Monitorizarea traficului (A)

Ciot Tudor-Calin
Grupa A2, Anul II

7 Decembrie 2021

1 Introducere

Acest raport tehnic descrie posibilitatea unei implementari a aplicatiei de monitorizare a traficului in cadrul materiei *Retele de Calculatoare*.

Proiectul doreste sa demonstreze capabilitatea de functionare a unei aplicatii de tip client-server concurent cu thread-uri care gestioneaza traficul si ofera informatii virtuale in timp real soferilor.

Functionalitati:

- Soferii pot raporta incidente din trafic in timp real catre sistem.
- Soferii participanti la trafic vor fi notificati in legatura cu incidentele raportate.
- Fiecare masina va trimite automat catre sistem informatii despre viteza cu care se deplaseaza.
- In cazul in care pe o anumita portiune de drum exista restrictii de viteza, soferii vor fi atentionati.
- Soferii pot vedea ce strazi sunt aglomerate.
- In plus, soferii care doresc informatii suplimentare(ex: vreme, evenimente sportive, preturi pentru combustibili, etc...), vor putea bifa aceasta optiune.

2 Motivatie

Am ales acest proiect pentru ca sunt pasionat de mic de masini, imi place sa conduc si sunt familiarizat cu o aplicatie asemanatoare bazata pe idei si funtionalitati similare, pe care o folosesc tot timpul cand conduc.

3 Tehnologii utilizate

Pentru acest proiect m-am gandit in a implementa o aplicatie de tip client-server concurent care va utiliza un protocol de transport TCP/IP, care permite conectarea simultana a mai multor clienti, transferul datelor fiind unul rapid si sigur. Fiecarui client i se va aloca la conectare cate un thread care ii va solutiona cererea, aceasta metoda fiind eficienta din punct de vedere al timpului de raspuns. In plus, vom utiliza si multiplexarea pentru a putea comuta intre socket-urile protocoalelor UDP (pe care il vom folosi deoarece viteza unui autoturism nu o putem aproxima exact si nu ne intereseaza asa mult integritatea datelor) si TCP pentru a trimite date sau comenzi in acelasi timp.

4 Arhitectura programului

Programul este alcatuit din 2 surse de cod, unul pentru client (*client.c*) si unul pentru server (*server.c*), implementate in limbajul C. Cele doua surse vor avea la baza protocoalele de transport TCP/IP concurent si UDP/IP pentru schimbul de informatii de la client catre server si invers. Pentru ca aceste doua protocoale sa comunice in mod eficient, vom folosi *select*() pentru a multiplexa socket-urile TCP si UDP.

Programul este proiectat sa monitorizeze traficul pe o anumita zona dintr-un oras. Spre exemplu, pentru inceput am ales zona Dacia din orasul Iasi. Informatiile despre strazi vor fi stocate intr-un fisier *xml* accesibil serverului. Clientul va introduce de la tastatura numele strazii pe care se afla, informatia fiind transmisa catre server si verificata. Viteza se va aloca in mod automat.

