UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Information and Communications Engineering

Final Dissertation

Estimating the number of people based on Wi-Fi probe request frames

Supervisors

Fabrizio Granelli

Daniele Miorandi

Student

Samuel Bortolin

Academic year 2019/2020

Acknowledgments

... thanks to my family, my girlfriend, my supervisors and all the U-Hopper team ...

Contents

Abstract

Sentence that describes the problem ...

The abstract is a short summary of the work describing the target, the subject of the thesis, the methodology and the techniques, the data collection and elaboration, the explanation of the reached results and the conclusion. The abstract of the dissertation must have a maximum length of 3 pages and must include the following information:

- context and motivation
- short summary of the main problem you have dealt with
- developed and /or used techniques
- reached results, the personal contribution of the student has to be highlighted

1 Introduction

Brief introduction of the internship and the project done at U-Hopper.[1]

1.1 Problem statement

In managing a company that provides services to physical customers, the most important aspect is how to manage the flow of customers to guarantee them optimal service. Overcrowding and long waiting times are serious problems caused by poor demand management. These problems spoil the service experience of the users and especially during this pandemic period due to COVID-19, it is important to avoid generating crowds and queues to avoid new contagions. The repetition of these events leads to the loss of customers and therefore to the loss of revenue for the company. This fact is increasingly relevant in this period of crisis with very low liquidity.

Offering an efficient and fast service is the key to increasing the number of customers in their own business. The solution is not to use all the available resources to try to meet these requirements, as excessive use of these leads to an increase in costs without leading to an increase in revenues. Of course, in order to calibrate the correct amount of service to offer, it would be necessary to know the demand. In particular it is necessary to know every time how many people require the service. Having this information is not trivial as it is highly variable over time and depends on several factors.

The purpose of this thesis is to create a system capable of providing useful information to a company's organizational departments to manage its resources more effectively and efficiently. In particular, the proposed system will provide estimates of the number of customers in real-time through the use of machine learning techniques. From this information, it will be possible to understand how the demand is distributed over time and what the peak hours are. This will clarify in which time slots it is necessary to increase the capacity of the service in order to be able to provide it to a greater number of people. In a dual way, this information is also useful for understanding when there is less demand and it is possible to reduce the capacity of the service, with the aim of saving resources.

The idea can also be extended to provide the same crowding data to its customers. With this data, they can plan better when to use the service offered, avoiding being in crowded situations or situations that require long waiting times.

1.2 Approach to the problem

This is the approach ...

1.3 Outline

In chapter 2 the state of the art is analyzed. In chapter 3 the methodology and the choices in the system design are presented. Chapter 4 presents how the system is implemented. In chapter 5 the proposed system is validated and the results are evaluated. In the end, chapter 6 reports the conclusions and suggests some ideas for future work.

[1]website: `https://www.u-hopper.com/`

2 State of the Art

In this chapter, the current state of the art is analyzed in the context of counting people in a particular place of interest. The various technologies presented in the literature for this purpose are discussed. Furthermore, the reasons why counting people through the sniffing of the Wi-Fi probe request frames is the solution with the best trade-offs are explained. In addition, several implementations and some use cases are presented. At the end of this chapter, it will be clear the choice to develop this type of system with the U-Hopper team to find a solution to estimate the number of people. Unlike other solutions, our system achieves high accuracy, has a low-cost implementation, allows the transmission of data in real-time and ensures user privacy.

2.1 People counting methods

Let us start by thinking about before the popularity of mobile devices and the development of all these new communication technologies that are making this world increasingly interconnected. Therefore, before it becomes possible to exploit the signals of people's devices to identify and count them.

In 2010, Pinna et al. compared some of the technologies of the time (e.g. infrared sensors, treadle switch-based systems, weigh in motion systems using load cells) to avoid the manual collection of the occupation information and promote automatic counting [18]. All of these methods presented in the paper have good accuracy (95% in optimal conditions of use for treadle mats and approximately 90% for the infrared sensors, 97% for load cells) but have high costs and need the installation and the maintenance of the sensors.

Other studies have been carried out to improve the performance of infrared sensor systems. Jia and Zhang described a system with subordinate nodes for collecting information from pyroelectric infrared sensors and a master node that when it receives information from a sensor has to determine whether a person is entering or leaving [9]. Such systems require sensors for each entry point, in the case of overcrowding or continuous flows of people, some sensors may not work properly and can lead to a wrong estimation. In these solutions, errors accumulate over time.

Mikkelsen et al. compared a a light sensor estimator based method with a Wi-Fi-based method to estimate the number of people present [13]. This simple Wi-Fi implementation uses a probability distribution of the number of Wi-Fi enabled devices that a person is carrying. They concluded that the Wi-Fi estimator is better at recovering after errors, while errors in light sensor estimator accumulate over time. These presented methods, in which errors accumulate over time, are not acceptable in a continuous monitoring system and this does not happen with the Wi-Fi solution and other solutions.

Many methods are proposed in the literature for processing images and videos captured by a camera device. Recognition and tracking of people by a stereo vision system to increase or decrease a people counter [2], a separated-aggregated framework based on deep learning to estimate the number of people from still images [28], a deep learning-based method for estimating crowd density and the total number of people in high-density crowd images [27], estimation of the people flows and then summing them to obtain people densities [11], a locate size and count CNN (Convolutional Neural Network) model is proposed to to resolve people in dense crowds [20]. These methods rely on the use of cameras to get the data to analyze and therefore have the problems related to these devices. Images can be noisy in dim light conditions and, due to the presence of obstructions or overlaps, people may not be detectable. Moreover, the implementation of these methods entails high costs due to video camera devices. However, taking photos or videos of unknown people is always a privacy issue as decreed by the GDPR (General Data Protection Regulation) in the rights of the data subject. Therefore, these types of solutions cannot be used to solve our problem.

A not very popular but interesting method of counting the number of people is to analyze the audio of a place with a microphone and count the different audio tones [10]. Afterward, Valle proposed a more sophisticated prediction model to estimate the occupancy of a room which borrows speech recognition tradition and is based on Gaussian mixtures and hidden Markov models [25]. These methods can be inaccurate due to high ambient noise and the limitations on the quality of the microphones must be considered during the analysis. In addition, other sounds from smartphones or radios must be filtered to avoid being counted as people. However, as in the case of video methods, the analysis of the voice of unknown people is always a privacy issue as decreed by the GDPR. Therefore, these types of solutions cannot be used to solve our problem.

As reported by wearesocial.com in the Digital 2020 report[2], 67% of the world population is a mobile phone user. In Europe, Internet users are equal to 84% of the population and 92% of total Internet users is a mobile phone user. In this new era dominated by smartphones, radio-frequency solutions to exploit the signals of these devices to count people are increasing. In recent years Wi-Fi, Bluetooth, Bluetooth Low Energy, LTE (Long Term Evolution) approaches have been developed.

Many researchers have investigated the possibility of estimating the number of people using the two most popular standards for wireless communication, i.e. Wi-Fi and Bluetooth. In 2014 Schauer et al. performed this type of comparison with the aim of creating a pedestrian flow estimation system. They stated that only a fraction of surrounding devices could have been tracked by periodical Bluetooth scans. Therefore Bluetooth-based estimations were less accurate, showing an average correlation to the ground truth of only 0.53 in the best case. In contrast to Bluetooth, Wi-Fi tracking provided a good approximation of crowd densities and pedestrian flows with an average correlation of 0.75 [21].

Bai et al. tried to detect devices with different approaches. The number of the sensed Bluetooth Low Energy devices was about $\frac{1}{3}$ of the sensed Wi-Fi devices. Bluetooth Low Energy data were sparse, and most of the sensed devices only appeared for a few seconds. The number of unique Bluetooth MAC addresses was less than $\frac{1}{10}$ of the unique Wi-Fi MAC addresses, so they gave up the possibility of using Bluetooth Low Energy and Bluetooth data. They concentrated on filtering the Wi-Fi data and obtained a correlation with the ground truth of 0.839 [1]. These papers show us the feasibility of using methods based on Wi-Fi, unlike the methods based on Bluetooth and Bluetooth Low Energy which do not provide satisfactory results in terms of accuracy.

Di Domenico et al. were the first to propose to use LTE signals of opportunity for applications different from location/tracking, i.e. to estimate crowd density within an environment relying on the analysis of variations of the LTE reference signal received power [5]. This approach is affected by the changing of positions that lead to a different superimposition of multipath components and, hence, to a different received power. The same number of people at different times can generate totally different values, thus leading to significant errors during the estimation process. They achieved an average accuracy ranging of 82%, but they tested this system only with a maximum of 5 people. As the number of people increases, the accuracy of the classification decreases. Therefore this approach cannot be used in overcrowded areas.

Shibata and Yamamoto used a sensor node to obtain time-series data of signal strength on a frequency band used for cellular communications. Then they analyzed them using several machine learning techniques to estimate the crowd density around the sensor node installation site [23]. This method did not provide the number of people but only the stages of occupancy, with three stages (Low, Normal and Crowd) the precision is 78% but with five (Low, Little_Normal, Normal, Little_Crowd and Crowd) is only 53%. From this type of information, it is not very clear how many people are in the place and listening to a reserved part of the spectrum is illegal.

