POLITECNICO DI TORINO

Master Degree course in Mechatronic Engineering

Master Degree Thesis

Urban Crowd Estimation via WiFi Probe Analysis

Supervisors

Prof. Claudio Casetti

Prof. Paolo Giaccone

Candidate

Giuseppe Perrone

Academic Year 2023-2024

Acknowledgements

This work was supported by the Horizon-JU-SNS-2022 Research and Innovation Programme under Grant Agreement No. 101095871 (TrialsNet project).
I would like to thank my supervisors, Claudio Ettore Casetti and Paolo Giaccone for their invaluable support and guidance throughout the drafting of this thesis.
Thanks to my mentors, Diego and Riccardo, for their availability and enthusiasm; without your help, I probably would have gone crazy.
Also, a special thanks to everyone who accompany me on this journey. All of this would be meaningless without all of you.

Abstract

Nowadays, cities are places where data streams are continuously and ubiquitously exchanged between smartphones, IoT devices, and other modern technologies. The constantly growing urban populations make it necessary to provide stakeholders like transportation managers and security agencies with accurate methods for crowd-monitoring. In this way, more efficient space construction processes can be designed, i.e., by optimizing resource allocation or enhancing security measures. Analyzing network traffic has proven to be an effective method to estimate people's presence in various specific areas. This thesis aims to provide a complete understanding of WiFi-based crowd-monitoring techniques, with their main advantages and drawbacks. We conducted an extensive experimental study of WiFi 802.11 probe requests, searching for new valuable information to enhance systems' performance and overcome their main limitations. The first phase of the research interested the content of the frames, particularly the Information Elements (IEs), fields that specify the characteristics of the source device. The findings were applied to ARGO, a network-based counting system developed in response to the Trialsnet European project. The conducted studies resulted in a new crowd-counting algorithm. It leverages additional fields of probe requests with respect to its predecessor and uses the OPTICS clustering algorithm to categorize and group Probe Requests based on the presumed source. The modifications applied resulted in an accuracy for crowd counting ranging from 83% to 93%. The results demonstrate the effectiveness of our adjustments, leading to an accuracy of up to 10 percentage points higher than the original. Additionally, the cluster quality has improved, as evidenced by higher values for homogeneity and completeness. In the second part of our work, we shift our focus to the time behavior of probe requests. We have examined how their sending rate varies according to the state of the channel, ranging from a non-congested state to a high-congested one. In the latter case, we observed a significant difference in the rates with respect to the ones recorded in isolated conditions. This analysis has shown that approaches used until now are not suitable, as they do not consider the impact of the congestion of WiFi channels in overcrowded environments. These observations have led to a new proposal for adapting the performance in diverse, specific contexts. This can lead to a more flexible algorithm but is feasible only when ground truth data are available, as the rate is tuned by minimizing the error relative to the effective ground truth. This research has shown that WiFi-based monitoring techniques offer a valid solution for crowd-estimation problems. The implemented system can provide accurate results, aligned with ground truth data. Furthermore, the affordability of this framework and its low power consumption make

this system suited for daily monitoring applications, enabling a more efficient resource allocation and preserving the privacy of people in the monitored areas.

Contents

Chapter 1

Introduction

Crowd monitoring includes all the processes related to the observation and management of groups of people in public spaces or events. It is essential in many domains:

- Where large mass gatherings occur, it can ensure public safety by preventing panic and improving emergency response capabilities.
- In the commercial domain, knowing the flow of people can provide information about the most frequented shops.
- In transportation, crowd-monitoring can identify the most used routes, improving public transport efficiency and resource allocation.

This thesis aims to develop a framework that can be used for crowd-monitoring applications. In particular, we focus on a Network-based Crowd-Monitoring system due to its significant potential and efficacy in large-scale implementations. The number of IoT devices is constantly increasing: according to [1], there were approximately 12.5 billion devices in 2010, and their number will reach 41.6 billion connected devices by 2025 [2]. The ubiquitous presence of smart devices generates vast amounts of transmitted data around us. Network-based crowd-monitoring systems utilize the transmitted data streams to gather information about crowd size in a monitored area.

Our focus is specifically on WiFi-based crowd-monitoring systems. These frameworks leverage the content of Probe Requests, i.e., frames standardized by IEEE and periodically broadcasted by any device with an active WiFi interface. Before Apple iOS 8, we could use these messages to track the presence and movement of any device, as each probe contained a globally unique identifier. However, to protect users' privacy, vendors started implementing MAC address randomization after 2014, randomizing the identifier to prevent privacy issues. We aim to develop a WiFi-based system to infer the crowd size within the area by analyzing these messages even with MAC address randomization.

We conducted extensive research to understand the structure of probe requests: we analyzed a vast dataset of probe requests to identify the principal information that could be obtained from them. The obtained insights are applied to an existing WiFi-based crowd-monitoring system [3] to enhance its accuracy. The results show that the WiFi-based crowd-monitoring systems can address crowd-monitoring applications, providing stakeholders with general knowledge about the crowd size without needing any sensitive information.

The first Chapter of this work talks about the principal systems for crowd counting. For each of them, we point out the main advantages and drawbacks. Chapter 3 talks about the sniffing system used for this work: it is developed by [3] and can collect all the probe requests transmitted by nearby devices. Chapter 4 analyses the WiFi-based counting system used as reference, analyzing its performance. Chapter 5 talks about the limitations of the previously presented algorithm and some applied improvements. These chapters show how WiFi-based crowd-monitoring can preserve people's privacy in monitored areas, granting, at the same time, the needed information.

The last Chapter analyses the time behavior of probe requests: we focus on the rate of the probes as a function of the source device, its phase, and the channel congestion. Here, it becomes clear that the behavior of these messages is almost unpredictable; it depends on many parameters, many of which are still unknown. Therefore, there is mention of a potential approach for these systems that could be calibrated in specific circumstances and could harness the best of this technology while minimizing errors to the greatest extent possible.

Chapter 2

People Counting Solutions

The main topic of this Chapter is the current state of art for people counting systems, which are frameworks able to estimate the number of people in a certain area. This Chapter provides an overview of different systems analyzing each of them on the basis of their performances, development cost, and compliance with privacy requirements imposed by the European Union [4]. Section 2.1 talks about “Image-based crowd-monitoring techniques”, i.e., video surveillance systems. Section 2.2 talks about all techniques which do not involve the use of recorded images , i.e., LIDAR and some other types of radio sensors, used to estimate the crowd size without exploiting any sensitive information.

2.1 Image-based Crowd-monitoring techniques

2.1.1 Video Surveillance systems

A video surveillance system aims to monitor the human activity in public or private scenarios using a system composed by multiple cameras strategically positioned to capture the whole environment.

