Design Document

COMP 6231: Distributed System Design

Assignment 1

Distributed Flight Reservation System (DFRS) using Java RMI

Name: Yongping Gao

Student ID: 27778317

Email: gaoyongping1990@gmail.com

Technique & Data Structures

The techniques I'm using is Java Remote Method Invocation(Java RMI) to implement communication between Server End and Client End and UDP/IP socket programming to implement the message passing between servers.

For data structures I am using HashMap to store Passenger Records in the server side.

I am using Json files (act like a database) to store all the information such as FlightRecords, PassengerRecords and Mangers information in the server and the client.

I am using plain txt file to save all the logs produced in the system.

Project Structure

There are 5 packages in the project.

1. CLIENT PACKAGE

In the client package, I design 3 classes, ManagerClient, PassengerClient and ServerInfo.

ManagerClient and PassengerClient are acting as the terminal for the users to let them choose the function they want to use.

ServerInfo is acting like a centre repo that storing all the server information such as server address, etc.

2. SERVER PACKAGE

In the server package, there are 3 server classes for each city. Each city has their own sub package storing their log file, flight records file and passenger records file. They all implement the DFRSInterface to do the RMI in Java.

3. MODEL PACKAGE

All the classes in this package is the basic models in the project, like Passenger, Manage, PassengerRecord, FlightRecord, Log, etc. They are all acting different roles in the whole project and they are basis of the project.

4. UTIL AND DRIVER PACKAGE

In the util package, there is one helper class to handle the JSON file processing. In the driver package, it's the entrance of the whole project (main function).

Testing Scenarios

1. HUMAN–COMPUTER INTERACTION

I design some test cases to test the interaction between human and computer by using Eclipse console. For example, when user input the wrong command which is not what the system wants, the system will give the hint to the user and direct the users to the right place.

2. PASSENGER BOOK FLIGHT TEST

I simulate the scenario when the passages want to book the flight. When the user type in the destination city and departure city, the client should query the database (JSON file of flight result in the server side) that are available and return to the user.

The user should choose the date they prefer. And if the flight is booked, the server will and a passenger record automatically.

3. PASSENGER RECORD

passenger record should be store in hash map (Passenger record JSON file) and grouped by passenger's first character of their last name.

4. MANAGER EDIT THE FLIGHT RECORD

The manager should be able to edit the flight record based on RecordID in this system. If the RecordID is not found, the system is able to add a new record in the system and stored in database (FlightRecord JSON file)

5. COUNT THE PASSENGER RECORD NUMBERS

The manager is able to get the total number of passenger records in these three servers. When the query happens, the server that the manager is currently logged in should be able to send request to other 2 servers by using UDP/IP, and then get the reply(count) from them. Finally give the result to the Manager.

The most difficult and important part

THE MOST IMPORTANT PART

The most important part of this system is that the communication between servers or server and client. Using RMI and UDP/IP programming technology is the key part to make the whole system work.

THE MOST DIFFICULT PART

The most difficult part of this system is how to design the whole project. How to make things elegant and efficient. How to make the whole system a bug free system.

And also for the coding part, the most difficult part for me is to handle JSON files. And how to save the object in JSON format and parse the JSON object to Java Object.

Reference

[1] Tutorial slides: Distributed_Sys_Tut_2_Socket_Programming.pdf
[2] Tutorial slides: Tutorial_2_JAVA_RMI.pdf
[3] Java API doc: https://docs.oracle.com/javase/7/docs/api/
[4] http://stackoverflow.com/questions/19796590/com-google-gson-internal-linkedhashtreemap-cannot-be-cast-to-my-object