Tema 2- Mersul trenurilor
(Rețele de calculatoare)

Racoviță Cristina-Gabriela (Grupa 2A4)

Decembrie 2024

1 Introducere

Aplicația Mersul trenurilor oferă și schimbă informații, în funcție de datele primite de la clienți. Pot fi afișate informații precum : status plecări/sosiri, întârzieri, estimare sosire, mersul trenurilor în ziua respectivă, plecări/sosiri în următoarea oră. Aplicația conține un server (multithreading) și clienți multipli. Informațiile pot fi actualizate în directă legătură cu detaliile date de clienți (cum ar fi întârzieri).

2 Tehnologii aplicate

În cadrul aplicației vor fi utilizate : protocolul TCP, fișiere XML și thread-uri.

Este de preferat utilizarea protocolului TCP în detrimentul protocolului UDP, deoarece protocolul TCP asigură o bună comunicare între server și clienți fără riscul pierderii informațiilor, care ar putea fi necesare în actualizarea datelor și nu numai.

Fișierele XML sunt necesare pentru organizarea datelor și pentru stocarea lor. În cadrul acestor fișiere se găsesc informații utile ce țin de traficul feroviar (întârzieri, sosiri, plecări).

Thread-urile au ca scop administrarea conexiunilor cu mai mulți clienți, fiecare client având alocat un thread, astfel încât cererile să poată fi satisfăcute în același timp, fără ca acțiunile unui client să interfereze cu cele ale altor clienți.

3 Structura aplicației

Componența aplicației include un server, un client și fișiere XML. Serverul este responsabil de procesarea datelor trimise de către clienți, cum ar fi actualizarea datelor sau prezentarea anumitor informații la solicitare(prin accesarea fișierelor XML). Clientul, reprezentând legătura utilizatorului cu aplicația, inițiază solicitări la care serverul răspunde. Instrucțiunile disponibile sunt: $info\_trenuri$

(oferă informații legate de trenurile care circulă azi), *info_tren_ora* (oferă informații legate de trenurile care circulă în următoarea oră), *ora_plecare*, *ora_sosire*, *intarziere*, *actualizare_intarziere*, *estimare_sosire*, *conectare_cont* (reprezintă logarea clientului în cont după nume și parolă), *deconectare* (ieșire din cont) și *gata iesim* (închidere client).

