2015-2016

Report

Indoor Positioning application in an Android terminal.

Group members :

Florent Jacquet
Simon Magnin-Feysot
Houda EL HILALI
Lin ZHU
Haishan FANG

Summaries

1. Introduction

This project aims to resume the knowledge of Wi-Fi indoor positioning System learnt in the LO53. In order for the project to take place we need:

- Access Points

- Android device

- A server

- A Wi-Fi connection

The project begins first with collecting all the information needed in order to have a RSSI map and a valid finger print. Afterwards the devices can be located easily.

To achieve this project, we use wireless access points with theRSSI, a positioning server running on Python, and an Android device.

For this project we chose to use Python 3 in order to discover a language that was new to some of us. Python was only used for the server part but we needed to use C for the access points configuration. And as we were asked to have a running android application Java was the language used under android studio ide.

Concerning the hardware requirements, we needed several TP-LINK access points, a laptop (to run the server) and an Android device.

Our work was managed with the use of the git tool in order to share the improvement of each of us on the project.

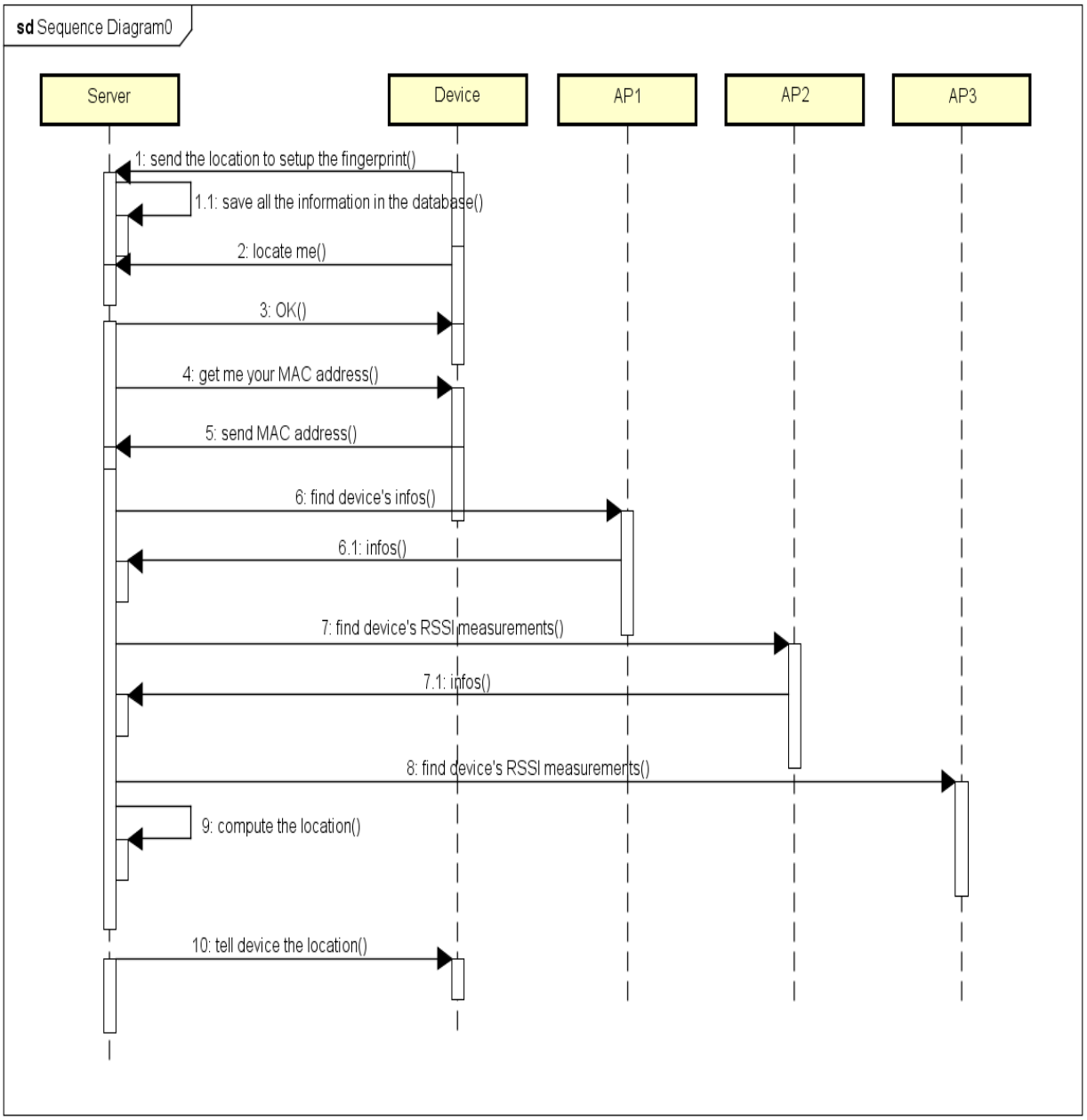
2. Importance of such project

Wi-Fi positioning system (WPS) is used where GPS is inadequate. The localization technique used for positioning with wireless access points is based on measuring the intensity of the received signal (RSS) and the method of "fingerprinting". Typical parameters useful to geolocate the WiFi hotspot or wireless access point include the SSID and the MAC address of the access point. The accuracy depends on the number of positions that have been entered into the database. The possible signal fluctuations that may occur can increase errors and inaccuracies in the path of the user. Anyplaceis a free and open-source Wi-Fi positioning system that allows anybody to rapidly map indoor spaces and that won several awards for its location accuracy. Like the last indoor positioning project of the MIT that its students were able to have an indoor positioning system using just one access point.

