POLITECNICO
MILANO 1863

Internet Of Things & Wireless Internet Joint Project: "Finding Nemo"

Luca Pirovano – 10569272

Luca Vecchio – 10565156

Professors:

- **Matteo Cesana (Internet of Things)**
- **Alessandro Enrico Redondi (Wireless Internet)**

Academic year: 2020/2021

Table of Contents

1 Introduction

1.1 Project Overview

Nemo is a young clownfish who lives with his father Marlin in an anemone in the Great Barrier Reef. Unfortunately, on the first day of school, Nemo is captured by a pair of scuba divers.

Marlin, who is a really anxious father, had already prepared a system to find his son. He had placed 6 special anchors in the four corners of the Great Barrier Reef and installed a hacked operative system on Nemo's phone.
Marlin had mapped all the Great Barrier Reef with the anchors, creating a fingerprint dataset composed of the Received Signal Strength Indicator (RSSI), a smartphone in some location of the Great Barrier Reef. Now, he only needs to capture the probe requests coming from Nemo's smartphone and map them into his fingerprint dataset. For this task, he needs you. Would you help Marlin in this adventure?

In this project, you are requested to find the coordinates (X,Y) of the position of Nemo using the dataset of RSSI values from the six sniffer-anchors. The dataset is, however, hidden and fragmented inside MQTT publishes/subscriptions and CoAP requests packets. Every correct MQTT and CoAP request will give you a piece of the dataset, and, in the end, you will be able to reconstruct it and eventually find Nemo's coordinates.

The dataset is a fingerprint dataset, and for each (X,Y) coordinates, it contains an array of RSSI values corresponding to the values captured by the anchors in that position. Once found the RSSIs emitted by Nemo's smartphone, you have to determine his location by comparing the RSSI measurement with the database's entries.

1.2 Used tools

- JetBrains Pycharm.
- git
- Python 3.8 with the following packages:
  - aiocoap: CoAP client API
  - certifi
  - chardet
  - numpy
  - matplotlib
  - idna
  - paho-mqtt: MQTT client API
  - parse
  - PyYAML
  - requests
  - urllib3
- Course's Ubuntu Virtual Machine.
- WSL with mosquito installed for subscribing to MQTT topic.

2 Implementation

In this section we will describe how we accomplished to project goals.

The entire code is available at: https://github.com/PiroX4256/IoT_WI_project_2021

2.1 Dataset retrieval

First, we needed the complete dataset of the RSSI signals from the map, in each even position.

This dataset was fragmented between MQTT and CoAP servers. We decided to retrieve it as follows.

2.1.1 MQTT

We subscribed to the MQTT wildcard topic (#) using the Mosquitto client on WSL.

In this case, several messages were containing valid positions, so we decided to automate the retrieving with the help of a simple python script, which is called mqtt_client.py.

In simple terms, what it does is:

- subscribe to the wildcard topic '#',
- listen to all incoming messages,
- parse the valid messages through a regexp which defines how they are formatted,
- put them into the local MQTT dataset file, which contains only the MQTT values.

Afterwards, we filtered out the invalid values (for example the ones with a positive RSSI or with the position out of the grid) and we started looking for clues to find the missing coordinates.

In fact, the MQTT topics were full of sentences which helped us finding the hidden CoAP URLs that contained useful information (see figure 1). See the CoAP section for more details.

Figure 1

2.1.2 CoAP

First, we retrieved the information from CoAP using an automated Python script, which is called coap_client.py.

In simple terms, it:

- launches a discovery on .well-known/core to retrieve the list of CoAP URLs,
- for each URL that it has found, it performs a GET, POST, PUT, OBSERVE and DELETE request,
- if the result of the request matches the parser regular expression (see figure 2) it parses and adds it to the local CoAP dataset file.

X,Y Position	A1	A2	A3	A4	A5	A6
(6,4)	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]

Figure 2

Of course, this procedure does not consider the hidden resources, which were found through a manual analysis of the MQTT topics, as described in the previous section.

Once the hidden resources URLs were found, they were reached through a GET, POST, PUT, OBSERVE or DELETE request through the Copper (Cu) instance on the VM's browser.

For instance, in the hidden resources we found:

- the position of several RSSI vectors, also by adding several query parameters (such as `answer=yes` or `gps=False` and so on),
- the RSSI of Nemo, by listening to `/root/BarrierReef/Nemo`.

2.1.3 Dataset merge

Finally, we merged the pieces of dataset found in MQTT and the ones found in CoAP in order to create a large complete dataset.