Another set of solutions developed exploits the attenuation of signals, which could be made by an IR-UWB (Impulse Radio Ultra Wideband) radar, Bluetooth Low Energy devices or RFID (Radio Frequency Identification) readers with antennas.

Choi et al. presented an approach using an IR-UWB radar which requires a preliminary detection of the clusters of people in the environment to set the parameter values for the algorithm. After collecting the data, through the use of statistical models they found the number of people that have the maximum likelihood from the minimum of 0 people to the maximum of Np people [4]. This type of approach needs to know the maximum number of people and works only in a restricted area depending

[2] `https://wearesocial.com/digital-2020`

on the antenna. In this case with an angle of 80° and a maximum distance of 5 meters, therefore it cannot be used for applications like ours that need to cover a wide space.

Brockmann et al. presented a method to count people in a queue using the attenuation of Bluetooth Low Energy signals [3]. Almost 98% accuracy, but a lot of devices are needed and it can be used only in situations where there is a queue with a predefined path where the sensors are located.

Gupta et al. proposed an algorithm to estimate the number of people that are crossing the RFID installation in both the directions, achieving 90% accuracy by real-time experiments for continuous movement up to 75 persons [6]. For realize a system like this a lot of RFID readers, antennas and tags are needed. An optimum distance is required between the readers, and with the tags. It works only assuming that people do not reverse directions while walking in the passage. However, if the density of the crowd increases beyond an extent, it tends to completely block the reading of the tag and therefore cannot be used in environments where there are large flows of people entering and leaving.

2.2 Wi-Fi probe request frames implementations

From the previous section, it is clear that the Wi-Fi solution is a step further compared to other solutions in the literature. By analyzing the Wi-Fi probe request frames, it is possible to better estimate the number of people in a certain place with lower costs. This method does not accumulate errors over time, does not require a predefined path where users must be and can cover a large area. Moreover, it is possible to anonymize the MAC address to ensure user privacy.

Handte et al. presented one of the first approaches to estimate crowd density by monitoring Wi-Fi probe request frames. They modified the firmware of some existing access points and created a Web service that allows the upload of the latest crowd density measurements [8]. The system was able to continuously detect around 20% of the people on average because in 2014 there were far fewer mobile devices than nowadays (only 49% of the population had a smartphone in Span in 2014) and they didn't have the advanced techniques of nowadays.

Mikkelsen and Madsen presented a system to anonymize the MAC address of the sniffed probe request frames, to sent them to a server and to analyze them putting two thresholds: minimum value of the RSSI (Received Signal Strength Indicator) and minimum detection time [12]. The ratio between the estimated number of devices, obtained by setting the two thresholds, and the number of people is around 50%. They said that the use of machine learning techniques could have ensured greater accuracy of estimates.

Oliveira et al. designed a specific device to monitor the presence of people by analyzing the Wi-Fi probe request frames [15]. Subsequently, they proposed a method for estimating the number of devices with a very strong correlation with the ground truth of the number of people in the environment, with a Pearson,Äôs correlation coefficient of 0.896 [16]. They said that the experiment should be replicated in other scenarios to test the versatility of the method. Moreover, they believed that one of the answers to get estimates closer to the ground truth values may be in the use of machine learning techniques.

Nishide filtered the collected data using a combination of RSSI, packet frequency, and the total time duration which the nearby device is detected. Filtering is performed individually for each parameter, and then the linear regression and correlation coefficients are calculated in different places [14]. He said that there may be other ways to accurately estimate the number of people using machine learning.

These papers show us the possibility of sending data in real-time and that there is a correlation between the devices obtained from the collected data and the presence of people. Furthermore, they suggest that the use of advanced techniques, such as machine learning, could have a positive impact on accuracy in solving the occupancy problem.

Wang et al. employed the Random Forest method to infer occupant counts using the Wi-Fi connection counts data [26]. The method was tested in a real office building with an average occupancy of 22-27 people and a peak occupancy of 48-74 people, the RMSE (Root Mean Square Error) is four people on the test set. For more than 70% of estimations, the errors are within two people counts, and for more than 90% of estimations, the errors are within six people counts. This paper confirms that the use of machine learning techniques has a good impact on estimating the number of people but this method does not have a communication system for the transmission of data and results in real-time.

2.3 Use cases of the Wi-Fi method

The Wi-Fi method has several use cases. A strength of this method is in fact the versatility of use for many application contexts.

Prasertsung and Horanont used the Wi-Fi probe request frames monitoring technique to identify the number of the customers visiting a coffee shop. They showed that the number of customers tends to increase on the average of 30% on a promotion day [19]. This can be used to explore how a promotion can drive customers into stores. Shen et al. proposed a shopping group detection system using Wi-Fi. Experimental results indicated that this method could is capable of detecting over 90% of the groups with an accuracy of 91% [22].

Using real-world data collected in a large social event by a network of passive Wi-Fi sensors Zhou et al. extracted patterns related to crowd behaviors [29]. Singh et al. proposed a first-hand application of Wi-Fi sensors and LSTM (Long Short Term Memory) neural network for crowd forecasting and large-scale public event monitoring [24].

The particular case of people estimation on public transport such as buses is a complex application because, unlike the estimation in a static place such as a shop, it is necessary to consider that the vehicle is in motion. Therefore, the presence of people waiting at the stops, people in the traffic, pedestrians and other things should be considered during the data cleaning phase. The esteem of people in public transport is a really interesting challenge that is dealt with in literature. This can provide some useful information to public transport managers, from which they can better manage the routes and reorganize the distribution of their vehicles.

Handte et al. presented a navigation system for bus passengers that has the ability to seamlessly interconnect bus passengers with the real-world public bus infrastructure. Using the occupancy classes to classify the prediction (low, medium, and high occupancy), they got an exact match accuracy of 61.9% [7]. This work provides an indication of the feasibility of real-time information but accuracy can certainly be improved with the use of machine learning techniques.

Oransirikul and Takada presented a method to predict the number of passengers at the bus stop by capturing Wi-Fi activity. They used a polynomial regression method with six independent variables with a degree of 2 [17]. It works well with an average MAE (Mean Absolute Error) of the prediction of 6, but they did not deal with transmission and analysis of data in real-time. Our approach is similar but used machine learning to determine the best polynomial approximation, i.e. the degree and the coefficients, using two variables: trend and seasonality of the number of devices detected.

3 System Design

In this chapter, the methodology and the choices in the system design are presented. Starting by describing the components of the system with their functionalities (i.e. the blocks in the system architecture) and then explaining the working logic of the developed system. After this chapter, it will be clear which are the main parts of this system and how they cooperate to achieve the project goal.

3.1 System architecture

In a place of interest, there are people/users with their devices that are sending Wi-Fi packets, if they have the Wi-Fi device turned on. The purpose of our system is to exploit these packets to infer the number of people present. The system architecture of the developed system is shown in figure 3.1. This is a distributed architecture, in fact, there are three main components on different platforms that cooperate over a communication network in order to achieve this goal.

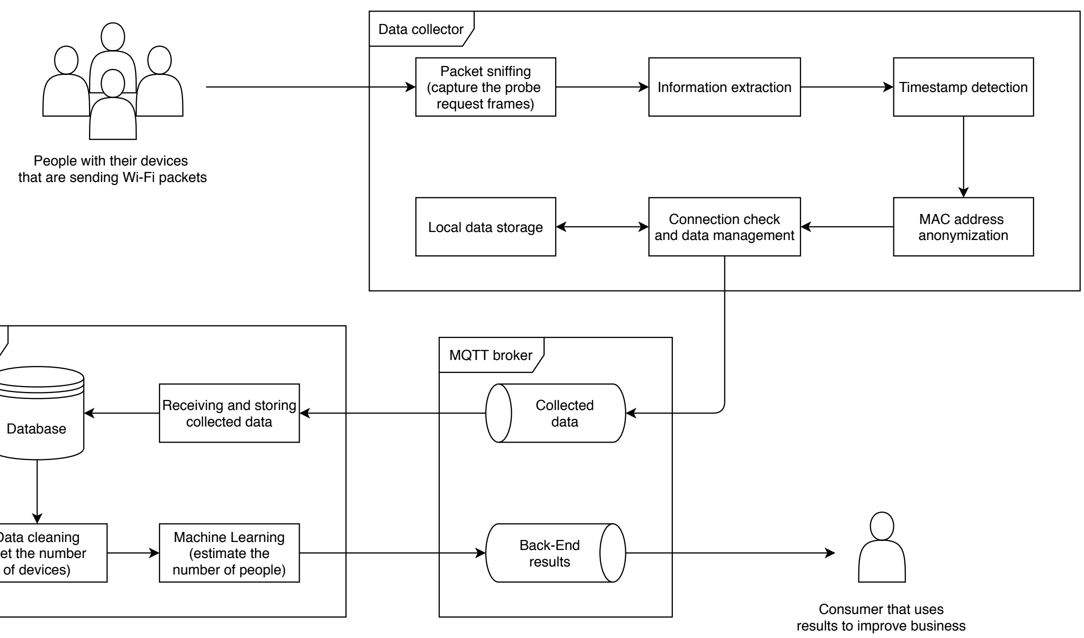

Figure 3.1: Architecture of the proposed system.