The presence of a system of this type is justified by the necessity to ensure the prevention of any emergency, utilizing a proactive approach rather than reactive. The cameras capture images or videos of the monitored scenario and all the collected images are shown in a central station. The connection with the central station can be done in three different ways: analog, digital, and wireless.

A system of this type can collect a large amount of data which can be analysed through computer vision techniques to reduce the human involvement in video monitoring.

2.1.2 Automated Video Surveillance Systems

An automated system processes the input images of the camera giving an output that depends on the application considered. This task is achieved through computer vision: a field of artificial intelligence whose aim is to enable computers to interpret and understand visual information from the real world. To address this issue many algorithms can be utilized, they vary in the features extracted from the images and on the computed output. Examples of computer vision algorithms used in crowd-monitoring can be found in [5], where authors talk about:

- Face recognition.
- Object classification.
- Motion detection.
- Image segmentation.

All these algorithms leverage machine learning and deep learning techniques to train models on large datasets. Starting from the trained model it is possible to recognize patterns and make decisions based on the visual input received.

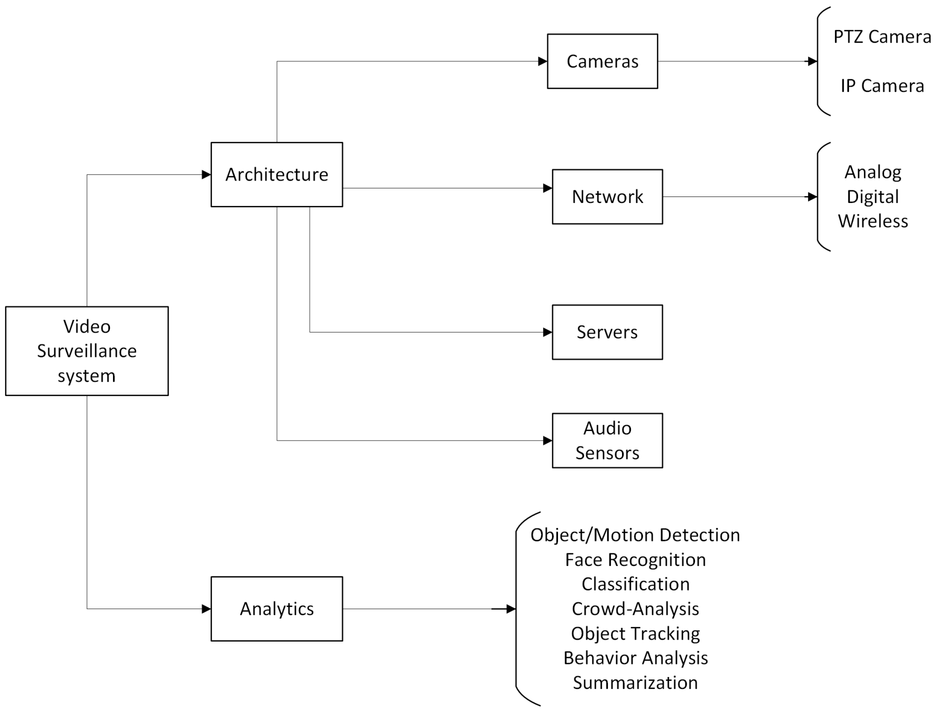


Figure 2.1: Video surveillance system architecture

Figure 2.1 shows the overall architecture of a Video-surveillance system. In the following Sections, we analyze some algorithms developed to address crowd-monitoring applications, pointing out the main pros and drawbacks for each of them.

Computer vision for Crowd-Counting

According the authors of [6], we can divide the methods for video surveillance-based crowd monitoring systems in mainly three different classes:

- **Counting by detection**: Authors of [7] define this approach as a “Direct Techniques”. The algorithm can recognize if an object is a person or not through a classifier. The final output is the number of objects classified as “Person”.
- **Counting by clustering**: More flexible than Supervised learning approaches. Counting by clustering focuses on the motion of the objects in the scene: different trajectories may be clustered together and the total number of clusters provides the objects’ count in the scene.
- **Counting by regression**: Explicit pedestrian detection is avoided to increase the computational efficiency of the system. The algorithm learns how to map the image’s features into the number of people in the scene. The first step of this type of approach is the features’ extraction, i.e., area of the region [8, 9], edge features [10–12] or texture features [13, 14]. The extracted features are correlated to crowd size based on a regression model.

The high computational cost of the first two approaches makes them not suited for real-time applications. In addition, the performances are not so good in case of not moving objects and in case of occlusions in the crowd.

In the latest years, Convolutional Neural Networks have been largely employed in this context. Despite the many advantages that these systems can offer, also in this case an high computational cost is required.

Pedestrian detection-based approaches

Object classification is a class of algorithms which allows computers to divide objects in different classes based on some characteristics of the image. Figure 2.2 shows some of the possible algorithms, each has its own way to address classification problems.

In this type of approach, a classifier utilizes features extracted from an input video to recognize people between the objects that are present in the scene. Different features can be considered to address this problem.

