Tema 2 - Rețele de calculatoare

Panainte Diana-Teodora, 2B3

10 ianuarie 2023

1 Introducere

Mersul trenurilor este o aplicație formată dintr-un server multithreading și un număr nelimitat de clienți. Serverul oferă informații despre mersul trenurilor pentru ziua curentă, despre plecări, sosiri sau intârzieri din ora următoare la cererea clienților și actualizează informațiile sale cu cele trimise de clienți despre posibile intârzieri ale trenurilor aflate în circulație și estimarea sosirii trenurilor în gara în care se află utilizatorul.

2 Tehnologii utilizate

Se va utiliza protocolul TCP, deoarece este nevoie de o comunicare fără pierdere de informație între server și fiecare client. Este important ca informațiile de la server să ajungă la client, deoarece întârzierile și estimarea sosirilor sunt esențiale pentru clientul ce le solicită. De asemenea, este necesară și asigurarea comunicării între client și server, prin anunțarea întârzierilor spre exemplu, astfel încât să se poată actualiza informațiile ce vor fi trimise mai târziu altor clienți.

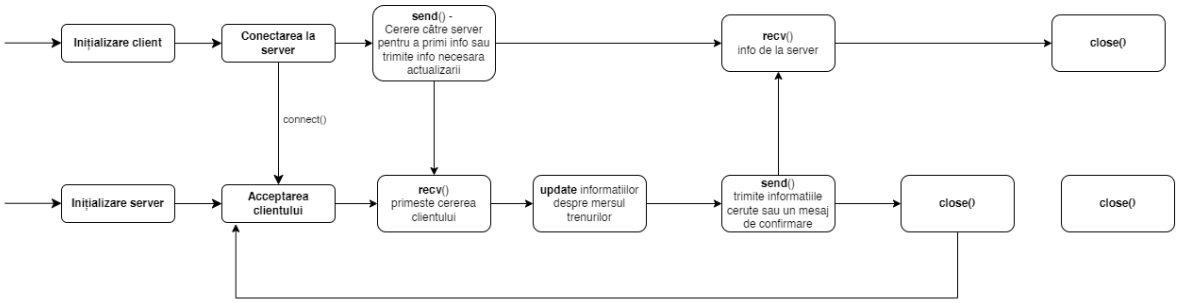
Se vor folosi thread-uri pentru a se putea gestiona mai mulți clienți în paralel. Astfel, fiecărui proces i se va asigna un thread pentru ca serverul să nu ia clienții pe rând, ci va putea răspunde tuturor simultan.

Informațiile despre mersul trenurilor, despre plecări, sosiri sau întârzieri vor fi citite din fișiere XML.

3 Arhitectura aplicației

Aplicația conține 3 fișiere: client, server și XML (pentru citirea datelor). În comunicarea client-server, clientul va cere prin comenzi informații de la server și va oferi informații despre întârzieri serverului. Comenzi posibile pe care le poate folosi clientul: *Arrivals*, *Departures*, *TrainTime*, *Delay*; Pe partea de server, acesta se va ocupa de partea de thread-uri, de gestionarea și actualizarea structurii de date (a vectorului) ce va păstra în memorie informații necesare din XML.

Diagrama de mai jos reprezintă conexiunea dintre server și client:



Mai jos este schema pentru fisierul XML, care va verifica și corectitudinea datelor.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Train">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="point">
          <xs:complexType>
            <xs:attribute name="type" type="xs:char"
                use="required" />
            <xs:attribute name="id" type="xs:unsignedShort"
                use="required" />
            <xs:element name="Station">
            <xs:attribute name="id_station"
                type="xs:unsignedShort" use="required" />
            <xs:attribute name="name_station"
                type="xs:char" use="required" />
            <xs:attribute name="arrival"
                type="xs:unsignedShort" use="required" />
            <xs:attribute name="departure"
                type="xs:unsignedShort" use="required" />
            <xs:attribute name="delay_departure"
                type="xs:unsignedShort" use="required" />
            <xs:attribute name="delay_arrival"
                type="xs:unsignedShort" use="required" />
             </xs:element>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4 Detalii de implementare

4.1 Crearea socketului

Prima dată vom crea un socket prin care se va realiza comunicarea client-server.