The first block of the system is a data collector which is located in the place of interest. It performs a preprocessing of the data (distributed work) with its functionalities: sniffing of Wi-Fi packets, capture the probe request frames; extract the useful information from these; detect the current timestamp; anonymize the MAC address using a hash function (no more privacy issues); check the internet connection; if there is no connection, save the data in the local storage; if there is a connection, publish the collected data to the dedicated queue.

The system uses the MQTT protocol for data forwarding, using an MQTT broker situated on a server, it must have a static IP address to be accessible, the broker has two dedicated queues: one for the collected data, published by the data collector and received by the subscribed receiver in the Back-End part, and the other one for the Back-End results published by the Back-End part after the analysis and received by the subscribed consumer.

The Back-End part is situated on a server. Initially, it deals with collected data receiving and storing in a database. Then, it deals with data cleaning: RSSI thresholding, remove random encounters (and randomized MAC addresses are removed with this), make a blacklist to remove the ever-present devices or devices revealed too many times during the day and therefore get the number of devices present in the place of interest. Finally, it deals with data analysis using Machine Learning: fit the degree and the coefficients of the polynomial approximation using the trend and the seasonality of the number of devices detected to get the number of people present and publication of the results to the dedicated queue.

At the end of this processing, there is the consumer who receives the results of the Back-End, i.e. the number of people in the place of interest, and could use this information to improve his business.

This architecture is scalable, it could admit many sensors physically distributed in different environments for data collection, to provide a practical example some use cases are shown in figure 3.2. It is important to locate them properly in the environments to cover all the areas of interest. All sensors publish data to the same MQTT broker and then all data is forwarded to the same Back-End. Cleaning and analysis will be performed according to the data source, each sensor will publish data in its own reserved queue and will be analyzed adequately to the characteristics of the use case for which it was used. In the end, results are sent to the respective consumer through an appropriate queue.

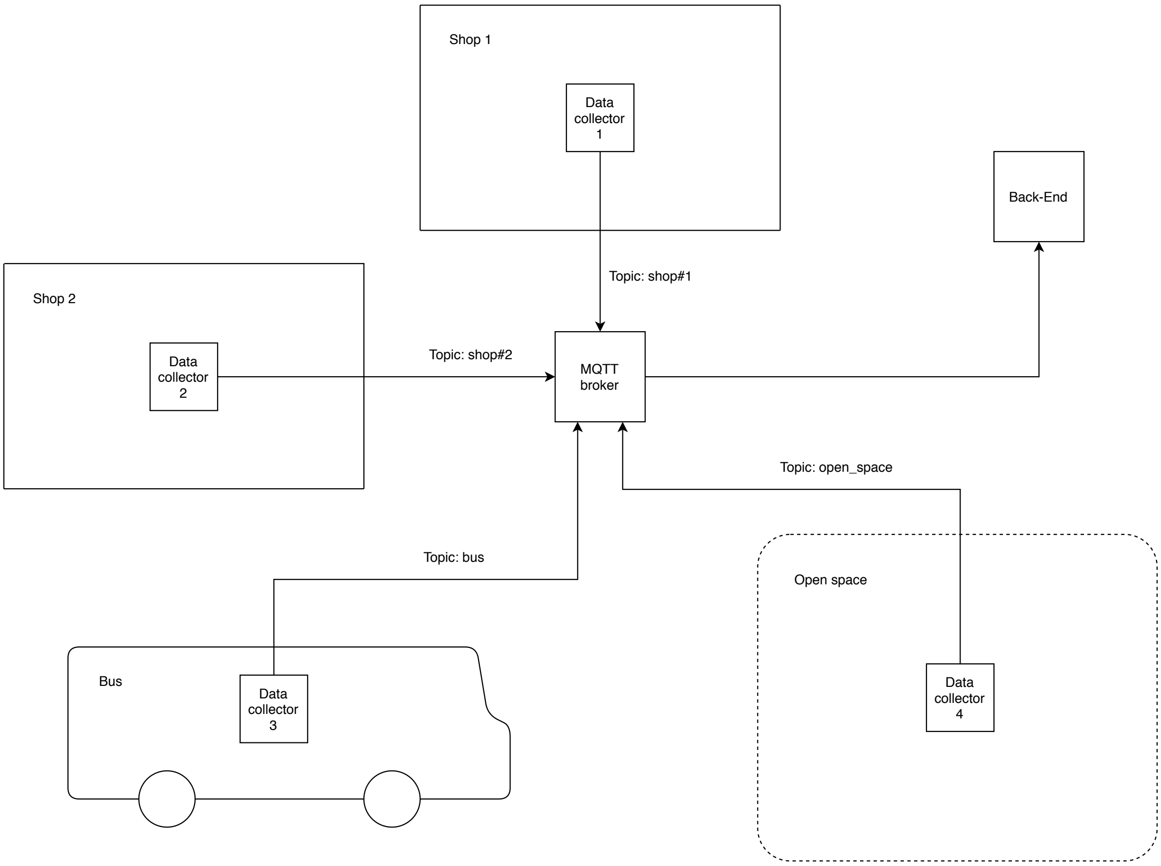

Figure 3.2: Presentation of a possible implementation with different use cases.

3.2 Data collection

The main block I worked on is the data collector. I developed the logic behind this block, with particular attention in the research of the connection and in the management of the data collected.

The flowchart representing the data collector logic is shown in figure 3.3. When the data collector is turned on, the system searches for the connected Wi-Fi dongle and puts it in monitor mode to perform the packet sniffing. Initially, a packet counter is set = 0 and the actual timestamp is saved. When a packet is received, it checks if it is a probe request frame and if it is not, it is thrown away. When a probe request frame is captured, the following information are extracted: RSSI, SSID (Service Set

Identifier), MAC address, sequence counter. The timestamp when the packet was revealed together with the other information are saved and the packet counter increases by one.

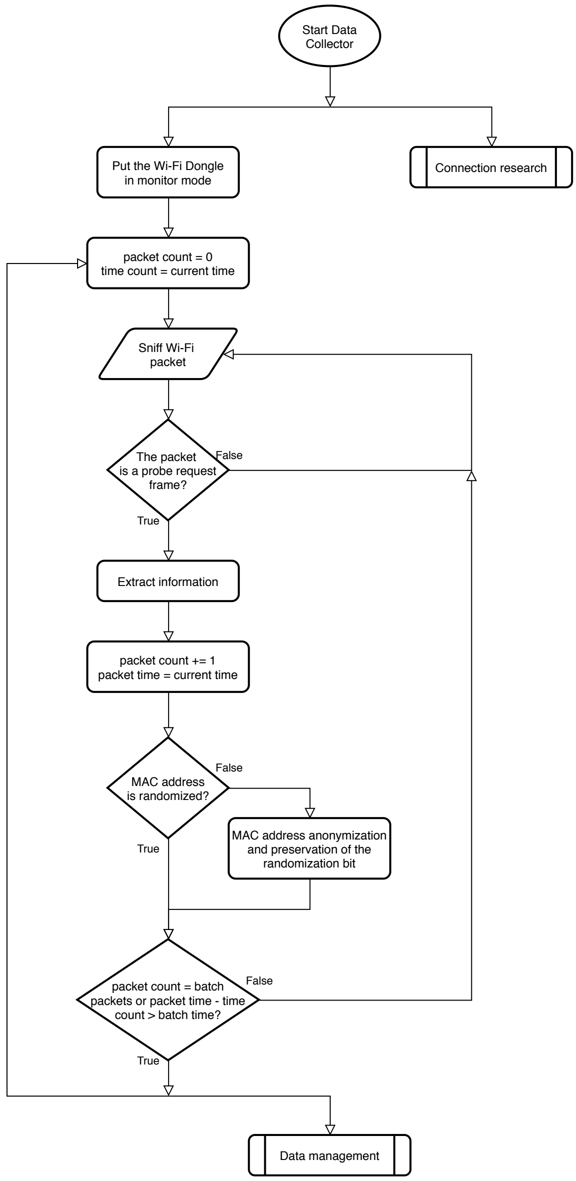

Figure 3.3: Flowchart representing the data collector logic.

Among the information that can be extracted from the MAC address before performing the anonymization there are the OUI (Organizationally Unique Identifier), from which the manufacturer of the device can be identified, and the local bit, the 7th bit of the MAC address, from which it can be seen whether the MAC address is randomized or not.

After extracting the information, there is the anonymization of the MAC address to preserve the privacy of the users and comply with the GDPR. This is made only for the MAC addresses that are not randomized, i.e. the real MAC addresses of the devices present. Anonymization is performed at this point in the architecture because then there are no more privacy problems, this work does not have to be done from the Back-End and for reasons of data security in case of a data breach during transmission. This process consists of hashing the MAC address using the BLAKE2 cryptographic hash function[3] (BLAKE3 is faster but still under development). BLAKE2 is faster than MD5, SHA-1, SHA-2 and SHA-3, and provides security superior to SHA-2 and similar to that of SHA-3. BLAKE2 supports keying and salting, and can output digests from 1 up to 32 or 64 bytes depending on the version. This hash function is the ideal for changing hash results every 24 hours or predefined time,

[3]website: `https://blake2.net/`

simply modifying the key or adding a different value of salt. This is made for privacy reasons and to avoid tracing the MAC address although it is not the real one but always the same after hashing. If the anonymization changes the state of the local bit, the previous state is forced/imposed to preserve this information on randomization, which can be used in the cleaning phase.

