UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Wi-Fi Positioning System

LO53 - P2016

Paul-Émile BRETEGNIER
Antoine LAMIELLE
Fabien LESUEUR
Vincent MÉRAT

Département informatique
Logiciel Embarqué et Informatique Mobile

UTBM
90010 BELFORT
https://www.utbm.fr

Supervisors
Frédéric LASSABLE

Contents

Introduction

In order to put into practice the knowledge and skills we have acquired during this semester in LO53, we had to develop a system to locate mobile devices on a floor via Wi-Fi.

For this project, we had to build a server able to do necessary process to locate the device, configure access points to give them the ability to send location information, and of course an app for mobile device, to give it the faculty to ask its position, and to put the data it received on a map.

This report include our approach for this project, the different steps of our brainwork to locate mobile devices and explaination about the different modules (serveur/AP/mobile app).

1 Global mechanics

1.1 Global Workflow

Our system works like the examples seen in class.

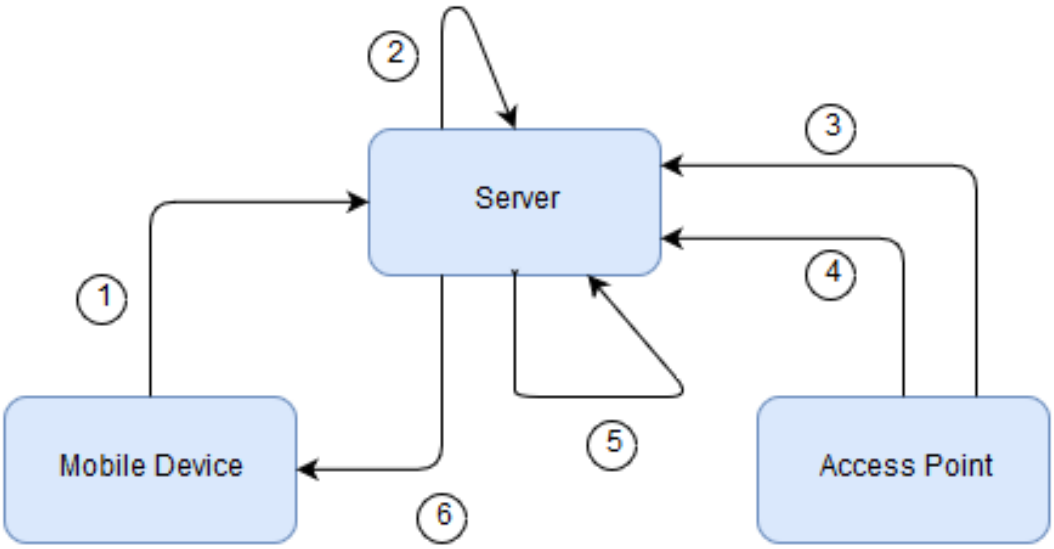

Fig. 1.1: Normal operation of the system

1. The mobile device ask its location via a HTTP request (FINDME)

2. The server add the MAC address of the device in its list of MAC address (if it's not already there)

3. The AP ask frequently the list of MAC address to monitor to the server

4. The AP regularly sends RSSI of the devices in the list to the server

5. The Server saves the data received, compute the location of the mobile device

6. The Server sends the location to the mobile device

1.2 Calibration

To calibrate our system, we use the Android App in calibration mode. There are two way to calibrate : first, we can do it step by step following instructions on the screen, or manually, by putting the cursor on our current location and click the add calibration point button.

In step by step mode, the user has to go first on the origin point. Then he can click on calibrate. The app will ask the step of calibration (in meters). The user will have to go to different points indicated on the screen and do a calibration on each on them. After all this process, the system is ready to locate a device.

In manuel mode, the user do the number of calibrations he wants at the position he wants. This methode is less accurate because the area isn't clearly criss-crossed.