Zona Dacia

Diagrama functionalitatii intre *Client* si *Server*

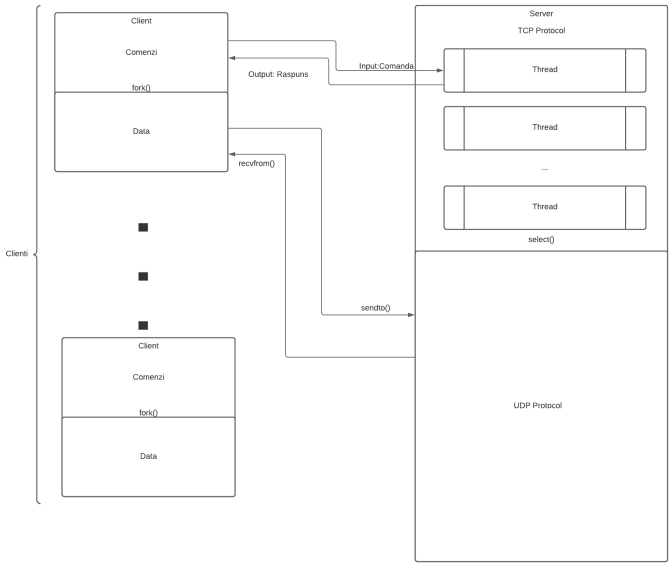


Diagrama pentru *Client* simplificata

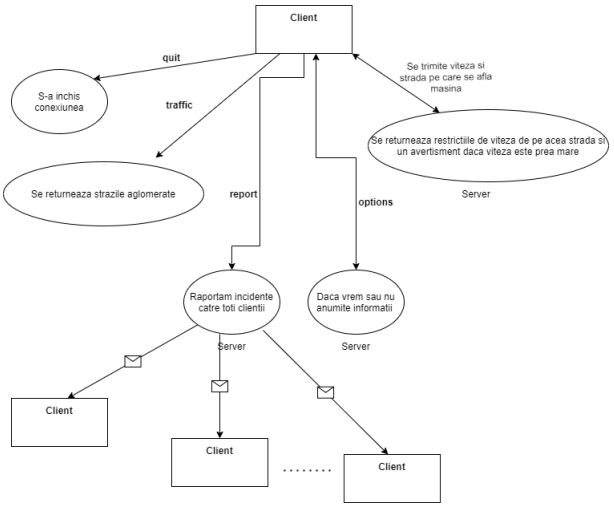
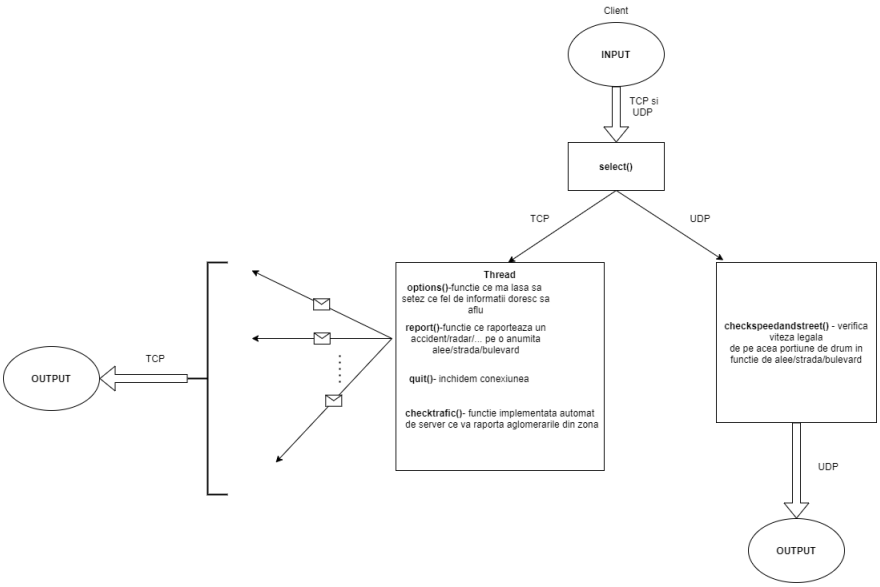


Diagrama pentru *Server* simplificata



5 Detalii de implementare

Pentru inceput, ne vom focusa pe partea de client. Clientul la pornirea programului este rugat sa isi seteze locatia de plecare. Viteza se va aloca in mod automat si va fi trimisa o data la un minut catre server (prin acest lucru am urmarit simularea conceptului de viteza prin gps). Cu ajutorul fork() vom creea un proces copil in care vom pasa aceste informatii si le vom trimite cu ajutorul protocolului UDP inspre server.

Pentru a simula parcurgerea unui drum dintr-un punct A in B vom creea o functie care ne va extrage dintr-un fisier *xml* un numar care va reprezenta id-ul unei strazi o data la o perioada de timp(pentru a simula parcurgerea acelei strazi). Numerele din fisierul *xml* vor fi asezate astfel incat masina noastra sa aiba un drum plauzibil in reateaua stradala (de exemplu sa nu putem sa ajungem de pe Bulevardul Dacia imediat pe Strada Tabacului).

Dupa ce am setat locatia de pornire, vom putea introduce anumite comenzi. Comenzile disponibile sunt:

- **report** - Cand vom introduce aceasta comanda programul ne va intreba ce dorim sa raportam. Dupa ce am selectat ce dorim sa raportam raspunsul se va duce la server, de acolo informatia fiind transmisa catre toti utilizatorii.
- **traffic** - Cand vom introduce aceasta comanda programul va trimite cererea la server, serverul returnand strazile care sunt aglomerate.

- **options** - Cand vom introduce aceasta comanda serverul ne va intreba daca dorim sau nu informarea in legatura cu vremea, evenimentele sportive sau preturi pentru combustibili. Dupa ce am selectat **Da** sau **Nu** raspunsul va pleca spre server, serverul oferind informatiile dorite.

- **quit()** - Cand vom introduce aceasta comanda, vom instiinta serverul ca vrem sa inchidem conexiunea. Acesta ne va scoate din lista socketurilor active si v-a inchide conexiunea.