We noticed that, for each anchor, there were 5 samples of signal, and we decided to make an average of them to obtain a representative signal value from the various measurements.

2.2 Nemo Localization

We combined different metrics to track Nemo's position.

2.2.1 Nemo RSSI

The Nemo RSSI vector found in CoAP is: `[-61,-61,-57,-58,-57,-58]`.

By looking at it, we can conclude several things:

- since the signal with respect to antenna 1 and 2 is weaker than the once w.r.t antenna 3,4,5 and 6, we can state that Nemo cannot be for sure in the left half of our grid.
- By looking at the signals, we can see that Nemo should be on the central line of the grid, since it is more or less the same between antennas placed in opposite parts, for example 1 vs 2, 3 vs 5 and so on.

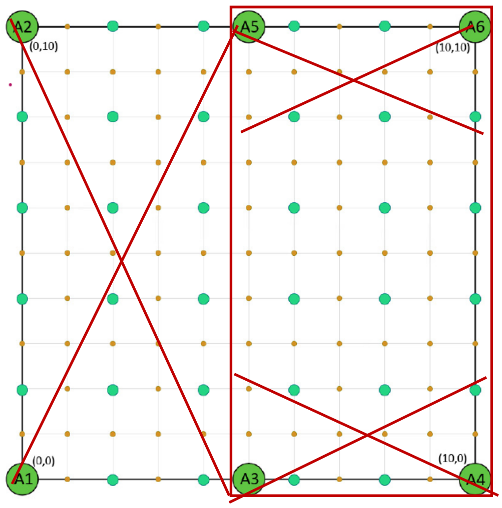

Figure 3

- By looking at Figure 3, we can then conclude that Nemo cannot be in the cancelled areas.

2.2.2 Comparison with other points' RSSIs

The first step for each metric is to compute the average signal for each entry in the dataset, for each anchor.

2.2.2.1 Metric 1: Taxicab Geometry

The first metric we decided to use in order to compare our dataset RSSI vectors is the so-called "Taxicab Geometry".

In fact, we took the Nemo RSSI vector and we compared it to the other points RSSIs by computing the Manhattan distance between each anchor RSSI value and Nemo's RSSI.

The one with the lowest distance value is the nearest to Nemo.

Example:

Nemo RSSI: [-61,-61,-57,-58,-57,-58]

Another RSSI: [-63.2, -63, -60, -58, -58.2, -54.2]

Difference: [2.2, 2, 3, 0, 1.2, 3.8]

Sum of differences: 12.2

Using this metric, we restricted the search field of Nemo's perimeter to four points highlighted in Figure 4. Of course, it can also be in the odd positions.

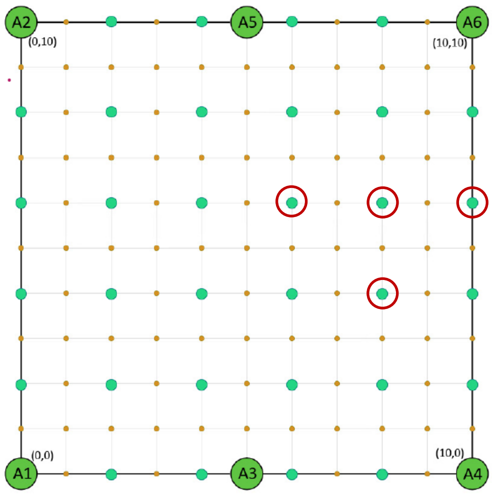

Figure 4

The result is consistent with respect to the observation we made in the previous section. In fact, following this metric, Nemo should be in the central-right part of the grid, probably slightly to the left of the point at coordinates (8, 5).

2.2.2.2 Metric 2: polynomial regression[1] + centroid[2] positioning

We then decided to adopt the method of polynomial regression, which consists of finding a correlation between RSSI and distance for each anchor, and then estimating Nemo's position based on the strength of its signal from each anchor.

---

[1] It is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x.

[2] The geometric centre of a polygon is called the centroid. The average values of the polygon's vertexes are the coordinates of the centroid point.

This method should be more accurate than the previous one, since it considers each antenna as a separate element, which makes sense because in the reality every antenna can have different physical parameters that influence their signal strength.

The first step is to calculate the polynomial regression of the RSSI distribution with respect to the distance from the anchor to each point in the grid present in the dataset.