3. Diagrams

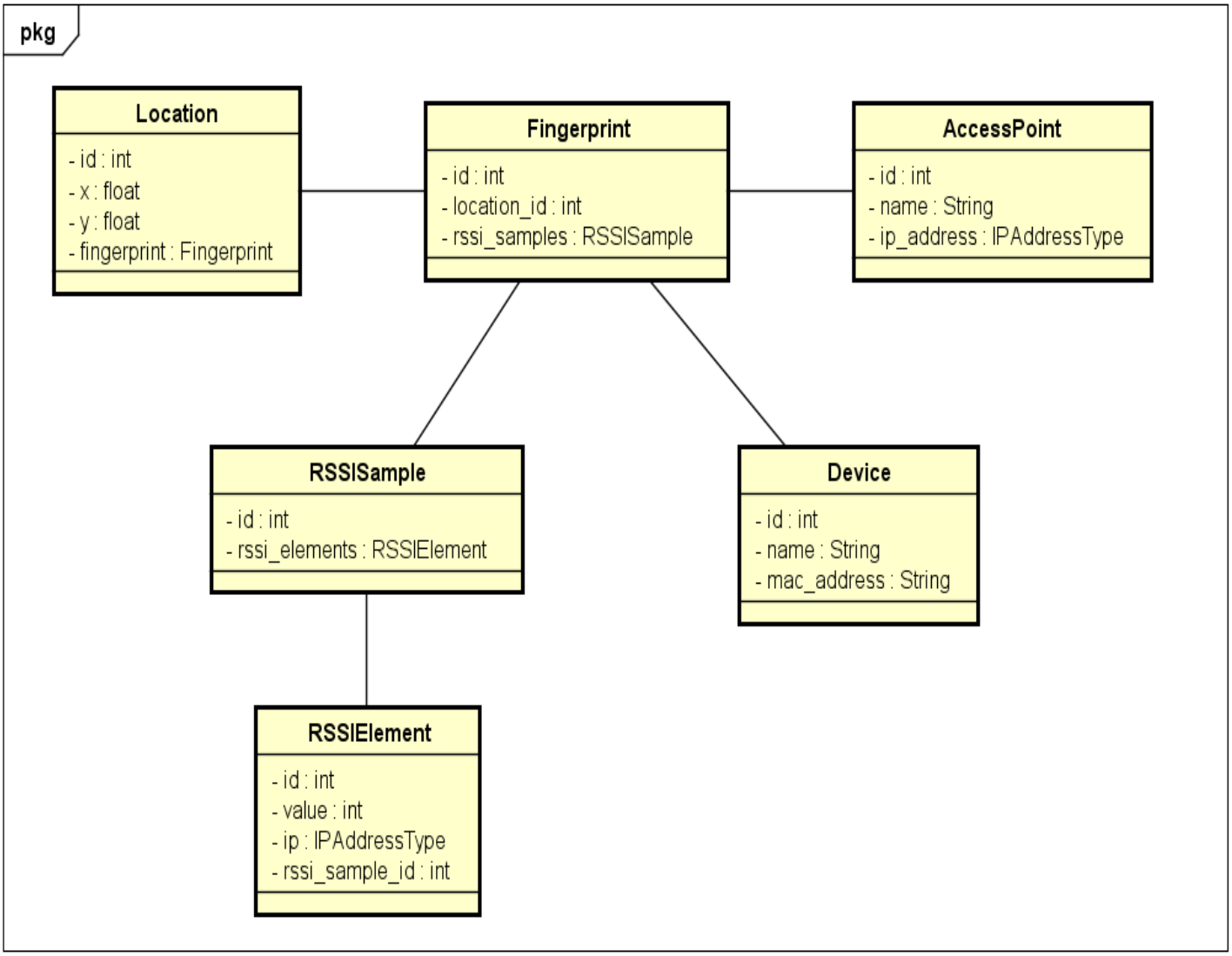
3.1. Global sequence diagram

The image below describes the different communications between the device and the server and also between the server and the access points.



3.2.Database mode

For our data base we choose to have 5 table were we can store information about Access Points in order to manage them. Also to store the devices information. And as every positioning system we have also fingerprint, RSSISample, Location and RSSIElement tables that will enable us to store information that will help us to locate a device.



4. Technologies used

4.1.Python:

The server was developed in Python 3. It used the web micro framework Flask to handle the HTTP communication, and the ORM provided by SQLAlchemy, using SQLite, to manage the database. The server is mainly divided in two parts: the setup server, and the localization server. Some part, like the models, and utility functions are shared by both parts. Each server provides a REST API to communicate with using HTTP.

4.1.1. Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed.

Flask is called a micro framework because it does not presume or force a developer to use a particular tool or library. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frameworks related tools. Extensions are updated far more regularly than the core Flask program.

We choose Flask because it’s easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

4.1.2. SQLAlchemy

SQLAlchemy is an open source SQL toolkit and object-relational mapper (ORM) for the Python programming language released under the MIT License.

SQLAlchemy provides "a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language". SQLAlchemy's philosophy is that SQL databases behave less and less like object collections the more size and performance start to matter, while object collections behave less and less like tables and rows the more abstraction starts to matter. For this reason, it has adopted the data mapper pattern (like Hibernate for Java) rather than the

active record pattern used by a number of other object-relational mappers. However, optional plugins allow users to develop using declarative syntax.

We choose SQLAlchemy because it enables us to easily manage the data base.

4.2.JAVA for Android