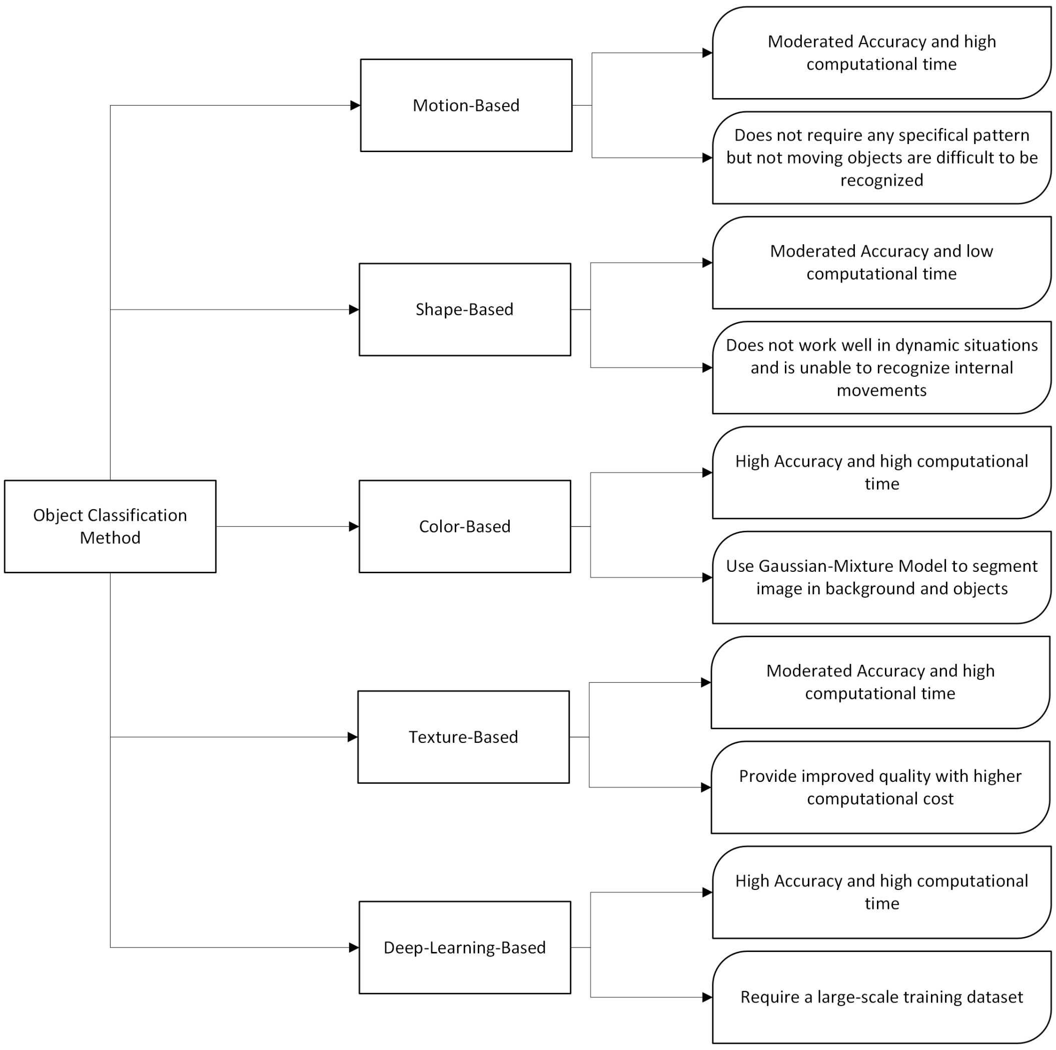

Object Classification Method
Motion-Based
Moderated Accuracy and high computational time
Does not require any specifical pattern but not moving objects are difficult to be recognized
Shape-Based
Moderated Accuracy and low computational time
Does not work well in dynamic situations and is unable to recognize internal movements
Color-Based
High Accuracy and high computational time
Use Gaussian-Mixture Model to segment image in background and objects
Texture-Based
Moderated Accuracy and high computational time
Provide improved quality with higher computational cost
Deep-Learning-Based
High Accuracy and high computational time
Require a large-scale training dataset

Figure 2.2: Algorithms of Object Classification

In [15], authors assume that the appearance of an object can be identified starting from the local intensity gradient. The overall image is divided into smaller regions called "cells". For each cell, the algorithm computes the direction of gradient, and creates a histogram of the orientations for each region. The histograms are then normalized for better invariance to lighting conditions. The normalized histograms are called HOGs and they constitute the feature vector of the image.

In [16], authors utilize low-level features i.e., gradient responses in different directions, which are averaged to obtain a mid-level feature more informative. Authors call features of this type shapelets since they are considered an accurate description of the shape of an object.

In [17] three different kinds of features are used:

- Two-rectangle feature: the difference between the sum of the pixels in two adjacent regions (Figure 2.3 A-B).
- Three-rectangle feature: the sum of the pixels within two outside rectangles subtracted from the sum in a center rectangle (Figure 2.3 C).
- Four-rectangle feature: the difference between the sum of pixels in one diagonal pair of rectangles and the sum of pixels in the other diagonal pair (Figure 2.3 D).

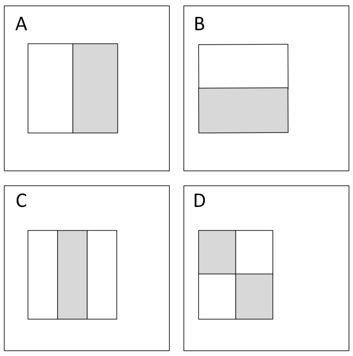


Figure 2.3: Rectangular features

These types of features are chosen in order to obtain a computational efficiency as high as possible.

After features extraction a classifier utilizes the extracted data to categorize the objects in the scene between two classes, the "Person" class or the "Not Person" one. The mainly used classifier for this approach are:

- **SVM**: This is a linear classification algorithm, which classifies points according to where they are with respect to an hyperplane.
- **Random Forest**: the Decision Tree (*DT*) is a machine learning model characterized by a tree-like structure. It consists of nodes and edges, each of which symbolizes a potential decision and its possible outcomes. Multiple *DTs* can be aggregated into a Random Forest classifier. In this case, each singular tree is trained on a random subset of the training data and features. The final prediction is based on the majority of the output of the individual trees.

The performance of the classifier, however, is extremely scenario-dependent: the proposed architecture works quite well for sparse crowds but in a more crowded scenario, where occlusions are more likely to occur, the results achieved are not good enough to justify their extremely high computational cost.

Trajectory clustering-based approaches

These types of approaches relies on the assumption that motion features of the same object, i.e., direction or velocity of the motion, are relatively uniform over time. Different trajectories, characterized by similar features can be grouped to represent different moving entities. This methodology has been adopted in [18] where authors modify the *KLT* *tracker* algorithm [19] to tackle the problem of occlusions in human crowds. The KLT tracker is an Algorithm that aims to find the motion parameter of an object from image I to a consecutive image J (Figure 2.4 - 1). Computed parameters are then utilized in the clustering phase to identify how many moving objects are present.

The main problem of its standard version is related to occlusion in the crowd: the algorithm is not able to track the features continuously. In case of occlusion, new trajectories' starting points need to be recreated. In the newly implemented version, features are recreated at fixed time locations and trajectories are propagated forward and backward in time in case of occlusions (Figure 2.4 - 2). These modifications increase the computational efficiency and obtain trajectories that are longer and smoother than the ones originally obtained with *KLT tracker*.

