Human Mobility Tracking through Passive Wi-Fi: a case study of Madeira Island

MASTER DISSERTATION

José Miguel Santos Ribeiro
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

July | 2016

Human Mobility Tracking through Passive Wi-Fi: A case study of Madeira Island

Report of Dissertation

Master in Informatics Engineering

José Miguel Santos Ribeiro

July 2016

Supervisor:

Prof. Dr. Duarte Nuno Jardim Nunes

Madeira Interactive Technologies Institute
University of Madeira
(Portugal)

(Left in blank intentionally)

Abstract

In the busy city environment, it is increasingly important to analyze and understand people's patterns and behaviors revolving their commutes and times spent in specific parts of towns. A non-intrusive positioning system is essential to be able to make this kind of analysis, which can be taken into account in urban planning, tourist tracking or passer-by counters among many other applications.

Wireless technologies have been used recently as a way to track objects and people movements usually with high precision on an indoor scale. This dissertation presents the development of a low cost tracking system based on non-intrusive passive Wi-Fi technology. Here we take advantage of the Wi-Fi devices carried by people to assess the passerby of locations, and map the devices to different locations in time.

The project focused in the use of a Wi-Fi scanner to collect information about the passerby's devices, to analyze the data in appropriate time intervals, and providing a way to visualize such information. The study occurred in Madeira Island over the course of 6 months, where 36 scanners were placed on strategic places of the island, in order to collect data from both locals and tourists. The data was processed regarding the location counts, the connections between locations, and events.

Keywords: Mobility; Passive Tracking, Wi-Fi.

Resumo

No ambiente atribulado das cidades, é cada vez mais importante analisar e perceber os padrões e comportamentos das pessoas relativamente às suas caminhadas, e tempos despendidos em certas partes das cidades. Desta forma, um sistema de posicionamento não intrusivo é essencial para fazer este tipo de análises, que podem ser levadas em conta no planeamento urbano, no rastreamento de turistas, ou contadores de pessoas num certo local, entre muitas outras aplicações.

As tecnologias sem fios têm sido utilizadas recentemente como maneira de localizar objetos e movimentos de pessoas, normalmente com alta precisão numa escala pequena dentro de edifícios. Esta dissertação, apresenta o desenvolvimento de um sistema de localização não intrusivo baseado na tecnologia Wi-Fi. Tomámos partido dos dispositivos Wi-Fi utilizados pelas pessoas para contabilizar a quantidade de pessoas que passam num local, e mapear os dispositivos em várias localizações numa escala temporal.

O projeto foca-se no uso de um dispositivo analisador de Wi-Fi para recolher informação sobre os dispositivos que passam por perto, e analisar os dados em intervalos de tempo apropriados, com uma via para disponibilizar esta informação. O estudo foi feito na ilha da Madeira durante 6 meses, em que 36 analisadores foram colocados em pontos estratégicos da ilha, para recolher dados tanto de pessoas locais como de turistas. Os dados foram processados relativamente às contagens diárias, às ligações entre os diversos pontos, e análise de eventos.

Palavras-chave: Mobilidade; Análise Passiva, Wi-Fi.

Table of Contents

List of tables

List of Figures

1 Introduction & Motivation

With the widespread availability of mobile devices, location technologies based on them became popular, and used in a wide range of applications. One interesting aspect of location aware technologies is to understand human dynamics. Considering that many people spend major parts of their days travelling in cities, either working or doing errands, this particular area is interesting and raises some challenges about the technologies and methods used to understand human dynamics in urban areas. The availability of large mobility datasets, captured through GPS in modern smartphones, Wi-Fi access points and cell towers is becoming common place. However, these datasets usually depend on complex and expensive infrastructures that are under control by major telecom or software corporations. Despite the technical feasibility of accessing this data, urban planners, public authorities (for instance tourism and civil protection) and local communities (including small businesses) cannot easily deploy these systems and access the data to improve the services they provide to the citizens.

Indoor location based services exist mainly to facilitate the people's lives inside buildings, either enhancing the experience or helping them to perform tasks faster, while the outdoor services are mainly used to help people navigate during travel. In the last few years, the number of smartphones has increased in detriment of older generation phones, and the demand for mobile data consumption is ever growing. Wi-Fi has become a strategy of offloading the mobile carriers from the data consumption and is used in combination with mobile data for a richer experience and better battery life. This allows for a great opportunity to exploit that trend, by using the publicly available information delivered by the devices when trying to connect to Wi-Fi networks. Previous research and several commercial systems provide Wi-Fi location information to companies and organizations. Usually such systems rely on the central management of a large Wi-Fi network (for instance in a hotel, a company or University campus). To the best of our knowledge this was the first attempt to conduct a large scale deployment of a Wi-Fi monitoring infrastructure spanning two Island of around 270 000 inhabitants. By placing Wi-Fi scanners in specific locations (for instance the airport) we were able to correlate our data with reliable ground-truth, thus ensuring the reliability of our system to actually detect how many people pass by a specific location.

We envision our system to support a large number of new applications that take advantage of a distributed system that is community-based to track people through Wi-Fi signals. We believe that human mobility information should be available to the communities that try to improve the urban environment or simply understand how they can improve the planning of their operations. This type of information can develop into new applications that change the way advertisements, events, and transportation routes are created for different places, thus adjusting to the user demands. The need to understand which other places do people come from, and where do people go to, is also relevant and poses an interesting challenge to this project. The purpose of this system is currently to detect number of passerby's connections and major events in the Islands of Madeira and Porto Santo. However, our long-term goal is to create an open-access platform that could power many different services, from helping small businesses plan their operations to new and innovative storytelling applications.

The system developed in the context of this dissertation emerged form the opportunity to develop a low-cost passive Wi-Fi infrastructure that could be used by the local authorities in Madeira Island. It started from the need of the Tourism Board to evaluate the number of people attending specific events and is expected to evolve into a large scale mobility sensing platform currently used by several city halls, the airports and ports, the nature reserve park and also several small businesses, including restaurants, bars and shops. The system is currently used mainly for research purposes but several innovative ideas for new services that could rely on a low-cost community based platform are being considered.

In this document we disclose the implementation details, starting from the hardware, the database, the web client, and the analysis methods. Then, we present the details of the deployment, results and statistics regarding the overall system and locations, and finally data relative to the devices and information gathered.

1.1 Beanstalk Project

This thesis is part of a larger project called Beanstalk[1], which is a MADEIRA14-20 FEDER funded project, consisting in a multidisciplinary team of engineers, designers, artists and web analytics working at the Madeira Interactive Technologies Institute (M-ITI), in partnership with the Madeira Promotion Bureau (AP Madeira).

This project is divided into two components – the first of which focuses on the creation of a platform where it is possible to keep track of the flow of people in the Islands of Madeira and Porto Santo. The second component consists in the development of a location based storytelling experience, using everyday mobile devices, that capitalizes on the previously collected data.

The goal is to design and deliver research on the wide topics of tourism and new marketing trends, striving to build prototypes and test beds in order to gain a better understanding of what both locals and tourists visit, how and when, and further complement this, with a transmedia experience that can potentially stimulate visits to the islands and the local economy.

As we explained previously the tracking platform quickly evolved from these initial premises of keeping track of the flow of people in the islands and evolved into a more ambitious sub-project. The low-cost hardware used to conduct the passive Wi-Fi scanning, together with the unexpected motivation of small-businesses to access data about their own locations, provided an opportunity for quickly growing the platform. In addition, the richness of the information collected and the high reliability of the results when compared with the ground truth, enabled the system to evolve substantially from the initial requirements of simply counting people.

1.1.1 Team

Through the course of the project the different members had specific roles regarding both the storytelling and the tracking components. The members related to the interactive technologies, were responsible for developing the story telling side, along side with the designers and 3D artists. The members responsible for the web design, planned the various websites of the project, including the website presented further in this document.

Personally, I was responsible for the implementation of the tracking software, alongside with the creation and maintenance of the database, also developing of the web API and web interface to visualize the results. The management members acted as a bridge between the project and the third party entities for the deployment, facilitating the communication and the permits for the physical installations.

[1] http://beanstalk.m-iti.org/

(Left in blank intentionally)

2 Related Work

In this chapter we present literature work that motivated us during the development of this project. We start by presenting the different types of mobility analysis and then their tech approaches to solve similar issues. Finally, we review how the collected data was analyzed in those previous studies.

2.1 Mobility Tracking

The various methods of mobility analysis on urban environments were conducted previously in order to estimate: daily commutes, usual travel distances and route planning. Focusing on the non-intrusive methods, a study from M. Dixon [1], used the data from mobile phone call records from Ivory Coast to assess the mobility patterns of the citizens, by trying to map the routes between the GSM communication antennas providing this information through a web based prototype tool to visualize the data. The data was analyzed using a k-medoids clustering algorithm [2] to separate the antennas into smaller clusters, and then, simple graphs were made to analyze the distance between calls, average daily distance and the call volume per user. The users were then mapped to the closest main roads to the antennas, and their traveled paths were inferred from there. The data visualization involved a map, that could be explored by clusters, and the ability to visualize the daily statistics for selected days.
While this study[1] presents results compared to weekends and holidays, their limitations include the implication of a phone call being made, the high distance between the antennas, and travels that include at least the distance between two cellular antennas.

Several studies have attempted to locate or count the number of people in specific locations. Most of them use wireless technologies, with a large part of them exploring technologies such as RFIDs [3] or Bluetooth [5].

The GSM technology was also explored before by Timothy Sohn in [4] by analyzing radio signals with signal strengths, cell IDs, mobile network code, mobile country code, location area code, and channel numbers from fixed sources, and then estimating the movement of the users. The captures were performed with a sample rate of 1 Hz with a Audiovox SMT 5600 mobile phone recording readings of nearby cell towers. The mobility modes were estimated by the signal strengths measured from the connections with different cell towers. The authors then divided the mobility modes into *stationary*, *walking* and *driving*, with a group of data collectors that logged their lives through a period of 78 days. The system attempted to predict the user's modes between *stationary*, *walking* and *driving* and also n attempt was made to estimate the daily step count of a user. The overall overall prediction accuracy was 85% for the three modes. They present as possible applications for this type of system examples of social-mobile applications and computer supported coordinated care. The disadvantage of this system is the need for the user to have a special phone to track the tower signals for later analysis.

The Bluetooth technology was explored in a study from A. Baniukevic [5], where it was attempted to create a hybrid system between Wi-Fi and Bluetooth. While this system allows

for precision, it can only cover small distances when compared to Wi-Fi. It proposes the use of the hybrid method to overcome the limitations of both technologies, allowing for the Wi-Fi detection to be used when away from a Bluetooth antenna, and, when near one, allow for the high precision traceability that Wi-Fi can not provide.

The authors also presented a table with a non exhaustive list of pros and cons for each technology:

Table 1 - Wi-Fi vs Bluetooth comparison [5]

	Wi-Fi	Bluetooth
Pros	Widely deployed and available infrastructure	High positioning accuracy
	Good coverage of the indoor space	Simple position estimation
	Quick signal scanning	Low energy consumption
Cons	High infrastructure dependent	Extra infrastructure investment
	Time-consuming radio map creation	Extra specific mobile devices
	Low positioning accuracy	Slow signal scanning
	Complex position estimation	
	High energy consumption	

The limitations of both Bluetooth and RFID, are that they are usually deployed in indoor environments and need more complex algorithms often recurring to device triangulation with information from various external devices that use the technology in question. Also, these technologies are usually detected by each device individually, which requires the users to install a third party software [6] to enable this type of data capture in each individual smartphone.
The main barrier of the Bluetooth technology for non-intrusive sensing is that, not only the Bluetooth feature needs to be turned on, it also has to have the functionality "discoverable" enabled, and that feature is disabled in most modern smartphones for security reasons - often needing explicit permission from the users, or only enabled while in the discovery screen[2].

Because of these limitations, other researchers looked at Wi-Fi technology to gather the same type of information in a non-intrusive way. Users place enormous value on Wi-Fi connectivity and the big overall demand for data, makes them use this technology evermore. This makes it ideal to capture people's information around cities, not only due to its widespread use, but also because it does not require the user to install any apps on the phone, nor be connected to a specific device or network.

A study has been done using the Wi-Fi technology to capture human mobility information in highly crowded areas such as football games and universities campuses [7]. The technique used in the campus was different, as they used the functional access points that already captured the data into a RADIUS database. Hence, they just used the database to filter the data by the desired

[2] http://developer.android.com/guide/topics/connectivity/bluetooth.html#EnablingDiscoverability

packets and analyzed the flow of people moving inside the campus through the various access points.

A white paper was released by Cisco in 2013 [8] , where an experiment was done using cisco Wi-Fi equipment to log the data of people attending to venues in retail stores, hotels, conference facilities, shopping malls, schools, and city centers. In this experiment, the data was analyzed with Cisco Mobility Services Engine, which allowed the filtering of the number of visitors, the amount of time they spend, and the frequency of their visits in the place. Due to the high density of APs in small areas, the authors were able to process this information according to the RSSI of each device through various access points, and triangulate an estimate position of the device in coordinates mapped into the real world. With this information it was possible to gather crowding factors in a venue, waiting times in lines or stores, dominant directions, speed, frequency, and users' usual paths. Another factor used to distinguish visiting users from the people that work in the venues and are constantly there, is the discovery of devices in the entrance gates. They divided the venues into zones, either manually, or with undisclosed mathematical models, being able to generate cluster models due to the high density of access points, and possibly inform the user of highly concentrated areas. The movement analysis was done between each pair of zones, calculating the number of devices that moved from one zone to the other and the median duration and speed. The filtering used was mainly the date and time to compare the main events of the venue, in addition of other filters such as labeling the detected people as visitors or the shop owners.

Another experiment was conducted by X. Hu in the University of Notre Dame [7] where an ultra-dense dataset of Wi-Fi data was collected from two football games. The goal of this study was to analyze the energy waste while devices are trying to connect and discover networks. This study provides guidance about the scanning method, with details and statistics about the scanning process. The targeted stadium had no publicly Wi-Fi available and seated roughly 80 000 people, which the authors claim to represent the true picture of an ultra-dense environment. The data was captured near the five entry gates of the stadium with Linux laptops coupled with multiple high range Wi-Fi adapters. The software used to capture the requests was *tcpdump* to capture the requests, later processed using *tshark* and Python scripts. The information was later stored in a MySQL database and analyzed regarding the number of devices per minute, the time interval between each appearance, the received power traces, and the type of scanning made by the devices.

From the university of Hasselt, a short paper reveals the technicalities behind an experiment called WiFiPi [9]. This experiment was deployed in two venues covering a music festival and a university campus, with the aim to gather involuntary information of mass events' visitors. The paper describes the technical details of the implementation, the techniques for placement and justifies the benefits of Wi-Fi technology over Bluetooth. The deployment was done with a Raspberry Pi computer with a Wi-Fi dongle that supported monitor mode, and a cell phone to provide internet connectivity. The scanning was also done using Python scripts with *scapy* – a packet manipulation package. The deployment included 15 trackers which captured around 36 000 different devices in the total of both experiments.

The results focused on the analysis of mobility between pairs of locations, and the stationary times spent at each place, trying to assess the most popular singers in the music venue. They stated having some technical difficulties occurring during the first experiment, where power failures were common, and both the detector software and the Raspberry Pi suffered from what the authors called "childhood diseases". These problems resulted in malfunction of some detectors, preventing them from detecting more devices.

Several applications are suggested in the results, including simulation of mobility based on real data from hops between locations with logged time intervals, to identify a user in certain premises and adjusting the conditions to his preferences or automatically logging into services. Other suggestions covered the privacy aspects referring to the locations of users, for user profiling, and they discuss a hashing technique used to anonymize the mac addresses.

The following table (Table 2) resumes the previous work and studies that attempted to track people's location with different technologies and environments.

Table 2 - Previous studies details

Study	Location	Deployed Scanners	Detected Devices	Technology/ Hardware	Capture Technique
Neural networks and RFID for indoor location [3]	University Corridors	3-9	-	RFID	-
Mobility Detection Using Everyday GSM Traces [4]	People's daily routines	3	-	GSM	Audiovox SMT 5600 mobile phone
Hybrid Indoor Positioning [5]	Apartment building; Shopping Center	2	-	Bluetooth; Wi-Fi	-
Cisco's Wi-Fi-Based Location Analytics [8]	Venues	-	-	Wi-Fi; Cisco Access Points	Cisco Mobility Services Engine (MSE)
Large Scale Movement Analysis of Wi-Fi data [8]	University Campus	550	5 989	Wi-Fi; Campus infrastructure	RADIUS server with network logins
Is There WiFi Yet [7]	2 Football Stadiums		183 146	Wi-Fi; Laptops	Python; tshark
WiFiPi - Tracking of Visitors in Mass Events [9]	Music Festival; University Campus	15	137 899	Wi-Fi; Raspberry Pi	Python; Scapy

From table 2, we conclude that the area of the deployments was consistently small, only covering parts of particular buildings, and the hardware used in these studies was either micro embed computers, or laptops. They often do not focus on the capture of the data, but rather in the analysis, where not all of them disclosed the hardware used, but the ones which did, used the Python programing language with third party data capturing packages.

2.2 Mobility Analysis

There are several techniques to analyze and synthetize mobility information from the tracking data. A group at the University of Minho [10] analyzed large scale movements from Wi-Fi location data. The method used by the authors analyzed the logins in the network form the devices in the university campus. This was possible because the campus network requires a login process that uses a RADIUS protocol for authentication. The goal was not to improve the positioning technique directly, but rather to conduct a movement analysis using the positioning information. The goal of this analysis was to produce knowledge in terms of movement patterns. The logins in the network were recorded to a MySQL database and later processed by the team. Those events were translated into data structures named *stays* and *space leaps*. The *stays* represent the permanency of a device in one location for a certain amount of time, and are associated with a MAC address, a start and end timestamp, and a location. The *space leaps* represent the movements between one locations to another, represented by the mac address, location, and timestamps. The *space leaps* are further divided into *Elementary Movement,* where the later has the same representation, it has a different semantic meaning to identify the leaps that occurred within a small specified time interval, this denoting a sequential movement, instead of a simple leap between locations that can occur with hours or days apart.

Further analysis was made to interpret the leaps into the *flow* structure retrieved from the leaps representation. The flow structure calculates the number of people as inbound and outbound traffic for each location.

An advanced method used in a study from [11] used the information broadcasted from 8000 Wi-Fi devices in Australia to perform what the authors called SSID profiling. This technique, involves analyzing the captured information, focusing on the SSIDs (names of the saved networks on the devices) in an attempt to associate different devices with social connections. It uses algorithms used on texts similarity, where each SSID is considered as a word. Those connections attempted to locate people that visit the same places, share the same interests, or family bonds. The connections were assessed with probabilistic algorithms based on the popularity of the SSIDs, and weighing the intersection of the same SSIDs through different devices accordingly to their popularity. They conclude that the less popular the SSID is, the more important the connection is, whereas an SSID that is popular among many devices may not represent a relevant connection between those devices. The algorithms used were Jaccard Index [12], which returns the ratio of the intersection between the elements of two sets by and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union. It also used the Adamic similarity [13], which assesses the frequency of an SSID, from where the popularity is extracted. The IDF-similarity [14], Cosine-IDF[15] and Psim-q, which compared the differences between the SSIDs of two devices, which also represents the likelihood of a relevant social connection. Some problems were also stated regarding the non uniqueness of the SSIDs, which would yield false results.

Part of our data analysis followed the methods mentioned in [10], due to the disclosure of the methods used and the database structures that supported this processing of data, and the methods mentioned in [11] were only tested as possible future work, as they were not the primary goal of the project, and arose as advanced techniques.

2.3 Mobility Ground Truth

The majority of the reported research projects used mobility data to discuss technical issues, privacy concerns or the potential of tracking technologies to understand human mobility dynamics. For instance in [11] and [10], the authors focus was to expose the privacy issues of the networks discovery process, either by exploring the SSIDs, in order to infer social links, or to explore common paths and concentrations around a venue.

The authors also explored the deployment techniques, focusing on the technological side, like the energy efficiency of the network discovery process [7], or the analysis of data focusing on its usability [9] to simulate multi-hop networks, and explore relationship graphs between the users.

