"Train's Time List" Documentation

Lazar Alexandru Constantin

Alexandru Ioan Cuza University, Bulevardul Carol I nr. 11, Iași, Romania

Abstract. This is a documentation paper for a computer science networks project. It contains information regarding the server that updates and/ or offers data in real time for all clients that are registered for: train's time list, departure/ arrival status, delays and estimated arrival. The server reads data from XML files and updates its data(delays and estimated arrival) on clients' request(signal-receive). The logic is processed by the server, the client only request information about the departure/ arrival and sends data to the server about possible delays and arrival estimates.

Keywords: TCP · Concurrent Server · Command Design Pattern · Command Queue · Threads · Sockets.

1 Introduction

"Train's time list" is included in the project list of the Computer Network class. It has many real life applications, representing the blueprint for an app that could be used by any company that provides transportation/ delivery services. It keeps track of the arrival/ departure status, delays and estimated arrival.

2 Used Technologies

This project contains a multi-threaded, server that communicates with it's clients using TCP (Transfer Control Protocol) and sockets, thus ensuring correctness of transmitted data and minimal delay for the multiple simultaneously connected clients through the use of threads. The downside of using TCP is that the "Three-way Handshake"(both sides synchronize (SYN) and acknowledge (ACK) each other, then send data. The exchange of these four flags is performed in three steps—SYN, SYN-ACK, and ACK) adds delay, but the upsides more than compensate, because the application relies on real-time data, thus a reliable, ordered and error-checked data stream is required.

3 Application Architecture

3.1 Application Logic Diagram

The application contains a finite number of clients, a server with multiple threads and a database. The client sends a command to a server thread. The command is sent to a dispatcher that sends the requests to a common queue, then returns a response to the client through the thread.

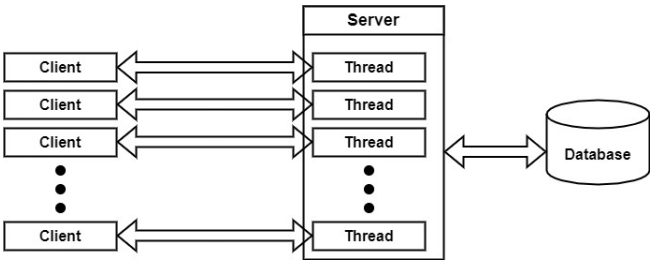

Fig. 1.

3.2 Simplified Command Pattern Diagram

This figure better describes how the communication between the components works.

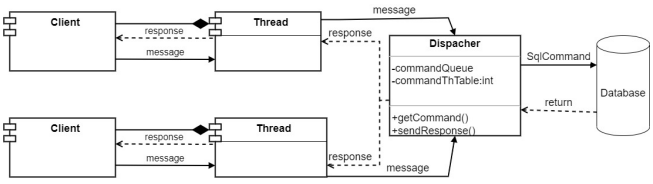

Fig. 2.

Observation: Database and Dispatcher are still in development.

4 Implementation Details

4.1 protocol

The protocol for the client consists of the next commands:

- **help** this command provides the list of all possible commands that could be sent to the server, accompanied by a description for each command.The list of commands vary depending of the access type of the user. This command can be executed by both a user and an admin.
- **register:type:username:password** this command adds a new user to the database. The type can be either user or admin. This command can be executed by both a user and an admin.
- **login:username:password** this command loggs in the client and provides access to certain commands that respect the protocol. This command can be executed by both a user and an admin.
- **trainList** this command provides the list of all trains ID number that depart/ arrive today, accompanied by their departure time.

  This command can be executed by both a user and an admin.
- **departure:type** where type could be either a train ID or all.

  If the command has the form **departure:trainID** the departure time for the train will be displayed, with the delay mentioned (could be +0), only if the train departs today.

  If the command has the form **daprture:all** the departure time for all of the trains will be displayed, with the delay mentioned (could be +0)., if the train departs in the next hour.