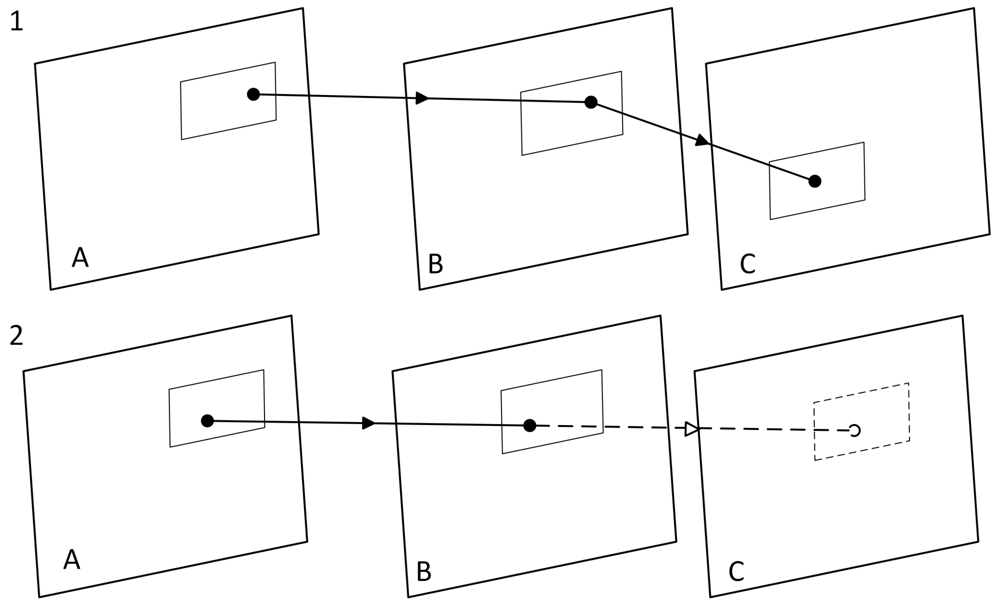


Figure 2.4: Features' propagation for KLT tracker

All the found trajectories are then clustered together to identify similar patterns in extracted data. Additional information can further reduce the complexity of the problem i.e., two trajectories too distant cannot be considered as part of the same object if the *bounding box* of the object is known. The number of clusters at the end of the process coincides with the number of people in the crowd. The main problem of this strategy is the assumption of motion coherency over time and space: false estimation may arise in case of static people in the scene or in case of similar motion between two objects [20].

Feature-based regression approaches

Instead of isolating singular individuals and continuously tracking specific motion parameters, Feature-based regression approaches use a collective description of the crowd. This collective approach helps simplify computational processes and allows for a reduced resource demand. The first application of this system can be found in [21] but the general pipeline remained approximately the same over the years (Figure 2.5).

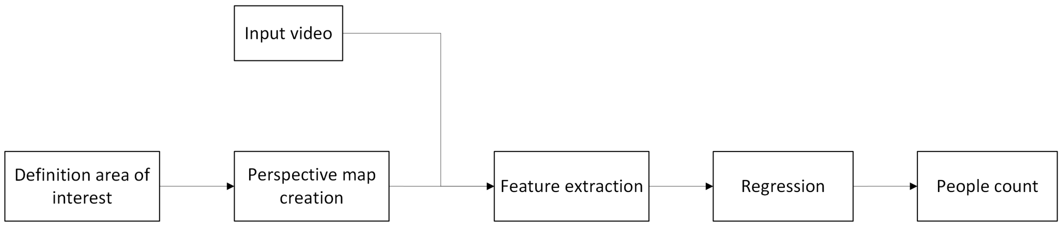


Figure 2.5: Feature Regression Pipeline

After background subtraction, different highly informative parameters can be extracted from the foreground segment of the input image. Five key features are highlighted in [11]:

- Size: Extension of the area of interest i.e., the pixel count used in [21] or the density map introduced in [22].
- Shape: Length of the perimeter and orientation of the area of interest.
- Edge: the change of intensities over the pixel of the image.
- Texture: contrast and homogeneity of the image.
- Key-Points: any other possible point of interest i.e., corners and angles [23].

Features can be extracted either from the entire region of interest, in this case, we talk about holistic regression, or from individual regions of the image, referred to as the local regression approach. Histogram-based approaches can be used too: local information is accumulated in histogram bins to construct a global feature.

The features extracted are given as input to a regression model which gives as output the people count. Many studies used different types of models over the years, a review of the used models is given in Table 2.1. However, all these strategies show poor performances in the case of crowd images with high density and scale variation effects in the images making the accuracy even lower.

Implemented in:	**Regression model:**
Crowd monitoring using image processing [21]	Linear Regression
Counting people with low-level features and Bayesian regression [24]	Gaussian processes Regression
Bayesian Poisson regression for crowd counting [25]	Bayesian Poisson regression
Flow mosaicking: Real-time pedestrian counting without scene-specific learning [26]	Polynomial regression
Feature mining for localised crowd counting [27]	Kernel ridge regression

Table 2.1: Implemented regression model

CNN for crowd-counting

A neural network is a machine learning model which is inspired by the functioning of the human brain. Exactly as its biological counter part, it is composed of interconnected artificial neurons organized in different layers. Each layer can communicate with the previous and the following ones, and this feed-forward process enables the network to learn from complex data to make predictions or decisions. In recent years, researchers focused on Convolutional Neural Network (CNN) for crowd-counting applications. The interest for this technology is mainly due to its capability of automatically extracting the most suited features to use in the counting phase [28]. A Convolutional Neural Network is used to process data with a grid-like topology, i.e., images, and its general structure is shown in Figure 2.6.

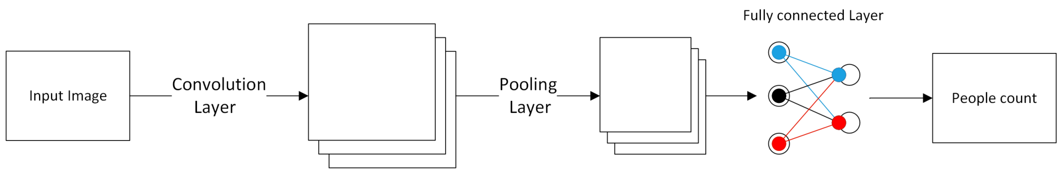


Figure 2.6: CNN general architecture

The image is given as input to the convolution layer which is the core block of the overall architecture. It contains a set of kernels whose parameters are computed during the training phase. Kernel are small matrices applied to input data for feature extraction. Each kernel slides over the image and, for each possible input section, dot product is calculated. This process is called *Convolution* and its output is called *Activation map*. The final output of the layer is composed of all the activation maps generated through this process. The activation maps are then given as input to the pooling layers.

Pooling Layers simplify received data performing an aggregation operation to reduce their size i.e., taking the maximum or average value of each input window [29, 30].

The number of layers inside a CNN defines its depth. Typically the first layers detect the low-level features of an image, i.e., lines, while the latest layers detect more informative high-level features i.e., shape and objects. The output layers are then responsible for mapping these highly informative features to the desired output [31].