While these projects succeeded at these goals, they do not provide ground truth for the captured data, i.e., they don't compare the data obtained by the mobility tracking with the actual flow of people near the tracking systems. This kind of information is hard to obtain, and is traditionally achieved by traditional methods such as manual counts of people in a location, through different hours of the day. This is usually performed by a person on the ground visually counting the occupants of a chosen location. This technique can also be improved from doing the same manual counts from video footage. It has the advantage of being stored and visualized many times, and does not need the person counting to be in the target location. It has the disadvantage of missing people, if the camera is not placed appropriately.

The video footage mentioned above can also be fed into video analysis software that performs the head counts. It has the advantage of being an automated task, and produces quick results when compared to a person. The disadvantages are in the picture conditions and lighting variations that have to be taken into count, otherwise, it can easily produce false results, which can be difficult to detect. The problem with these approaches is the accuracy of the counts, when the places considered are large and its limits are hard to cover or even to define.

In this chapter we presented the related work in which we based our hardware, the capturing software and the data analysis, presenting studies from different environments that used similar techniques of capturing this type of data. Next we present how the capture of data is done and the methods, techniques and technologies used to implement and deploy the system.

3 Methods, Techniques and Technologies Used

In this chapter we present the platform we developed in order to collect mobility information for analysis of the passerby's counts, events and connections. We start by presenting the logic of the components behind our approach, and then how our infrastructure was deployed and the different components involved.

3.1 Mobility Tracking with Probe Requests

The method to capture the data, was to explore the network discovery process used in Wi-Fi, presented in the following paragraphs, explaining how the data is captured and the components of such data.

As specified in the IEEE 802.11 protocol, each device can implement two ways to scan for available networks: active scanning and passive scanning. The passive scanning relies on the access point side, where it broadcasts itself in messages with a default interval, for any device that listening in that channel. In order for devices to detect the access point, they need to scan all channels for a certain period of time, to ensure they find all the available networks. This method is not the most suitable for mobile devices due to the scanning period in each channel, which needs to be enough to cover the broadcasting interval of the access point broadcasts, and due to this limitation, access points can be missed during the scan.

The mechanism to determine when to search for networks is not defined by the IEEE 802.11 specification [16] and is, therefore, left to vendors to implement. As it was stated by [17], *The fact that the algorithms are left to vendor implementation provide vendors an opportunity to differentiate themselves by creating new and better performing algorithms than their competitors*. This accounts for the differences noticed when broadcasting SSIDs, or the time intervals between requests, and the types of scanning that are performed across different devices and systems.

Instead, these devices use a combination of both passive and active scanning, which consists in skipping the scan period, and directly send broadcast frames – called probe requests - to listening access points for each channel in a sequence. This is done very quickly, and the access points that receive the requests, respond back to the device with the necessary information to join the network: such as sub-protocols, encryption and their physical MAC address. Although the device that requests the connection still has to wait in each channel for a response, this method has been proven to be faster than passive scanning and thus better for mobile devices. With this method, although it requires the devices to send data, the reduced scanning time allows for more energy savings and lower association times [7]. When the device also emits an SSID, this usually leads to only the APs with that SSID to respond, but is depends on the manufacturer's implementation.

Passive scanning benefits from not requiring the client to transmit probe requests but runs the risk of potentially missing an access point (AP) because it might not receive a beacon during

the scanning duration. Active scanning has the benefit of actively seeking out APs for association, but requires the client to actively transmit requests [14].

The active scanning is enabled in a large number of operating systems, and the devices send these requests periodically, in order to ensure the connection with the access point with the strongest signal in the area. While the devices are trying to connect or discover networks, they try to connect to some of the networks saved previously in the device, leading to a broadcast of saved networks in each channel [11].

Because these frames are broadcasted before any connection is established, no encryption is used, and the information contained in these messages is sent in plain text, which can be captured by any device listening in those channels [11]. The following diagram (figure 1) shows the composition of a probe request:

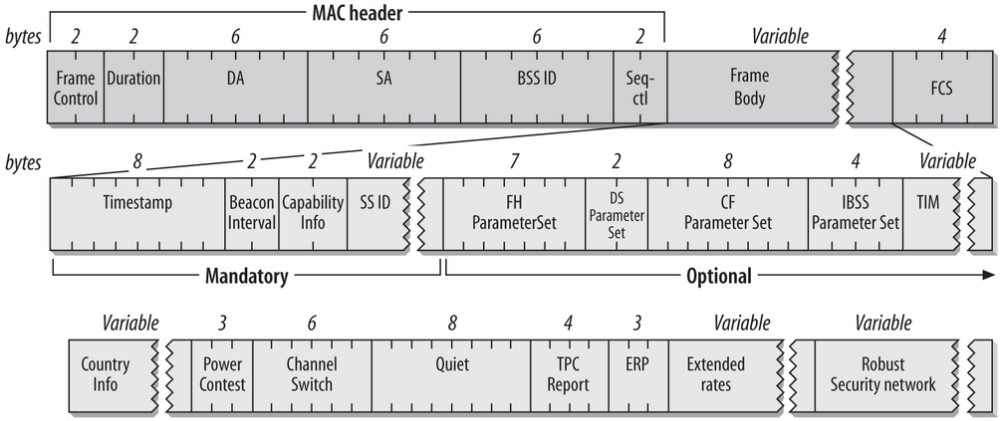

Figure 1 - Probe Request structure [16]

Those frames, directed to all access points, contain general information about the device and the network that they are trying to connect to, the more relevant being:

- **MAC address (SA)**: is used to identify the device that expects a response from the AP. From this parameter, the brand of the Wi-Fi can be easily identified by associating it with the address ranges set to each maker, and in many cases determine the brand of the device itself.
- **BSSID**: mac address of the destination AP (almost always ff:ff:ff:ff:ff:ff, like the destination address – DA – because it is sending to all listening access points).
- **SSID (Service Set Identifier)**: is the destination network identifier to which the device is trying to connect. Not present when the device is only scanning and not trying to connect to any network, and always present if the destination network is hidden. This field was present in 52% of the total captured frames.
- **RSSI (Received Signal Strength Indicator)**: this parameter is not present in the frame itself, but is derived by the capturing device upon reception of the frame by analyzing the power level received by the antenna resulting in an index between 0 and -255, where a lower index represents a stronger signal.

With this technique we can store the information about the devices' mac address, saving real information about a real identifier, and the locations of its detections through time. Since this poses a privacy issue, we decided to anonymize the MAC addresses to prevent the link between the data saved in our database and the real devices. This was done through the use of a hash function with the algorithm SHA-256 applied to the mac addresses before the database insertion.

3.1.1 Probe request broadcast interval

As stated in the related work (section 2), the broadcast frequency can change depending on the device manufacturer implementation of the IEEE 802.11 protocol. This phenomenon was also detected in our system by saving all the requests and analyzing the time intervals for each mac address. The results of this analysis are displayed in Figure 2, with 81% of requests occurring under 2 minutes.

This was also confirmed in a short experiment where we analyzed 5 android smartphones (including LG, Samsung, Wiko) and one iPhone that broadcasted probe requests with an interval of 10 seconds when not connected to any network, and an interval from 1-2 minutes when already connected. In the iPhone, the interval reached 5 minutes when already connected to a network.

The frequency of the broadcasts and the device status determines the success rate of capturing a frame. Primarily, the system works only when either the Wi-Fi and/or the location services with Wi-Fi scanning (standard mode in Android and iOS [11]) are active. With those prerequisites fulfilled, the frequency depends on:

1. The device being already connected to a network or not;
2. The device being on standby or awake modes;
3. The option to keep Wi-Fi on during standby/sleep mode is turned on for when the device is in that mode;
4. If the user is on the Wi-Fi selection screen;
5. The manufacturer/model of the device, and it's operating system;

The former study [7] analyses the frequency of probe requests, and concludes that the great majority has a frequency below 23 seconds with a significant peak at 45 seconds.

The following graph (Figure 2) represents the probe request intervals distributed by groups of time from captures made at M-ITI between the dates of 2016-01-29 and 2016-02-12, showing similar traits.

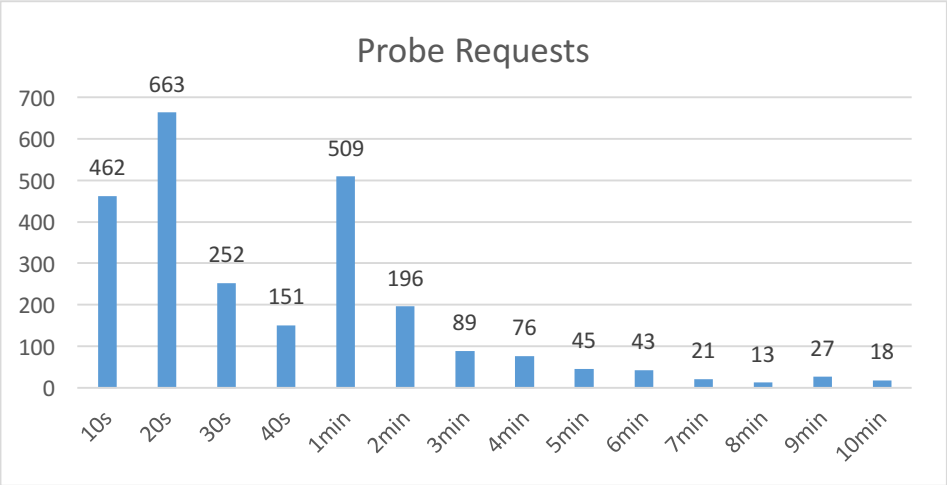

Figure 2 - Probe requests interval distribution

This was important to see the different intervals that occur between each request, and understand the limitations of the system, where a device that emits requests with an interval of 5 minutes can be missed by the system, if the user is moving quickly for instance.

3.2 Mobility Data Analysis

In order to analyze the data, we stored he collected information in a main table with all the information broadcasted from the device. To analyze the results, daily and hourly procedures were created to be able to provide results without the need of querying directly the main table containing the probe requests. The time intervals of day and hour were the ones considered appropriate to detect events, and people counts.

3.2.1 Connections

To analyze the mobility, we used a similar approach as the one used in [10], due to the disclosure of the methods used and the database structures where the data was analyzed by performing daily summary queries. The analysis summarized captured probe requests by date and hour, and stored in summary tables the results relative to:

- The daily number of distinct mac addresses in each place;
- Hourly number of distinct mac addresses in each place;
- Number of leaps, moves, and flow for each place;

Meta-data was also updated daily, relative to:

- Number of SSIDs per mac address;
- Number of mac addresses per SSID;
- Number of mac addresses per device vendor;

Note: To analyze the stay times, and the leaps, we queried the probe requests ordered by mac address and time of capture. This allowed us to interpret the location changes through time.

Instead of doing it for the total dataset, this was done only for the last 90 days, since it is a resource consuming operation. This means that a leap that is spaced apart 90 days between appearances is rendered invalid for the system.

3.2.2 Visitors

As the system progressed we explored how the data could be used to differentiate visitors from locals. With the goal of finding out the types of users in each place, we came up with a method to assess the number of visitors of the system, which we represent as the visitors of the Island. From the information analyzed for each device (MAC address), regarding its first and last appearance in our system, we calculated the number of days between those dates. With that, we did a distribution and concluded that a large number of devices appear under a 15 days' interval, and after that, the number of days decreases significantly. In following charts (figures 3 and 4), we present the distribution of age, comparing the first 3 months of capturing (figure 3) against the total data captured (figure 4). We notice that the distribution stays consistent, apart from the devices detected in only one day.

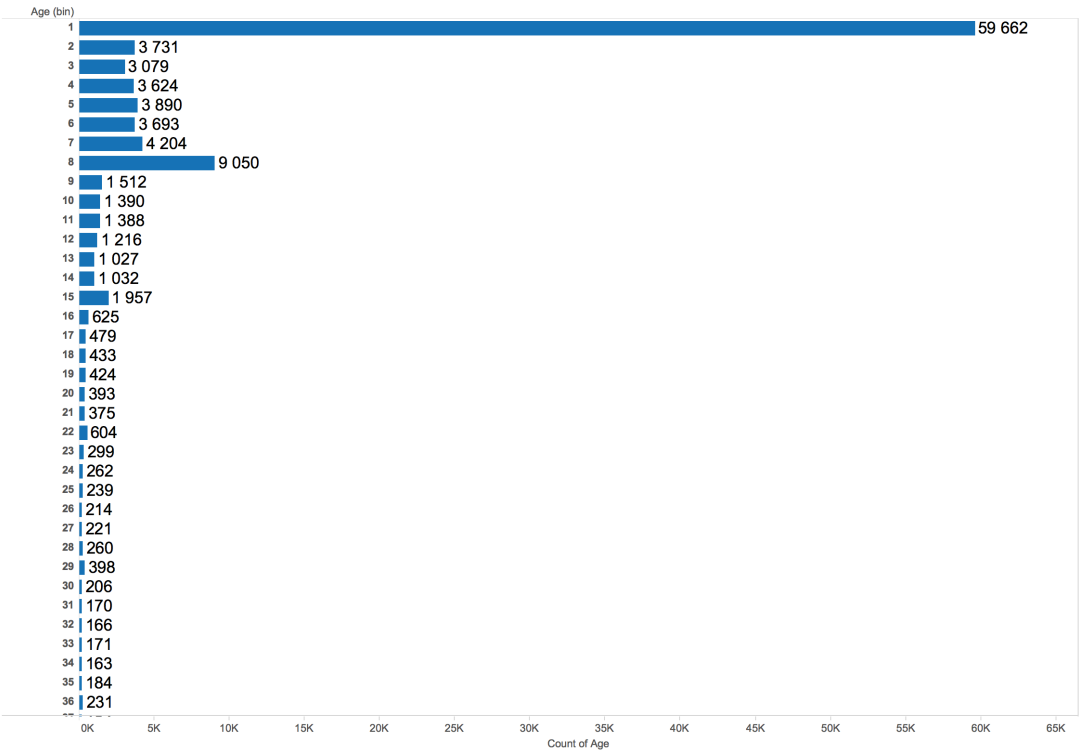

Figure 3 - Distribution of device age from November 2015 to February 2016

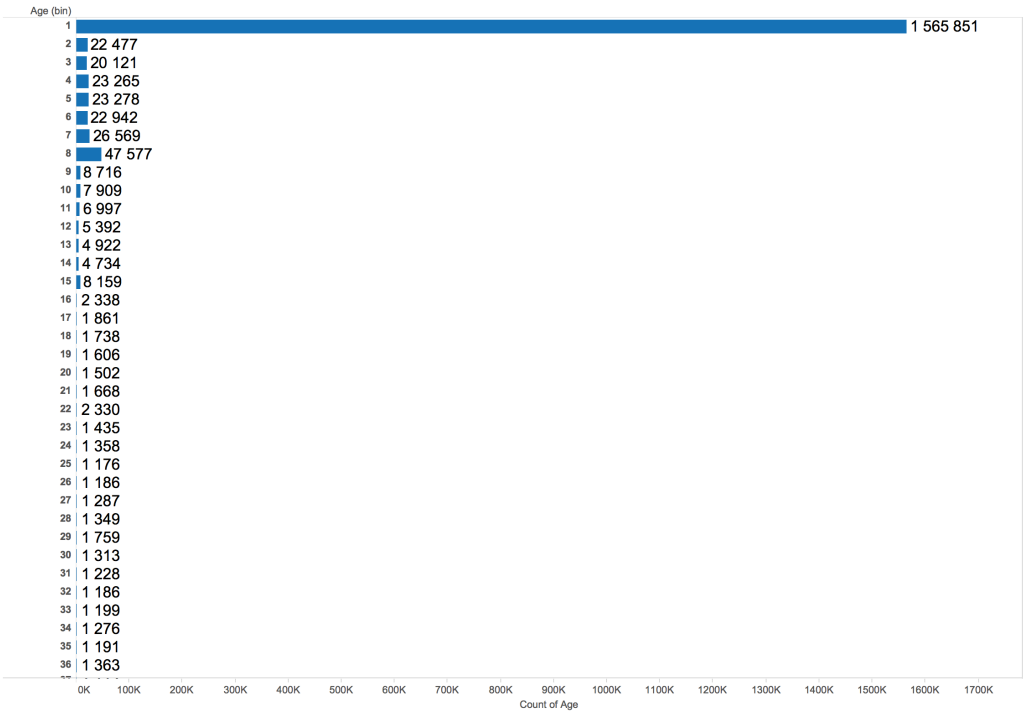

Figure 4 - Distribution of device age from November 2015 to June 2016

Due to the abnormal number of devices that only appear in the system one day, we decided to ignore them when assessing the visitor counts, and decided to define the threshold for a visitor as a device that appears in the system in an interval between 2 and 15 days, referring the rest as locals.
The disparity of the number of devices that are only detected in one day, is further analyzed in the results (section 5.2.2) and mentioned in future work, and can be explained by both the people appearing in the airport, and not appearing in any other router.

3.2.3 Events

After the system was installed we started noticing spikes in the data denoting recurrent or unique events. For instance, at the airport and port the system captures the flows of people corresponding to the arrival or departure of planes and ships. Similarly, and as we expanded the network of scanners, we started noticing unusual events corresponding, either to patterns of flow (weekdays vs. weekends) or sudden events (a concert, an exposition or even a football match). Therefore, we implemented an event detection algorithm trying to automatically identify abnormal occurrences in the data. This was done as an attempt to find the abnormal amounts of people in certain days that represent possible events that occurred in a location. Three types of detections were used:

- **Daily events** - Which compares each day with all the other days within the selected dates;
- **Weekday events** – Which compares each day with only the same weekdays days from other weeks;

- **Hourly events** - Which displays the number of events within 15 minute intervals for each day.

Both the formulas (Equation 1) and (Equation 3) for the daily and weekday events, are based on event detection from centralized energy monitoring [18], which relies on a threshold method, that is defined accordingly to the sum of the average and the standard deviation for the selected population (days).

$$t(P) = E(PX) \pm \sigma(PX) * p$$

$$f(Pi) = \begin{cases} 1, & i \geq t(P) \\ 0, & i < t(P) \end{cases}$$

P – place
PX – population of daily counts of one place
p – percentage of the standard deviation, that defines the thresholding
t – threshold for the population of one place
Pi – count for the iterated item

The p value, influences the distance threshold, between a value and the average (measured in the amount of standard deviations). When each value is compared against the calculated threshold, it is classified as being an event or not. This formula is based on a simple threshold, where the main purpose is to have a function to the threshold depending on the standard deviation and average, thus making the threshold automatic for every location based on the defined value of *t.*

The formula used for the hourly events (Equation 3) used a window thresholding, with a dimension output.

$$ds(X_i) = \frac{X_i - W_{li-1}}{\sigma(W_{li}) + X_i} \quad (3)$$

W_{li} – window of length l with the previous counts, starting from item i
X_i – count for the iterated item

The window length chosen was 3 and the *ds* function returns a number between -2.4 and 2.4, which, when compared against a tresholding value, returns the likelihood of an event. The detected events were later displayed in the visualizations against the ground truth, corresponding to a list of official events of the island gathered by the project team from official tourism websites.

3.2.4 Web scrapping

As an additional technique for collecting ground truth we used the web scrapping technique to obtain the flight information of the local airport about flight arrivals and departures, times, and locations. The scrapping consists in analyzing the structure of an HTML page[3], and retrieving the relevant information by running queries and selectors in the xml structure of the file.
This was done with Node.Js and the module *cheerio*[4], as a way of obtaining ground truth for the events detected on a daily basis, because the information about the flights is provided by the website for the previous day only.
The ground truth data was stored in the database and subsequently represented in the visualizations of the mobility trends. This enabled the possibility of verifying if a specific spike corresponded to a known event from the ground truth data scrapped from the airport website.

3.2.5 Vendor discovery

A MAC address is used to uniquely identify each network card, and each network card manufacturer is granted a certain range of addresses by the Institute of Electrical and Electronics Engineers (IEEE). From that range, the manufacturers program their production cards with the MAC address in a read only memory.