Android applications are developed using the Java language. As of now, that's really the only option for native applications. Java is a very popular programming language developed by Sun Microsystems (now owned by Oracle). Developed long after C and C++, Java incorporates many of the powerful features of those powerful languages while addressing some of their drawbacks. Still, programming languages are only as powerful as their libraries. These libraries exist to help developers build applications.

Some of the Java's important core features are:

- It's easy to learn and understand
- It's designed to be platform-independent and secure, using virtual machines
- It's object-oriented

Android relies heavily on these Java fundamentals. The Android SDK includes many standard Java libraries (data structure libraries, math libraries, graphics libraries, networking libraries and everything else you could want) as well as special Android libraries that will help you develop awesome Android applications.

Java was only used for the device application part using Android studio.

4.3.C Access Point

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations. By design, C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems. That's why it was the perfect language to configure the access points.

5. Code explanation

5.1.1. Access Point

The access points need to be turn into the monitor mode in order to sniff every packet available around it. We can do it using the web interface of the access point or writing a script shell (mon.sh) to do it in a terminal through ssh.The access point application was programmed in C. The project imposes this language but anyway, the AP system is so minimalist the C is the best choice.

We used the base code given on Moodle. This code already getsWi-Fi packets and analyses them in order to get the MAC address and the RSSI. We used also the given rssi_list.h file and we wrote the corresponding rssi_list.c. Several modifications were done, in fact, we don't store the rssi in mWatt but in db, the conversion is done by the server. We assume that the server is here to do the most bigger part of the computation, and the AP and mobile just send minimal information.

To communicate with the server, we just use TCP socket and not HTTP (wich requires the additional libcurl, and doesn't provide anything essential for our program), so the use of TCP protocol was minimalist but fits perfectly to our goals. But we needed to define a little bit more than just the use of TCP.

