Traffic monitoring system

Robert-Adrian Găină (2E4)

Faculty of Computer Science "Alexandru Ioan Cuza" Iași

1 Introduction

Traffic decongestion and fluidization is a modern problem that can be attenuated with the help of navigation software. This paper serves as the technical report of the application named Traffic monitoring, a project recommendation from Continental for the Computer Network course at Faculty of Computer Science. The scope of this application is being able to monitorize the traffic upon a certain area by providing informations to the drivers. In addition to that, the drivers can report temporary incidents on the road for the others or modify informations in the system.

This report contains detailed informations of the used technologies, application architecture, implementation details as well as improvements that could be added in the future.

2 Application architecture

The application will be based upon the **Client-Server** model with a **concurrent server**. In this way, we are avoiding the starvation problem that appears when using an iterative server with an increased number of clients.

The project implementation is based on the Transmission Control Protocol (**TCP**). This connection oriented method was chosen in order to assure a safe transport of data between the clients and the server. For this type of application we must be sure that we are not losing any data packets during the transfer nor receiving them in a different order. In addition to this, it is more reliable to keep a constant connection between the clients and the server in order to send traffic updates to the participants. Therefore, the TCP protocol will be used for implementing the application.

The concurrency pattern used will be **threading.** For each client the server will create a new thread that will execute the commands required. Threading was chosen over forking because of the common memory zone present in the technique, allowing the server to send information faster to the clients.

The information transmitted between the client and the server will be done using **sockets**. Each client will create a parent and a child process after the connection is established. From this point the client-server communication is assured by the parent process while the server-client is managed by the chil. (Write to server in parent read from child).

All the traffic details will be stored into a **SQLite** database. This database can be modified over time by the clients and each update of it will be sent to the connected clients.

3 Implementation details

3.1 Traffic mapping

This app will monitor the traffic of one city split into neighbourhoods. Each neighbourhoods contains a set of streets with each street containing a set of informations (F.I. speed limit, sport events, gas stops with gas prices, weather info, accidents etc...). Any client can report new incidents on a certain street or neighbourhood that will be registered into the database. Any incident can be removed from the database if a number of clients report the incident as a false alert. Every time a driver enters a new street he will get informations from the server about the area as well. Additionally, after a fixed period of time on a street the driver will get the updated information.

3.2 Estabilishing a connection

A connection between a client and a server can be established only by the client. When we want to create a connection the client will initiate the connection request. When receiving the connection request the server accepts it and creates a new thread that will receive further commands from the client or send updates to the client.

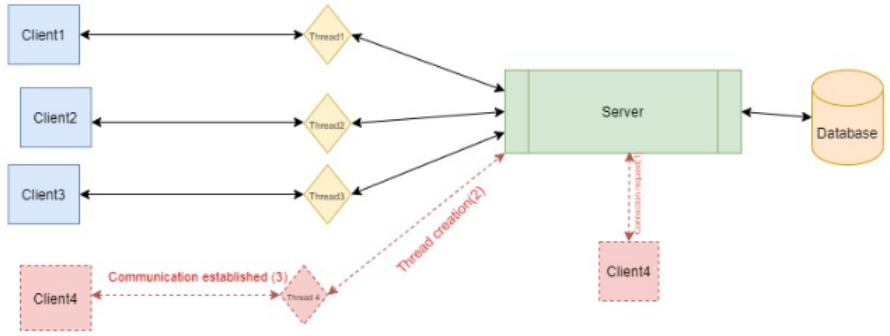

Figure 1. Client connecting to the server

3.3 Client-Server communication

In the application we define two types of communication operation: **Interogations** and **updates**. An interogation is a request that doesn't alter the database while an update modifies it.

1. The types of interogations are:

- Data request from a client that changes the neighbourhood/street
- Data request from a client about a specific zone/information
- Regular update of informations (every ~1 minute)

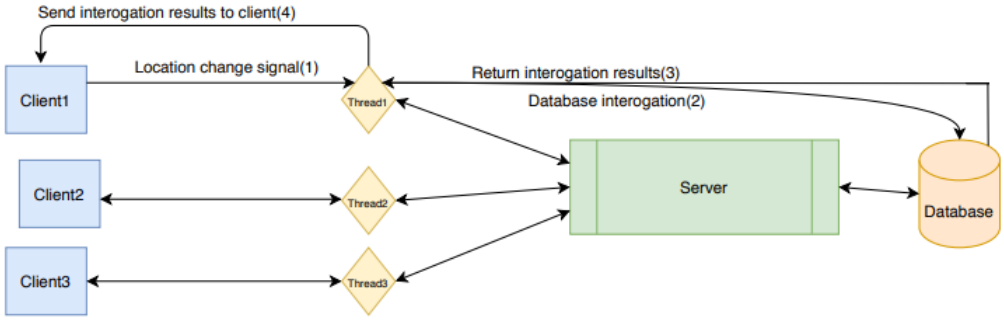

Figure 2. An interogation initiated by the client

For the first two type of operations the client sends a request to the server asking for informations. Whenever the localization detects

that the driver changes to a new street/neighbourhood it will send an automated request to the server asking for informations about the street/neighbourhood. Additionally, the client can manually send a request for a certain zone. According to the command received, the server will search in the database and return the informations required back to the client.