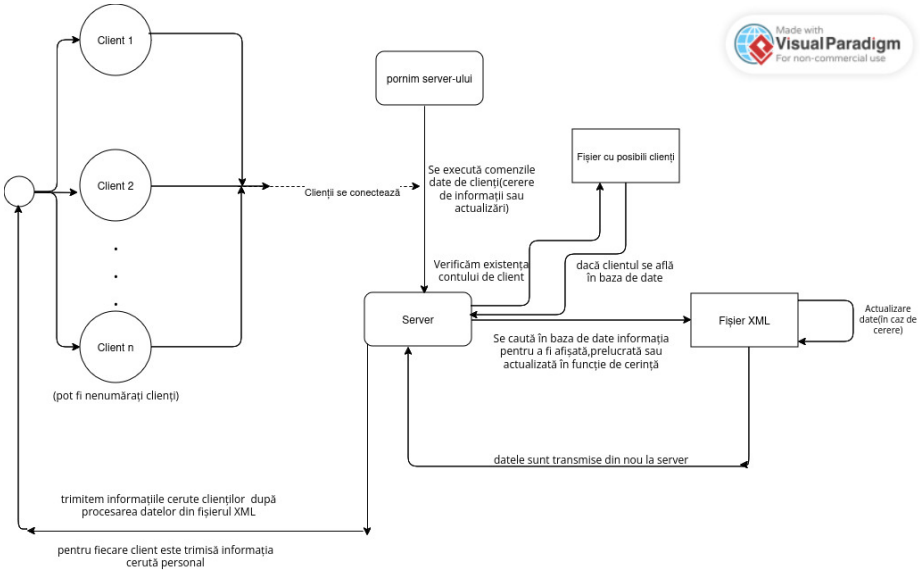

Figure 1: Reprezentare schematică a comunicării

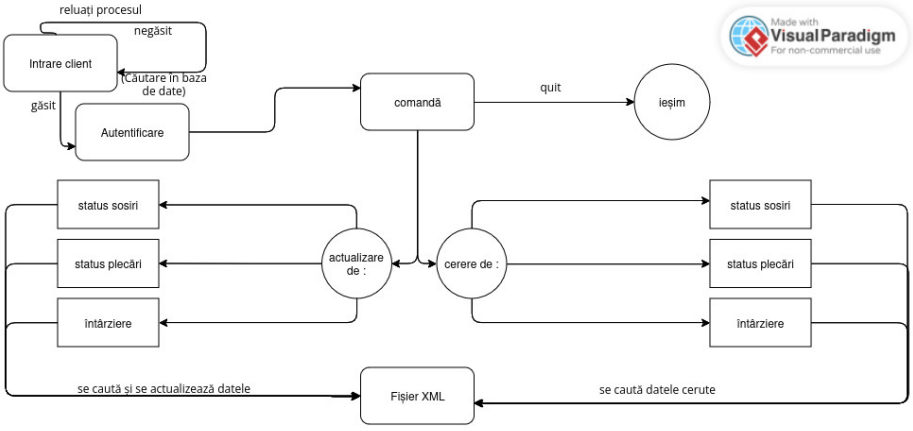

Figure 2: Reprezentare schematică a funcționalității

4 Aspecte de implementare

4.1 Socket - creare

Primul aspect de implementat este crearea socketului care îndeplinește conexiunea dintre client și server.

Creare socket

```
int sd;
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
  {
    perror ("[server]Eroare la socket().\n");
    return errno;
  }
```

4.2 Legătura cu serverul

Are loc stabilirea familiei de socket-uri, acceptarea oricărei adrese, utilizarea oricărui port utilizator și conectarea la server.

Conexiunea cu serverul

```
server.sin_family = AF_INET;

server.sin_addr.s_addr = inet_addr(argv[1]);

server.sin_port = htons (port);

if (connect (sd, (struct sockaddr *) &server,sizeof (struct
  sockaddr)) == -1)
  {
    perror ("[client]Eroare la connect().\n");
    return errno;
  }
```

4.3 Bind, listen și accept

În această secțiune se face atașarea socketului (bind), punerea serverul să asculte pentru a detecta dacă vin clienți să se conecteze și acceptarea clientului.

Bind, listen, accept

```
1   if (bind (sd, (struct sockaddr *) &server, sizeof (struct
     sockaddr)) == -1)
2    {
3      perror ("[server]Eroare la bind().\n");
4      return errno;
5    }
6
7   if (listen (sd, 6) == -1)
8    {
9      perror ("[server]Eroare la listen().\n");
10     return errno;
11   }
12
13   if ( (accept (sd, (struct sockaddr *) &from, &length)) < 0)
14   {
15         perror ("[server]Eroare la accept().\n");
16         continue;
17   }
```

4.4 Creare thread-uri

Concurența clienților va fi gestionată prin utilizarea thread-urilor.

Creare thread-uri

```
1   td=(struct thData*)malloc(sizeof(struct thData));
2   td->idThread=i++;
3   td->cl=client;
4   pthread_create(&thr[i], NULL, &treat, td);
```

4.5 Comenzi

Verificarea comenzilor este făcută cu funcția strncmp, modalitate aplicată tuturor comenzilor.

Exemplu comenzi

```
1    if(strncmp(sir,"gata iesim",10)==0)
2         {
3            printf("Terminam conexiunea cu clientul-din server la
      cererea clientului\n");
4            close(tdL.cl);
5            break;
6         }
7    if(strncmp(sir,"conectare_cont",14)==0)
8         {
9            FctConectareCont(sir);
10        }
```

Un scenariu real de utilizare al aplicației este reprezentat de dorința unui student de a găsi informații legate de trenul cu care trebuie să se întoarcă acasă, iar programul oferă datele necesare satisfacerii cererii clientului.

5 Concluzii

Alte posibile adăugiri la aplicație pot fi: implementarea calculării posibilului preț al biletului în funcție de tipul călătorului (de exemplu: dacă este elev biletul este gratuit, dacă este student i se aplică reducere) sau oferirea de informații suplimentare (precum: linia la care se află trenul, numărul de locuri disponibile rămase).

6 Referințe bibliografice

- https://edu.info.uaic.ro/computer-networks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
- https://edu.info.uaic.ro/computer-networks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
- https://edu.info.uaic.ro/computer-networks/files/7rc_ProgramareaInReteaIII_En.pdf
- https://online.visual-paradigm.com/
- https://edu.info.uaic.ro/computer-networks/files/4rc_NivelulTransport_En.pdf
- https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines