CZ4013

Distributed Flight Information System

Author:
Pavel Jahoda (N1800740K)

1 General overview 1

2 Request and reply design 1

3 Marshalling 1

4 Invocation semantics 1

5 Services 1

General overview

The Distributed Flight Information System is based on client-server architecutre. Both of these programs are written in Python. At the start of the server, the server binds a socket and waits for new connection from the client. A separete method is called to establish connection. This allows us to develop a secure client-server handshake. After the connection is established, client can send requests to the server which performs the requested service, such as looking up flights based on their source and destination. Multiple invocation semantics (at-least-once and at-most-once) are implemented.

Request and reply design

Each request from the client is at the beginning represented as an array of objects. The array consists of identifier of the requested service, number of objects, unique numbers representing each object's type (including the error message) and the objects itself. This array is then send to the marshalling methods where they are converted into an array of bytes. After the server receives the request (array of bytes), unmarshalling method is called. The unmarshalling method converts the array of bytes back to the array of objects.

Based on the identifier of the requested service, the server then performes the service and reply is send back in a similar way. When the server needs to send an error message (for example, when no flight with queried ID exists), it does so by sending an array that consists of the requested service id and special number that indicates that an error message is being send.

Marshalling

Our goal in the marshalling process is to convert objects into array of bytes. First, we convert more "complex"objects (instances of classes such as flight object) into array of data types such as strings and integers. These integers and string are then converted into numbers between 0 and 127 indicating their ascii code and then converted into an array of bytes. At the beggining of such array there is an indication of the objects type (unique number) and number of bytes which is used to represent this object in an array.

In our array of bytes, we have indication of number of objects, unique numbers indicating their type and we also have number of bytes needed to represent a particular object. All these information give us the ability to perform unmarshalling and reconstruct all the objects.

Invocation semantics

Services