The MAC address is composed out of 6 bytes (256 bits), where each byte is represented by 2 hexadecimal characters, consisting in a total of 12 characters separated in groups of two (each group representing one byte). They are represented in a string separated either by hyphens (-) or colons (:), resulting in a 17 characters string as the following example: 1A:2B:3C:4D:5E:6F.

The first 3 bytes represent the manufacturer or organization, and the last 3 bytes represent the card identifier for that organization, as represented in figure 5. Because of this design, it is possible to know the manufacturer of the wireless card which is captured by our system.

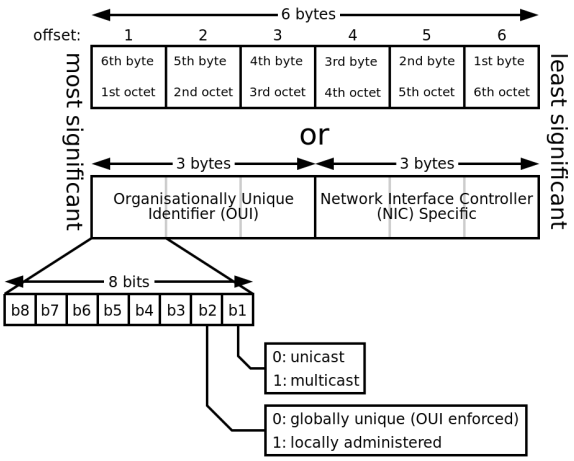

Figure 5 - MAC address composition

[3] http://www.ana.pt/en-US/Aeroportos/Madeira/Funchal/Arrivals/FlightInformation/Pages/FlightInformation.aspx?Day=-1
[4] https://github.com/cheeriojs/cheerio

Using the mac addresses received as a reference to the device, we used a free API from macvendors.co[5] to discover the manufacturer of the device which was broadcasting it, and plotted chart X with the device vendor distribution from all the devices captured by the system. This is done for every new mac address found in the system, and stored in the database.
Another API was also tested from macvendors.com[6], but imposed a daily limit of requests that was not compatible with our needs, and that was one of the reasons that led us to choose the former one. Other reasons include many mac addresses that were not found in macvendors.com but were found by macvendors.co.

3.3 Implementation Technologies

In this section we briefly describe the main implementation technologies used in this project, in particular for the server side components which includes the database and the middleware. The system was deployed using a LINODE virtual private server running Ubuntu 14.02 LTS, with 4GB of RAM, and a 2.5 GHz dual core CPU.

MySQL - The database used was MySQL, with InnoDB engine. This is a relational database, that ensures integrity over the captured and processed data. Since the nature of the data is primarily a time series, it has well defined fields that do not need the flexibility of a schema-free or object database, having a smaller impact on disk space.

Node.js – is a runtime environment to develop server side applications written in JavaScript, and uses the latest Google's V8 JavaScript engine, making it reliable and up to date. One of the main flexibilities is its modularity. The core libraries are small, meaning that most functionality is added through packages, this makes it very light, and ensures that package updates do not require core updates. Its asynchronous nature is ideal for deploying web services, due to it's high speed on replying to many clients, and its low resource consumption under high loads.

The web server is implemented through the express module, which is a framework for deploying HTTP web services, handling many parameters, the most relevant for us being caches, cookies and sessions.

JSON Web Token (JWT) – is a JSON-based open standard for transferring messages between parties in web application environment. It is cross language and it is developed for node.js as a module and ensures that a message is digitally signed using private keys to encode and decode the information. The tokens are designed to be compact, URL-safe and usable especially in web browser authentication process. It is used to transfer authenticated messages between an identity provider and a service provider. In this project it is used to upload the data from the routers to the server without requiring a login on the system, using instead an authenticated message. This ensures that we don't receive data from other sources than the routers.

[5] http://macvendors.co/api
[6] http://www.macvendors.com/api

Other technologies were considered for the web services, such as *php* and *python* and other types of encryption. We chose the *Node.Js* platform due to its high availability and performance on many requests, and due to its native manipulation of JSON objects, which has become a web standard. The encryption methods considered were based on AES algorithms, but simple client authentication was chosen instead with JWT through simple requests (without the use of the VPN), due to the non private nature of the data, and to reduce the overload of the processor, to process a huge amount of requests.

In the following figures 6, 7 and 8, we present the server loads for the month of May, in a total of 36 routers sending information about the captures probe requests. The peaks verified each day, represent the daily queries performed at midnight to filter the obtained data into auxiliary tables and the peaks verified occasionally represent manual database operations and code testing during the development of the project.

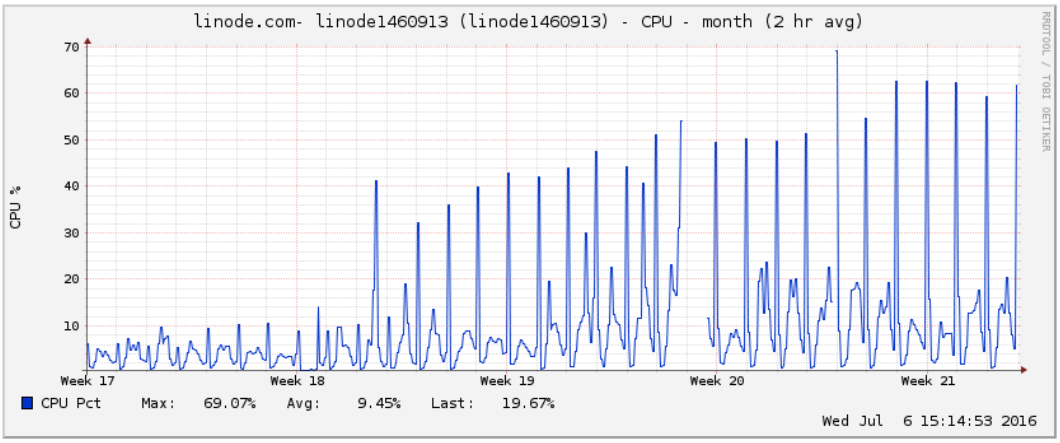

Figure 6 - CPU usage increase over deployment in May 2016

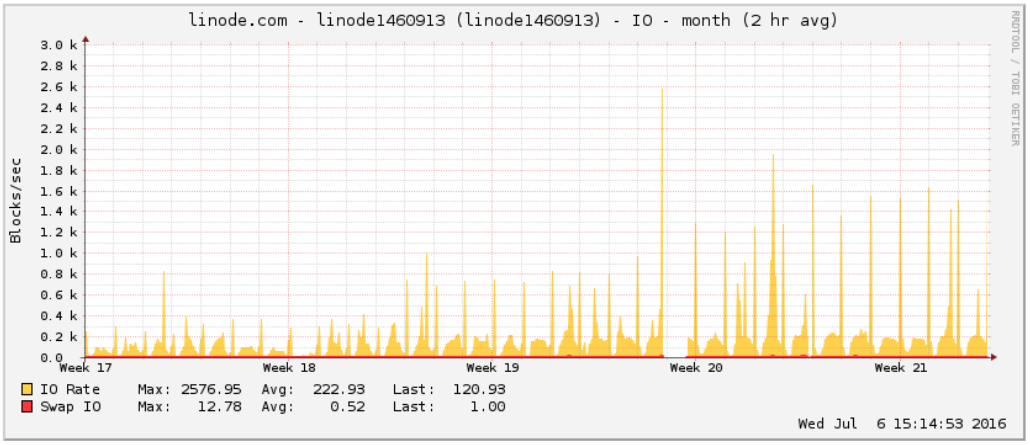

Figure 7 - Disk input/output over March 2016

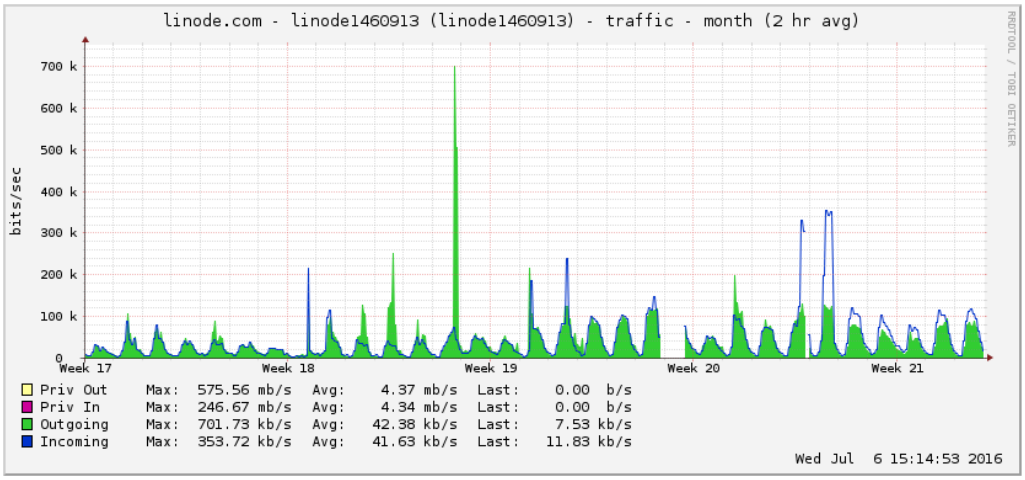

Figure 8 - Ipv4 network usage over March 2016

This is relevant to understand the current load of the system with 36 scanners, and estimate how it could possibly grow to more devices or making decisions to upgrade the hardware.

3.4 Visualization

For the client-side of the system our main concern was guaranteeing the coexistence of different packages, and few dependencies from server, eliminating the dynamic file generations by the server. For this, we used the API created to get general data for analysis, and adapted the data on the client to fit our needs. This way, the data from different services could be meshed together to create richer visualizations.

jQuery - is a JavaScript library that not only makes JavaScript easier to implement and read, but also adds functionalities to solve cross browser compatibility problems. It provides tools to manipulate and navigate through DOM elements, control events and animations. jQuery UI was also used as an extension to the former library, providing a set of user interface widgets, animations and themes created on top of the jQuery.

Highcharts – is a charting library written in pure JaaScript, offering an easy and friendly way of adding interactive charts to web sites or applications. Adaptable to modern browsers with graphs are rendered in SVG, also supporting legacy browsers, displaying them in VML. Highcharts ha as modular codebase, and is also the foundation of its two sister products, Highstock and Highmaps. It was used in the website to create graphs displaying the daily and hourly counts, among heat maps and inbound and outbound connectivity ratios.

Leaflet – is an open source JavaScript library to build interactive maps for the web. It is broadly used due its simplicity, high performance and customization. It supports various types of layer services, and GIS (Geographic Information System) formats, such as the ones from OpenStreetMaps[7]. It is built on a modular structure allowing plugins to extend its

[7] http://www.openstreetmap.org/

functionalities. In this project it was used to display the locations in a map, to show the connectivity between the different places, and animations for most active in a given timespan.

D3 – it is a JavaScript library to develop customized data visualizations using vector graphics for web browsers. The library is based on the creation of simple geometrical shapes, that when combined with user input events and custom CSS, result in fast and highly customizable diagrams, charts or animated content. The reason to use this library arose from the need to build a custom chord graph visualization for place connectivity, that was not supported by the Highcharts library.

Cytoscape.js – it is a tool for building graphs based on nodes and edges. It is often used for biological applications, and network graphs. The edges were also used to visualize the connections between places, allowing for the user to explore the data as a map, with zooming tools and clickable objects.

Other options were considered to display the maps instead of leaflet, such as google maps and Mapbox[8] (a branch of leaflet). Leaflet was chosen due its open source nature, and not needing the use of an API key, being able to run completely without the creators' dependencies.

To display the charts, the google charts and D3 were considered, but when compared against the Highcharts library, google charts provides less visualizations and less customizations. D3 on the other hand, offers a free canvas to build complex visualizations from scratch. Although being even more flexible than Highcharts, it has a steeper learning curve, and does not provide boilerplates for simpler visualizations.

All these libraries and frameworks were joined together in the client side of the web environment, due to the common use of JavaScript. All of them have have good documentation on the web which makes the development and integration simple, and do not generate conflicts between each other. Due to their modularity, they can also be extended with features to fulfill the developers' needs.

In the next chapter, we present the system implementation where the technologies mentioned in this chapter are used.

[8] https://www.mapbox.com/

4 The Beanstalk Tracker

In this chapter we present the hardware and software implementation details of the system. We start by explaining the system as a whole, and then move to the router/tracker and its operating system. We then present the database structure, data analysis and the web services that support the client-side visualizations.

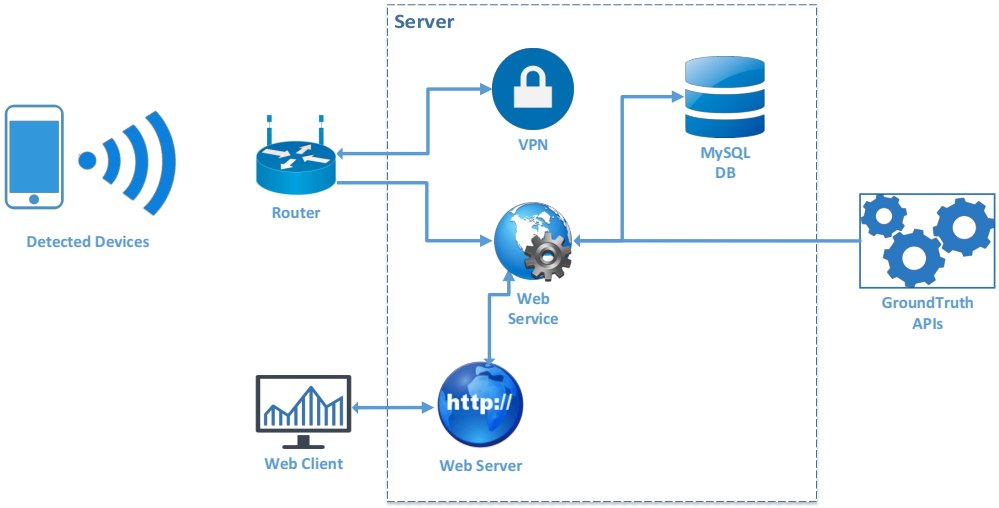

Figure 9 - System Components

The tracking system represented in figure 9 comprises the capture of the probe requests from the passerby devices through a modified commercial Wi-Fi router. The router operates in monitoring mode and sends the probe request information to a MySQL database. The server side components perform the calculations and optimizations required for analyzing the captured data and provide the results through a web server to the clients. The routers are connected to a VPN allocated on the server to allow their remote management, and the scripts processing the data interact with several extern services and APIs.

Following, we present the design of the router modules:

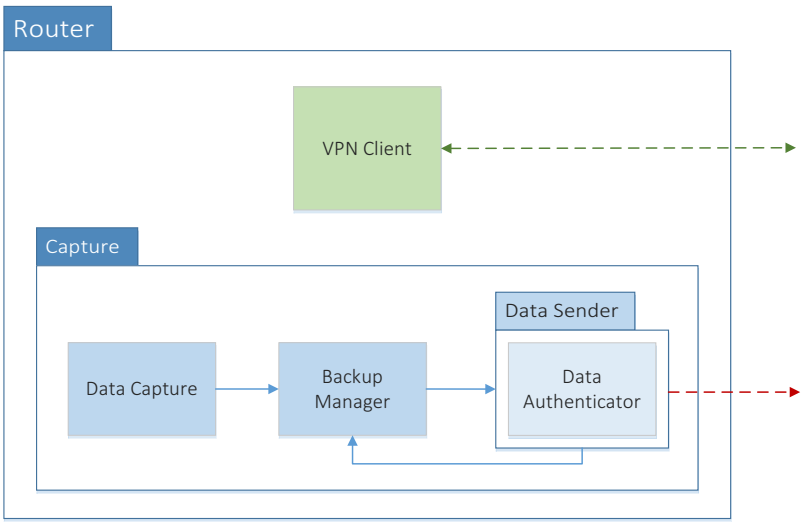

Figure 10 - Router Modules

With the modules presented in figure 10, we allow the data to be securely sent to the server through he authentication component. When the data sending fails, the data is stored into a backup, that is later sent when the connection is available. Adding to this, there is a VPN client software to access it.

In figure 11, we present the modules from the server, that include the data receiver module that serves as a middleware to insert the collected data into the database. The database is represented in the MySQL component, which has routines to process the collected data. It also has the web client services, to serve the https requests to visualize the data. The VPN server is allocated in the server to communicate with the clients in the boxes.

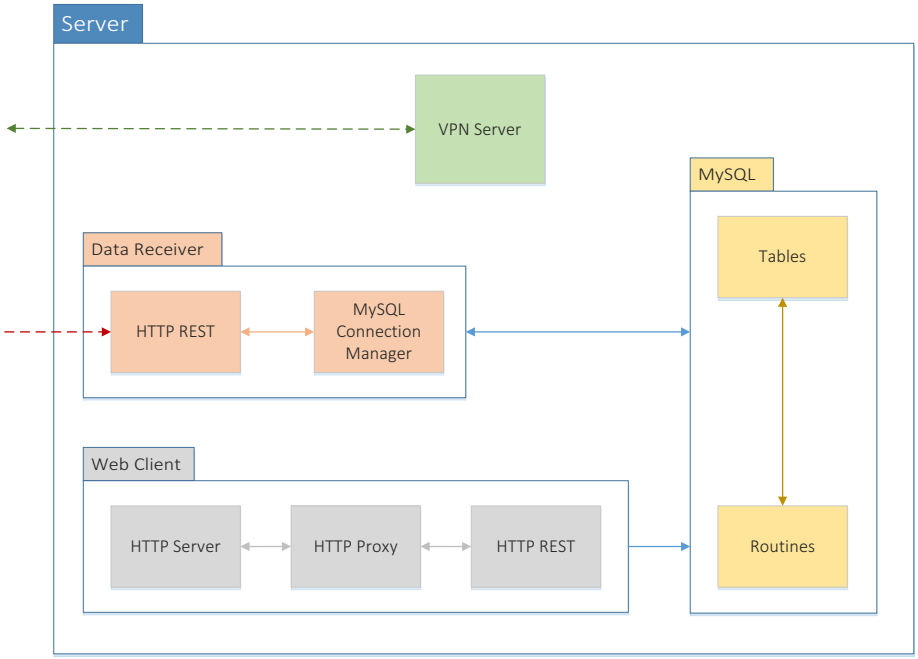

Figure 11 - Server Modules

4.1 Tracking platform

There are many solutions to implement Wi-Fi monitoring. From, commercial solutions costing more than 800€[9] to custom-made solutions based on Arduino. After considering all the options (table 4) we decided to use a commercial home router to capture the data. Since we were targeting multiple locations and wanted a robust solution with antennas, we opted for installing Linux on an existing and cheap commercial router costing around 45€. To capture the data we opted for the TP-Link MR3240v2 home router was used with a compatible open source operating system - OpenWRT – Barrier Breaker 14.07.

In figure 12, we show the first round of 10 routers deployed and on figure 13 the router in 5 de Outubro street. Below, there is a table with the technical hardware specifications.

[9] http://www.libelium.com/products/meshlium/

Figure 12 - TP-Link MR3420 v2 Router

Figure 13 –Router positioned at 5 de Outubro Street

Table 3 - TP-Link MR3420 v2 Hardware Specifications

Interfaces	1x USB 2.0 port	
	1x 10/100 Mbps WAN port	
	4x 10/100 Mbps LAN port	
Wireless features	Standards	IEEE 802.11b/g/n
	Frequency	2.4 - 2.4835 GHz
	EIRP*	< 20 dBm
	Antenna	2x 5 dBi
	Antenna Type	Omni Directional, Detachable, Reverse SMA
Hardware	CPU	Atheros AR9341 @535 Mhz
	ROM/Flash	4 MB
	RAM	32 MB
	Wireless	Atheros AR9341
Others	Certification	CE, FCC, RoHS
	3/4G modem compatibility	LTE/HSPA+/HSUPA/HSDPA/UMTS/EVDO

*EIRP (Equivalent Isotropically Radiated Power)

Both the internal memory and the swap memory were expanded with an external flash drive with a minimum of 512MB (256MB for storage and 256MB for swap). This flash drive was used in the USB port that was originally designed for the 3/4G modem announced as a feature of this router.

