Documentaţie Proiect : Reţele de Calculatoare

1. Mersul trenurilor (A) [Propunere Continental]

Mersul Trenurilor este un proiect tip server-client asemanător sitului CFR călători.
Proiectul este împărţit în 2 componente :
-client : o interfaţă grafică care ajută la folosirea aplicaţiei şi afişarea datelor (cere informaţii de la server)
-server: face legătura între cererile oferite de client si baza de date, actualizând baza de date atunci cand este necesar.

2. Tehnologii utilizate:

- Modul de comunicare între client şi server se realizează prin comunicaţii TCP (Trebuie să ne asigurăm că informaţiile sunt transmise şi primite de client în ordine , integritatea informatiilor)
- Threaduri: Fiecare client este legat de un thread care se ocupă de comunicare şi de datele cerute de către client. Cum datele despre trenuri sunt partajate de toţi clienţii.
- Datele sunt stocate într-un fişier xml iar ele sunt încărcate la iniţializarea servărului în memorie.
- Serverul se bazează pe o implementare “command queue”.
- Memory management : atunci când un client închide conexiunea Invoker-ul utiliyat trebuie oprit.

3. Arhitectura aplicaţiei :

3.1 Utilizarea aplicaţiei

- Clientul se conectează la server
- Serverul crează un thread nou care se va ocupa de client pană la terminarea conexiunii de catre client
- Clientul cere de la server informaţii despre anumite trenuri (in funcţie de staţie de plecare, de ora de plecare sau de ID tren) / oferă servărului informaţii despre o întarziere a unui tren (tren ID + intârziere)
- Serverul stochează cererile oferite de client într-o coadă şi oferă câte o cerere pe rând la threadurile responsabile de clienţi
- Threadurile execută comenzile şi transmit clienţilor înapoi informaţiile cerute / sau un mesaj de confirmare in cazul cererilor de tip “Delay”.

3.2 Detalii de implementare

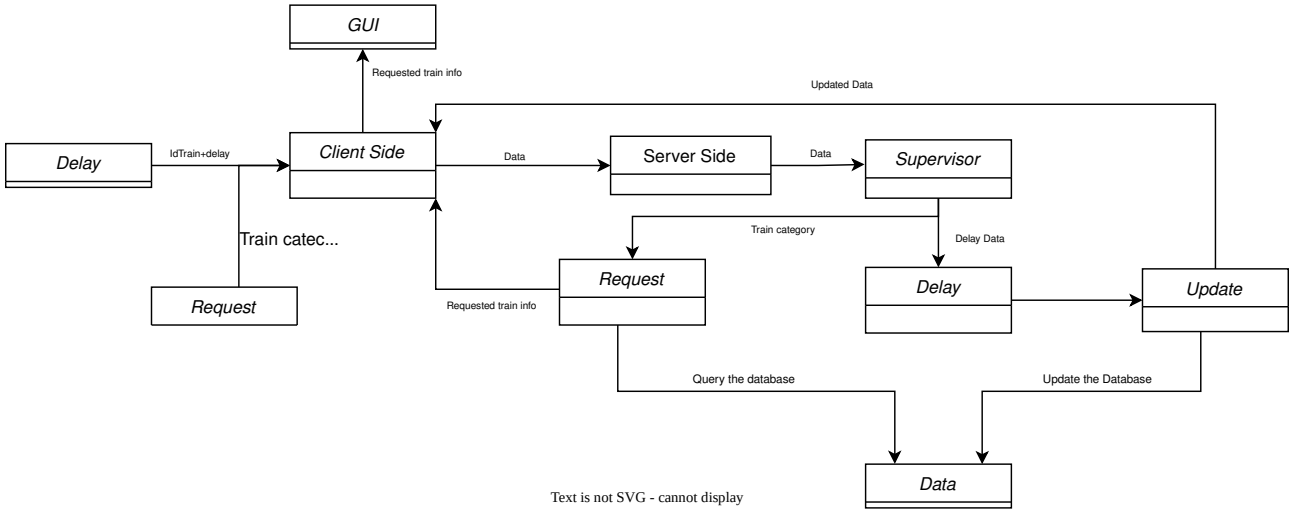
Implementarea clientuli şi serverului sunt facute ca doua aplicaţii separate pentru a putea fi rulate pe diferite calculatoare.
Iniţializarea serverului constă în citirea datelor dintr-un fişier xml şi memorarea lor întro structură care conţine informaţiile despre toate trenurile.