All the clients will ask for automatical updates of informations at every 1 minute.

2. The types of updates are:

- A client signals an information that is not in the database already to the server
- A set of clients signal the absence of an event that is present in the database

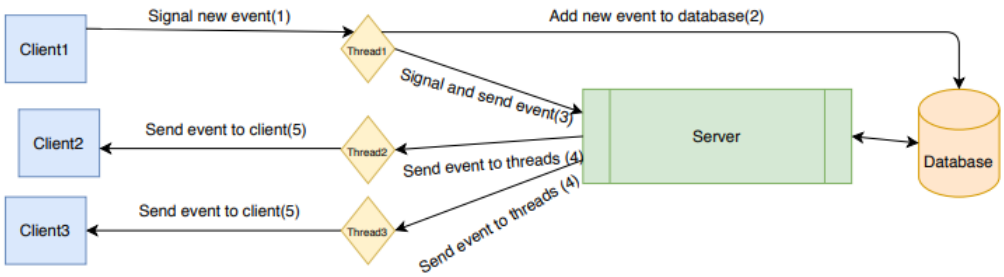

Figure 3. An update triggered by a client

–

When a client signals an event that is not registered by the server the thread responsible with the communication with that client will modify the database and signal the other clients by using the list of socket desc riptors about the change.

Since some of the events are not permanent or fixed (F.I. accidents) each temporary event can be deleted. When an user comes across a temporary event he will be notified and allowed to signal the absence of the event. When a number of consecutive clients report the incident as being absent it will be removed from the database.

For some interogations the user will also be able to change the data (F.I gas prices) when near the place.

3.4 Use cases examples

In this section we will present how the application should work in the final version.

In order to connect each client should already know the adress and the port where the server allows connection. For every client that sends a connection request the server will create a thread that should only respond to that specific client. After the connection is established the thread will wait for commands.

Connection example:

Two clients send a connection request. $\longrightarrow$ Server creates two threads

Clients send a request each to the server $\longrightarrow$ The thread assigned to each client solves the request

One client randomly disconnects $\longrightarrow$ The thread assigned to that client will close

Commands available to the user:

- Login: - connects the user to its account
- Extra: - allows the user to ask about what type of information he can receive in traffic. There are 3 levels for this settings. 0- only important informations, 1- additional events/places on the street, 2-aditional events/places on the neighbourhood where the street is part of
- Change: -allows the user to send a change of street/neighbourhood (This is mostly used automatically by the geolocalization system)
- Search: -search in the database for events/places based on a pattern.
- Alert: -allows the user to signal the presence/absence of accidents or other temporal events. This will insert the event into the database along with a column that is decremented/incremented based upon the feedback of other drivers. (If the value becomes 0 the event is dropped from the database)

Alert example:

Client signals an accident that is not registered by the application on a street $\longrightarrow$ thread responsible for the client adds the accident

in the database as a temporal event (that can be removed after 3-5 people marked it as absent event) and to a common memory zone for all the threads to see $\longrightarrow$ all the threads will pick up the command and send in to their assigned clients $\longrightarrow$ threads will mark that they have notified their clients $\longrightarrow$ when all threads notified their clients the accident will be removed from the common memory zone.

Elimination of alert example:

Client asks about informations upon a street where an accident was recently reported $\longrightarrow$ The server will send the related informations about the street and ask for the confirmation of the accident$\longrightarrow$ if the accident is marked by the client as absent the server will decrement the value detailed before otherwise increment $\longrightarrow$ If the value is 0 the accident is dropped from the database.

4 Conclusion

This application can help mitigating the daily traffic inconveniences and also signaling flawed street networks.

This application could benefit from the following improvements:

- Being able to configure a starting and ending point in order to send alerts only present on the route planned
- The updates/informations could be sent only if they are from the neighbourhood where the driver currently is.
- A graphic interface to ease the experience of the user.
- Implementing street network flows on the map allowing us to see how crowded it is.
- A localization system.
- A gps speedometer system.

References

1. General informations about computer networks
   https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. Project description [Monitorizarea traficului (A) [Propunere Continental]]
   https://profs.info.uaic.ro/~computernetworks/ProiecteNet2020.php
3. TCP vs UDP
   https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html
4. Thread vs Fork
   http://www.geekride.com/fork-forking-vs-threading-thread-linux-kernel
5. Waze (Application functionality inspiration)
   https://www.waze.com/