The wireless interface was assigned to a single network, and configured to monitoring mode. This ensures that no network name is broadcasted, denying the possibility to gain access to the device wirelessly. It also hides the router from other capturing devices, since it does not broadcast packets, and simply monitors the surrounding traffic. Both the control web page and SSH (Secure Shell) accesses were secured with a 16 char alphanumeric randomized password.

Because the scanning from the target devices is done sequentially in every channel supported by the device, listening to the probe requests works regardless of the monitored channel. Two methods could be implemented for this: either monitor in a single channel, thus losing the probe requests for other channels, or perform channel hopping. In this technique, the channel of the interface is changed frequently between the most used channels. The problem with this approach is the time it takes to reconfigure the interface, which although being less than a

second, if performed frequently, could lead to a major capture loss, being possible to completely ignore many devices. As stated in [9] that "*the cost of channel hopping does not overweigh its advantages, resulting in fewer devices detected in total*".

For this study, we chose to listen to the channel 11 since it is one of the three recommended non-overlapping channels (1, 6, 11) [19]. Therefore we reduce the interference from other of channels and frequencies, while checking one of the most common factory setting increasing the changes that more devices could be operating in this channel. Though the channel doesn't influence the capture, because as explained above, the devices cycle through all the channels in the active scan.

A custom script in Python was added to run at the boot of the router, to start capturing in the wireless interface in monitoring mode. This basic script uses the capturing package *Scapy,* to capture packets and filter them for the type 0 (management) and the subtype 4 (probe requests). It also takes into account the big amount of repeated messages (same mac address and SSID) from the same device, and doesn't send to the database repeated captures in the configurable interval of two minutes to avoid data duplication. It is also programmed to record the data internally in the flash drive storage, when the connection to the receiving service is not available - caused by server down time, or internet unavailability - and sends the internal backups when the connection is restored. The probe requests detected are sent to a remote database through a web service with JWT authenticated messages in HTTP requests.

4.1.1 Hardware options considered:

There are several pieces of hardware available on the market that allow this type of data gathering, and the common requirements of them are: 1) A Linux based operating system (not mandatory); 2) Wi-Fi card + antenna(s) 3) Internet connectivity.

Table 4 – Hardware options comparison

	Raspberry Pi 2	Beaglebone	Arduino Yun	TP-Link MR3420
CPU	ARM 900 Mhz	ARM 720 Mhz	ATmega 32u4	Atheros AR9341 @535 MHz
RAM	1 GB	256 MB	2.5 KB	32 MB
Storage/Flash	-	-	32 KB	4 MB
Ethernet	1x10/100 Mbit/s	1x10/100 Mbit/s	1x10/100 Mbit/s	4xLAN 10/100, 1xWAN 10/100
Wi-Fi card	-	-	-	Atheros 300Mbps
Wi-Fi antenna	-	-	-	2x Omnidirectional
External casing	-	-	-	Yes
USB	4x2.0	1x2.0	1x2.0	1x2.0
Price	35€	45€	50€	45€

While all these options are relatively low priced, they all need extra components in order to capture data, such as the wireless card, wireless antennas, and the need to build or buy custom boxes to accommodate the hardware. And those were the reasons to choose the option from TP-Link, due to providing a ready to deploy solution.

4.1.2 OpenWRT

OpenWRT is a Linux distribution for embedded devices, which provides a fully writable file system with package management. This allows users to access settings beyond the available factory page provided by the vendor, extending its functionality.

It is also used to implement advanced functionalities that can usually only be found on expensive and professional equipment. With the installation of the necessary packages a simple home router can implement advanced features, such as a RADIUS server for authentication, VLANs for testing network configurations, setting up VPNs, perform sniffing on the network or even setting up VoIP system.

Being open source, allows for constant development and updates made by the community. This ensures that security risks are patched regularly, as well as the custom packages being developed to extend functionalities.

4.1.3 Remote Management - VPN

The need to control the routers remotely arose as a requirement to perform maintenance checks, and possibly modify the running software without having to travel to each individual location. We chose to use a VPN software that was compatible with both OpenWRT and the server's operating system. The official recommendation was OpenVPN. It is an open source software that implements VPN techniques to create point-to-point connections with standard security protocols. It allows authentication with pre-shared keys, certificates or username/password using TLS technology.

This VPN service was installed both on the server and on the routers as a package. This ensures that the routers can be accessed from the server, without having to configure any port forwards on their network. The network defined in the VPN is of class A, with a subclass 1 having the network address of 10.1.0.0 and a netmask of 255.255.0.0. The server is assigned to the address 10.1.0.1.

The server includes a set of private and public keys used to authenticate the server to which the device is connecting to. The public key is present in the routers and login credentials are used to authenticate them based on a username and password (16 characters alphanumeric). The key used to authenticate the server is used with the RSA algorithm of 2048 bits, and all the connections are encrypted with a cryptographic cipher BF-CBC (Blowfish - cipher block chaining).

Once connected to the VPN, each device is identified by its username, and can be accessed through SSH, allowing total control of the operating system and its processes, or through a web configuration interface to use the available routing and firewall settings. This allows us to perform maintenance checks, script modifications, or any other type of software configuration that could be normally done when connected directly to the device.

4.2 Database

An important backend service is the database used to store the data originating from the trackers and also the analysis calculations performed on the server. The database is logically divided into three main sections:

- Tables that hold data relative to the capture of the probe requests, store the results of the processed data and daily analysis;
- Tables relating to the users, and permission information to access the web pages, API, and data visualizations;
- Tables relating to the daily summaries to optimize the data visualization queries.

If the following subsections we discuss the different sections of the database separately.

4.2.1 Data capture and processing

The structure of the capture and processing (figure 14) relies on the central table of the **probe requests**. The probe requests are divided into components, and which are linked to the probe requests table through foreign keys. This keeps data in the 5th normal form, without data redundancy, and with separated concepts.

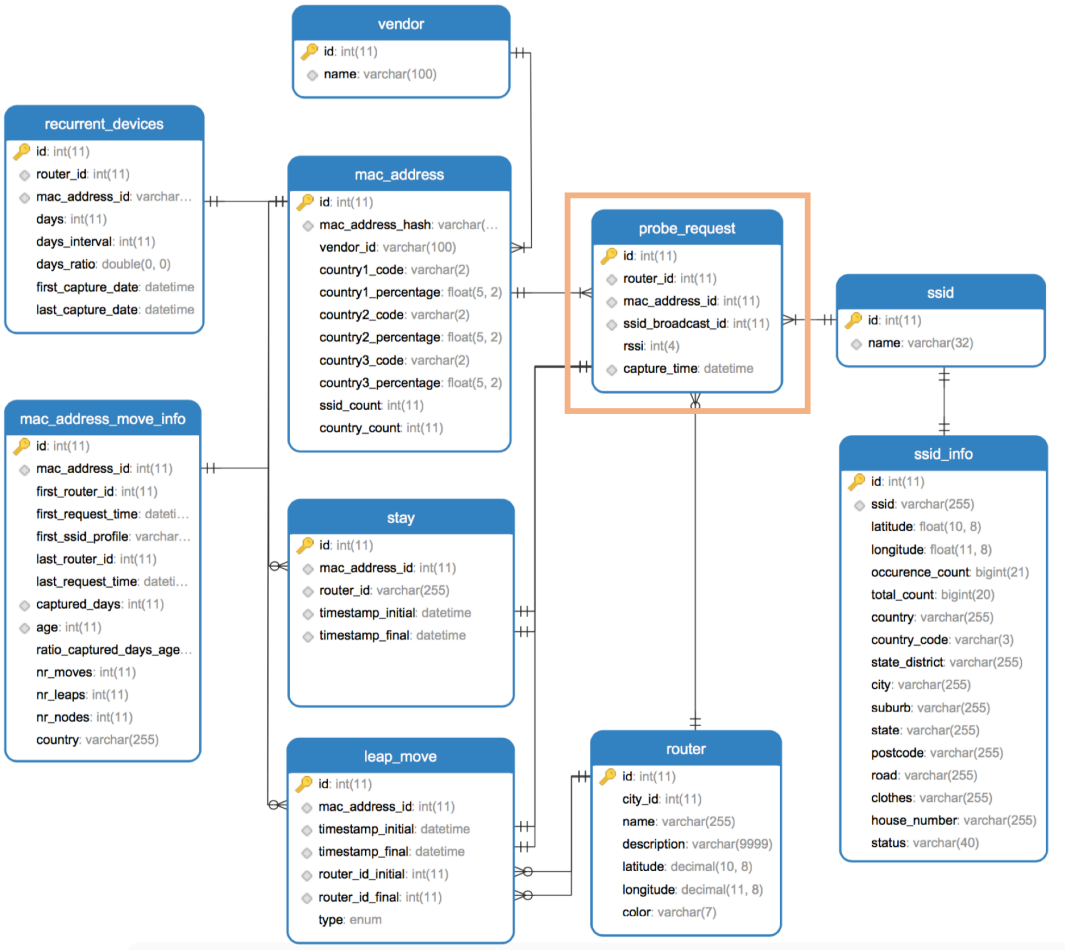

Figure 14 - Capture and process database schema

Table 5 - Capture and process database tables description

Table	Description
router	Contains the information about each individual router, linked to a city id, and a pair of geographical coordinates.
city	Identifies each individual city with an id and a name.
probe_request	Stores the packet's information captured by the routers in the form of foreign keys, linked to a mac address and an SSID
mac_address	Stores information of the mac address, linked to the other tables through foreign keys
mac_address_move_info	Stores information updated daily regarding the first and last capture times, locations, and the number of leaps, locations and total number of days in the system
ssid	Stores information about the SSID, and its geolocations
vendor	Contains an identifier for every vendor detected by the system.
leap_move	Contains data relative to the movements of between routers by the same device. With the timestamp and initial place, and timestamp and final place.
stay	Contains the data relative to the periods of the same device being on the same location for a time interval.
recurrent_devices	These are devices that spend great periods of time in the same location with a defined periodicity.
ssid_info	Contains the information relative to the SSID's broadcasted by the devices such as their original coordinates, country and region.
country	Contains a list of 252 official countries defined by their 2 letter country code and English name retrieved from http://www.iso.org/

The daily queries performed to increase the performance for the data visualizations, are stored in the following tables (figure 15):

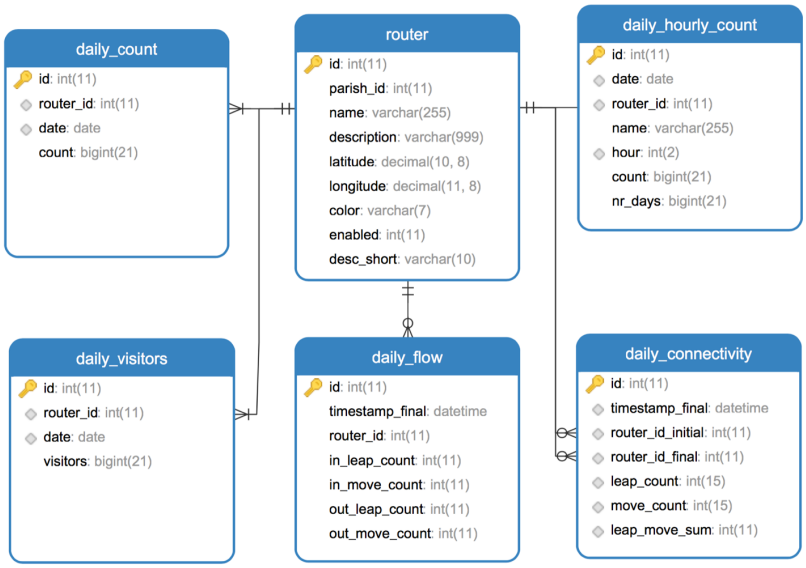

Figure 15 - Daily summary tables

4.2.2 Users and Permissions

This structure implements a system of users, and permissions, which are linked together via groups. Each user belongs to one or more groups, and each group can have access to multiple pages, services and routers. This is represented in figure 16:

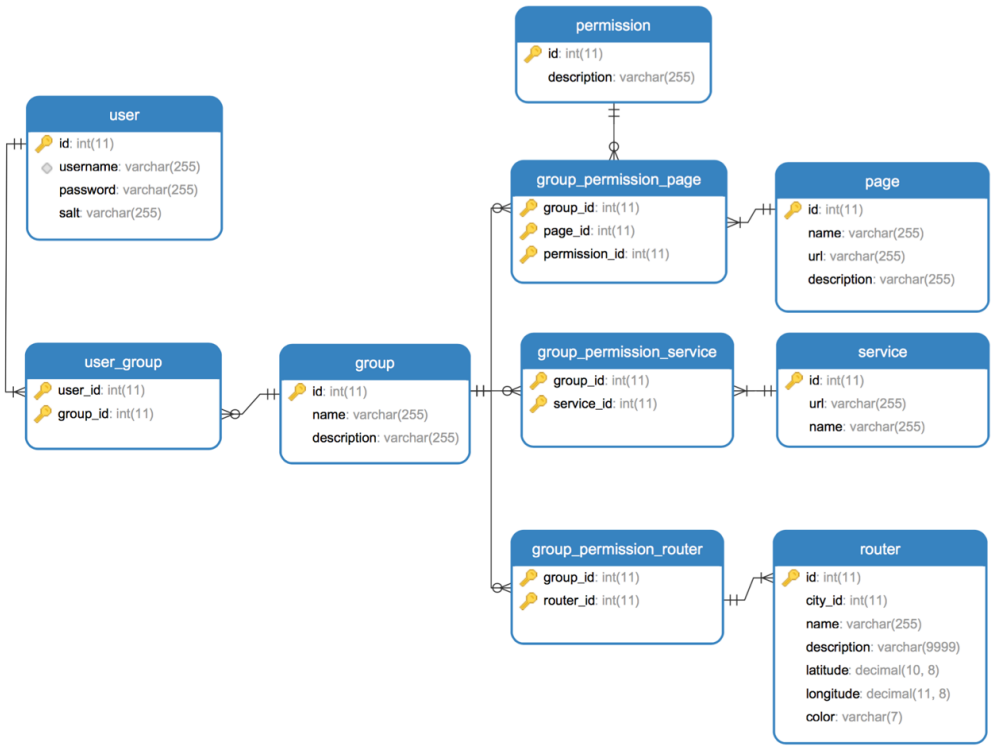

Figure 16 - User database schema

Table 6 - User database tables description

Table	Description
user	Contains the user private data, such as user name and hashed login credentials.
user_group	Contains the association between a user and zero or more groups.
group	Contains the individual groups, to be associated to the user and the permissions.
service	Contains the web services names and urls
group_permission_page	Contains the association between the group and the pages, and the type of access, to which the users in the group are able to access,
group_permission_service	Associates each group with a series of web services
group_permission_router	Contains the association between the group and the routers, to which the users in the group are enabled to retrieve data from.
router	Described in table 3.
page	Contains the records of web pages to which each group of users has access to.
access_log	Is used to record the accesses made to the system, including the logins, and the API calls, recording each user and the data accessed.

4.3 Data Visualizations

In this section, we present the data visualizations developed for the project and the API supporting them. The technologies used were Node.Js for the web services, serving the results in JSON format to the ajax requests from the user interface. Below, in figure 17, we present the sequence diagram of a request from a web client.

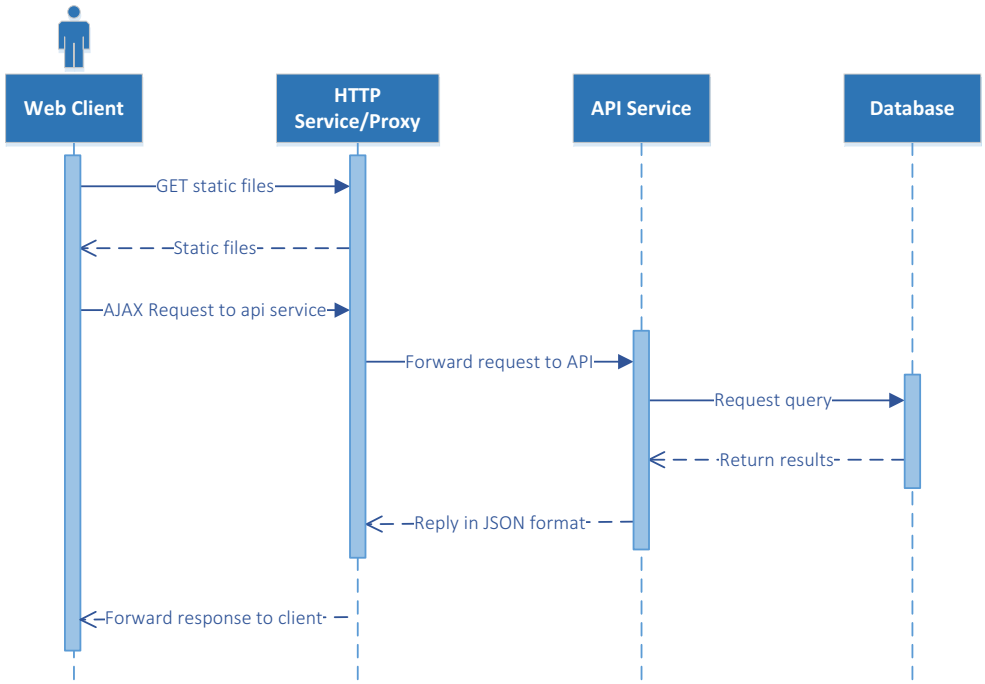

Figure 17 - Sequence of web client requests

The files used by the client, are static instead of pre-processed by the server, meaning that the dynamic parts of the interface are built on the client side with JavaScript. The client itself makes ajax requests to the API to retrieve the necessary data, and renders it dynamically on the page, allowing for the exploration of the data without constant page switching/reloading and a faster experience to the user.

This allows the code to run on any HTTP server, without dependencies on the server providing the static files. The only configuration necessary being a proxy pass to the API, so that the session cookies are passed to the domain for user authentication purposes, since cross reference requests do not allow to send cookies.

4.3.1 API

The processed data from the database is delivered to the web clients through a REST API. The technology used was the node.js, as already used for the service receiving the probe requests. This API runs on port 8001, and is accessed by a proxy-pass configured in Apache web server with the alias /api/. All the responses are delivered in JSON format, including the data, and error messages.

To use this API, the users needs to make a login before being able to access any other service provided by it. Following, we present the web pages that compose it, and the API services used to power the visualizations.

The following services are provided, where the variables in brackets, represent variables passed by url. The date variables accept the format of YYYY-MM-DD, and are not always required, resulting in responses that account for all the summaries since the routers have started capturing data.

Services available:
/api/login

- used to authenticate a user in the system for further access to the services.

/api/has-login

- returns the login status from the session cookies present or not in the request.

/api/get-routers

- used to retrieve meta-information for each router, such as geographical coordinates, color, description or city.

/api/get-correlation/startDate/{**date**}/endDate/{**date**}

- returns an array of elements with source, destination and counts of leaps.

/api/get-daily-count/minutes/{**integer**}
/api/get-daily-count/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns an array with the daily counts of one router (or all if id is omitted) filtered between the dates (if specified).

/api/get-daily-visitors/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns the daily count of visitors for the selected router between the selected dates.

/api/get-place-count/minutes/{**integer**}

- returns the count of people for each router in the last specified minutes.

/api/get-hourly-count-per-weekday/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns the average count of devices per hour of the day for each of the weekdays

/api/get-hourly-stay-average-per-weekday/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns the average stay times in seconds per hour of the day for each of the weekdays

/api/get-events/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns an array of the events detected by the system for the selected router between the selected dates.

/api/get-event-gt/id/{**integer**}/startDate/{**date**}/endDate/{**date**}

- returns an array of the events ground truth gathered in the airport scrapping and/or the touristic events for the region of the selected router between the selected dates.

4.3.2 Web Pages

The following visualizations and data interactions were created by the design team of the project, and implemented with the client side JavaScript language, in interaction with the API from the previous topic. Processing is done both by the server and the client, where the data delivered by the API is sometimes generic and not dependent on the specific design implementations. Following, in figure 16, there is a navigation map and details about the pages and elements composing them.