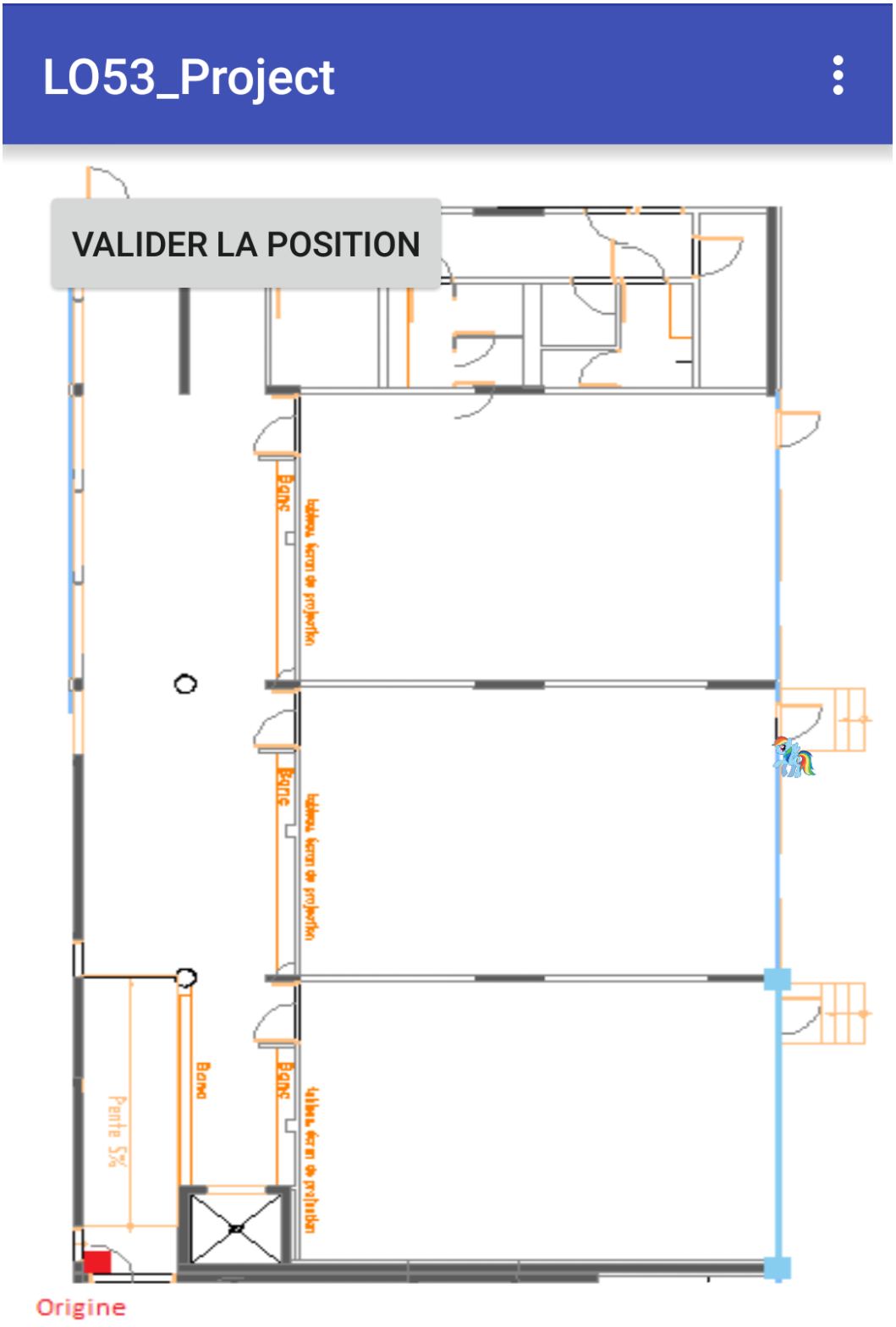

Fig. 1.2: Calibration Screen

2 Modules mechanics

All the communication between server, APs and mobile devices are done in HTTP. In that way, there is a uniqueness in the communication protocols between the elements. It is also, a very simple and effective solution since lots of libraries exists in all languages to work easily with HTTP. In addition, it's easy to test it via a web browser or cURL.

2.1 Server Side

We choose to implement the server with pyton and the library Django using a MySQL database (see below the database model). It has multiple roles.

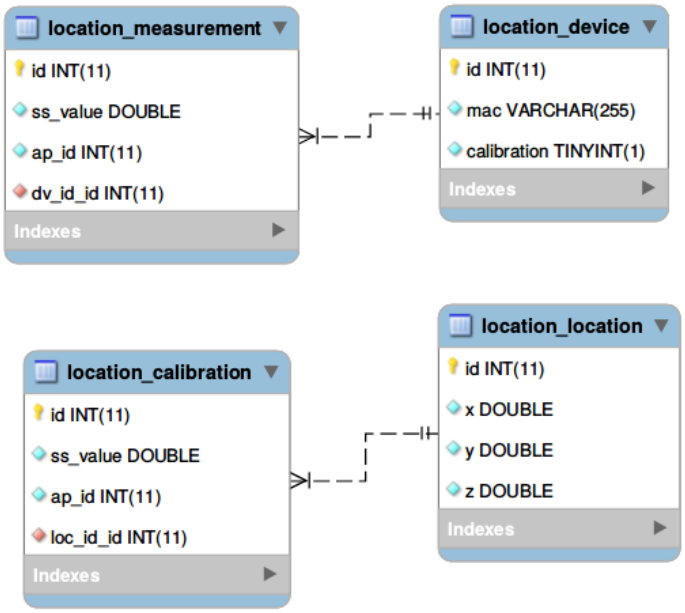

Fig. 2.1: Database model

First, it as to manage the list of MAC address that we want to monitor and locate. When a device wants to be located and sends a request, its MAC adddress is added to the list.