  This command can be executed by both a user and an admin.
- **arrival:type** where type could be either a train ID or all.

  If the command has the form **arrival:trainID** the departure for the train will be displayed, with the delay mentioned (could be +0), if the train arrives today.

  If the command has the form **arrival:all** the arrival time for all trains will be displayed, with the delay mentioned (could be +0), if the train arrives in the next hour.

  This command can be executed by both a user and an admin.
- **timeList** it displays all departures for today.

  This command can be executed by both a user and an admin.
- **arrivalStatus:type** where type can be either a train ID or 'all'.

If the command has the form **arrivalStatus:trainID** it will be displayed if the train has arrived or not.

If the command has the form **arrivalStatus:all** it will be displayed if all of the trains have arrived or not.

This command can be executed by both a user and an admin.

- **departureStatus:type** where type can be either a train ID or 'all'.

If the command has the form **departureStatus:trainID** it will be displayed if the train has departed or not.

If the command has the form **departedStatus:all** it will be displayed if all of the trains have arrived or not.

This command can be executed by both a user and an admin.

- **addDelay:trainID** the arrival and/or departure time is modified based on the 3 of 4 possibly achievable combinations of the arrivalStatus / departureStatus flags.

This command can be executed by both a user and an admin.

- **estimatedArrival:trainID** it displays the estimated arrival time (which is the arrival time + delay).

This command can be executed by both a user and an admin.

- **addTrain:ID:date:departure/arrival**. It adds a train to the database. This command can be executed only by an admin.
- **deleteTrain:ID:date** This command can be executed only by an admin.

4.2 Inter-component communication

Client-Thread Communication The Client sends the raw input data to the thread and prints exactly what was sent to the thread. This improves the security of the app, SQL injection or execution of unwanted commands being impossible.

Thread-Dispatcher Communication The communication between these components is based on a hash table which converts all of the commands into integer numbers. Each command has it's own associated prime number in their hash map and the username/ passwords are encrypted using their own hash map. Thus 2 strings of numbers will be sent to the dispatcher, one being a product of prime numbers and the other being the hased username + password. The dispatcher sends back hased data to be decrypted by the thread.

Dispatcher Database Communication The Dispatcher decrypts the message from the Thread, converts it into data about the thread and into SQL commands and then operates the database. The response from the database is saved in the dispatcher, which will dispatch the message to corresponding thread(s) afterwards.

Database management The server has an internal clock. The data that is sent/ received to the client has its own timestamp according to the server time, not the client time. The database is updated only on client requests, not regarding to the real time clock.

5 How the project can be improved

For the time of implementation, the protocol or the database is not yet implemented. But, the functionalities that were described in this paper do not provide a secure encrypted data transmission. The client sends data to the server that is not encrypted, thus opening the server to a lot of security faults.

Also, the app would be better if it would include more features for the admin, like shutting down the server remotely, deleting/ adding users etc.

References

1. Socket programming in C, https://www.csd.uoc.gr/ hy556/material/tutorials/cs556-3rd-tutorial.pdf. Last accessed 7th of December 2021
2. Transmission Control Protocol, Wikipedia https://en.wikipedia.org/wiki. Last accessed 7th of December 2021
3. Command pattern, Wikipedia https://en.wikipedia.org/wiki. Last accessed 7th of December 2021
4. Command queue, Wikipedia https://en.wikipedia.org/wiki. Last accessed 7th of December 2021
5. Computer Networks, UAIC https://profs.info.uaic.ro/ computernetworks. Last accessed 7th of December 2021
6. Scince Direct, Three way handshake https://www.sciencedirect.com/topics/computer-science/three-way-handshake. Last accessed 7th of December 2021
7. Linux Manual Page https://man7.org/. Last accessed 7th of December 2021
8. Diagram program used https://app.diagrams.net/. Last accessed 7th of December 2021
   https://man7.org/
9. LNCS Homepage, http://www.springer.com/lncs. Last accessed 7th of December 2021 Oct 2017