor afla pe aceeași stradă la un offset apropiat unul de altul , serverul va trimite tuturor clienților un mesaj precum ca pe acea strada

are loc o aglomerare în trafic .

Pentru clienții care doresc și informații despre evenimente sportive, preturi combustibil sau vreme , serverul va trimite clientului respectiv informațiile dorite , care vor fi citite de asemenea dintr-un fișier .xml .

```
void checkaglomeration(){
  int i,j;
  char buf[300];
  bzero(buf,sizeof(buf));
  int t,nr;
  char str[300];
  for(i=0;i<maxc;i++){
if(clienti[i].active){
  t=1;
  for(j=i+1;j<=maxc;j++){
    if((pos[i].x==pos[j].x)&&(pos[i].y==pos[j].y)&&((((pos[i].offset-pos[j].offset)<=50)||((pos[j].offset-pos[i].offset)<=50)))){
      t++;
    }
     if((pos[i].x==pos[j].y)&&(pos[i].y==pos[j].x)&&((((pos[i].offset-pos[j].offset)<=50)||((pos[j].offset-pos[i].offset)<=50)))){
      t++;
    }

  }
  if(t==3){
    nr=pos[i].offset/50;
    bzero(str,sizeof(str));
    for(j=0;j<nr_streets;j++){
      if(st[j].x==pos[i].x&&st[j].y==pos[i].y){
        strcpy(str,st[j].name);
        break;
      }
        if(st[j].x==pos[i].y&&st[j].y==pos[i].x){
        strcpy(str,st[j].name);
        break;
      }
    }
    snprintf(buf,sizeof(buf),"agl: Aglomeratie pe %s nr.%d",str,nr);
    for(j=0;j<=maxc;j++){
      if(clienti[j].active){
        if(write(clienti[j].fd,buf,strlen(buf))<=0){
         perror ("Error at write()");
         return;
        }
      }
    }
  }
}
  }
}
```

Aceasta functie o folosesc in server pentru a verifica daca exista aglomeratie in trafic, verificand daca sunt utilizatori la aceeasi adresa la locatii apropiate.

5 Concluzii

O îmbunătățire pe care as putea adăuga-o proiectului ar fi ca , clientul sa își seteze niște locații prestabilite , cum ar fi locația de acasă , de la serviciu sau de la facultate.

O alta imbunatatire pe care as adauga-o proiectului ar fi o interfata grafica , pentru a fi mai usor clientului sa actioneze.

O alta îmbunătățire ar fi folosirea unei rețele stradale reale , cum ar fi cea folosita de Google Maps , care ar putea face aplicația una foarte utila în viata de zi cu zi .

O alta îmbunătățire ar fi ca utilizatorul sa selecteze o locație unde vrea sa ajungă , iar interfața grafica sa îl ghideze ,prin afișarea drumului sau a direcției de mers.

6 Bibliografie

https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
https://www.codeproject.com/Tips/82005/A-count-down-timer-in-c
https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-select