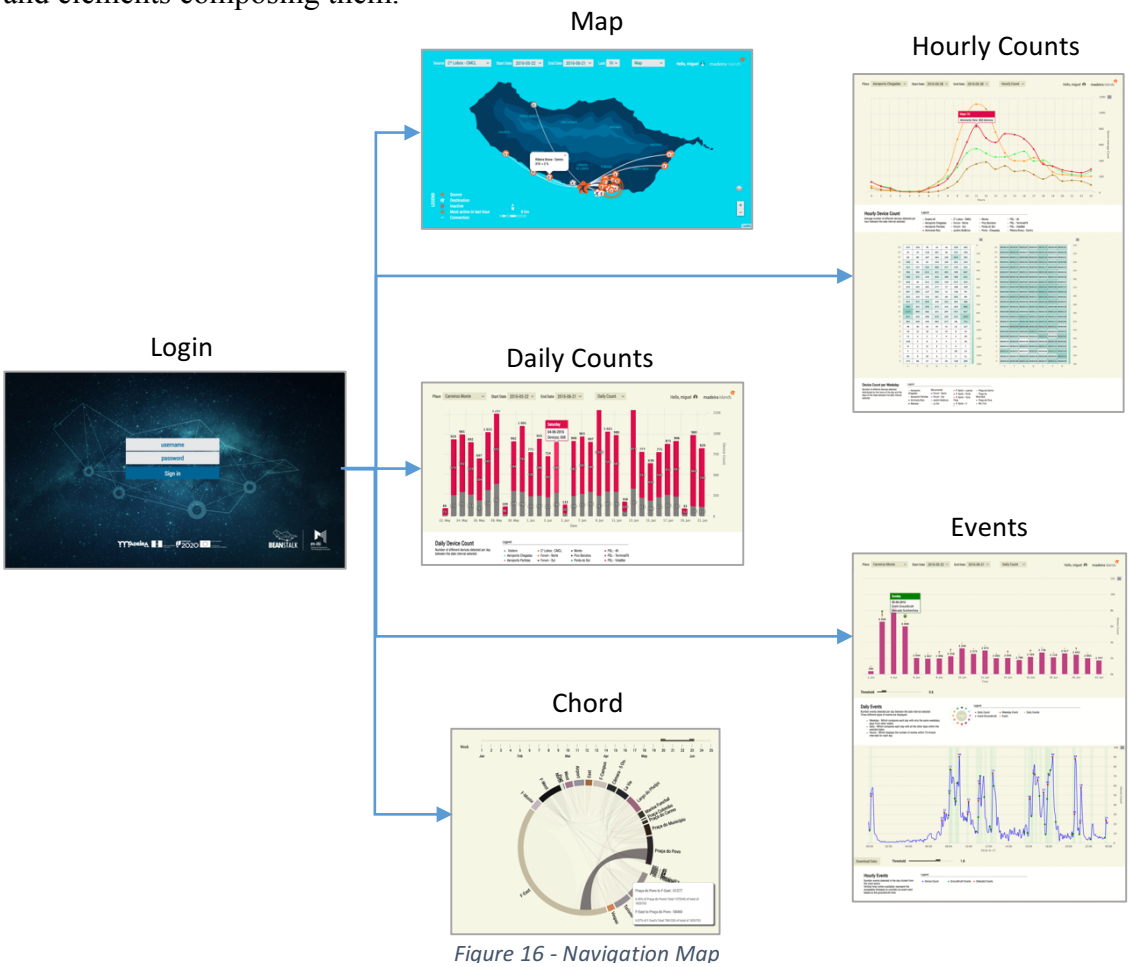

Figure 16 - Navigation Map

The filtering of the displayed data is done through basic date picker controls, which allow the selection between two dates, and the place to target the data source.

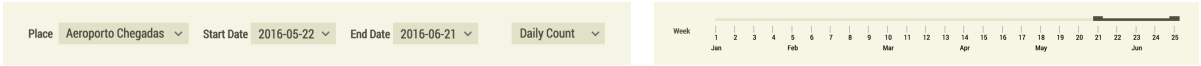

Figure 18 - Filtering place and dates by day and weeks

4.3.2.1 Login

The system was accessed with credentials that were given to the associated entities, and had different page permissions depending on the entity, which usually could only access the locations associated with their institution.

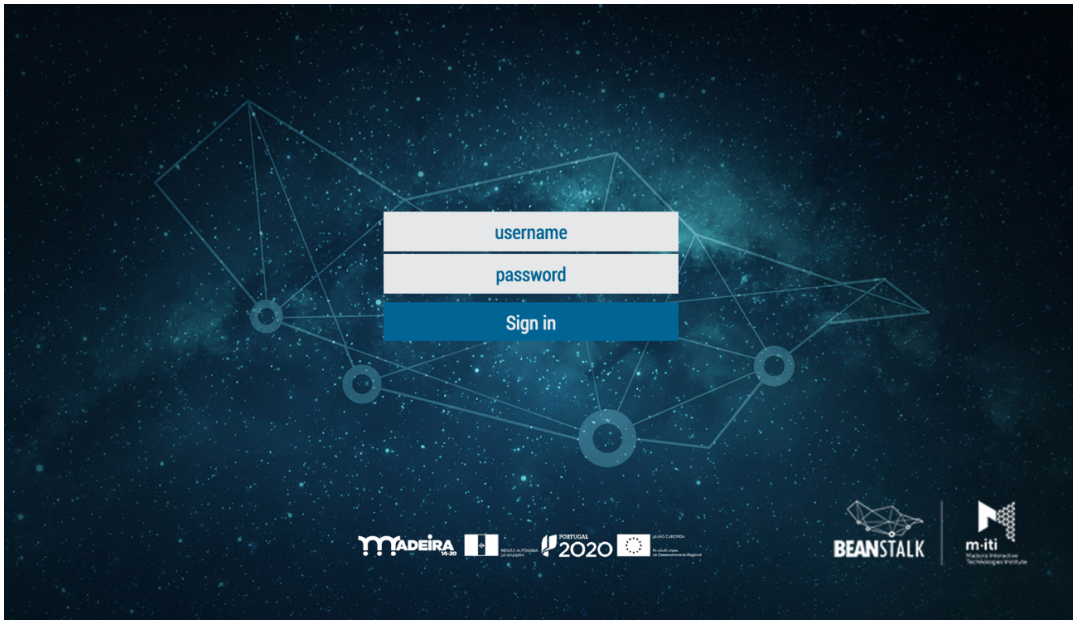

Figure 19 - Login web page

4.3.2.2 Map

In this page, we gather different types of information together in an abstract map of the island. The visualizations include the installed routers geotagged, current status of each router, the amount of leaps between the selected place and the rest the places with most activity in the last hour. The size of the flower icon is directly proportional to the number of leaps from the selected place. The map is built with the **Leaflet** framework and OpenStreetMaps layers.

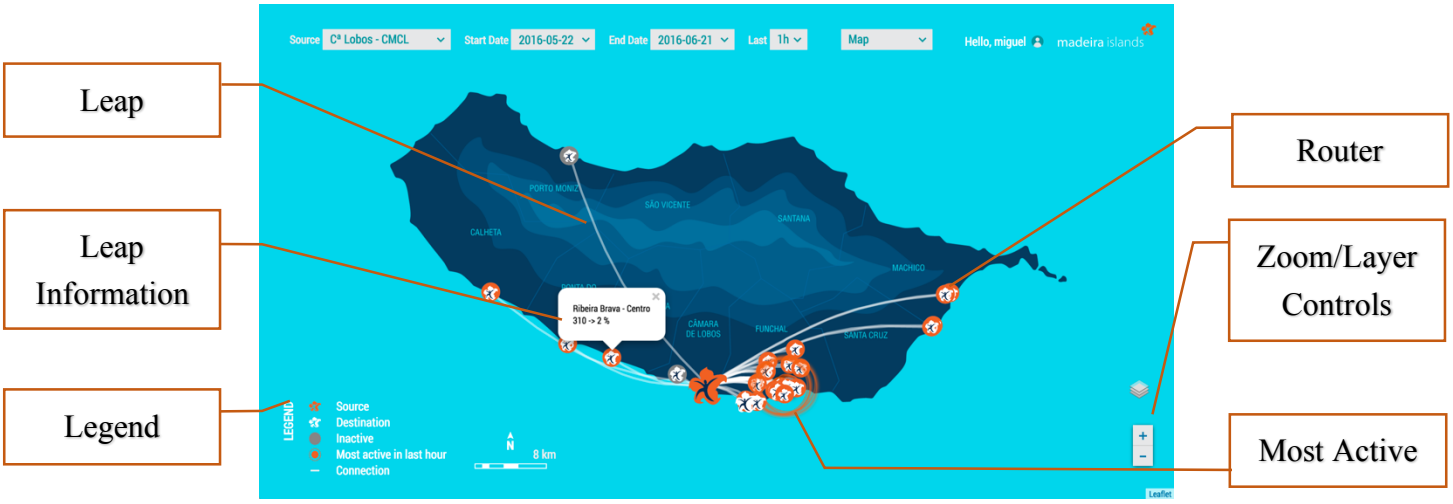

Figure 20 - Map web page components

Interaction

- A source place can be selected from the select control on the top bar, or by clicking directly on a router icon.
- When hovered, each router displays additional information about the leap originated in the selected place.
- Zoom can be applied with the on screen controls, through the mouse or pinching action.
- Other layers from OpenStreetMaps can be chosen from the layer control.

Web services

- /api/get-routers – used to retrieve geographic coordinates and names for each router
- /api/get-correlation – used to draw the leaps information between each router
- /api/get-place-count – used to display the places with most activity

4.3.2.3 Daily Data

In this page, we present the daily counts of people for the selected router. The data is also enriched with the display of the visitor counts as grey. It is already possible to view the difference of the Sundays compared to the rest of the weekdays in the selected place. The graph is powered by **Highcharts** framework.

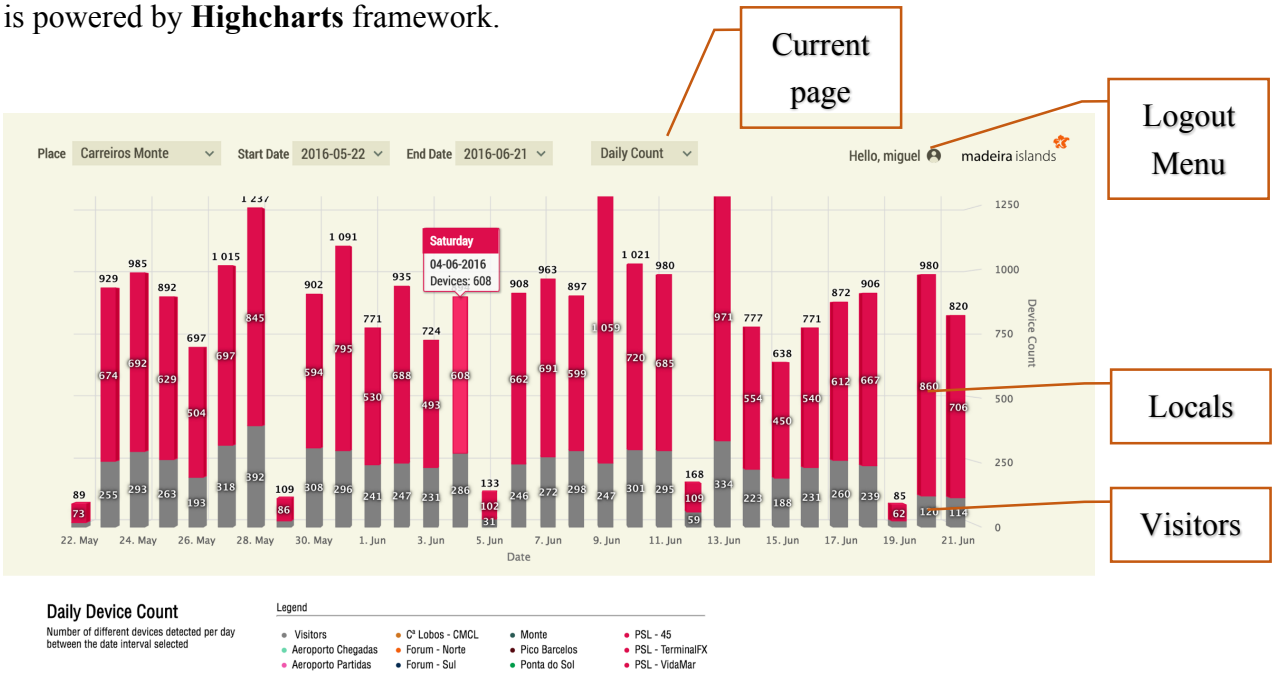

Figure 21 - Daily Counts for Carreiros do Monte between 2016-05-22 and 2016-06-21

Interactions

- Hovering over a column, displays the date, the day of the week, and the count of the hovered series.

Services used

/api/get-routers – used to retrieve the colors and names for each router
/api/get-daily-count – used to build the total daily count chart
/api/get-daily-visitors – used to display the number of visitors for each day

4.3.2.4 Hourly Data

Three types of representation are displayed here. With two visualizations for the the hourly count of devices and one for the the hourly stay time, both powered by Highcharts framework.

With figure 22 we represent a spline graph, where we show the average device count in the different places, spread by the hours of the day between a chosen date interval. We can spot a general tendency to have bigger activity from 7 a.m. to 9 p.m. It is possible to select only the desired places to perform comparisons between them.

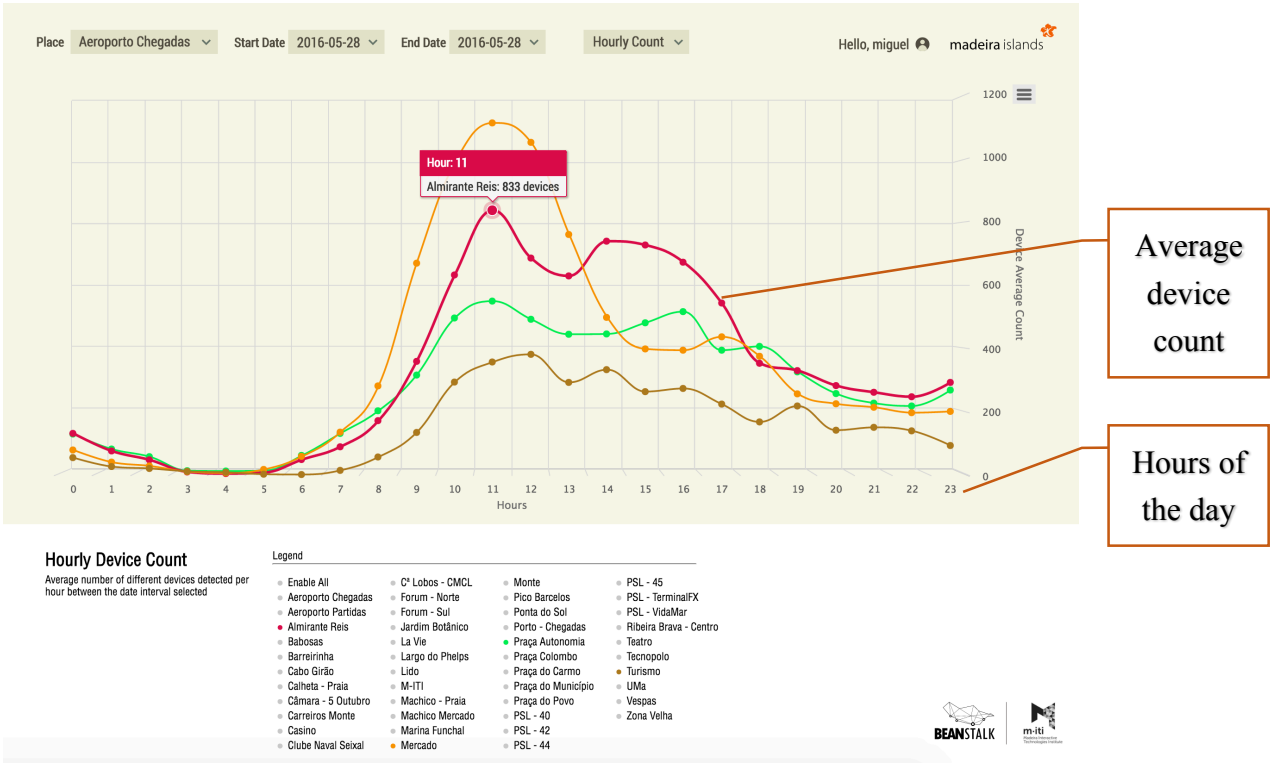

Figure 22 - Average hourly counts

Interactions

- Enable or disable places from the controls in the legend.
- Click on a series line, to select a single series.
- Hover over a series, to display the device count average for that hour.

Services used

- /api/get-hourly-count/ - to display all the routers in the spline chart

The heat map in figures 23 and 24 represents the device count in a single place against the different days of the week, and across the 24h hours of each day. It is queried directly to the data tables, and provides live information. This visualization allows to see tendencies towards certain days of week in certain hours of the day. It can be useful to store owners or to plan events that need a certain amount of public. Since this graph can display all places at once, the place selection control is used for the next two heatmaps in figures 21 and 22.

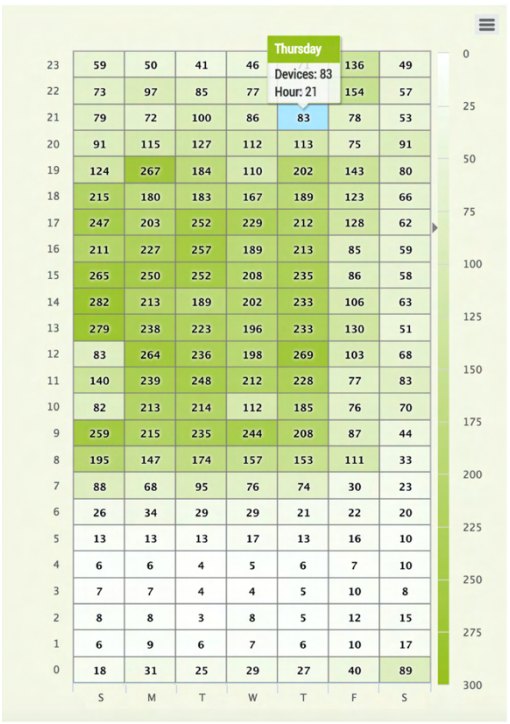

Figure 23 - Hourly average **count** per weekday for Mercado

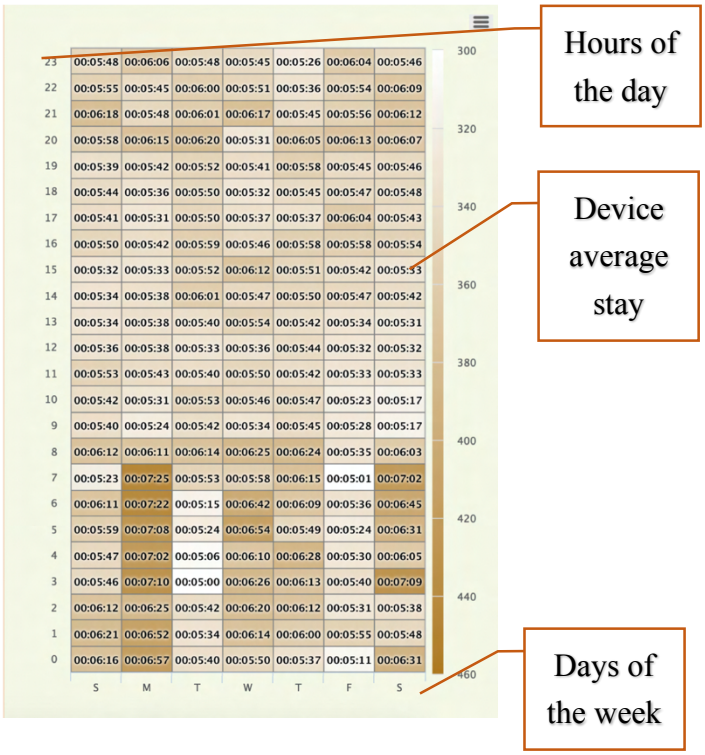

Figure 24 - Hourly average **stay** time per per weekday for Praça do Carmo

Services used

- /api/get-hourly-count-per-weekday/ - used to build the matrix of day/hour counts
- /api/get-hourly-stay-per-weekday/ - used to build the matrix of day/hour stay times

4.3.2.5 Events

This page represents the events detected by the system, alongside with the official events from Madeira island, and the flight information from the airport. This information is shown per day, and it is also possible to select a day and visualize the time when the events were detected. The ground truth events that have a specific time associated instead of just a date, such as the airport flights information, are also displayed in the hourly events graph, in comparison with the events detected by the system. The reason of choosing the airport as a source of ground truth is explained in chapter 6.