Crearea socketului

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

4.2 Conectarea la server

Conectarea la server

```
server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl (INADDR_ANY);
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 pentru socketuri și acceptarea clientului

Bind pentru socketuri și acceptarea clientului

```
if(bind(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1){
    perror("[server]Eroare la bind().\n");
    return errno;
}
if(listen(sd, 5) == -1){
    perror("[server]Eroare la listen().\n");
    return errno;
}
if(accept(sd, (struct sockaddr *)&from, &length) < 0){
    perror("[server]Eroare la accept().\n");
    continue;
}
```

4.4 Crearea de thread-uri

Clienții vor fi serviți în mod concurent prin folosirea de thread-uri.

Crearea de thread-uri

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

4.5 Estimarea timpului de sosire a trenului

Estimarea timpului de sosire se calculează în funcție de timpul oficial de sosire și întârzierea posibilă.

Estimarea timpului de sosire

```
char arrival_time[101],delay_arrival[101];
strcpy(File,"Path/mersul_trenurilor.xml");
if((file = fopen(File,"r"))!=NULL){
    citeste_xml(File,"arrival",arrival_time); //citete din xml timpul de sosire i
    citeste_xml(File,"delay_arrival",delay_arrival); //eventualele întârzieri trimise
    de client
    Trimite_catre_client(arrival_time+delay_arrival); // se trimite ctre client
    estimarea timpului de sosire
}
```

4.6 Cautarea prin XML pentru a afla mersul trenurilor intr-o statie precizata de client.

Cautarea prin XML

```
void searchXmlOneStation(int statie, char mesaj[LITERE])
{
  /*initializez mesajul de trimis catre client cu ""*/
  strcpy(mesaj, "");
  FILE *fd;
  char type[10] = "", id[10] = "";
  fd = fopen("trenuri.xml", "r");

  char *buffer = NULL, *copie = NULL;
  size_t len_buffer = 0;
  bool ok = 0, station = 0;                                    /*ok e pt cand am
    gasit un type si id; station e pt cand am gasit un tag <station>*/
  bool okStationName = 0, okArrivalTime = 0, okDepartureTime = 0; /*toate astea sunt
    pt a afisa toate info corecte cand compar verifStatie cu stationName*/
  int j, ct = 0;
  char stationName[100] = "", arrivalTime[10] = "", departureTime[10] = "";
  char verifStatie[100] = ""; /*din statie, care e nr, gasim numele luat din orase.txt
    si apoi salvam in verifStatie*/
  copie = findCity(statie);
  strncpy(verifStatie, copie, strlen(copie) - 1);
  // printf("verifStatie: %s; strlen(verifStatie):
    %d;\n",verifStatie,strlen(verifStatie));

  getline(&buffer, &len_buffer, fd); // parsing the xml until discovering the category
  while (strcmp(buffer, "</trainList>") != 0)
  {
    // parsing the xml until discovering the category
    getline(&buffer, &len_buffer, fd);
    if (strstr(buffer, "<type>") != NULL)
    {
      strncpy(type, buffer + 8, 2);
      getline(&buffer, &len_buffer, fd);
      strncpy(id, buffer + 6, 4);
      getline(&buffer, &len_buffer, fd);
      ok = 1;
    }
    if (ok == 1)
    {
      // printf("AM AJUNS INTR-UN TRAIN\n"); /*am ajuns intr-un train*/
      strcpy(stationName, "");
      strcpy(arrivalTime, "");
      strcpy(departureTime, "");
      okStationName = 0;
      okArrivalTime = 0;
      okDepartureTime = 0;
      while (strstr(buffer, "</train>") == NULL)
      {
        if (strstr(buffer, "<station>") != 0)
        {
          station = 1; /*AM AJUNS INTR-O STATION*/
        }
        if (station == 1)
        {
          station = 0;
```

Continuare