These systems showed their high potential starting from their first implementation in [32]: a CNN can extract automatically all the features of interest, giving the most suited weight to each of them. It is possible to introduce in the training phase different images, i.e., trees and buildings, to reduce the false positive rate and it is possible to counteract some of the major problems of the classical approach i.e., scale variations and occlusion [33]. However, despite the very high performances achieved in Crowd Counting by this approach, CNN cannot be used in many situations due to the extremely high computation times and the requirements of a very large training dataset.

Video Surveillance consideration and drawbacks

Despite the high accuracy that these systems can achieve, they are all limited by some problems related to the system's architecture itself: i.e., scale variation effect and non-uniform people distribution reduce the effectiveness of these algorithms; the monitored area is quite small and, in case of multi-camera systems, processing the data become more computationally expensive due to the data-integration problem [34]. The considered scenario may introduce some problems too i.e., in an outdoor context changing lighting conditions must be taken into account, and further limitations are introduced by the interactions between the objects in the scene i.e., occlusions [5, 28, 35]. However, while some of these effects may be counteracted, as done in [36] with the scale variation effected, many other problems cannot be overcome at the moment: the large amount of data to be transmitted requires an extremely high transmission rate and the hardware must be able to maintain the computational time sufficiently low to avoid excessive delays. Furthermore, collected images and video could contain many sensitive information which can lead to privacy issues too [37]. All these reasons make these strategies prohibitive in many real-life applications.

2.2 Sensor-based Crowd-monitoring techniques

Over the years, several other solutions have been proposed in order to address the problems afflicting systems described in Section 2.1. Rather than counting directly people in the scene, it may be convenient to use some indirect information coming from other types of sensors. In this way, it is possible to estimate the crowd size without necessarily knowing any information classified as sensitive by the European Union [4, 38].

2.2.1 Infrared sensors

Infrared sensors are able to detect and measure the infrared radiation emitted or reflected by nearby objects. Their functioning depends on the specific type which is considered.

We can find two types of infrared sensors:

- Active Infrared Sensors: they can emit infrared radiation which will be reflected by obstacles and collected back. The sensor can then estimate both the presence of a nearby object and its distance.

- Passive Infrared Sensors (*PIRs*): they are extensively used in security. PIRs do not emit any radiation but they will receive the ones emitted by nearby objects.

In crowd-monitoring applications, passive sensors are commonly employed. These sensors can be integrated in camera-based systems to give extra data to be used in processing.

However, many researches have attempted to build a PIR-based monitoring system, characterized by a low-cost architecture and very low power consumption.

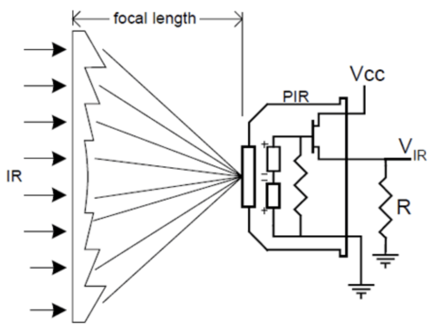
Passive Infrared Sensor in Crowd counting

Passive infrared sensors (*PIRs*) are used for their capability to monitor a quite spread area, without capturing any sensitive information. Their functioning is based on the fact that any object emits infrared radiation, the higher its temperature the higher the radiation which is emitted. *PIRs* are constructed with pyroelectric sensors that can detect radiations emitted by a nearby object. When a person passes in front of a background, the detected infrared radiation detected increases instantly. If the radiation change exceeds a certain threshold, an output is generated.

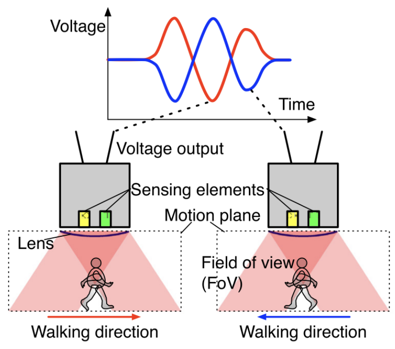
These types of sensors are typically very cheap and their power consumption is very low. Furthermore, their structure can be modified in order to detect not only presence but motion too. A PIR-based motion detector is divided in two halves. The two parts are interconnected and, if one of them detects a higher infrared radiation than the other, an output is generated based on the sense of motion. An example of the structure of a motion detector is shown in Figure 2.7 along with an example of its functioning.

The output can be a binary value, i.e., 1 if motion is detected 0 otherwise, or a signal, in this case the sensor is able to provide more detailed information about the shape, the dimension of the object, and the motion characteristics.

In general PIR sensors are more suited for human detection rather than people counting; their characteristics, however, make them appetizing for this type of applications too.



a) Building blocks of a PIR sensors, reproduced from [39]



b) functioning of a PIR sensor. Reproduced from [40]

Figure 2.7

A very simple people-counting implementation can be seen in [41], where two binary *PIRs* sensors are applied on both the side of a door: if one detects motion, the system check whether motion is detected by the other one within a limited time window. If so, the count of people in the room is adjusted on the basis of the sense of motion i.e., if the person detected is entering or exiting. For this approach, however, the limitations are quite intuitive: the system does not address the case of multiple people entering in the room at the same time.

This problem was solved by [42]. Here, two shelves are appended on the ceiling on each side of the door. Each shelf supports a system composed by 8 *PIRs* and an ultrasonic sensor able to trigger the *PIR* array. A Convolutional Neural Network then uses the data collected by the system to estimate the number of people entering the room. The results in this case where quite good, with an accuracy of around 90%.

Some attempt were done to apply PIR-based systems in a more spread scenario, using a whole room as location. These applications have shown multiple limitations: in [43] for example authors used an array of thermal sensor to build the heat map of a room and to detect number and position of the people in the monitored area. The map displays areas with higher temperatures (ideally corresponding to the presence of people) compared to cooler areas (floor or objects). The average accuracy of the proposed system was very low, around 41%.

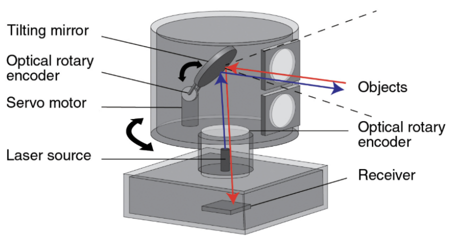
In [44] a system of PIRs array was built to detect the number of people in the room and their direction of movement. The results were quite good but in the considered scenario there were only 4 people in the room. The achieved results are not representative for overcrowded situations as the accuracy is expected to decrease as the number of people increase.

Furthermore, the application in outdoor environments is unfeasible due to the short transmission range of PIR sensors and due to the disturbances introduced by sunlight and heating.