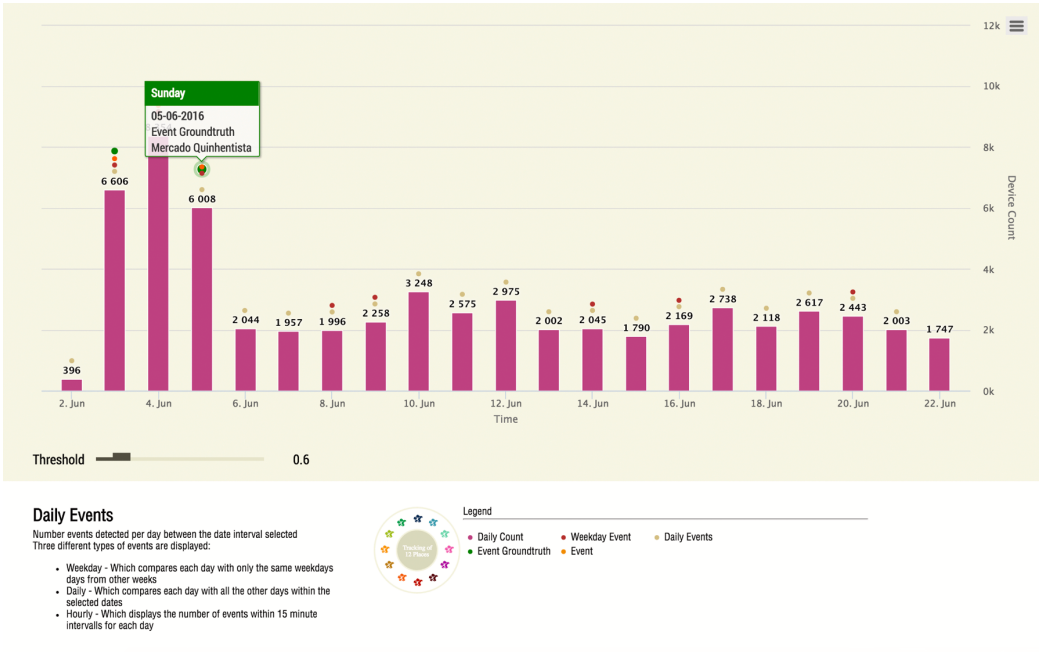

Figure 25 - Daily events

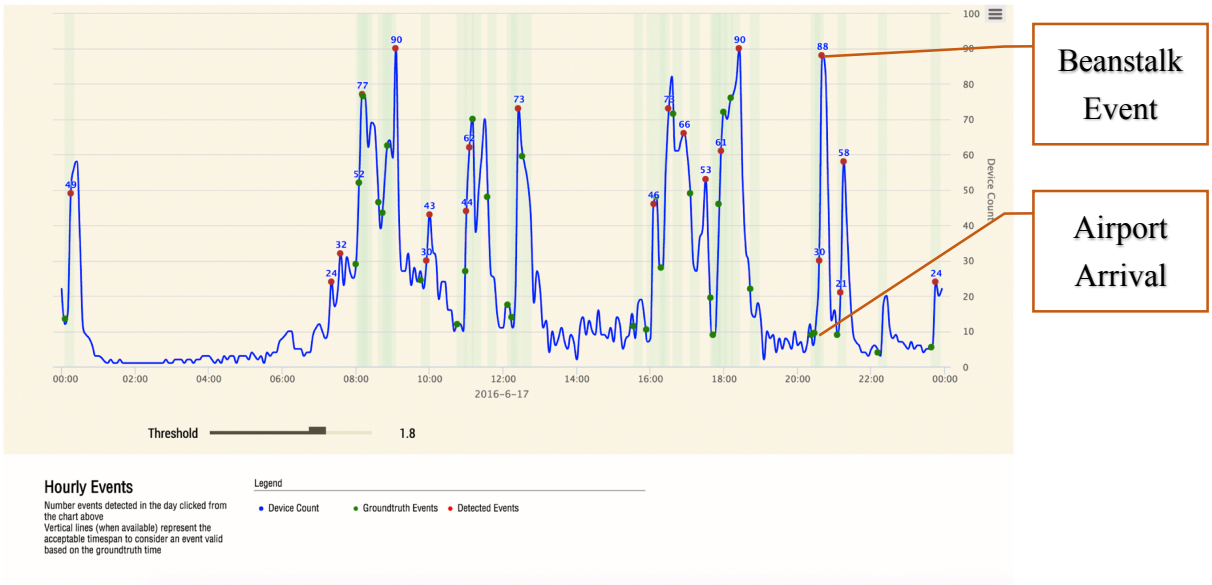

Figure 26 - Airport hourly Events

We should note that the events detected by the system, have a usual delay between 5 to 15 minutes to the ground truth, which represents the delay between the plane landing and the arrival of the passengers to the luggage room. The hourly ground truth information is only available for the airport, as the other places only have events that last all day with no specific time.

Interactions

- Hover graph, to display the event and ground truth information.
- Click on a daily event to display the hours of the day, and the detected events.
- Choose between spline or bar chart.

- The threshold control, allows to adjust the threshold for the detection algorithms in real time, also aiding in the future to understand the best threshold for each location.

Services used

- /api/get-daily-count – used to build the series relative to the daily counts
- /api /get-events – used to build the series relative to the events detected by the system
- /api /get-event-gt – used to build the scatter series of the event ground truth
- /api/get-hourly-count – used to build the hourly events chart

4.3.2.6 Connectivity

In this page, we provide a chord diagram that represents the moves and leaps between the different regions, and the routers themselves. The minimum time interval is a week (due to the small counts that occur in a day being less significant to analyze), and a new dedicated control was added to filter the weeks of the selected year. The diagram allows a visual exploration of the major leap counts providing the user with a tooltip for further details of the hovered connection.

It is powered by the D3 library, and contains movement animations so that the user can clearly understand what happens throughout his interactions with the system.

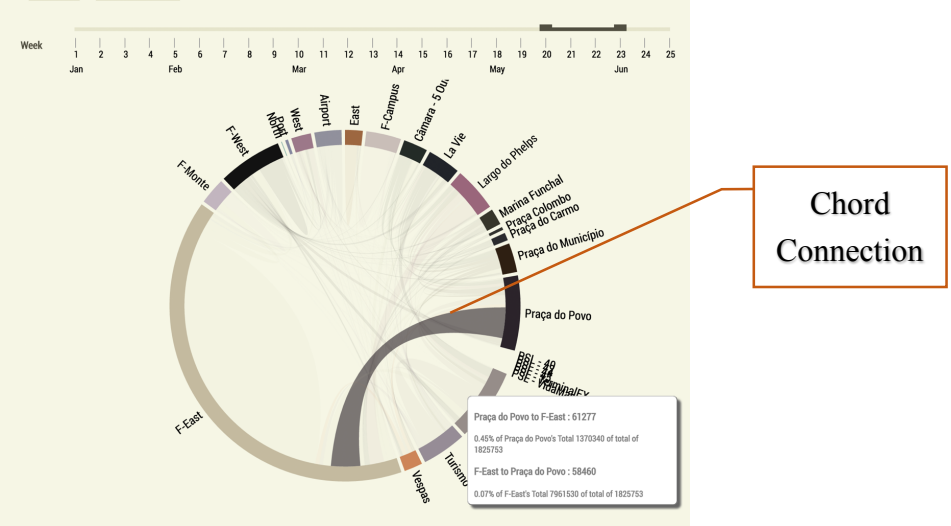

Figure 27 – Chord Diagram between the 20th and 23rd weeks of 2016

Interactions

- Hover on a chord line, displays information about the number of leaps between the origin and destination places or regions.
- Clicking on a region, converts that region into the routers belonging to it.
- Clicking on a router, converts it back to the region it belongs to.

Services used

- get-connectivity – to get the number of leaps between each place for the given weeks

(Left in blank intentionally)

5 Results

In this chapter we discuss the deployment of the project, and cover some general statistics about the system and the collected data, presenting an analysis of the collected data per location, and comparing it with the ground truth sources. We will also briefly show more advanced methods of data analysis, and how they could be improved and further implemented in the future.

5.1 Deployment Timeline

The deployment was done progressively in Madeira island starting in Nov 27 of 2015, aiming to capture data in the different parishes and reaching places all around the island. In association with APM, the locations were also chosen based on touristic and local attractions based on the top 20 locations from TripAdvisor[10], view points and places with high density of people. The installations were done indoors in governmental buildings, and private establishments (such as restaurants and cafes and placed in strategic locations aiming to capture the outside devices). This strategy was used to take advantage of relying on the existing internet connections.

With the goal of monitoring the touristic movements, it was also decided to install routers on the entrance points of the island, such as the port and airport. Following, we present an extensive list of the chosen locations in a deployment timeline:

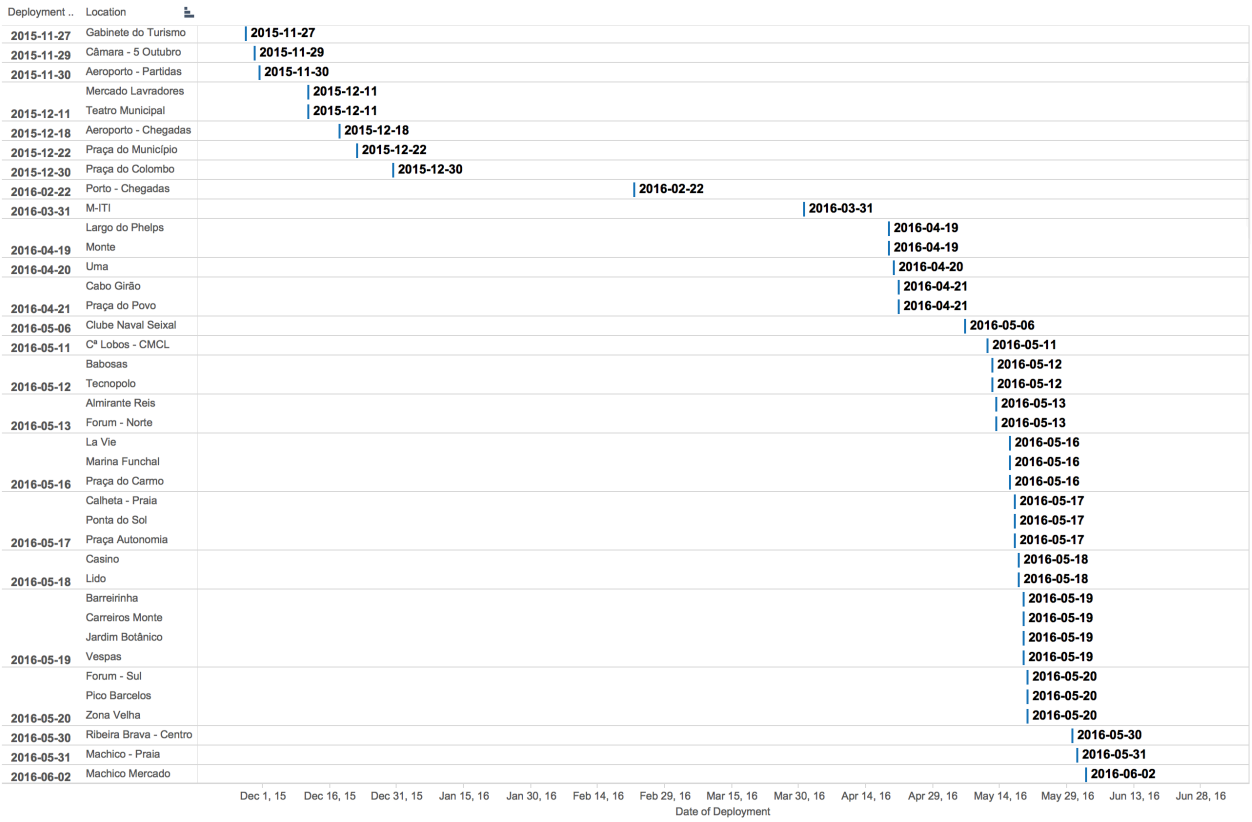

Figure 28 - Development timeline

[10] https://www.tripadvisor.com/

5.2 General statistics

The following results, are representative of 36 locations during the dates between 2016-05-20 to 2016-06-20. Because some of the routers (last 3) from figure 28 were deployed close to the time of writing of this report, and don't have yet a full month of data, we decided to leave them out of the statistics.

Over the course of the whole deployment, there were detected:

- 22 079 983 probe requests
- 1 738 923 unique mac addresses
- 353 798 unique SSIDs
- 4 538 877 leaps

In a database that occupied 7.2 GB of data, and 10.7 GB of indexes.

The daily average count of probe requests, with 36 routers communicating analyzed between 2016-05-20 and 2016-06-20 was of 359 766, averaging roughly 10 000 per router.

5.2.1 Routers per device

In the chart below (figure 29), is depicted the number of devices (vertical axis) and the number of routers where they appeared (horizontal axis). The large majority of devices only appear in a single location, and that information is better clarified in the last column of table 7 displaying their locations. This is important to understand the number of places where most of the devices appear, and see the distribution of how many leaps are done by each device.

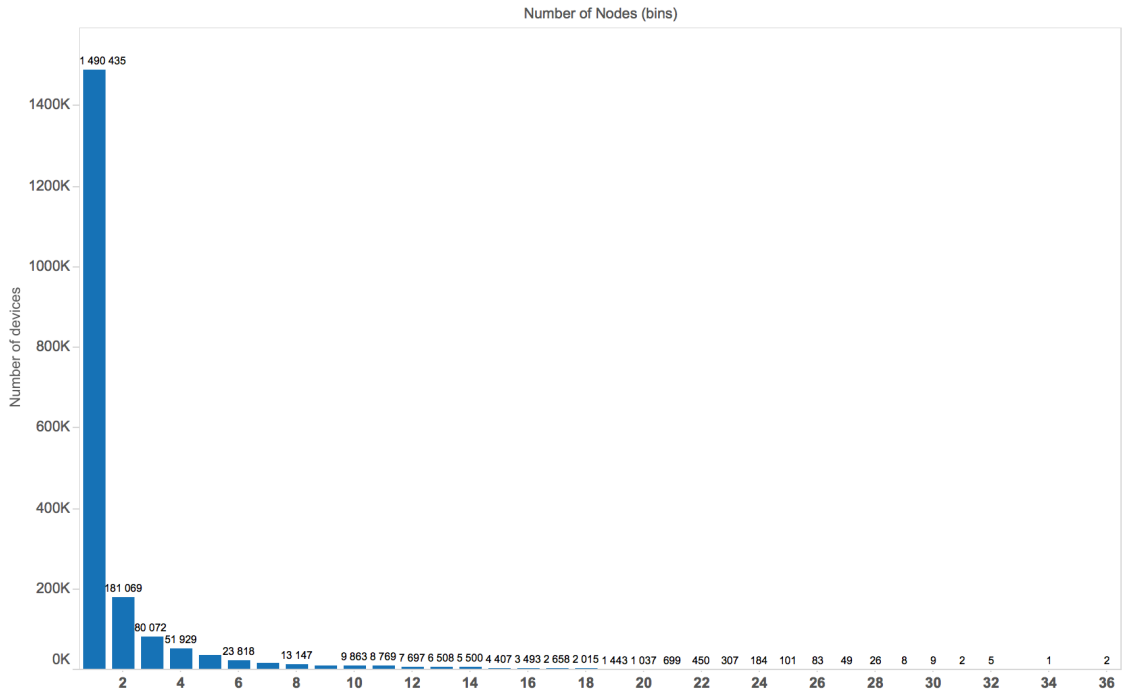

Figure 29 - Number or routers per device

5.2.2 Location Statistics

On table 7, we present the statistics and counts for each router and present a brief analysis of these results. The data relative to the daily averages, was collected between 2016-05-20 and 2016-06-20.

Table 7 – Location statistics

Location	Daily Average count	Daily Average Visitors	Daily Average Inbound Flow	Daily Average Outbound flow	Devices first appearance	Devices last appearance	Only appearance
Aeroporto Chegadas	4797	981	803	1705	130795	18628	170649
Aeroporto Partidas	1483	571	836	322	26483	117447	77120
Almirante Reis	4885	1365	3282	3358	65369	7063	71851
Babosas	580	193	334	333	12939	1510	13268
Barreirinha	1202	192	524	474	23469	1143	22804
Cabo Girão	986	384	265	269	13282	856	12528
Calheta - Praia	1389	323	408	345	42084	1919	41575
Câmara - 5 Outubro	1985	188	1969	2064	11828	6512	15333
Carreiros Monte	803	226	394	378	15332	859	14693
Casino	1535	239	1160	1113	25197	1495	24774
Clube Naval Seixal	394	63	67	67	7389	335	8139
Cª Lobos - CMCL	1112	184	502	493	11492	1390	12312
Forum - Norte	1307	92	813	808	14478	1237	16133
Forum - Sul	3007	516	1865	1685	42104	3290	39266
Jardim Botânico	675	276	379	374	13211	868	12426
La Vie	4935	484	3034	2921	77818	4761	79420
Largo do Phelps	4418	713	3602	3588	49380	4892	52609
Lido	2332	651	1147	1074	36057	3391	33309
M-ITI	209	7	148	154	1367	249	1718
Machico - Praia	1227	250	688	648	14479	1243	13207
Machico Mercado	2965	568	1034	897	33167	3207	30509
Marina Funchal	1019	234	1309	1294	5551	1507	5328
Mercado	5597	1176	4924	5008	60984	23276	77414
Monte	2046	566	816	827	52472	3071	58412
Pico Barcelos	643	151	501	468	6932	710	6230
Ponta do Sol	751	163	240	223	15633	921	15427
Porto - Chegadas	354	17	205	240	10176	9706	28442
Praça Autonomia	6180	1274	7900	7820	82554	10071	73206
Praça Colombo	285	66	220	251	2549	4586	5542
Praça do Carmo	1055	69	560	565	15718	674	17729
Praça do Município	3151	504	2335	2513	34697	13940	44732
Praça do Povo	5685	1112	5769	5802	68537	8253	68514
PSL - VidaMar	2456	672	382	314	585	0	0
Ribeira Brava - Centro	634	206	228	194	5740	725	5097
Teatro	7169	1387	6340	6514	94755	40056	111956
Tecnopolo	340	15	539	538	2326	364	2418
Turismo	3686	855	3608	3717	52248	26643	65736
UMa	1119	42	895	915	9602	1136	11116
Vespas	4148	847	2452	2333	65007	4864	61692
Zona Velha	5012	1080	7113	6980	46468	7232	39046
Total	**93556**	**18902**	**69590**	**69586**	**1300254**	**340030**	**1461680**
Average	**2339**	**473**	**1740**	**1740**	**32506**	**8501**	**36542**
Std deviation	**1952**	**411**	**2065**	**2076**	**29972**	**19523**	**35515**

The router that registered the highest amount of devices were the Teatro, Praça da Autonomia, Praça do Povo, Mercado, Largo do Phelps and La Vie, which coincide with the busiest areas of the island. Whereas the routers that captured the least amount of unique devices were the ones in M-ITI, Clube naval do Seixal and Praça do Colombo.

As expected, the routers where more devices first and last appeared in the system, were the airport arrivals and the airport departures respectively.

Many devices, also only register a single appearance, and the most common locations where that to happened were the Airport, Almirante Reis, Mercado and the Teatro.