Once the pre-processing of the data is completed, we decided to process the collected data in batches and therefore there are to check 2 batch criteria, based on a maximum number of packets and a maximum time since the last batch transmission. These parameters have to be set according to the use case/case study, once one of these is reached it is possible to decide what to do with this data by running the data management flowchart, shown in figure 3.4.

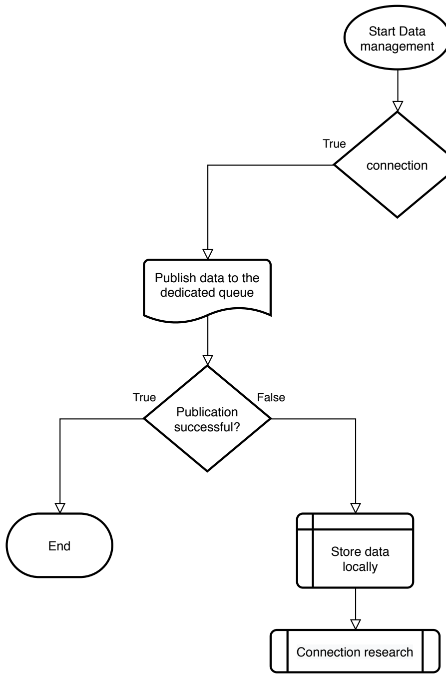

Figure 3.4: Flowchart representing the data management logic.

Higher values of the parameters allow us to avoid a continuous transmission of data and to fill the queue on the broker or in the lack of a connection, to open the file to save the data every time a packet is detected. On the other hand, choosing smaller values of the parameters reduces estimation delay, making the system more responsive. In our experiments, we used intermediate values with a maximum number of packets of 50 and a maximum time since the last batch transmission of 60 seconds, which is a good trade-off between congestion and transmission delay.

When the batch is ready, the algorithm for managing what to do with the data starts. If there is no connection, the collected data is stored locally to be sent later when a connection is found. Instead, if there is a connection with the MQTT broker, the collected data is published to the dedicated queue. If the publication is successful, the flowchart ends. Otherwise, it is stored locally, subsequently the system searches again for a connection.

When the data collector is turned on, the system searches also for a connection with the MQTT broker. Figure 3.5 shows the flowchart explaining the logic behind this. Initially, the connection is set = False and the system checks whether the data collector has a peripheral with an IP address for Internet access. When it has an IP address, a MQTT Client with its credentials tries connecting to the MQTT broker. When the broker is reachable and accepts the client connection, the connection is set = True and if there is data stored locally, it can be published by running the flowchart for the publication of the stored data, shown in figure 3.6.

The data is read from the local storage and is published to the dedicated queue. If the publication is successful, the data is deleted from the local storage. Otherwise, it remains stored locally and the system searches again for a connection.

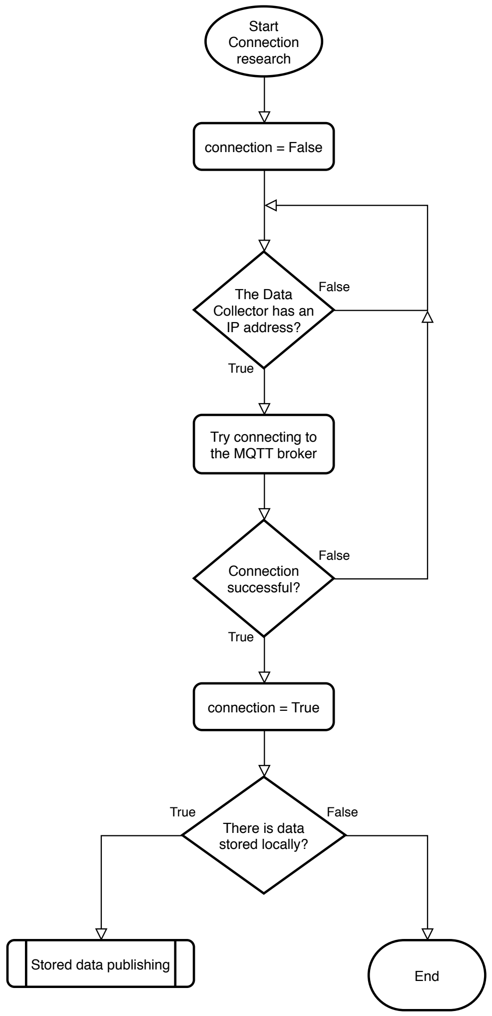

Figure 3.5: Flowchart representing the connection research logic.

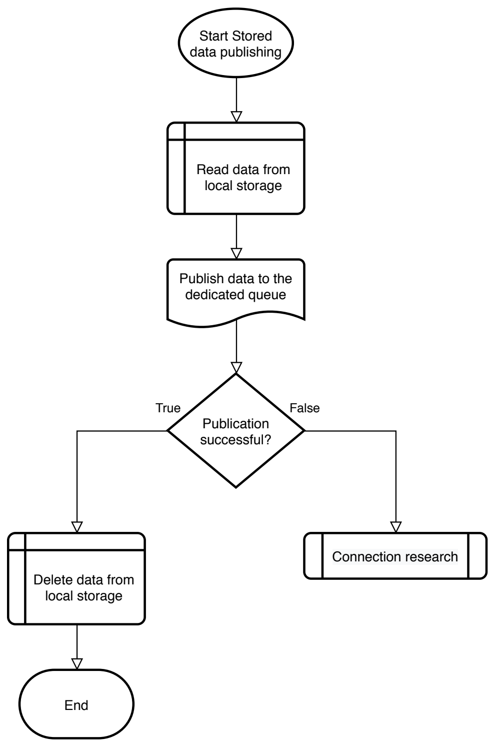

Figure 3.6: Flowchart representing the stored data publishing logic.

3.3 Data forwarding

MQTT[4] stands for Message Queuing Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol. It is designed to be bandwidth-efficient and to use little battery power. It is efficient in distributing information to one or many receivers. Information is organized in topics. Topics are treated as a hierarchy, using a slash (/) as a separator. These principles make it the ideal protocol for the emerging world of machine-to-machine/Internet of Things and for mobile applications where bandwidth and battery power are limited. It is useful for connections with remote locations, as in our case to send a huge amount of data to the Back-End part. Moreover, it can easily scale from a single device to thousands.

The MQTT protocol defines two types of network entities: a message broker and clients. An MQTT broker is a software running on a computer that receives all messages from the clients and then routes them to the appropriate destination clients. An MQTT client is any device that runs an MQTT library and connects to an MQTT broker over a network. For security reasons, the MQTT broker can be configured to require clients to use a username and password when connecting. If they match allowed credentials, clients can publish/subscribe to the topics. Otherwise, the connection is refused. When a publisher has new data to distribute, it publishes this data to the broker. Any client that wants a copy of that message will subscribe to that topic. The broker then distributes the data to any clients that have subscribed to that topic. Multiple clients can receive the message from a single broker. Similarly, multiple publishers can publish topics to a single subscriber.

[4]website: `http://mqtt.org/`

The protocol uses a publish/subscribe architecture in contrast to HTTP with its request/response paradigm. The main difference to HTTP is that a client does not have to pull the information it needs, but the broker pushes the information to the subscribed client, in case something new has been published. The broker also keeps track of all session information as clients connect and disconnect. If this connection is interrupted by any circumstances, the MQTT broker can buffer all messages and send them to the client when it is back online, setting the clean session bit = False. If clean session bit is true, then all subscriptions will be removed for the client when it disconnects.

Each subscription/publication to the broker can specify a quality of service measure. These are classified in increasing order of overheads:

- 0: At most once - the message is sent only once and the publisher and broker take no additional steps to acknowledge delivery to the subscribers (fire and forget, with no confirmation).
- 1: At least once - the message is re-tried by the sender (publisher or broker) multiple times until acknowledgment is received (by the broker or the subscribers) (acknowledged delivery).
- 2: Exactly once - the sender (publisher or broker) and receiver (broker or the subscribers) engage in a two-level handshake to ensure only one copy of the message is received (assured delivery).

For these illustrated features, the MQTT protocol is used in the system for data forwarding. A broker situated on a server, with a static IP address to be accessible and with two dedicated queues is used, one for the collected data, published by the data collector and received by the subscribed Back-End part, and the other one for the Back-End results published by the Back-End and received by the subscribed consumer. The broker is set up to allow only connections from the data collector, the Back-End and the consumer. They have their own credentials for authentication.

The clean_session is set = False. If this connection is interrupted by any circumstances, the MQTT broker can buffer all messages and send them to the client when it is back online. The QoS is set = 2 both in publish and subscribe. To ensure exactly one copy of the data, no loss, no duplicates to clean.

3.4 Data cleaning and analysis

The Back-End part is situated on a server and receives and stores the collected data. The two main Back-End functionalities are data cleaning and data analysis.

The flowchart representing the data Back-End logic is shown in figure 3.7. Initially, the Back-End subscribes to the collected data queue to receive the data when published by the data collector. Input data is managed by adding corollary information for future analysis, e.g. randomization and day of the week. Then this data is stored in a database.