Pe partea de server, vom multiplexa cu ajutorul lui **select()** intrarile protocoalelor TCP si UDP. Pe partea UDP vom avea o singura functie (**checkspeedandstreet()**) care va procesa datele primite de la toti clientii si va returna un avertisment daca viteza este depasita in functie de tipul de strada pe care se afla clientul atunci. Tipurile de strazi si limitele de viteza:

- **Alee** - 30 km/h

- **Strada** - 50km/h

- **Bulevard** - 60km/h

Pe partea TCP, vom aloca cate un thread fiecarui client care se conecteaza pentru a putea rezolva cererile in mod concurent. Fiecare comanda va avea cate o functie alocata care va rezolva solicitarea. De exemplu, functia **checktraffic()** va folosi un vector de frecventa in care fiecare casuta va reprezenta id-ul unei strazi, iar cifra din casuta va reprezenta numarul actual de masini de pe acea strada. O strada devine aglomerata daca numarul de masini acceptate depaseste o anumita limita. Limitele sunt:

- **Alee** - maxim 3 masini pentru a nu fi aglomerat

- **Strada** - maxim 5 masini pentru a nu fi aglomerat

- **Bulevard** - maxim 10 masini pentru a nu fi aglomerat

Codul functiei **checktraffic()**:

```
9   void checktraffic(int v[],int n)
10  {
11      FILE *fp;
12      char line[256];
13      int id;
14      for(int i=0;i<n;i++)
15      {
16          if(i<3 && v[i]>3)
17          {
18              fp=fopen("strazi.xml","r");
19              fseek(fp,8,SEEK_SET);
20              while(fgets(line,256,fp)!=NULL)
21              {
22                  //printf("%s",line);
23                  strcpy(line,line+8);
24                  //printf("%s",line);
25                  if(strncmp(line,"<id>",4)==0)
26                  {
27                      strcpy(line,line+4);
28                      char aux[2];
29                      strncpy(aux,line,1);
30                      //printf("%s\n",aux);
31                      id=atoi(aux);
32                      //printf("%d\n",id);
33                  }
```

```
34                  if(strncmp(line,"<nume>",6)==0)
35                  {
36                      if(i==id)
37                      {
38                          strcpy(line,line+6);
39                          char aux2[20];
40                          int k=strlen(line);
41                          strncpy(aux2,line,k-8);
42                          printf("%s\n",aux2);
43                          memset(aux2,0,20);
44
45                      }
46                  }
47
48
49              }
50          }
51          if((i>=3 && i<=6) && v[i]>5)
52          {
53              fp=fopen("strazi.xml","r");
54              fseek(fp,8,SEEK_SET);
55              while(fgets(line,256,fp)!=NULL)
56              {
57                  //printf("%s",line);
58                  strcpy(line,line+8);
```

```
60                  if(strncmp(line,"<id>",4)==0)
61                  {
62                      strcpy(line,line+4);
63                      char aux[2];
64                      strncpy(aux,line,1);
65                      //printf("%s\n",aux);
66                      id=atoi(aux);
67                      //printf("%d\n",id);
68                  }
69                  if(strncmp(line,"<nume>",6)==0)
70                  {
71                      if(i==id)
72                      {
73                          strcpy(line,line+6);
74                          char aux2[20];
75                          int k=strlen(line);
76                          strncpy(aux2,line,k-8);
77                          printf("%s\n",aux2);
78                          memset(aux2,0,20);
79
80                      }
81                  }
82              }
83          }
84          if((i>=7 && i<=9) && v[i]>10)
```

```
if((i>=7 && i<=9) && v[i]>10)
{
    fp=fopen("strazi.xml","r");
    fseek(fp,8,SEEK_SET);
    while(fgets(line,256,fp)!=NULL)
    {
        //printf("%s",line);
        strcpy(line,line+8);
        //printf("%s",line);
        if(strncmp(line,"<id>",4)==0)
        {
            strcpy(line,line+4);
            char aux[2];
            strncpy(aux,line,1);
            //printf("%s\n",aux);
            id=atoi(aux);
            //printf("%d\n",id);
        }
        if(strncmp(line,"<nume>",6)==0)
        {
            if(i==id)
            {
                strcpy(line,line+6);
                char aux2[20];
                int k=strlen(line);
                strncpy(aux2,line,k-8);
                printf("%s\n",aux2);
                memset(aux2,0,20);

            }
        }
    }
```

Acest cod poate suferi modificari!

6 Concluzii

Prima imbunatatire a acestui program poate fi o interfata grafica atractiva care sa atraga cati mai multi utilizatori. Pe partea de functionalitate, am putea adauga o functie care sa implementeze un GPS. Introducem punctul de plecare si punctul de sosire, apoi in server trebuie sa adaugam un fisier in care fiecare drum are un anumit cost. Selectam drumul care corespunde punctelor noastre si are costul minim.

O alta imbunatatire o reprezinta schimbarea arhitecturii serverului si clientului. In loc sa facem un fork() in client si sa folosim doua protocoale, putem sa folosim doar protocolul TCP/IP si select() cu OOB data transmission.

In concluzie, arhitectura gandita este una optima deoarece ofera o servire concurenta si sigura pe partea protocolului TCP, dar si rapida pe partea protocolului UDP.

7 Bibliografie

1. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S7/servUdp.c
2. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S7/cliUdp.c
3. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S9/servTcpCSel.c
4. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
5. https://www.geeksforgeeks.org/tcp-and-udp-server-using-select/