```
while (strstr(buffer, "</station>") == NULL) /*nu ajung la finalul unui station*/
          {
            if (strstr(buffer, "<stationName>") != 0)
            {
              /*deci gasesc stationName aici*/
              /*prelucrare stationName*/
              okArrivalTime = 0;
              okDepartureTime = 0;
              j = 0;
              char aux[100] = "";
              for (int i = 17; buffer[i] != '<'; i++, j++)
                aux[j] = buffer[i];
              aux[strlen(aux)] = '\0';
              strcpy(stationName, aux);
              okStationName = 1;
              // printf("Am gasit numele statiei: %s.\n",stationName);
            }
            if (strstr(buffer, "<arrivalTime>") != 0)
            {
              /*gasesc arrivalTime; prelucrarea arrivalTime*/
              okArrivalTime = 1;
              getline(&buffer, &len_buffer, fd); /*acum suntem la linia cu hour*/
              j = 0;
              char aux[100] = "";
              for (int i = 11; buffer[i] != '<'; i++, j++)
                aux[j] = buffer[i];
              aux[strlen(aux)] = '\0';
              strcpy(arrivalTime, aux);

              getline(&buffer, &len_buffer, fd); /*acum suntem la linia cu minutes*/
              j = 0;
              strcpy(aux, "");
              for (int i = 14; buffer[i] != '<'; i++, j++)
                aux[j] = buffer[i];
              aux[strlen(aux)] = '\0';
              strcat(arrivalTime, ":");
              strcat(arrivalTime, aux); /*concatenez minutele la ore*/
              // printf("Am gasit arrivalTime si stationName: %s in %s.\n",
    arrivalTime, stationName);
            }
            if (strstr(buffer, "<departureTime>") != 0) /*gasesc departureTime;
    prelucrarea departureTime*/
            {
              okDepartureTime = 1;
              getline(&buffer, &len_buffer, fd); /*acum suntem la linia cu hour*/
              j = 0;
              char aux[100] = "";
              for (int i = 11; buffer[i] != '<'; i++, j++)
                aux[j] = buffer[i];
              aux[strlen(aux)] = '\0';
              strcpy(departureTime, aux);

              getline(&buffer, &len_buffer, fd); /*acum suntem la linia cu minutes*/
              j = 0;
              strcpy(aux, "");
              for (int i = 14; buffer[i] != '<'; i++, j++)
                aux[j] = buffer[i];
              aux[strlen(aux)] = '\0';
              strcat(departureTime, ":");
              strcat(departureTime, aux); /*concatenez minutele la ore*/
              // printf("Am gasit departureTime si stationName: %s in %s.\n",
    departureTime, stationName);
            }
```

Continuare

```
if (strcmp(verifStatie, stationName) == 0 && okStationName == 1 && okArrivalTime == 1
   && okDepartureTime == 1)
          {
            /*inseamna ca am gasit statia ceruta de client*/
            //printf("Trenul %s%s ajunge in %s la ora %s si pleaca la ora %s.\n",
   type, id, stationName, arrivalTime, departureTime);
            strcat(mesaj, "Trenul ");
            strcat(mesaj, type);
            strcat(mesaj, id);
            strcat(mesaj, " ajunge in ");
            strcat(mesaj, stationName);
            strcat(mesaj, " la ora ");
            strcat(mesaj, arrivalTime);
            strcat(mesaj, " si pleaca la ora ");
            strcat(mesaj, departureTime);
            strcat(mesaj, ".\n");
            okStationName = 0;
            okArrivalTime = 0;
            okDepartureTime = 0;
          }
          getline(&buffer, &len_buffer, fd);
        }
      }
      getline(&buffer, &len_buffer, fd);
    }

    // printf("contorul este: %d\n",ct);
  }
  if (ok == 1)
  {
    ok = 0;
  }
 }
 fclose(fd);
}
```

5 Concluzii

În acest moment, aplicația are în plan afișarea mersului trenurilor pe ziua curentă, așa că s-ar putea îmbunătăți prin afișarea programului săptămânal. De asemenea, se pot afișa sosirile și plecările pentru toată ziua în loc de următoarea oră. Deoarece se știe timpul în care ar trebui să fie într-o stație și posibilele întârzieri, s-ar putea afișa și stația în care este trenul, sau ultima stație prin care a trecut. Mai mult, s-ar putea verifica și întârzierile trimise de client, ori printr-un mijloc obiectiv, precum un oficial de gară care poate trimite și el timpul de întârziere, sau un al doilea client, care observă întârzierea independent de primul.

Ar mai putea fi util și afișarea trenurilor care circulă între 2 stații precizate de client, pentru a îi putea oferi informațiile pe care le caută mai ușor și pentru a oferi o mai bună experiență de folosire pentru utilizator.

6 Bibliografie

$https://tex.stackexchange.com/questions/10255/xml-syntax-highlighting$
$https://profs.info.uaic.ro/\ computernetworks/cursullaboratorul.php$