Then, it has to add the RSSI it received from the APs to its database

Finaly when it receive a requestion FINDME from a mobile device, and if it had enough RSSI information about it, it will compute its position with the data it has, and send the coordinates to the device via HTTP.

When the server receive a FINDME request, it can answer 3 different codes:

- ACK : It's the first answer when it receive the first request from the device
- WAIT : The processing is still in progress
- MAP : Process finished, the server send the coordinates via HTTP

2.2 Access Point configuration

For this project, we use TP-Link access point with OpenWRT, We've code our program in C with an external library (libpcap) to manage paquets.

The Acccess Points are regularly collecting the list of MAC address from the server via HTTP (request CHECK). In that way, they collect only data about theses MAC addresses.

When they capture a paquet, they read the MAC Address of the sender from its header. If it's a MAC address of the list, the AP cast the paquet. In that way, the RSSI is extrated from the paquet and then send back to the server.

In the code below, we iterate until we find the flag IEEE80211_RADIOTAP_DBM_ANTSIGNAL in the paquet to get the RSSI value.

```
1  /* Reset id_flag to start again the iteration */
2          id_flag = 0;
3          do {
4             flags = *(((u_char*)rtap_head->it_present) + (id_flag*4)) |
5                *(((u_char*)rtap_head->it_present) + (id_flag*4+1)) << 8 |
6                *(((u_char*)rtap_head->it_present) + (id_flag*4+2)) << 16 |
7                *(((u_char*)rtap_head->it_present) + (id_flag*4+3)) << 24;
8
9             /* Process for each field */
10            for(field = IEEE80211_RADIOTAP_TSFT; field <=
                  IEEE80211_RADIOTAP_VHT; field++) {
11               if(flags & (1 << field)) {
12                  /* Update the offset */
13                  if((offset \% radiotap_field_sizes[field].align) != 0) {
```

```
                    offset += radiotap_field_sizes[field].align - (offset
                        \% radiotap_field_sizes[field].align);
                }
                /* Get the RSSI value if the field corresponds */
                if(field == IEEE80211_RADIOTAP_DBM_ANTSIGNAL) {
                    rssi = *((unsigned char *) rtap_head + offset) - 0x100;
                    printf("\\nmac: \%s | rssi: \%d", mac_string, (int)
                        rssi);
                    send_rssi_to_server((int) rssi, mac_string);
                }

                offset += radiotap_field_sizes[field].size;
            }
        }
        id_flag++;
    }while(flags & (1 << IEEE80211_RADIOTAP_RADIOTAP_NAMESPACE));
```

2.3 Android Application

The mobile device need an app to communicate with the server, to do that we've developed an Android App. It has two main role. First to help for the calibration, then to show the position of the device.

For the calibration, the app send CALIBRATE request to the server with, inside, its current coordinates. the number of calibration points depends on the step of calibration selected at the begining.

To locate the mobile, after sending a FINDME request (i.e http://trips.loc/api/findme/68:5b:43:62 the app read the answer via JSON, when the code is MAP, the coordinates are sent with it.

```
{"code":"ACK"}
{"code":"WAIT"}
{"code":"MAP", "x"=36,"y"=72;"z"=3}
```

3 Results

During our tests, we have worked with three AP to locate a mobile device : two in room H010, one in room H011. With this configuration, we had a location precision around two meters on the whole floor. To increase its accurency, we could add more AP (five would be optimal) in the 3 classrooms to have a better coverage.

We had few problems with the detection of the mobile device due to the selection of the right channel of monitoring.

Conclusion

This project was interesting and complex. We had to deal with all the componnents and theirs languages : the server with python django, the access points with C and the mobile device with Android. They had to communicate with each other in a very simple and efficient way. We choose to use HTTP for its simplicity and its ease it implement. The major difficulty was to enable the AP to retrive data about the mobile devices we wanted.

We could improve our system by adding maps mamagement in the server: it will send it to the device (the first time) with the first location. In that way, we could locate the device on multiple area.

Bibliography

[Car] Tim Carstens. *Programming with pcap*. URL: http://www.tcpdump.org/pcap.html (visited on June 16, 2016).

[Tea] Django Team. *Django Project*. URL: https://docs.djangoproject.com/en/1.9/ (visited on June 16, 2016).

[WSU] WSU.edu. *Packet Capture With libpcap and other Low Level Network Tricks*. URL: http://eecs.wsu.edu/~sshaikot/docs/lbpcap/libpcap-tutorial.pdf (visited on June 16, 2016).