When a time slot is over, the data of the current time slot is read from the database. An RSSI-based threshold is applied to delete data from devices too far from the data collector (to be adapted to the case study, the position of the data collector has to be taken into consideration, as well the environment in which they are located, for cleaning and analysis). Random encounters are removed, as shown in the flowchart in figure 3.8. Unique devices are extracted with their occurrence timestamps. If there are devices that appear only once or for a short time, they are removed. These parameters have to be adapted to the case study: MIN_T=20 MIN_RSSI=-100 (and randomized addresses are removed with this).

Then a blacklist is made to remove the ever-present devices or devices revealed too many times during the day, as shown in the flowchart in figure 3.9. A list of occurrences is created with a maximum interrupt time of delta_t. If there are devices that appears many times or for a long time, they are blacklisted. These parameters have to be adapted to the case study: MAX_T=7200 MAX_OCC=10 DELTA_T=300. At this point, it is possible to get the number of devices present in the place of interest for each timestamp based on the occurrences of the devices not in the blacklist. The presence of a device is increased by a probe compensation time set as t_p=15 seconds added before the first Wi-Fi probe request detection of the MAC address and after the last one. This is done to consider the time that passes between the actual presence of the device and the actual transmission of the first Wi-Fi probe request.

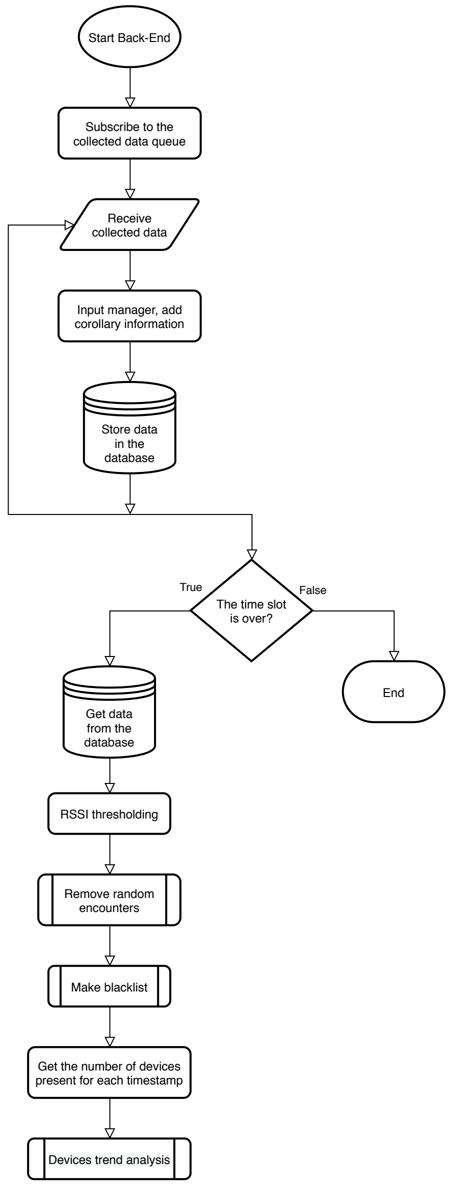

Figure 3.7: Flowchart representing the Back-End and the cleaning logic.

Finally, data analysis is performed, as shown in the flowchart in figure 3.10. Trend (raising/lowering) and seasonality (repetition of the components) of the number of devices are extracted using a decomposition of the corresponding temporal series. Once these two variables are known, it is possible to apply to them the correct polynomial approximation relating to the current time slot to get the forecast of the people present in the place of interest.

I did not develop the part of machine learning, but I integrated an existing one into this project to get the final results, i.e. the number of people in the place of interest for each timestamp. Then the results, when available, are published to the dedicated queue (Back-End results) to be received by the subscribed consumer who could use this information to improve his business.

3.4.1 Preparation of the model

The best degree and coefficients of the polynomial approximation are calculated during the preparation of the model that consist in a training phase for each time slot of each day of the week, using manually-collected ground truth. Figure 3.11 shows the flowchart of the logic used in that phase.

In the timestamps where the ground truth is collected by the number of devices revealed in these timestamps, we get the trend and the seasonality of the annotated dataset. Using this dataset we train a regressor to find the best polynomial approximation for each considered time slot. We initially

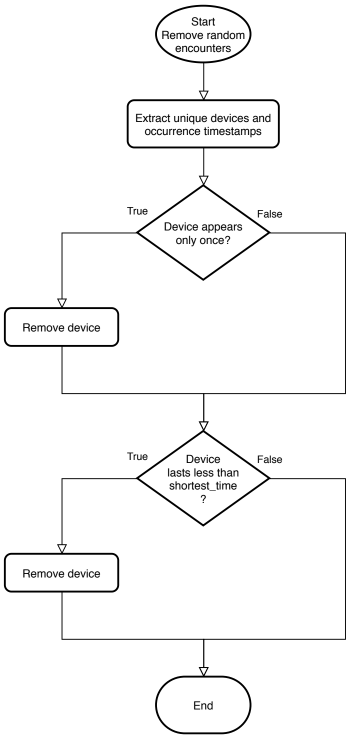

Figure 3.8: Flowchart representing the logic of removing random encounters.

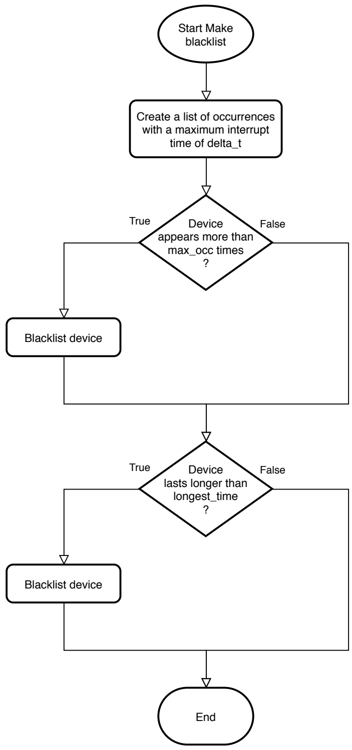

Figure 3.9: Flowchart representing the logic of making the blacklist.

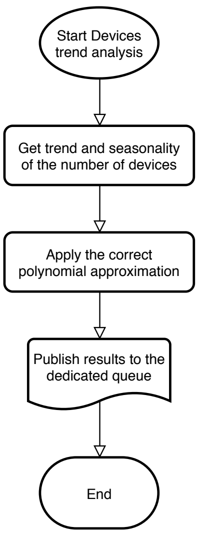

Figure 3.10: Flowchart representing the data analysis logic.

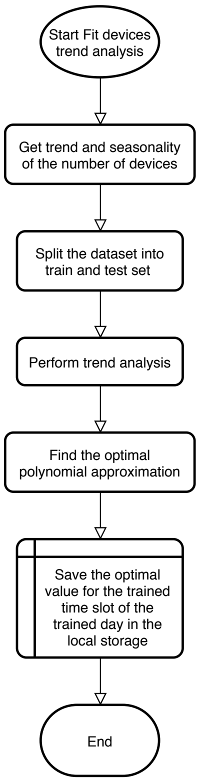

Figure 3.11: Flowchart representing the ML training logic.

divide the dataset into train set and test set. The Machine Learning model performs a trend analysis to find the best polynomial approximation for each time slot. These values of the coefficients are stored to be used in the different time slots when data are collected to get the correct estimate of the number of people.

There are 3 main time processes to manage:

- Probe revelations from the data collector: a random point process with different MAC addresses.
- Presence of devices: a cadlag step function based on the revelations of the MAC addresses, obtained after the cleaning of the collected data. Figure 3.12 show how the presence of the devices is obtained.
- Collection of the ground truth: a sampling of the number people present, a random point process asynchronous with respect to the probe revealed and devices revealed. Figure 3.13 shows how the ground truth is collected.

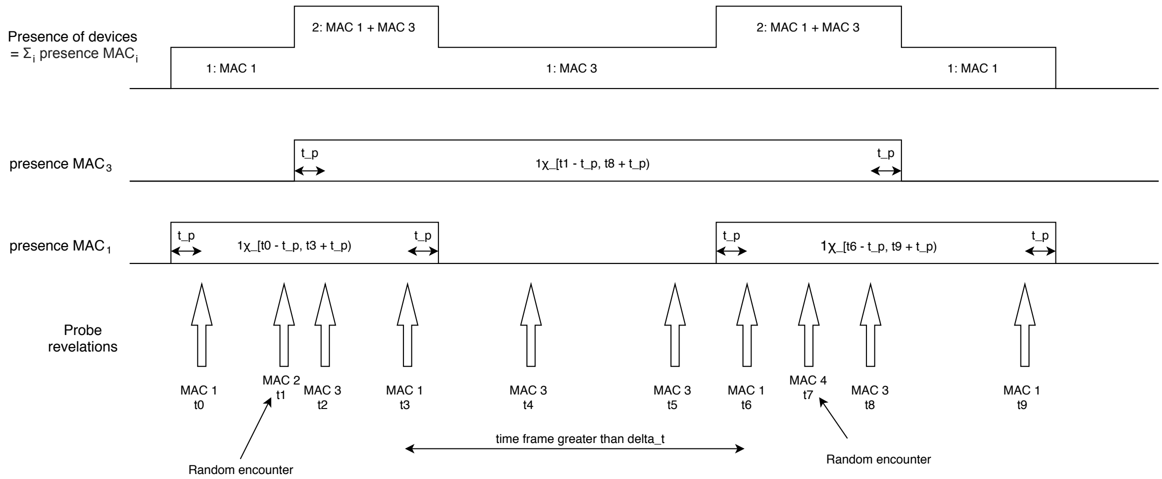

Figure 3.12: Illustration of the collection of the ground truth.