După ce clientul se conectează acesta poate să :
- ceară informaţii de la server
- oferă informaţii despre întârzieri

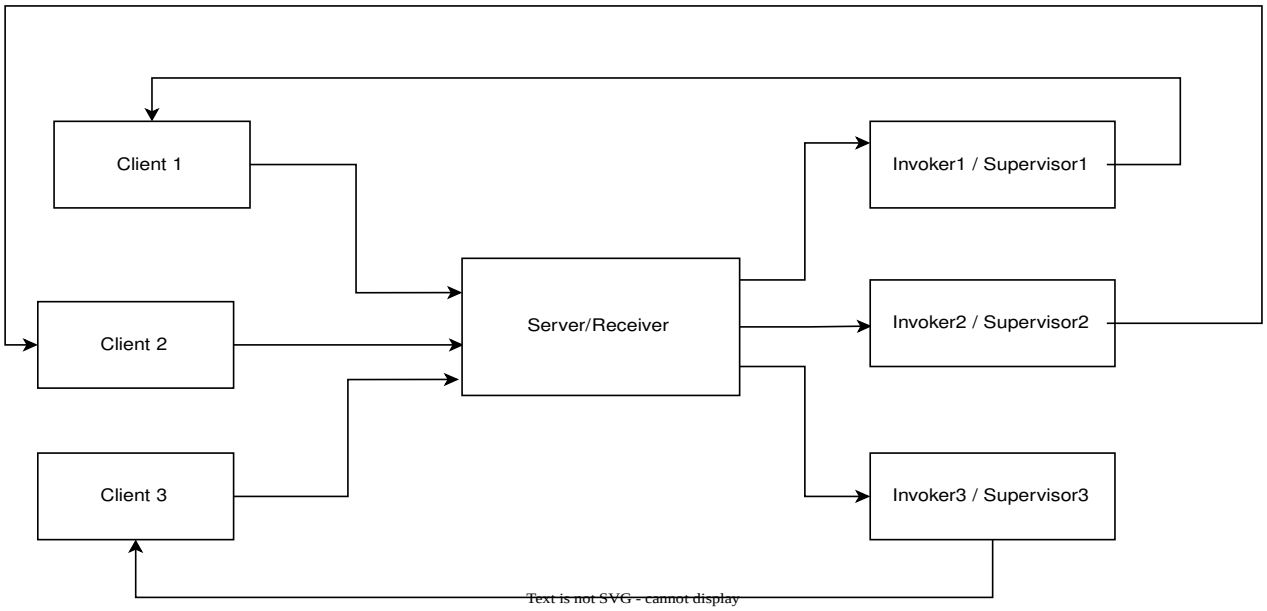
Serverul pune comanda într-o coadă comanda care urmează să fie executată
Un alt thread se ocupă de apelarea comenzilor :
- apelează obiectul responsabil de clientul specificat de comandă
- actualizează datele în datele păstrate local cât şi ale clienţilor cu date învechite
- se ocupă de trimiterea datelor înapoi la client

Comunicarea dintre client şi server continuă până când clientul închide aplicaţia moment în care serverul goleşte threadul responsabil de comunicarea cu clientul şi îl face deschid pentru a putea fi folosit de un client nou conectat.
Conumicarea dintre client şi server se realizează astfel:

-clientul cere de la server informaţii : &<IDTren>&<StatiePlecare>&<TimpPlecare>& sau
<ID><Staţie>*<Oră>*
-actualizează informaţiile din baza de date: #<IDTren>#<Intârziere>
-un tren ajunge sau pleacă din staţie : ^<ID Tren>^



Img 1



Img 2

4. Detalii despre implementare :

4.1 C/C++

Principalele limbaje de programare folosite sunt C++ pentru server şi C împreună cu GTK (interfața grafică) pentru client.

4.2 Utilizare a claselor şi a proprietaților acestora (inheritance)

```
class BaseCommand
{
public:
  virtual void execute(int fileDescriptor, RequestedData requestedData){};
  virtual void execute(RequestedData requestedData){};
};
class Request : public BaseCommand
{
public:
  void execute(int fileDescriptor,RequestedData *requestedData)
  {
    //Do stuff
  };
};
```

Ing 3

4.3 Utilizarea comenzilor de stabilire a conexiunii şi comunicare prin socket pentru comunicarea in rețea.
4.4 Utilizarea threadurilor în server pentru a putea răspunde la mai mulți clienți în paralel
4.5 Utilizarea de structuri pentru memorarea datelor

```
struct RequestedData
{
  char ID[10];
  char startStation [20];
  char timeStart [5];
```

Img 4

4.6 Implementarea funcţiilor

-Request : Returnează clientului numărul de rezultate care se potrivesc cu cererea urmat de rezultate, unul dupa altul
-Delay : Actualizează baza de date din server
-Update : Obligă toți clienții care au o inregistrare afectată de Delay să ceară datele din nou.

5. Imbunatatiri:

- O implementare a unui sistem de login astfel cererile de intarziere putand fii oferite doar de personalul calificat.
- Abordare diferită a funcției Update (foarte ineficient metoda aplicată)
- Implementarea mai multor Receivers pentru a îmbunatăți performanța.

Bibliografie:

>https://learn.microsoft.com/en-us/archive/msdn-magazine/2004/september/distributed-system-design-using-command-pattern-msmq-and-net
>https://www.bogotobogo.com/DesignPatterns/command.php
>https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php