**Design Document**

Distributed Systems Group Project

**Team Members**

Ben Targan, Connor Wilding

**Description**

The project aims to design and implement a three-tiered client-server system to serve a user’s lookups by using RPCs. The system consists of three components: the client, the places server, and the airports server.

Given a client’s query the system is required to find the five nearest airports for the specified location.

**Design Considerations**

As stated in the brief, **performance** and **usability** are our primary goals. To increase performance, a balanced **K-D Tree** will be constructed from the airports data. Lookups are performed by maintaining a set of five closest records and pruning tree branches that are further those in the set. We are constructing a **prefix tree** from the places2k data; lookups are performed by traversing the tree with the supplied city name. If there are multiple results, the state is used to decide on the appropriate result.

**Client**

* User supplies three arguments: places server machine name, city, and state.
* The client program uses RPC to contact the places server with location information.
* If an error is encountered, display to user else return 5 closest airports to the given location.

**Places Server**

* Initializes prefix tree from places2k data file.
* Traverses tree using city, then state if there is ambiguity.
* Sends result to the airports server with the lat/lon of the location.

**Airports Server**

* A balanced KD tree is constructed from the airports\_locations file.
* Queries k-nearest neighbors by pruning tree branches.
* Returns the 5 nearest airports.

**IDL File**

/\*\*

\* File: airport-lookup-stubs.x

\* Authors: Ben Targan, Connor Wilding

\* Date: 10/10/20

\*\*/

const N\_RESULTS = 5;

const NAME\_MAX = 65;

const STATE\_MAX = 2;

const AIR\_CODE = 4;

const ERR\_MSG = 255;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Places server

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct location {

double lat;

double lon;

};

struct place {

string name<NAME\_MAX>;

string state<STATE\_MAX>;

location loc;

};

struct places\_req {

string name<NAME\_MAX>;

string state<STATE\_MAX>;

};

struct places\_res\_data {

place orig\_request;

airports results;

};

union places\_res switch(int err) {

case 0:

places\_res\_data res;

default:

string err\_msg<ERR\_MSG>;

};

program PLACES\_PROG {

version PLACES\_VERS {

places\_res PLACES\_QRY(places\_req) = 1;

} = 1;

} = 0x27699174;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Airports server

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct airport {

location loc;

double dist;

string code<AIR\_CODE>;

string name<NAME\_MAX>;

string state<STATE\_MAX>;

};

typedef airport airports[N\_RESULTS];

union airports\_res switch(int err) {

case 0:

airports res;

default:

string err\_msg<ERR\_MSG>;

};

program AIRPORTS\_PROG {

version AIRPORTS\_VERS {

airports\_res AIRPORTS\_QRY(location) = 2;

} = 1;

} = 0x37699174;

**Grading Criteria (REMOVE BEFORE SUBMISSION):**

Graphical user interface, text, application

Description automatically generated