Step function for the presence of the different MAC addresses:
presence $\mathrm{MAC}_1 = 1 \cdot \chi_{[t_0 - t_p, t_3 + t_p)} + 1 \cdot \chi_{[t_6 - t_p, t_9 + t_p)}$
presence of devices $= \sum_i$ presence MAC_i
Dirac delta function for sampling the number of people presence:
$\mathrm{GT}(\mathrm{t}_i) = g(t) \cdot \delta(t - t_i)$

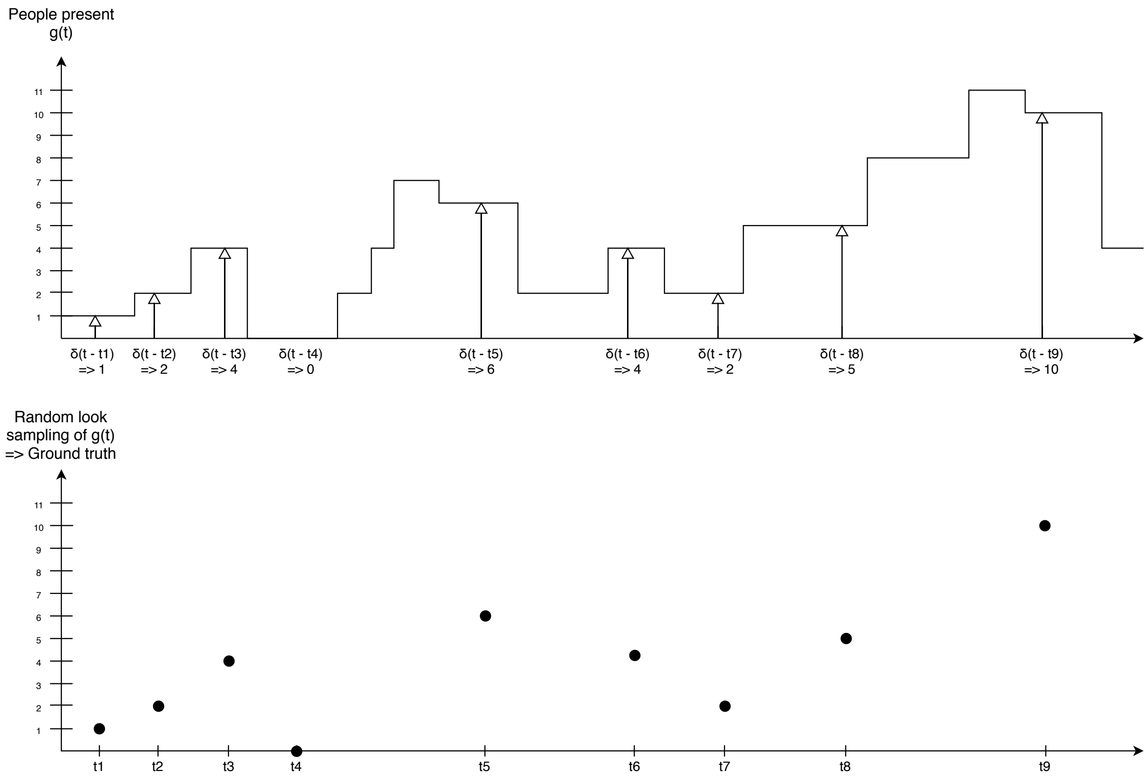

Figure 3.13: Illustration of the collection of the ground truth.

4 Implementation

This chapter presents how the components of the system have been implemented. Starting by describing the implementation of the data collector on a Raspberry Pi and then explaining how the Mosquitto MQTT Broker, the MongoDB database and the two Back-End parts, one for receiving and storing the data and the other one for analyzing the data, has been implemented using Docker containers. After this chapter, it will be clear how the system was implemented for the testing and validation, that are discussed in the next chapter.

4.1 Data collector implementation

Data collector implemented on a Raspberry Pi model 2B using:

- A EDUP 802.11n Wi-Fi Dongle (Range $\sim$ 10 meters) is used with the Scapy library for Wi-Fi packets sniffing and information extraction.
- A Real Time Clock Module is used for timestamp detection.
- From hashlib Python module BLAKE2s, an implementation of the BLAKE2, is used for MAC address anonymization.
- JSON Python module is used to store data locally and to read and publish them later.
- Another EDUP 802.11n Wi-Fi Dongle (Range $\sim$ 10 meters) is used with the Netifaces library for managing the connection.
- Paho-MQTT library is used for running the MQTT client and the data forwarding.

Client(client_id="name", clean_session=False), connect("broker_address", port=1883)
username_pw_set("username", password="password")
publish("topic", payload=json.dumps(data), qos=2)
subscribe("topic", qos=2)

4.2 Tool for ground truth management

I developed two scripts to manage the ground truth. One for collecting the ground truth and made the revelations in the place of interest. When the number of people present is written on the input of the program, it saves this information with the timestamp, the day of the week on a file. The other script is used to load the collected ground truth in the MongoDB using PyMongo MongoClient for accessing the database and storing data.

4.3 Server implementation

Docker Containers running on the server:

- As MQTT broker we used Eclipse Mosquitto[5], an open-source MQTT broker.
- As database we used MongoDB[6] for its simplicity in handling JSON files.

[5] website: `https://hub.docker.com/_/eclipse-mosquitto`
[6] website: `https://hub.docker.com/_/mongo`

- For running the receiver MQTT client we used Paho-MQTT for receiving the data and for running a MongoClient we used PyMongo for storing data on MongoDB.
- In the analyzer, we used PyMongo and Pandas to read and manage the data stored in MongoDB with a MongoClient to get the data and a Pandas DataFrame to clean the data and get the number of devices.
- Sklearn library is used to analyze the cleaned data and the ground truth. This library implements the machine learning part.

Finally, I developed a script to calculate the metrics, e.g. error information, and compute the graphs, e.g. histograms, error distribution and comparison of revealed devices and of people estimated with the real number of people present.

5 Evaluation

In this chapter, the proposed system is validated and the results are evaluated. Starting by describing how the system is tested at home to assess the feasibility of the proposed method. Subsequently, where and how the system is tested for experimental validation. Finally, an evaluation of the achieved results is presented. The data collector implemented on a Raspberry Pi has been placed in a place of social interest. In this place of social interest, I manually collected the ground truth for training and evaluating the model. The Mosquitto MQTT Broker, the MongoDB database and the Back-End part for receiving and storing the data in the database have been executed on the U-Hopper server. The final Back-End part for analyzing the data has been executed on my computer to adapt the parameters and train the Machine Learning model. After this chapter, it will be clear how the system has been validated and what the results achieved by the proposed system are. Overall conclusions are discussed in the next chapter.

5.1 Experimental validation

Due to the complex situation of this period due to COVID-19 and the limitations imposed on people, it was necessary to test the functioning of the system at home and verify the feasibility of the first part of the proposed method, i.e. detect the devices in the area. The tests were conducted on different days with the aim of revealing how many devices are in a determinate room of the house. As can be seen from figure 5.1, $\sim$ 66.000 Wi-Fi probe request frames are collected and the values of RSSI of the detected devices are divided into two main ranges. A value of RSSI in range -35 $\div$ -69 indicates that the device is close to the data collector, i.e. in the kitchen. Instead, a value of RSSI in range -69 $\div$ -91 indicates that the device is away from the data collector, i.e. not in the kitchen. Random encounters that do not belong to the house are rare and there are randomized MACs related to devices that are not connected to the home Wi-Fi network.