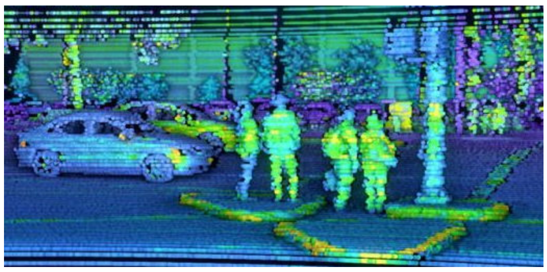
2.2.2 Laser scanner in crowd counting

A laser scanner is a device able to emit a laser beam whose reflection is then collected back, as shown in Figure 2.8. Through the beams' *time-of-flight* the sensor is able to estimate the distance of the objects in the scene, which are then converted in a set of *(x,y,z)* coordinates. The collection of point so obtained is called cloud-point.

The coordinates of all the points can be used to reproduce a 3D representation of the whole environment. An example of a LiDar system and the produced output are the one shown in Figure 2.8.



a) LiDar scanner's scheme. Reproduced from [45]



b) Environment reconstruction through a LiDar scanner. Reproduced from [46]

Figure 2.8

The obtained reconstruction can provide more information with respect the classical image-based systems, without being subject to any prospective distortion. An approach which uses this architecture, is able to count the people in the area just by counting the occupied spaces in the reconstructed image [47]. It is also possible to apply object classification algorithms on the images to recognize the human beings in the scene: as done by [48] which has shown the effectiveness of SVM algorithms in recognizing humans.

In these type of systems there is no collection of sensitive data and they are insensitive to lighting condition. Pre-Processing of the data is less computational expensive with respect to camera-based systems and very high accuracy can be achieved. The main limitation of these systems is due to the hardware and maintenance costs. Very high costs make these systems not adequate in many real-life application.

A less expensive approach is using a 2D-Lidar, as done in [49, 50], where the point cloud consists of a 2D-image. The architecture required in this case is less expensive with respect to the standard one, and learning the activity pattern is much easier with respect to the 3D-case, yet depth information are lost. Performances achieved by 2D-Lidar are then comparable with the one of classical camera-based systems.

2.3 Network-based Crowd-monitoring techniques

The widespread presence of mobile devices, constantly connected to wireless or mobile networks, suggests the possibility of using information coming from these sources for crowd-monitoring applications. The basic concept is that mobile devices continuously send and receive signals to and from nearby base stations or wireless routers. These signals can be potentially used to infer the crowd-size and other information related to presence and movement of present people.

An approach of this type offers several advantages. Firstly, it is a cost-effective implementation as it does not require additional infrastructure. Secondly, no user actions are required for participation in monitoring; information is passively collected. Finally, these techniques provide extensive coverage, as most people own and use mobile devices, making possible the monitoring of large urban areas or mass events.

2.3.1 Bluetooth-based Crowd-counting

Bluetooth is a short-range communication technology used to facilitate data exchange between devices. Bluetooth creates a Personal Area Network (*PAN*) known as a *Piconet*, which is composed by one master device and up to seven slave devices. The master has the role of managing the communication and synchronization among the devices, making the data-exchange possible.

The starting of a Bluetooth-based communication between devices is composed of two phases. The first phase is called inquiring, where a Bluetooth device actively searches for some nearby potential slave node. The second one is called paging. In this part of the connection process, the paging device establishes a connection with the target device. Once the connection is established, the devices can begin exchanging data [51].

During connection process, many sensitive information are exchanged between master and slaves i.e., the name of the device and the Bluetooth MAC Address, a globally unique identifier of the device.

Information of this type could be used to uniquely identify nearby devices, making possible the construction of a Bluetooth-based crowd-monitoring system.

One of the first systems adopting this technology in crowd-monitoring has been implemented in [52]. In this system a Bluetooth scanner has been applied on the ceiling of a bus. The scanner periodically sends inquiry messages, searching for Bluetooth devices in discoverable mode within the area of interest. According to the standard, a device must then respond with a inquiry response, containing the globally unique identifier of the device. The content of the inquiry responses are collected and correlated with the information about the bus route. The correlation is performed off-line and gives as output the origin/destination matrix of each discovered device.

A further attempt of implementation of Bluetooth-based crowd-monitoring has been done in [53]. During a religious event, where 1.5 million people attended, a set of Bluetooth scanners has been employed to monitor the crowd. Scanners captured a series of inquiring responses during the festival period, using the same approach previously discussed. It has been possible to reconstruct the general behavior of the crowd i.e., the more crowded areas and the general direction of movement, by studying the data collected from the scanners.

This technology has shown interesting results for crowd-monitoring applications, yet it does not perform well for what concerns crowd-counting: counting accuracy depends on the number of devices in discoverable mode, whose percentage is quite low, ranging between 7% and 11% [54].

Furthermore the short range of Bluetooth communication (around 10 m for common devices) introduces scalability challenges to monitor more wide areas.

2.3.2 WiFi-Based Crowd-counting

A standard WiFi network is composed of an Access Point (*AP*) and some stations connected to it. The AP acts as a gateway between the connected devices and a wired network, allowing the exchange of information between them. In order to be associated to an AP, devices can adopt two ways: a passive discovery mechanism and an active one.

Devices adopting a passive discovery mechanism wait on each channel of the spectrum, waiting for a Beacon message from the *AP*. If no Beacon frame is received within a certain time-window, devices can switch to another channel repeating the process once again. This methodology, however, is not efficient as it requires the device being passively listening on each possible channel. This is why active scanning mechanisms has been introduced to reduce the time required for connection.

During an active scanning phase, devices send a particular frame, called *probe request*. Probe requests are broadcasted to all the nearby devices and, if an *AP* is present, it responds with a Probe response message. A device employing active scanning, periodically sends probe requests in group of variable length. Each of these groups is called "burst" [55].

Probe request structure

Probe requests are IEEE 802.11 WiFi management frames. Their structure is specified by [56] and it is shown in Figure 2.9.

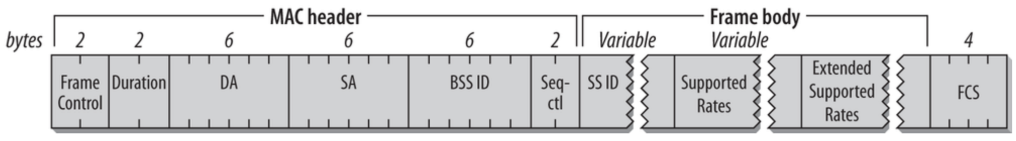


Figure 2.9: Basic structure of a Probe request frame. Reproduced from [57]

Probe requests are composed of two sub-fields: the MAC header and the frame body.