5.2.3 SSID per mac address

In the following chart (figure 30), we represent the distribution of the number of SSIDs that each device broadcasts. A significant amount of devices only broadcasts a single SSID, and over 90% of the devices can be found bellow the 3 SSIDs mark. This result can be unexpected at first glance, as most people should have more than one SSID stored in their devices, but since the requests implementation are left to the vendors to decide, not all of the stored SSIDs are broadcasted. It is relevant to the understanding of the SSID analysis, and to understand the distribution among the overall devices.

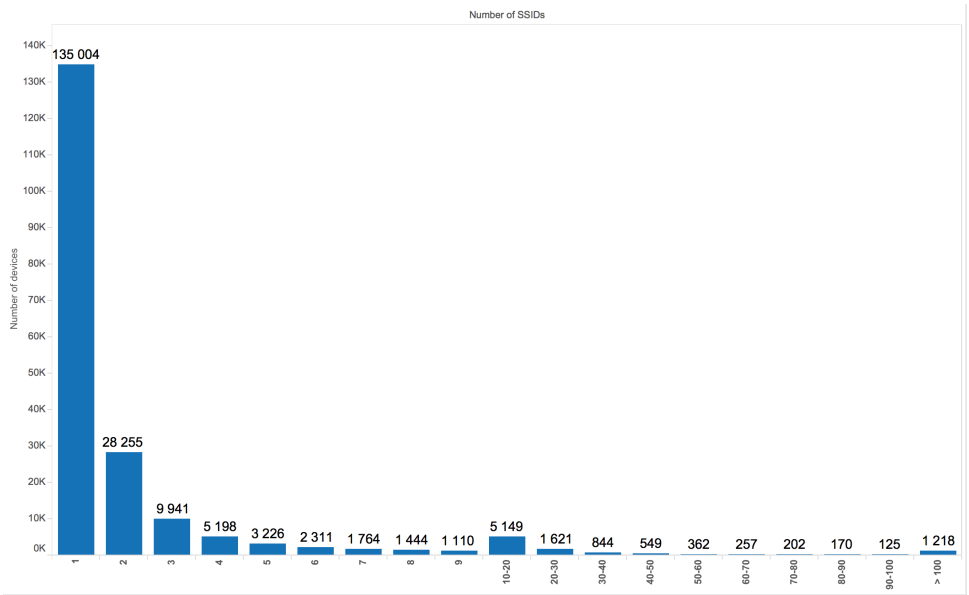

Figure 30 - SSID per MAC address

5.2.4 MAC address per SSID

In the following chart (figure 31), we display the SSIDs popularity (SSIDs shared among many devices). One of the most common SSIDs is _VINCI Airports Wifi, which is the network present in the Portuguese airports, this being common among many devices that are captured at the airport, which also is one of the locations with high count of devices.

The first line represents the number of devices that send blank SSIDs, meaning that the blank/empty SSID is shared among many devices.

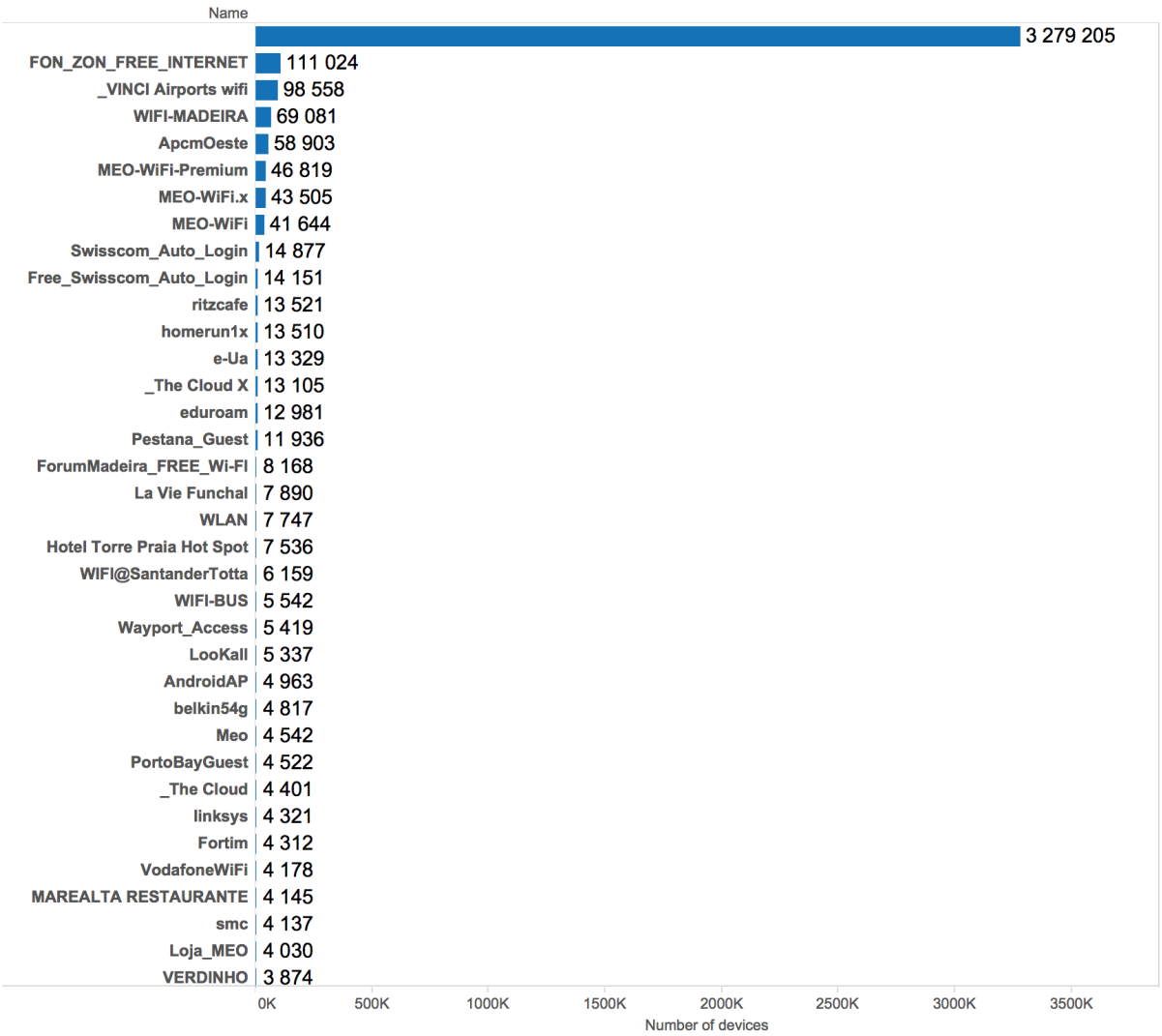

Figure 31 - MAC address per SSID

5.2.5 MAC address per vendor

After conducting the vendor discovery for each MAC address, we found that the most common brands are Samsung and Apple, but to note that the large majority of the devices wasn't found in the API used to retrieve this information as is represented in figure 30 as "No vendor". This is further explained in section 6.2 (Future work) where we analyze the effect of MAC address randomization.

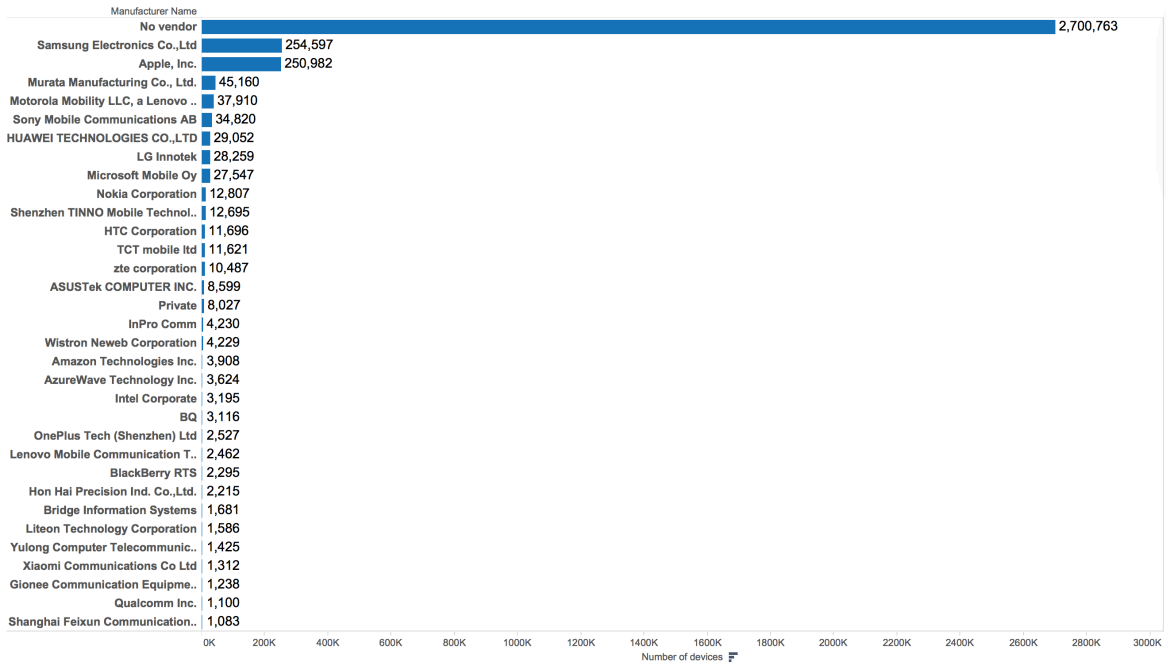

Figure 32 - MAC address per vendor

This information can be used to relate each place with the most popular brands of devices present, and understand the predominant types of devices across different places.

5.3 Common Paths

From the leap records, we were able to identify usual paths taken by the citizens, by filtering the moves that occur in the timespan of one day, and storing the number of times that a path was taken. The following diagram (figure 31) and table (table 8) represents the most usual paths taken and represented in a graph based diagram, where the numbers represent the order of the path. The following data is a sample from 10 days between 2016-06-10 and 2016-06-20.

Figure 33 –Example of common paths representation

Table 8 - Most common paths

Nr of repetitions	Node 1	Node 2	Node 3
252	Teatro	Turismo	Teatro
197	Turismo	Teatro	Turismo
234	Machico - Praia	Machico Mercado	Machico - Praia
88	Turismo	Teatro	La Vie
75	Machico Mercado	Machico - Praia	Machico Mercado
97	Machico - Praia	Machico Mercado	Machico - Praia
43	Teatro	Turismo	La Vie
43	Praça Autonomia	Zona Velha	Mercado
42	Zona Velha	Praça Autonomia	Praça do Povo
41	Praça do Povo	Praça Autonomia	Zona Velha
41	La Vie	Teatro	La Vie
39	Mercado	Zona Velha	Praça Autonomia
68	Tecnopolo	UMa	Tecnopolo
33	UMa	Tecnopolo	UMa
63	Mercado	Zona Velha	Mercado
33	Teatro	Turismo	Praça do Povo
26	Largo do Phelps	Mercado	Zona Velha
26	Turismo	Teatro	Vespas
26	Praça Autonomia	Praça do Povo	Vespas
26	Zona Velha	Praça Autonomia	Zona Velha
25	Praça do Povo	Praça Autonomia	Mercado
24	Teatro	La Vie	Teatro
23	Tecnopolo	UMa	Tecnopolo
23	Teatro	Turismo	Largo do Phelps
21	Largo do Phelps	Turismo	Teatro
21	La Vie	Teatro	Turismo
20	Teatro	Turismo	Praça do Município

5.4 Ground Truth

We propose a way of confirming the data, by comparing the records of people at the entrance points of the island, namely the port and airport. In collaboration with the airport entities, we were given access to the records of passenger counts and compare our results with these counts that represent the real number of passerby's in a confined space.

5.4.1 Daily Counts

The ground truth information was done with the airport with the collaboration of ANA, we were provided with the number of passengers from each arrival and departure for the first 10 days of the first five months of 2016 (plus the last five days from December 2015) in a total of 55 days. With the data aggregated by day, we then compared the beanstalk daily counts with the ground truth.

The following results show the relationship between the Beanstalk vs ANA daily counts for the whole ground truth data.

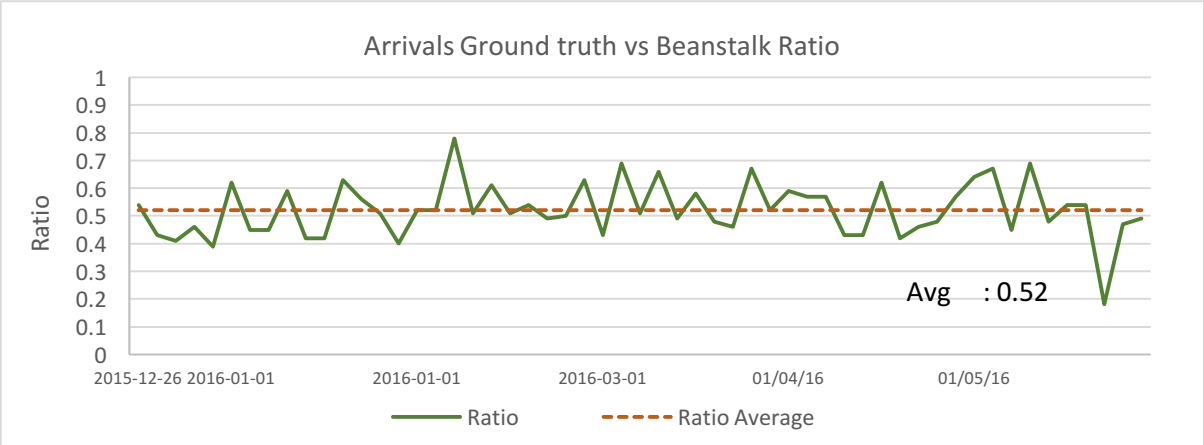

Chart 1 – Airport arrivals ground truth vs beanstalk - Ratio

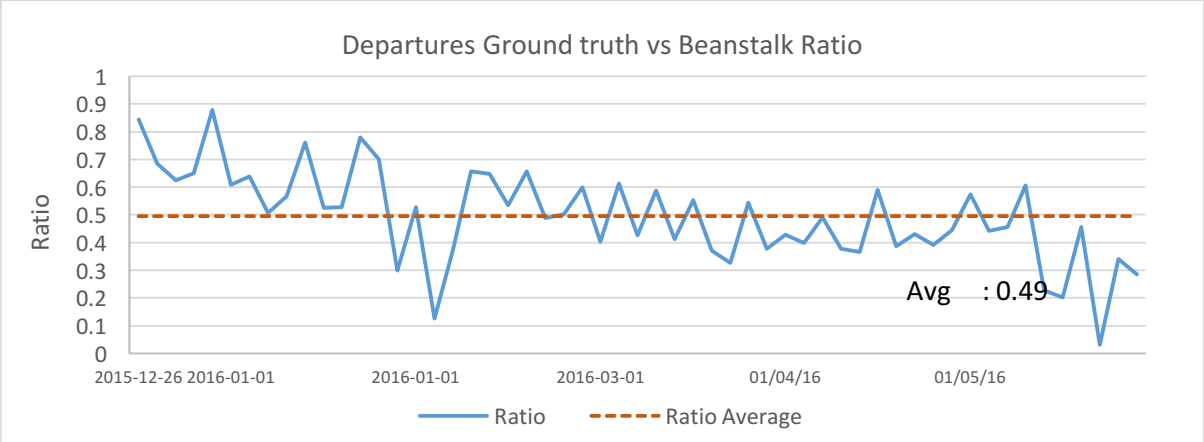

Chart 2 - Airport departures ground truth vs beanstalk - Ratio

This shows an average of 52% of passengers detected, with a standard deviation of 10% for arrivals and an average of 49% of devices detected, with a standard deviation of 17% for

departures. This difference can be explained by three main factors: the amount of population who do indeed use Wi-Fi, and the type of visitors that arrive to the island by plane (including the age and the development status of the source country). Another major factor is the known public warning to use the *airplane mode* while being in a plane. This can lead to people enabling this mode some time before entering the plane or some time after, leading to this discrepancy in data.

Next, we show the correlation between the ground truth and the Beanstalk tracker, displayed in a scatter plot, allowing for the estimation of the regression function. This was done for the daily counts of both arrivals and departures for the first ten days of April 2016.

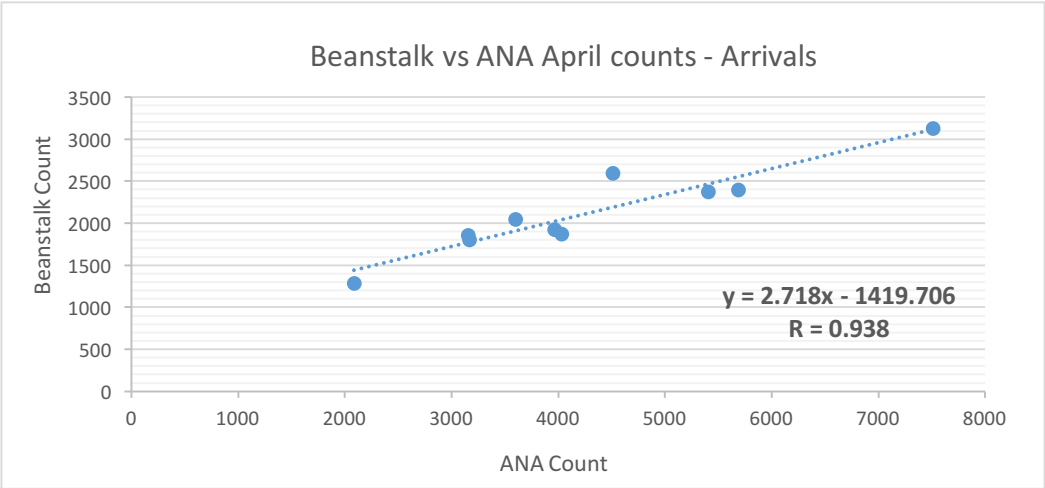

Chart 3 – Beanstalk Arrivals

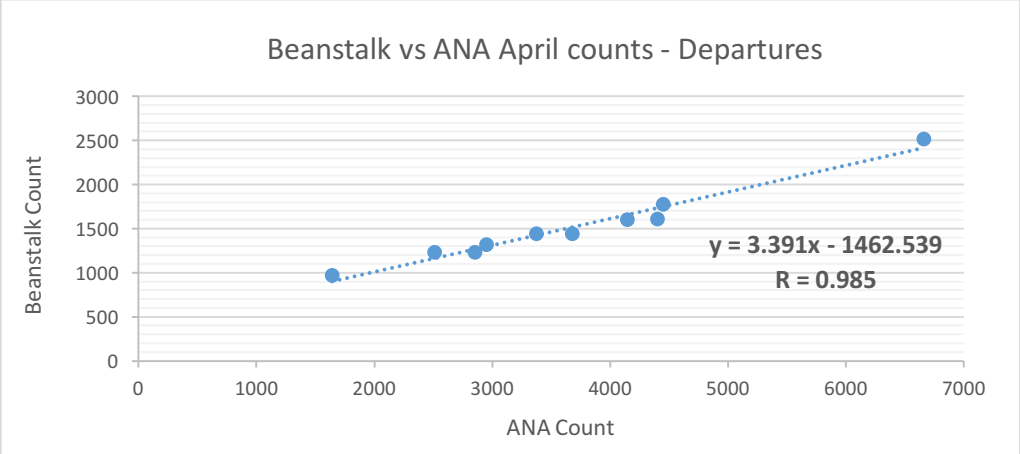

Chart 4 - Beanstalk vs ANA April counts - Departures

Based on this analysis, a linear regression was derived, and from that, we attempted to estimate the real number of passengers for the whole period of ground truth data, from the Beanstalk counts. The following charts display those results with an estimation based on the total data, and to prevent a function shaped to the data, we also used an estimation based only on data from April alone to predict values for the total period.

A Persons's correlation was performed between the sets of the airport passenger counts and the Beanstalk counts. The two variables were strongly correlated, $r(10) = 0.938$, $p < .05$, for the arrivals, and , $r(10) = 0.985$, $p < .01$ for the departures.