Devices		Day 1			Day 2			Day 3		Comments
	# probes	RSSI range 1	RSSI range 2	# probes	RSSI range 1	RSSI range 2	# probes	RSSI range 1	RSSI range 2	
Wi-Fi gate	6410	-75 ÷ -81		4959	-65 ÷ -77		4819	-65 ÷ -75		omnipresent, send 2/4 probe every ~ 30 sec, static, ~ 6/7 m away
Smart TV	6	-83 ÷ -89		/			/			static in the living room, ~ 6/7 m away
PlayStation 4	13660	-71 ÷ -83		5	-71 ÷ -75		5	-75 ÷ -81		static in the living room, ~ 6/7 m away
iMac	294	-83 ÷ -91		197	-79 ÷ -87		59	-75 ÷ -85		static in my bedroom, ~ 10 m away
MacBook	32	-87 ÷ -91	-61 ÷ -67	/			/			range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
Mom's Samsung	38	-75 ÷ -91		3	-73 ÷ -79		/			far, not in the kitchen (Wi-Fi usually turned off)
Grandma's Samsung	/			/			39	-71 ÷ -81	-57 ÷ -65	range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
Thomas's Samsung	103	-75 ÷ -89	-57 ÷ -69	49	-71 ÷ -79	-51 ÷ -69	166	-73 ÷ -81	-47 ÷ -69	range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
Dad's iPhone	94	-75 ÷ -91	-55 ÷ -65	1547	-77 ÷ -83	-49 ÷ -67	1170	-69 ÷ -83	-35 ÷ -67	range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
Mattia's iPhone	1377	-77 ÷ -91	-59 ÷ -75	978	-75 ÷ -85	-49 ÷ -65	2051	-73 ÷ -85	-49 ÷ -67	range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
My iPhone	40	-79 ÷ -91	-65 ÷ -67	54	-73 ÷ -85	59	98	-73÷ -83	-53 ÷ -67	range 1 → far, not in the kitchen; range 2 → nearby, in the kitchen
Printer	/			1	-81		/			static in my bedroom, ~ 10 m away
Other Wi-Fi dongle	/			13084	-19 ÷ -23	-37 ÷ -63	12188	-19 ÷ -23	-37 ÷ -61	another Wi-Fi dongle, 1/2 probe every 6/7 sec → value swings sometimes
Samsung Galaxy J3	17	-85 ÷ -91		/			/			non-home device
Samsung Galazy A20e	/			1	-83		/			non-home device
Randomized MACs	626			568			1190			probes with randomized MAC address, vague values of the RSSI
Total	22698			21445			21785			average of 22000 Wi-Fi probe request frames for ~ 18 hours

Figure 5.1: Presentation of home test results.

Subsequently, with the reduction of restrictions on personal mobility, it was possible to conduct an experimental validation and a data collection in a place of social interest with moderate mobility. The case study is to estimate the customers present in a Cafe, whose owner allowed me to conduct the experimentation in this delicate moment.

The data collector implemented on a Raspberry Pi model 2B has been placed in this Cafe and connected to an electrical source. In this place of social interest, I manually collected the ground truth with a random look strategy on my computer for training and evaluating the model.

The Mosquitto MQTT Broker, the MongoDB database and the Back-End part for receiving and storing the data in the database have been executed on the U-Hopper server.

Experimental environment considerations: Wi-Fi dongle (with a range of ~ 10 meters) can cover all the Cafe area and detect all the devices of the people. The Cafe does not have a Wi-Fi network but, when I go there to collect the ground truth, it uses the hotspot from my phone to send data to the server.

The final part of the Back-End for data analysis was executed on my computer to adapt the parameters of the cleaning part for the case study. After cleaning the data and obtaining the presence of devices, it is possible to prepare and train the Machine Learning model with the collected ground truth and the number of devices revealed in the timestamps when the ground truth was collected.

5.2 Evaluation of the results

Once the Machine Learning model is trained with the collected ground truth and the number of devices revealed at the timestamps when the ground truth is collected, the estimates of people are generated in these timestamps. We calculated the following parameters for evaluating the proposed system estimates and satisfy the KPIs (Key Performance Indicators), taking as the ground truth the number of people I manually annotated, with a random look strategy of some hours per day in different time slots:

- Mean Absolute Error = mean(abs(people_present - people_estimated))
- Mean Squared Error = mean(square(people_present - people_estimated))
- For each revelation scaled_MSE_trend += $\left(\frac{people\_present - people\_estimated}{people\_present}\right)^2$ and in the end Scaled_MSE_trend/revelation = $\frac{scaled\_MSE\_trend}{revelation}$
- For each revelation scaled_MAE_trend += abs$\left(\frac{people_present - people_estimated}{people_present}\right)$ and in the end Scaled_MAE_trend/revelation = $\frac{scaled\_MAE\_trend}{revelation}$

In figure 5.2 and 5.3 some results and useful processing information are reported.

	2020-06-16	2020-06-17	2020-06-18	2020-06-19	2020-06-20	2020-06-21		Total1
Probe captured	24011	19329	22533	23041	19062	11238		119214
Ttotal MACs	1489	873	1281	1307	1447	1187		7584
MACs only registered once	909	504	753	852	1002	634		4654
MACs lasted shorter than 20 seconds	443	280	395	344	342	447		2251
MACs occurred more than 10 times throughout the day	2	3	3	3	2	0		13
MACs lasted longer than 7200 seconds in any of it's occurrences	4	2	5	5	4	3		23
MACs remained	131	84	125	103	97	103		643
Manual annotations	116	61	63	51	37	12		340
							Mean Absolute Error	1.461
							Mean Squared Error	4.039
							Scaled_MAE_trend/count	0.448
							Scaled_MSE_trend/count	0.700

Figure 5.2: Illustration of the results and useful processing information of the first week.

	2020-06-23	2020-06-24	2020-06-25	2020-06-26	2020-06-27	2020-06-28		Total2	Total	
	45730	56835	58203	61900	28941	19768		271377	390591	
	1579	2260	2293	2282	1182	715		10311	17895	
	333	490	509	462	242	106		2142	6796	
	1060	1568	1543	1577	805	517		7070	9321	
	0	2	1	2	0	0		5	18	
	6	5	6	8	3	3		31	54	
	180	195	234	233	132	89		1063	1706	
	76	67	69	63	31	14		320	660	
							Mean Absolute Error	1.461	1.461	
							Mean Squared Error	4.039	4.039	
							Scaled_MAE_trend/count	0.448	0.448	
							Scaled_MSE_trend/count	0.700	0.700	

Figure 5.3: Illustration of the results and useful processing information of the second week.

Finally, in figure 5.4, 5.5, 5.6, 5.7, 5.8 there are some graphs created to show the Machine Learning results in terms of errors and comparison of the estimates between revealed devices and the ground truth.

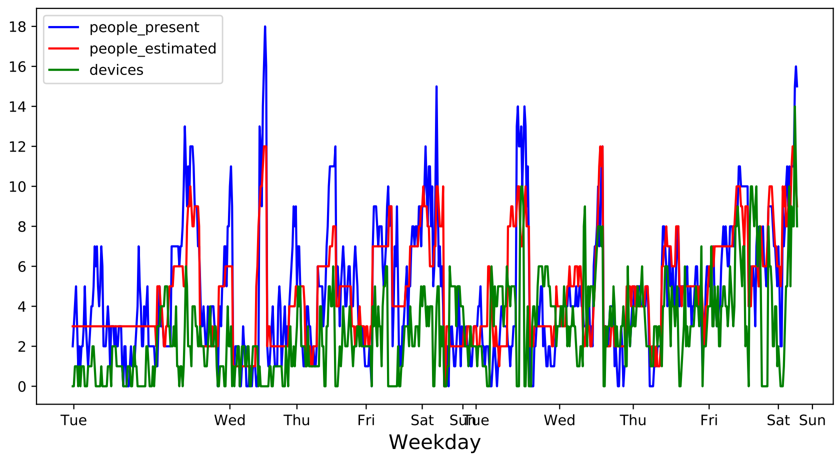

Figure 5.4: Comparison between estimates, revealed devices and ground truth.

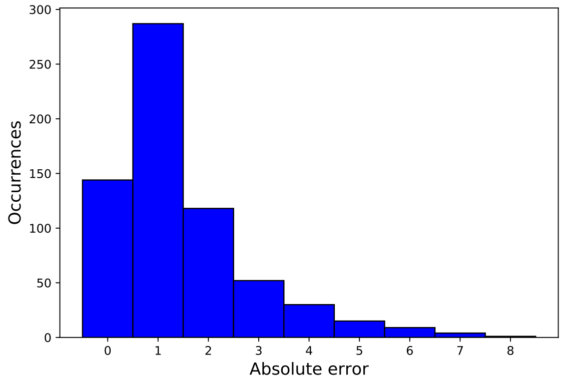

Figure 5.5: Graph illustrating the distribution of the absolute error.

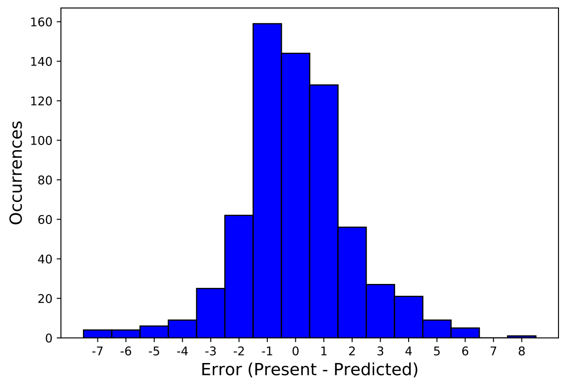

Figure 5.6: Graph illustrating the distribution of the error.

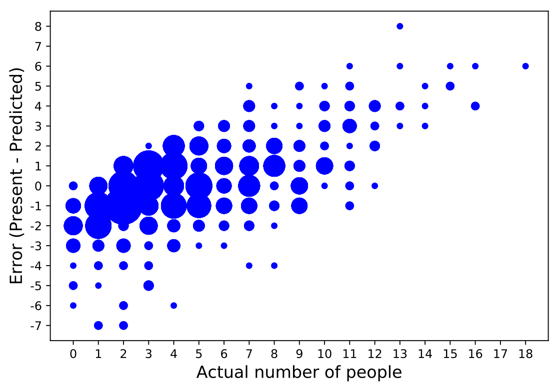

Figure 5.7: Scatter plot illustrating how the error is distributed in relation to the present number of people.