The MAC header contains many information which are essential for the correct addressing and management of the frame i.e., destination address, source address, length of the message. The fields composing the *Frame body* give some additional information about settings and characteristics of the device.

In literature many researches focus their attention on *PRs* and their content, highlighting the possibility of using these information for crowd-monitoring applications. An overview of PR-based crowd-monitoring is given in Section 2.4 along with a more detailed description of the used fields.

2.4 Probe Request-based People Counting

2.4.1 MAC address-based algorithms

One of the most important information which can be obtained from probe requests is the MAC address (Media Access Control address). It is contained in the MAC header and it is a globally unique identifier assigned for communication in a local area network. The structure of the MAC address can be seen in Figure 2.10.

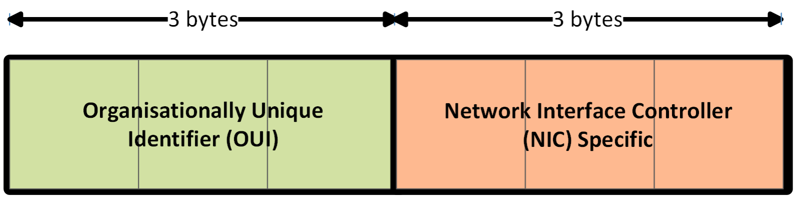

3 bytes
3 bytes
Organisationally Unique Identifier (OUI)
Network Interface Controller (NIC) Specific

Figure 2.10: Structure of a MAC address

The first 3 bytes of the MAC address represent the Organizationally Unique Identifier (*OUI*). This part identifies the manufacturer or organization that has produced the network device. These identifiers are assigned by IEEE to manufacturers, ensuring uniqueness of each of them.

The NIC-specific subfield is used to uniquely identify the network interface card and it is assigned by the manufacturer itself.

Given the uniqueness property of MAC-address crowd-counting is trivial: the number of devices present in the area is equal to the number of different captured addresses. An example of code for counting devices is presented in Algorithm 1.

Algorithm 1 Algorithm to compute crowd-size starting from MAC address

```
Require: .pcap file with the capture
Ensure: $Number\_of\_devices$ Number of the devices in area
1:  $MAC\_list \leftarrow []$
2:  for $packet$ in $capture$ do
3:      $src \leftarrow MAC\_address(packet)$        ▷ Get the packet source address
4:      if $src$ in $MAC\_list$ then
5:          continue
6:      else
7:          $MAC\_list \leftarrow insert(src)$
8:      end if
9:  end for
10: $Number\_of\_devices \leftarrow length(MAC\_list)$
```

In Line 1 an empty list is initialized. It is used to store the unique MAC addresses seen within the area. Each packet of the capture is then analysed extracting the MAC address of the packet's source. If the $MAC\_address$ is already appeared in the area, the algorithm analyses the next packet. If the MAC address has never been seen before, it is saved in the MAC_list. Finally, once all packets have been processed, the algorithm determines the number of unique devices in Line 10 by counting the length of the MAC_list.

However, as many researches have evidenced, the transmission in clear of the packet's source address introduces a lot of privacy issues. It could be possible to track any movement of any device by just searching for its MAC address [58]. For these reasons the MAC address has been classified by [4, 38] as a sensitive information and many smart device vendors started to implement a MAC address randomization mechanism to protect user's privacy.

With the introduction of MAC address randomization, the transmitted source address is no more a global unique address, but it is substituted with a locally administrated one. A locally administrated MAC address can be distinguished by a globally unique one by watching the seventh bit of the first byte. If the bit is set to 1, the MAC address is a randomized MAC address, which cannot be related to user's identity.

The first implementation of MAC address randomization was done in 2014 by Apple iOS 8 and, since then, it has been implemented by more and more manufactures [59]. For these reasons, an approach like Algorithm 1 cannot be used for crowd-monitoring as the same device may change its MAC address every few probe requests [55].

2.4.2 Information elements-based algorithm

Information elements (*IEs*) are fields optionally located in the frame body of a probe request. They encapsulate various pieces of information that are crucial for the communication between wireless devices and networks.

The fields' structure is specified by [56] and each *IEs* plays a particular role in conversation. The structure of a generic information element is reproduced in 2.11. They are composed of a type identifier, length, and data.

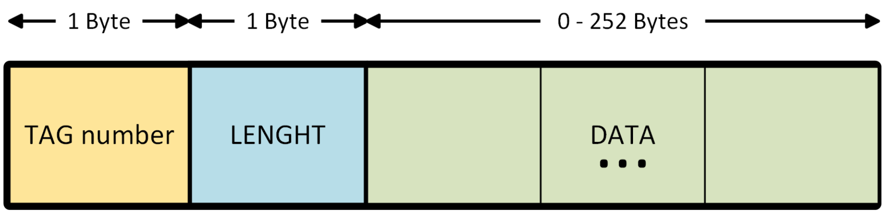


Figure 2.11: Structure of an Information Element field

The TAG number uniquely identifies the type of contained information. Each *IE* is associated with a specific TAG number; for example, the TAG number of the SSID field is 0. The length sub-field indicates the length in bytes of the data in the data sub-fields, which contain the transmitted information.

Some types of *IEs* that can be found in Probe Requests are:

- **SSID**: It contains the list of preferred network of the source device. This could be a specific SSID, in this case we talk about directed probe request, or a wildcard SSID to query all nearby networks.
- **Supported rate and extended supported rates**: These fields provide a list of the data rates which are supported by the device. With this information, the network is able to check whether it can satisfy client's data rates requirements.
- **HT Capabilities (High Throughput Capabilities)**: Informs the network if the client can support higher data rates and advanced features like MIMO (Multiple Input Multiple Output).

- **VHT Capabilities (Very High Throughput Capabilities)**: It allows the network to verify whether the client supports IEEE 802.11ac features.

- **Extended Capabilities**: It informs nearby networks about additional features which are not included in other fields.

- **Vendor Specific**: This field is used by vendors to insert additional information not defined in the IEEE standard. Information here contained can be used by vendors to enhance performances, and ensure better interoperability within their ecosystems.

Since transmitted data depends on the particular devices sending them, many researches defined these fields as a device-dependent signatures [60–62].

Both [61] and [62], in particular, have shown that data carried by these fields remain stable over time for the same devices, and some of them are contained in many of the detected probe requests.

