An Indoor Place Detection Framework For Wifi Fingerprinting

Tony Kovanen
University of Helsinki
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
Fi-00014 University of Helsinki
tonykova@cs.helsinki.fi

ABSTRACT

Wifi fingerprinting is a widely used indoor positioning technique using received signal strengths of wifi hotspots to determine locations. This efficient technique has been chosen for positioning in many experiments [1], and it's applications are continuously researched. The purpose of the project is building a framework on top of which it is easy build experiments using wifi fingerprinting, and to test the accuracy of different positioning approaches. The fingerprinting framework introduced in this paper is a web application that can easily be deployed using any database, and on any server.

1. INTRODUCTION

Fingerprinting is a technique which uses signals of some sort to determine the location of a device (receiver). It can be used on any types of signals whose strength decreases when moving further away from the sender. Wifi fingerprinting is widely used because wifi signals carry relatively far away from the sender, they are measurable with almost any off-the-shelf device, and are often many in public buildings. All these characteristics promise a relatively good basis for building real-world applications.

Fingerprinting consists of two phases: calibration and estimation. Before positioning with wifi signals is possible, a database consisting of reference signal strengths is needed. The collections of these measurements is called calibration. When doing the actual positioning, the measurements in the database are matched against measured signal characteristics and the position is estimated using best matches (different ways of doing this). This is called the estimation phase. This means that the position information that is received using this technique is not any geographic location, but distances from different reference points, i.e. not absolute but relative.

Positioning with fingerprinting can be done with different kind of location measurements. The measurements can either be continuous, which means that each measurement is associated with coordinates, or discrete, which means that measurements are all associated with a grid cell. This grid cell can either be of fixed size, or it can be a topological partitioning of the area, e.g. a room.

2. FLUFFY

FLUFFY, The Flexible Urban Fingerprinting Framework, is web framework built to be used for experimentation with wifi fingerprinting. Fluffy is a web application that has basic user functionalities of a social web application as built-in features: registering, and sending and receiving friend requests. It also has an admin functionality which gives access to calibration of the positioning system and deletion and admin promototion rights over the user accounts. The built-in user interaction in FLUFFY is that friends can view each other's positioning histories.

FLUFFY has it's calibration measurements each saved to place. A place can have multiple measurements, so it can easily serve as a grid cell for a deterministic approach, or it can be easily changed to associate to coordinates in an area for a continuous approach. FLUFFY's positioning is a functionality available to any user. It has been built so that trying defined positioning approaches side by side, as well as adding new approaches is easy. This is achieved with a neat software architectural pattern (explained later). FLUFFY has two in-built approaches to estimation. The first one uses the best measurement from the database and the other uses individual grid cells.

2.1 Calibration

Wifi fingerprints are saved to FLUFFY's database in a pure mac address received signal strength pair format. However, when the fingerprints are used in positioning they are transformed into hyperbolic fingerprints to eliminate the need for manual device specific calibration. Hyperbolic fingerprints are formed by taking each unique pair of fingerprints and taking their logarithmic ratio:

$$\log(x/y)$$

Here x and y represent signal strengths two fingerprints. To keep these values positive, an absolute value of the aforementioned ratio is used.

2.2 Estimation

FLUFFY has two in-built estimation techniques. The first technique matches the users fingerprints to each fingerprint in the database and chooses the best match. Instead of regular fingerprints hyperbolic fingerprints are used to compensate for the lack of proper RSSI (Received Signal Strength Identifier, the format in which hardware spits out signal strength information) standards and differences in device accuracy. The grid cell that the best measurement belongs to is then picked to be the best match.

The other technique uses each measurement in a grid cell. Average of all errors between the user's measurement and all the reference measurement of a grid cell is calculated, and the grid cell with the smallest error is chosen.

The chosen error function is by default euclidean distance:

$$d(s, x) = \sqrt{\sum_{i=1}^{n}(s_i - x_i)}$$

More error functions can easily be implemented and used instead.

3. SOFTWARE ARCHITECTURE

Fig 1.1 can be found in file ArchitectureDiagram.PDF

As seen in Fig 1.1, FLUFFY is built using a 3-tier architecture using a Java based web application development framework: Spring, along with Hibernate and JPA2.0. On the top level of the software are the controller classes, which listen to specified (as well as unspecified) addresses and respond to HTTP requests to those addresses by invoking respectful service tier methods. The second layer of the architecture, service-tier, consists of service classes, which provide the services needed by the controllers. These services use the business logic of the application (Localization package in this application). The last tier, DAO (Data Access Object), is the link between the services and the database. Between all the layers is an interface layers, so that there are no dependencies between implementations of services in the controllers or DAOs in the services.

Another architectural design that FLUFFY follows is the Model-View-Controller architecture (MVC). This architecture divides the application to views (in this case .jsp, basically extended .html), controllers (explained above) and model, which contains the actual logic. A representation of this architecture can also be seen in Fig 1.1.

4. CLASS DIAGRAM

Class diagram can be found in file ClassDiagram.PDF

5. NAVIGATION FLOW CHART

Navigation flow chart can be found in file NavigationFlowChart.PDF

6. DATABASE DIAGRAM

Database diagram can be found in DatabaseUML.PDF

7. USECASE DIAGRAM

Usecase diagram can be found in UseCaseDiagram.PDF

8. EXPERIMENTING WITH ESTIMATION TECHNIQUES

Some on the fly experimentation was done with both inbuilt estimation techniques. The result showed that with the calibrating device at least (none else tested) there is no great difference between the two methods. However, in some situations where the user is located near, but not in the nearest grid cell, the average method tends to give better result (giving a grid cell that is much nearer). Still whenever the user is actually in the specified grid cell, both methods seem to work quite well.

9. CONCLUSIONS

Both estimation methods used in FLUFFY are not perfect but work quite well as long as the wifi ecosystem of all grid cells are fairly unique. I hope this framework will come in handy and I hope to develop it even further in the future. Instructions for deploying and using the software can be found in the file instructions.txt.

10. REFERENCES

[1] B. K. et al. Adnext: A visit-pattern-aware mobile advertising system for urban commercial complexes. 2011.