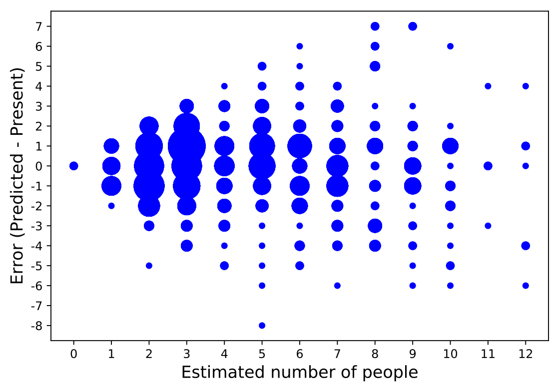

Figure 5.8: Scatter plot illustrating how the error is distributed in relation to the estimated number of people.

In figure 5.9 there is a graph created to show the Machine Learning results in terms of comparison of the estimates between revealed devices on all the time series when the Wi-Fi probe request frames are revealed.

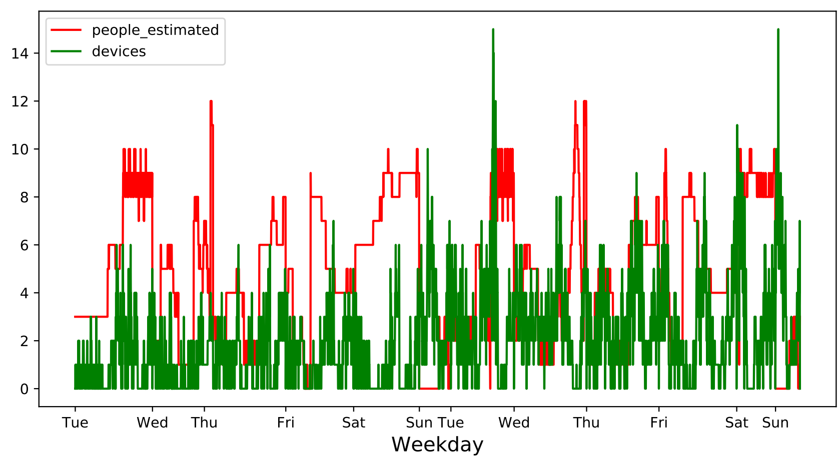

Figure 5.9: Comparison between estimates and revealed devices.

6 Conclusions

Write conclusions about the work done ...

6.1 Future work

Write about future work ...

Bibliography

[1] Lu Bai, Neil Ireson, Suvodeep Mazumdar, and Fabio Ciravegna. “Lessons learned using Wi-Fi and Bluetooth as means to monitor public service usage”. In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. 2017, pp. 432–440.

[2] Nicola Bernini, Luca Bombini, Michele Buzzoni, Pietro Cerri, and Paolo Grisleri. “An embedded system for counting passengers in public transportation vehicles”. In: *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. IEEE. 2014, pp. 1–6.

[3] Falk Brockmann, Marcus Handte, and Pedro José Marrón. “CutiQueue: People counting in waiting lines using Bluetooth Low Energy based passive presence detection”. In: *2018 14th International Conference on Intelligent Environments (IE)*. IEEE. 2018, pp. 1–8.

[4] Jeong Woo Choi, Dae Hyeon Yim, and Sung Ho Cho. “People counting based on an IR-UWB radar sensor”. In: *IEEE Sensors Journal* 17.17 (2017), pp. 5717–5727.

[5] Simone Di Domenico, Mauro De Sanctis, Ernestina Cianca, Paolo Colucci, and Giuseppe Bianchi. “LTE-based passive device-free crowd density estimation”. In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–6.

[6] Gaurangi Gupta, Vaishnavi Bhope, Japneet Singh, and AR Harish. “Device-free crowd count estimation using passive UHF RFID technology”. In: *IEEE Journal of Radio Frequency Identification* 3.1 (2018), pp. 3–13.

[7] Marcus Handte, Stefan Foell, Stephan Wagner, Gerd Kortuem, and Pedro José Marrón. “An internet-of-things enabled connected navigation system for urban bus riders”. In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 735–744.

[8] Marcus Handte, Muhammad Umer Iqbal, Stephan Wagner, Wolfgang Apolinarski, Pedro José Marrón, Eva Maria Muñoz Navarro, Santiago Martinez, Sara Izquierdo Barthelemy, and Mario González Fernández. “Crowd density estimation for public transport vehicles”. In: *EDBT/ICDT Workshops*. 2014, pp. 315–322.

[9] Yuchen Jia and Ju Zhang. “The application of wireless communication technology in bus overcrowding monitoring”. In: *6th International Conference on Wireless, Mobile and Multi-Media (ICWMMN)*. IET, 2015, pp. 11–16.

[10] Pravein Govindan Kannan, Seshadri Padmanabha Venkatagiri, Mun Choon Chan, Akhihebbal L Ananda, and Li-Shiuan Peh. “Low cost crowd counting using audio tones”. In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. 2012, pp. 155–168.

[11] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. “Estimating people flows to better count them in crowded scenes”. In: *arXiv preprint arXiv:1911.10782* (2019).

[12] Lars Mikkelsen, Radoslav Buchakchiev, Tatiana Madsen, and Hans Peter Schwefel. “Public transport occupancy estimation using WLAN probing”. In: *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE. 2016, pp. 302–308.

[13] Lars Mikkelsen, Hans-Peter Schwefel, and Tatiana Madsen. “Sensing quality and estimation of public transport occupancy during live operation”. In: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE. 2018, pp. 1–4.

[14] Ryo Nishide. “Filter efficiency analysis for extracting mobile device signals to estimate bus passengers population”. In: *Proceedings of the 7th IIAE International Conference on Intelligent Systems and Image Processing.* 2019.

[15] Luiz Oliveira, João Henrique, Daniel Schneider, Jano de Souza, Sérgio Rodriques, and Weiming Sherr. “Sherlock: Capturing probe requests for automatic presence detection”. In: *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)).* IEEE. 2018, pp. 848–853.

[16] Luiz Oliveira, Daniel Schneider, Jano De Souza, and Weiming Shen. “Mobile device detection through WiFi probe request analysis”. In: *IEEE Access* 7 (2019), pp. 98579–98588.

[17] Thongtat Oransirikul and Hideyuki Takada. “The practicability of predicting the number of bus passengers by monitoring Wi-Fi signal from mobile devices with the polynomial regression”. In: *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers.* 2019, pp. 781–787.

[18] Ivano Pinna, Bruno Dalla Chiara, and F Deflorio. “Automatic passenger counting and vehicle load monitoring”. In: *Ingegneria Ferroviaria* 65.2 (2010), pp. 101–138.

[19] Pichaya Prasertsung and Teerayut Horanont. “How does coffee shop get crowded? Using WiFi footprints to deliver insights into the success of promotion”. In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers.* 2017, pp. 421–426.

[20] Deepak Babu Sam, Skand Vishwanath Peri, Amogh Kamath, R Venkatesh Babu, et al. “Locate, size and count: Accurately resolving people in dense crowds via detection”. In: *arXiv preprint arXiv:1906.07538* (2019).

[21] Lorenz Schauer, Martin Werner, and Philipp Marcus. “Estimating crowd densities and pedestrian flows using Wi-Fi and Bluetooth”. In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services.* 2014, pp. 171–177.

[22] Jiaxing Shen, Jiannong Cao, Xuefeng Liu, and Shaojie Tang. “SNOW: Detecting shopping groups using WiFi”. In: *IEEE Internet of Things Journal* 5.5 (2018), pp. 3908–3917.

[23] Kyosuke Shibata and Hiroshi Yamamoto. “People crowd density estimation system using deep learning for radio wave sensing of cellular communication”. In: *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC).* IEEE. 2019, pp. 143–148.

[24] Utkarsh Singh, Jean-François Determe, François Horlin, and Philippe De Doncker. “Crowd forecasting based on WiFi sensors and LSTM neural networks”. In: *IEEE Transactions on Instrumentation and Measurement* (2020).

[25] Rafael Valle. “ABROA: Audio-based room-occupancy analysis using Gaussian mixtures and hidden Markov models”. In: *2016 Future Technologies Conference (FTC).* IEEE. 2016, pp. 1270–1273.

[26] Zhe Wang, Tianzhen Hong, Mary Ann Piette, and Marco Pritoni. “Inferring occupant counts from Wi-Fi data in buildings through machine learning”. In: *Building and Environment* 158 (2019), pp. 281–294.

[27] Wei Zhang, Yongjie Wang, Yanyan Liu, and Jianghua Zhu. “Deep convolution network for dense crowd counting”. In: *IET Image Processing* (2019).

[28] Youmei Zhang, Faliang Chang, Mengdi Wang, Fulei Zhang, and Chao Han. “Auxiliary learning for crowd counting via count-net”. In: *Neurocomputing* 273 (2018), pp. 190–198.

[29] Yuren Zhou, Billy Pik Lik Lau, Zann Koh, Chau Yuen, and Benny Kai Kiat Ng. “Understanding crowd behaviors in a social event by passive WiFi sensing and data mining”. In: *IEEE Internet of Things Journal* (2020).