We set the regression coefficient to 4, obtaining the following plot:

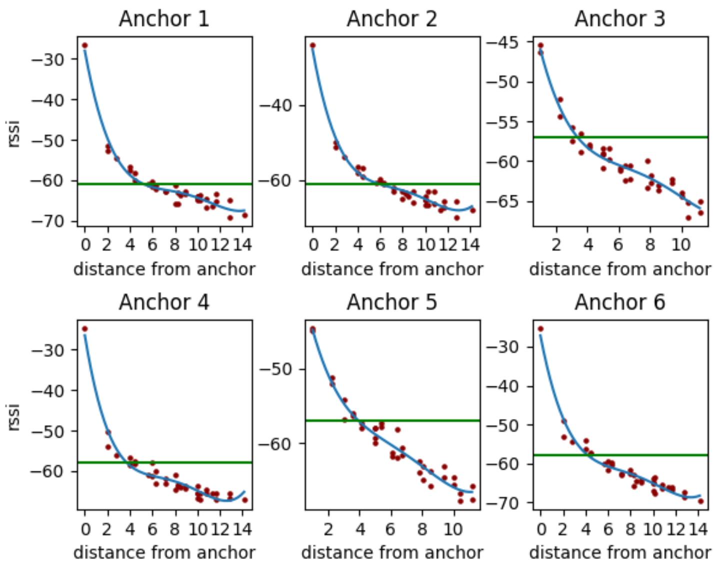

Figure 5

The blue line in Figure 5 represents the regression line, while the green line represents Nemo's RSSI value for each anchor.

The intersection between the blue line and the green line is our estimate of the distance between Nemo and the anchor point.

As we can see, this plot confirms the results obtained in the previous points, since its distance from anchor 1 and 2 is the same, but larger with respect to the other anchors (middle-right position).

We then applied the centroid positioning algorithm. First, we need to determine the region that contains Nemo, and then calculate the centroid of the region. The centroid acts as Nemo's location.

In Fig. 6, the measured RSSI values from Nemo to reference nodes can yield six distances. We can draw six circles using these distances from the corresponding reference nodes and we can get seven intersecting points, represented by the red dots (since the anchor 4 and 6 circles did not intersect, we counted their intersection with the grid border). The green dot is the centroid computed from connecting the red dots.

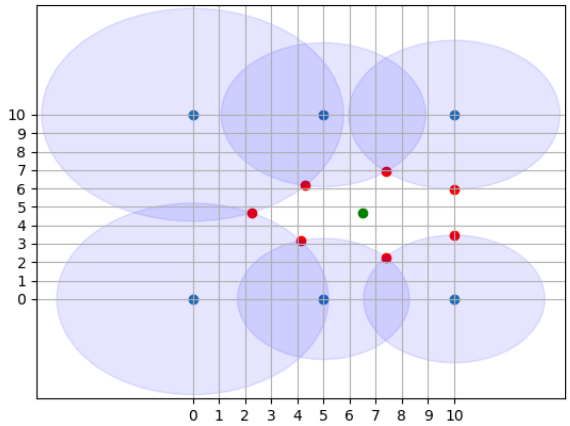

Figure 6

If we calculate the centroid of the polygon derived from connecting the red dots, we obtain an estimated position of Nemo, which in this case is still close to (7,5).

This result is very similar to the previous one (8,5), thus we decided to make an average of these value which results in a Nemo position which is between (7,5) and (8,5).

3 Conclusions

3.1 Dataset

We found the entire dataset which was fragmented in MQTT topics, CoAP resources and hidden CoAP paths.

3.2 Nemo RSSI

We found the Nemo RSSI by making a request to the following CoAP URL:
`coap://ec2-54-156-245-154.compute-1.amazonaws.com/root/BarrierReef/Nemo`

The RSSI vector of Nemo is: `[-61,-61,-57,-58,-57,-58]`.

3.3 Nemo Position

We found Nemo's estimated position according to the section 3.1.

Therefore, its position in the grid is **between (7,5) and (8,5)**.

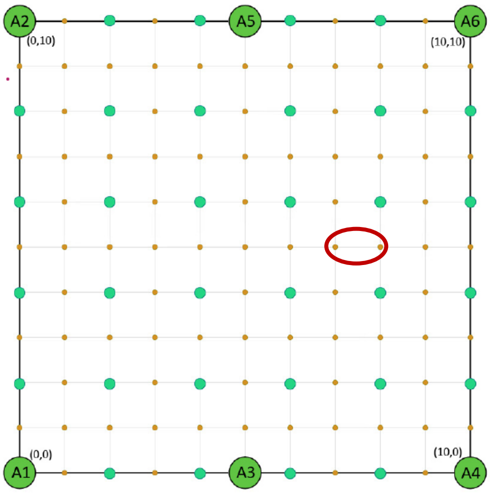

Figure 7