How they communicate?

- ✓ AP is launched and wait for TCP connection from the server — Server opens a connection and send a MAC address to the AP.
- ✓ AP gets the MAC, looking for the MAC in the linked list and return the corresponding RSSI.
- ✓ The server gets the RSSI and close the connection.
- ✓ AP waits for the next TCP connection.

We just have 2 threads running, one for the pcap function analyzing every packet in the air, and the second (the main thread) which is handling the TCP connection to communicate with the server. The second thread is also the main thread which is comparing the asking MAC address with all the addresses stored in the linked list. If the asking MAC address is not found, the AP will return 4000 to the server, because 4000 dBm is an impossible value to get with this APs.

The AP has a MIPS processor, which is different than our laptop processor we need to cross compile with the given toolchain. The main problem while debugging was that we didn't have GDB, the only solution was to print some traces.

5.1.2. Server

a- The shared part

The models:

The SQLAlchemy was chosen because it is easy to buildan ORM mapping some Python classes to tables in a database.

The following models were done:

- ✓ AP: storing the access points, with their IP and a name
- ✓ Location: storing x and y, plus a one-to-one link to the associated fingerprint
- ✓ Fingerprint: storing the link to its location, and a link to its RSSI sample
- ✓ RSSISample: storing its RSSI elements
- ✓ RSSIElement: storing the access point IP where it came from, and the associated RSSIvalue

The utility functions:

Again, there was two parts:

- the functions handling the computing of the location,
- and the ones taking care of the communication with the access points.

Computing locations: rssi_distance is the most important. It is the one that compute the distance between two RSSI sample, given a threshold. With some pythonic magic, it could handle RSSI either as dictionnaries, or as stored RSSISample classes. It returns the distance as a float. This value is given to the compute_location function, that just it erate though the stored finger prints, and find the one with the closest distance. Then it returns its location.

Communicating with the access points: The communication was made using TCP sockets, soit went easily with the socket package provided by Python. The first function is theget_rssi_element, that takes the access point IP, the device MAC, and an element dictionary to be filled. It opens the connection, sends the MAC, and wait for a 4 bytes value containing the RSSI value. Then it puts it into the dictionary at the key "IP". The second function, get_rssi_sample, takes only the device mac, and iterates though the access points to

create a thread for each one, and pass it the right values.Then, all the threads are launched almost at the same time, to ensure the device to be at the same location for each value received. The element dictionary filled is now returned, and it is our RSSI sample.

b- The setup server

The setup server is made to fill the database with every data needed to make the geo-positioning to work. It allows to initialize the database, add access points, and finally add localization, computing the corresponding fingerprint using the utility functions. The device's MAC address is stored in a web session, allowing to have many devices setting up the database at the same time.

c- The localization server:

It is almost the same as the setup server, but provide less function: it never writes the database, but uses it only in read mode. The only difference is that it performs the inversed operation concerning the localization: it gets the RSSI, then find a location given the fingerprint, rather than store a fingerprint corresponding to the given location.

5.1.3. Mobile application

In both applications we took in consideration that the server IP is fixed, because we thought that an external user won't be able to now the server IP and it would be unsecure if that person did. So rather than asking the user the IP server address and its port we fixed them.

a- The initialization app:

The main aim of the initialization up is to initialize the data base.

The app has two main activities:

-The location view asks for the coordination of the place where the user is in, in order to relate an rssi to that coordination

-The Access Point view that helps us to store information about the access points

b- The location app:

For the location app which is the main application of the project it has the activity where the building plan is shown with the marker put on the coordination of the location of the device.

The device has a direct connection to the server by an HTTP protocol. At first it sends an initialization request to check that the data base is ready then sends the mac address in order to be located. Afterwards, it sends a get location request to get the coordination and put the marker on the plan image checking the scale.

6. Conclusion

In this project, we successfully built an indoor-positioning system by several Access Points. We had several steps to achieve our project.

This project was an interesting subject to work on. It enabled us to learn some new technologies such as python 3 (with flask and SQLAlchemy). This indoor positioning project helped us also to accentuate what we learnt during this semester. We were allowed to see each part of a basic Wi-Fi positioning system and know the concept to monitor the communications and to connect the device, the APs and the server altogether.