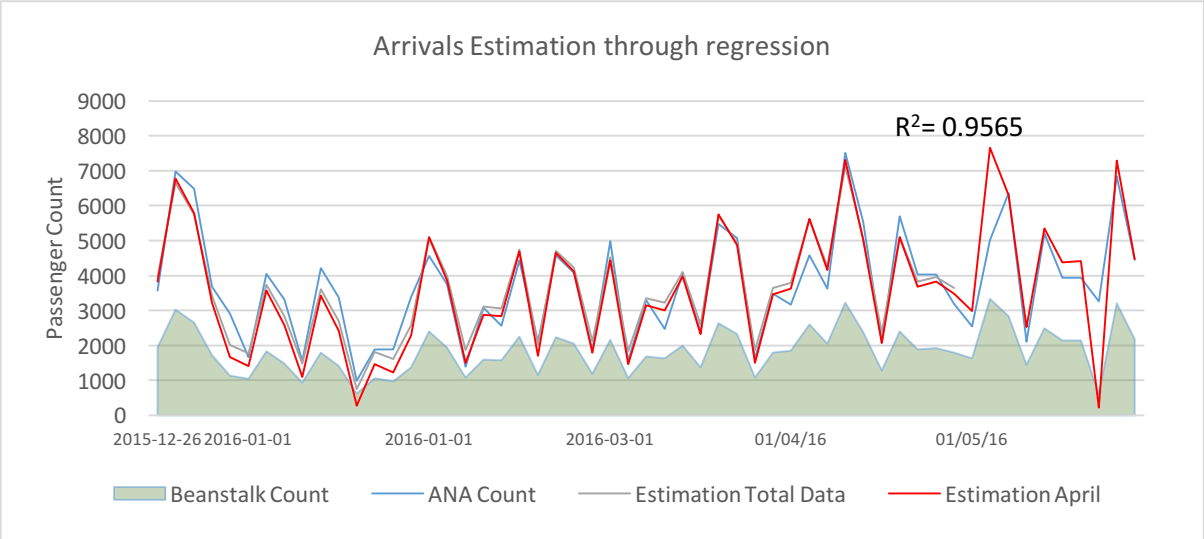

Chart 5 - Comparizon between estimated and real values for the airport arrivals

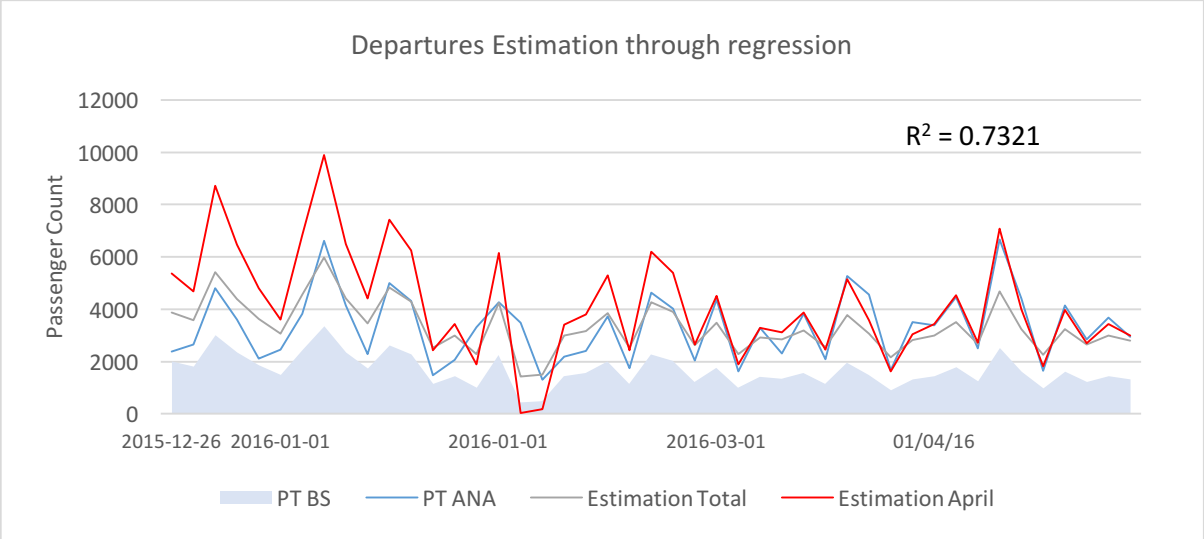

Chart 6 - Comparizon between estimated and real values for the airport departures

The arrivals achieved very consistent results throughout the whole period of collected data for both total estimations and April. On the departures however, while the estimation based on total is consistent, there is a major discrepancy for the data based on April alone, which mostly noticeable in the first 2 and a half months of data. This discrepancy could be explained by the abnormal amount of passengers occurring in the Christmas holidays.

The Pearson's correlation between the arrivals and the estimated values was calculated, with the following result $r(55) = 0.9565$, $p < .05$; and for the departures: $r(55) = 0.7321$, $p < .05$.

5.4.2 Events

The **hourly event** results were compared against the flight information from the airport scrapping, and tested with different thresholds for the algorithm with a 5 minutes sliding window. The following results are based on the inspections performed to the data, where every time a real event occurred, a visual peak in the stats was captured by the system and registered within a 15 minutes difference from the real event. (later for the arrivals, and earlier for the departures). The results following refer to the first ten days (except first month with only 6 days) of the first five months of 2016 and the days between 2016-05-01 and 2016-06-20, in a total of 96 days:

Table 9 - Event Detection Algorithm effectiveness

Place	Ground Truth Events	Threshold	Valid Events	Total Events	*TPR	**FPR
Arrivals	**3116**	2.14	3523	3973	1.13	0.28
		2.15	3267	3658	1.05	0.17
		2.16	3165	3589	1.02	0.15
		2.17	**3007**	**3472**	**0.97**	**0.11**
		2.18	2777	3211	0.89	0.03
		2.19	2646	3114	0.85	0.00
Departures	**3032**	2.11	3242	3203	1.04	0.03
		2.12	**3070**	**3016**	**0.99**	**0.03**
		2.13	2958	2988	0.95	0.04
		2.14	2819	2828	0.90	0.09
		2.15	2599	2493	0.83	0.20
		2.16	2499	2464	0.80	0.21

*TPR – True positive rate
**FPR – False positive rate

The best threshold for the arrivals was 2.17, based on achieving the higher TPR with the lowest FPR. But for the departures, that threshold changes to 2.12, concluding that the best threshold values vary depending on the location analyzed.

Note: When the same ground truth event is detected more than once by the system, it results in a number of valid events bigger than the ground truth events and an arithmetic TRP > 1.

For the **daily events**, although the algorithm works visually, as depicted in figure 25 (page 38), the ground truth official events were too broad, and when the Beanstalk results are compared directly to the number of events, the FPR rates become high, due to registered amounts of people that happened, but do not link to an official event, and some official events, that do not have enough impact to differentiate from an ordinary day on the captured devices. Though for the visual analysis performed, the most accurate *p (*percentage of standard deviation*)* is 1.15.

5.5 Advanced techniques

Other techniques can be applied on the data to extract more refined information other than counts and connections. Although these techniques are outside the project's original scope, they were partially implemented, and following, we present possible explorations of the data:

5.5.1 Linking devices in the wild

This technique [11] aims to infer social connections between the captured devices and thus inferring social links between people. This is done using the broadcasted SSIDs, and analyzing their rarity. With the assignment of a rarity factor to each SSID, based on their popularity among the devices, it is then possible to calculate similarity using algorithms (such as Adamic and IDF-similarity) that weigh the SSIDs shared between pairs of devices. The less common the SSID, the bigger the importance it will have in the similarity index.

The rarity factors were calculated for our data, but due to the large amount of MAC addresses and SSIDs, the results of such a comparison would return many millions of data (even on a smaller sample), since it compares all the mac addresses against each other, additionally the lack of data to back up the claims of the detected connections were the reasons why this method was not fully implemented.

5.5.2 SSID profiling

The SSID profiling [11] is a technique that consists in inferring previous places of where a person may have been, based on the location of the SSIDs broadcasted by the devices. Each SSID is first geotagged using for instance an API from wigle.net, then for each pair of coordinates, a street address is gathered from google maps API, and finally for every mac address, an estimation of possible countries of origin is done, by either selecting the country that has more SSIDs, or using other techniques that include assigning weights to the SSIDs, making some more important than others. This can be useful to differentiate tourists from locals, and provide a demographic idea of the types of people that frequent each place.

In the partial implementation of this feature, some problems were encountered related to the validness of the results and the following limitations also impose further challenges:

- Only a sample of the SSIDs stored in the devices are broadcasted and recent operating systems are progressively refraining from sending the network names on non hidden networks [20].
- An SSID can be shared among many access points of different networks, making the geotagging process unreliable
- Each SSID is only broadcasted in the radio channel stored previously for that network
- Limited wigle.net API, which has a 500 requests per day limit, may not find the geolocation of an SSID, since it is based on involuntary crowd source information from the users of their mobile application.

In this chapter, we presented the results and statistics of the collected data about the devices, SSIDs, paths, and the comparison between the collected data of two points with the ground truth of a confined place where we are certain of the number of people passing by it.

Next, we present some conclusions, and future work, also addressing the issue of privacy and how the companies are starting to apply measures to avoid this type of data collection.

(Left in blank intentionally)

6 Conclusions and Future Work

The project was aimed for tracking the mobility of people in Madeira island, and we have built and deployed a system that captures data, analyses it, and presents the results to the end users. The results allowed us to account for the counts of passerby's in a place, accompanied with ground truth to back up that information. We provided a strategy to detect tourists from locals, processed results about the connectivity between different places, and presented an algorithm for detection of events.

The data gathered and the results, show that this system allows to track mobility from people and derive other types of information regarding the devices and infer the demographics of certain locations.

6.1 Technology

The hardware used offered some technical problems regarding support for official operating systems from OpenWRT that had USB support for the installation of additional packages. As of now, the available version doesn't leave free space to install any additional packages for the USB drivers, and no extra functionalities can be added. Due to this we ended up using an unofficial release. Although the option was effective, because it included all the necessary requirements in a single package, a router that has official open source operating systems support should be preferred. Due to the low memory and the need of a USB flash drive, 4 of the 36 flash drives stopped working at random moments in time and required replacement. This unofficial version of OpenWRT also had random problems (which only affected a few devices), where the wireless interfaces would stop capturing, and a monitoring script was added to all of them to automatically restart the network services when not capturing for more than one hour.

The option to use MySQL as a database system, didn't create any opposition to our needs, and having a rich syntax, big amount of functions and event scheduler allowed easy manipulation of the data, although for future reference a database optimized for time series data, such as InfluxDB[11] may be a better option regarding the high amount time based data we captured.

The use of Node.Js for a backend technology revealed to be a good choice regarding its high modularity and the good community support and constant package development that makes development fast and reliable. Although for the data manipulation, other languages (php, python) may be more appropriate to perform sequential operations such as iterating through a table and performing sequential operations that require asynchronous time, where non native mechanisms and strategies need to be adopted due to the asynchronous nature of the language.

As it is, the website has only the necessary controls for manipulating the pages, locations and time intervals to visualize the data. But for future development, where more controls can be

[11] https://influxdata.com/time-series-platform/influxdb/

added, and the information can be more dynamic, the use of a front-end framework (besides jQuery) that natively supports the MVC pattern is encouraged.

6.2 Future Work

This project presents contributions regarding:

1) The capturing hardware and software used in a large scale deployment on an island and touristic environment;
2) The data analysis techniques, the processing and the integration with external APIs;
3) The event detection for data with different time intervals;
4) The confirmation of the data with reliable ground truth.

With this work and further results, we plan on writing two scientifically papers, leaving room for future work to be developed exploring more techniques of path finding or applications of graph theory, exploring other types of environmental sensors to enrich this information and gathering more demographic information through with other methods. The techniques for the SSID profiling and linking can be also explored.

This system can be deployed in other cities with more routers, and other strategies (such as big data) could be deployed depending on place geography, city planning, and the main data to be extracted. For a larger city, the system would need a high performance, and possibly a database the use of a database engine optimized for high amounts of writing operations to constantly store the data. Depending on the amount of routers, the use of a dispatcher to receive the requests can become a necessity.

Fast forwarding 5 years in time, the system as it is, would have expectedly captured over 220 million probe requests, and the database size would expectedly be over 150 GB of data. To cope with these amounts of data, strategies would have to be deployed to ensure the system performance, such as storing older raw data in separate databases/machines, and using the summary tables as reference to access the processed data summaries. For raw data processing, hardware upgrades would need to take place, especially in memory, to handle the large amounts of data processing and results from queries.

6.3 Privacy

This type of work raises privacy issues since it collects involuntary information about devices, that can be used to infer information about their users using the mac addresses and the SSIDs broadcasted [11]. Although the data collected is publicly directed at all listening devices, unencrypted, and no strategies of packet injection are used to induce the sending of these packages, the data collected raises still moral issues regarding the privacy of the users, where it often reveals the last Wi-Fi spots where the device was connected.

After our deployment, a paper [20] was published explaining the tactics used by modern day systems that try to change this information from being publicly broadcasted. The latest

operating systems from Apple, Android, Microsoft and Linux use a method called MAC randomization. The companies use this method with the goal of hiding the real MAC addresses of the devices, hindering analytics based on this approach. This is done to protect the identity of the users since the launch of iOS8, Android 6, and Windows 10. The following conclusions have been published on that paper[20]:

Table 10 - MAC address randomization through operating systems

Operating System	Enabled since version*	Conditions
iOS	8	Screen is off; (always since iOS9)
Android	6.0	Background scans
Windows[12]	10	Always
Linux	Kernel 3.18	-

All these operating systems share a common condition being that randomization is only done, if the hardware and drivers support it. It is important to mention that although these methods may have influenced our results, we still achieved a remarkable correlation with the ground truth data, estimation function, and event detections, possibly due to the prevalence of devices that are not running on the latest operating systems. The only results where this may be visualized, would be the number of mac addresses only detected in the system for a single day (figures 3 and 4, p18) and the number of mac addresses that could not be associated with a vendor.

These operating systems' features impose new challenges to be dealt with, regarding the methods for analyzing the captured information which affects the methods used for leaps, and unique device counts. Although these measures aim to prevent the passive tracking and involuntary information capturing, this paper [20] shows that the effectiveness of the randomization only covers part of the security issues, and other elements can still be explored such as Wi-Fi Protected Setup (WPS), which mainly uses a unique identifier. It was also reinforced that the current operating systems tend to broadcast less SSIDs, and that that tactic will be more popular in the future where the only SSIDs broadcasted will be ones from hidden networks. This is done because the access points require this information to be broadcasted, in order to respond to the probe requests, thus still leaving an opportunity to analyze SSID information. It further proposes algorithms to analyze the randomized mac addresses, grouping them clusters, taking into count the broadcasting interval and the analysis of various devices. They concluded that particular devices are linked to patterns in the randomizing, meaning that the process is not completely random.

[12] Although the MAC addresses are randomized, the same one is always used to connect to the same networks. This is achieved with a SHA-256 hash of the SSID, MAC address, internal connection id, and a random secret for each network interface. It assures that systems relying on fixed MAC addresses continue to work as expected, e.g., when authentication is performed based on the MAC address.

(Left in blank intentionally)

7 Acknowledgements

I would like to thank my advisor, Nuno Nunes for all the support and the ideas to analyze this type of data in such diverse ways.

This project, staff and hardware was funded by the Regional Project M1420-99-9999-FEDER-000035 in the MADEIRA14-20 FEDER scope.

A thanks to my project colleagues who helped in the deployment of some routers around the island and the establishment owners that allowed us to use their facilities and resources.

We would like to thank to APM (Madeira Promotion Association), to collaborate in this project and facilitate the installation of the trackers in governmental, touristic buildings and the island entry points such as the port and airport.

We would also like to thank to the airport management, Eduardo Teles for facilitating the delivery of the ground truth from the flights and passenger counts, which were essential to validate out data.

A thanks to the admins of wiggle.net, that agreed to change their API daily limitations to provide the coordinates for the detected SSIDs, and to the admins of MacVendors.co that also facilitated the use of their API for high amounts of data retrieval referring to the mac addresses vendor information.

Glossary

AES – Advanced Encryption Standard

AJAX – Asynchronous JavaScript and XML

AP – Access Point

JSON – JavaScript Object Notation

VPN – Virtual Private Network

References

[1] M. Dixon, S. Aiello, F. Fapohunda, and W. Goldstein, "Detecting Mobility Patterns in Mobile Phone Data from the Ivory Coast," *Bus. Anal. Inf. Syst.*, Jan. 2013.

[2] Kaufman, L. and Rousseeuw, P.J. (1987), Clustering by means of Medoids, in Statistical Data Analysis Based on the L_1–Norm and Related Methods, edited by Y. Dodge, North-Holland, 405–416.

[3] A. S. Martinez Sala, R. Guzman Quiros, and E. Egea Lopez, "Using neural networks and Active RFID for indoor location services," in *2010 European Workshop on Smart Objects: Systems, Technologies and Applications (RFID Sys Tech)*, 2010, pp. 1–9.

[4] Timothy Sohn, Alex Varshavsky, Anthony LaMarca, Mike Y. Chen, Tanzeem Choudhury, Ian Smith, Sunny Consolvo, Jeffrey Hightower, William G. Griswold, and Eyal de Lara. 2006. Mobility detection using everyday GSM traces. In Proceedings of the 8th international conference on Ubiquitous Computing, Berlin, Heidelberg, 212-224.

[5] A. Baniukevic, C. S. Jensen, and H. Lu, "Hybrid Indoor Positioning with Wi-Fi and Bluetooth: Architecture and Performance," in *2013 IEEE 14th International Conference on Mobile Data Management (MDM)*, 2013, vol. 1, pp. 207–216.

[6] A. K. Dey, K. Wac, D. Ferreira, K. Tassini, J.-H. Hong, and J. Ramos, "Getting Closer: An Empirical Investigation of the Proximity of User to Their Smart Phones," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, New York, NY, USA, 2011, pp. 163–172.

[7] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is There WiFi Yet? How Aggressive WiFi Probe Requests Deteriorate Energy and Throughput," *ArXiv150201222 Cs*, Feb. 2015.

[8] J. Little and B. O'Brien, "A Technical Review of Cisco's Wi Fi-Based Location Analytics White Paper," *Cisco, 2014*.

[9] B. Bonné, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: Involuntary tracking of visitors at mass events," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, 2013, pp. 1–6.

[10] F. Meneses and A. Moreira, "Large scale movement analysis from WiFi based location data," in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2012, pp. 1–9.

[11] M. Cunche, M.-A. Kaafar, and R. Boreli, "Linking wireless devices using information contained in Wi-Fi probe requests," *Pervasive Mob. Comput.*, vol. 11, pp. 56–69, Apr. 2014.

[12] Jaccard, Paul (1901), "Étude comparative de la distribution florale dans une portion des Alpes et des Jura", Bulletin de la Société Vaudoise des Sciences Naturelles 37: 547–579.

[13] Lada A. Adamic and Eytan Adar. Friends and Neighbors on the Web. SOCIAL NETWORKS, 25:211–230, 2001.

[14] J. Ramos, "Using TF-IDF to determine word relevance in document queries," *ResearchGate*, Jan. 2003.

[15] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.*, vol. 24, no. 4, pp. 35–42, 2001.

[16] M. Gast, *802.11 Wireless Networks:The Definitive Guide 2nd Edition*. O'Reilly Media, Inc., 2005, p. 206.

[17] P. Roshan and J. Leary, *802.11 Wireless LAN Fundamentals*. 2010. p 179.

[18] Luo, D., Norford, L. K. & Shaw, S. R., 2002. High Performance Commercial Building Systems Monitoring VAC Equipment Electrical Loads from a Centralized United Technologies Corporation. ASHRAE Transactions, 108(1), p.841-857..

[19] P. Fuxjager, D. Valerio, and F. Ricciato, "The myth of non-overlapping channels: interference measurements in IEEE 802.11," in *Fourth Annual Conference on Wireless on Demand Network Systems and Services, 2007. WONS '07*, 2007, pp. 1–8.

[20] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, New York, NY, USA, 2016, pp. 413–424.