In [63], authors create a crowd-counting system, named as BLEACH, based on IE signatures. They used a public dataset of probe requests [64], to train a logistic regression model. The trained model uses the information elements and the time-behavior of the bursts as input to compute the probability of different probe requests of being sent by the same devices. However, as authors of [65] highlighted, the dataset used for the training is quite old, being recorded in 2013. It does not take into account many modern devices characteristics, i.e., MAC address randomization. Furthermore a supervised learning approach, like the logistic regression is not flexible enough to take in consideration future changes in probe requests structure.

Another crowd-monitoring technique, named iABACUS, has been developed by authors of [66]. Also in this case, the aim of the algorithm was to identify whether two probe requests can be sent by the same device. In the developed algorithm, authors have employed the information elements received as a check i.e., only probe requests with identical or highly similar capabilities can potentially be associated. In this way the number of possible association is greatly reduced. The algorithm uses then information coming from other fields to verify the likelihood of two probes of being sent by the same source. This additional information comes from the sequence number field (SN). Originally sequence numbers were used to reassemble fragments of a MAC frame. They were assigned by a counter variable, incremented by one whenever a frame is sent out. Due to the monotonically increasing nature of this implementation, sequence numbers may be used to easily detect messages coming from the same device, even with MAC address randomization.

iABACUS considers any possible couple of probe requests, computing a similarity score for each of them. The score is inversely proportional to the difference between the sequence numbers between the two probe requests. The more the two *SNs* are similar the higher the score, hence they are more likely to belong to the same device. The algorithm examines all the possible combinations of probe requests meanwhile inserting in the same list probes with higher relation scores. However, the approach used by this algorithm is quite out-dated: it worked quite well in in its first implementations (accuracy of over 90% in dynamic case) but nowadays it is unfeasible.

Many researches have evidenced that monotonically increasing sequence numbers can lead to privacy issues as they make easier the recognition of a particular device [62,67]. For this, nowadays many vendors does not implement a monotonically increasing sequence numbers' series, adopting a more random approach instead. More in details, authors of [55] have noticed that, for each burst of probe requests, the sequence number of the first one is chosen randomly, while others are obtained increasing the counter variable by one or two from the previous *SN*.

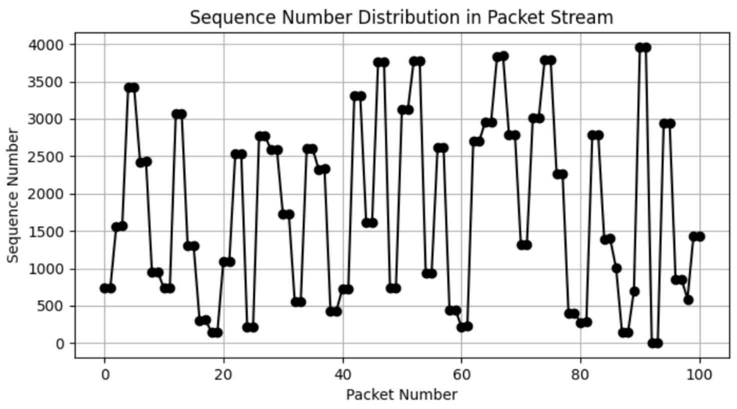


Figure 2.12: Sequence numbers of consecutive PRs sent by a Samsung Note 20

To gain better insights into the behavior of *SNs*, we analyzed one of the *.pcap* files in the dataset provided by [55]. We considered a Samsung Note 20 we extrapolated the sequence number associated to each probe request. The obtained numbers are shown in Figure 2.12 . Here, it is clear that the sequence has lost its monotonically increasing characteristic, showing a more random behavior. The choice of using sequence number as a discrimination element is then not applicable anymore.

Chapter 3

Crowd-Monitoring Framework

3.1 TrialsNet Project

TrialsNet is a European project that aspires to achieve a range of technical, performance, and productivity goals that will have a significant impact on the existing 5G network ecosystem. The objectives include trialing 6G applications, improving B5G networks to support these applications, introducing societal benefits, large-scale deployment of B5G networks, and achieving industrial and scientific impacts [68].

To achieve these goals, TrialsNet is conducting large-scale trials over three urban domains in Europe: Infrastructure, Transportation, Security & Safety; eHealth & Emergency and Culture, and finally Tourism & Entertainment.

To better describe and understand the specific requirements and challenges of the projects, thirteen representative use cases are developed in four geographic clusters (Italy, Spain, Greece, and Romania), involving an expansive range of users. The use cases

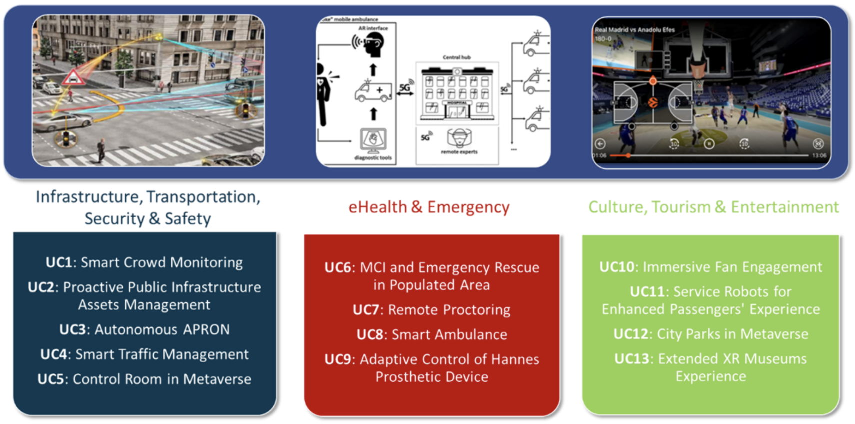

Infrastructure, Transportation, Security & Safety
UC1: Smart Crowd Monitoring
UC2: Proactive Public Infrastructure Assets Management
UC3: Autonomous APRON
UC4: Smart Traffic Management
UC5: Control Room in Metaverse
eHealth & Emergency
UC6: MCI and Emergency Rescue in Populated Area
UC7: Remote Proctoring
UC8: Smart Ambulance
UC9: Adaptive Control of Hannes Prosthetic Device
Culture, Tourism & Entertainment
UC10: Immersive Fan Engagement
UC11: Service Robots for Enhanced Passengers' Experience
UC12: City Parks in Metaverse
UC13: Extended XR Museums Experience

Figure 3.1: Use cases of Trialsnet project. Reproduced from [68]

so defined are depicted in Figure 3.1 and are currently being developed and tested in various locations and contexts, across different sectors and working conditions. This cross-functional application enables a complete and detailed evaluation of the Key Performance Indicators (*KPIs*) and Key Value Indicators (*KVIs*) of 5GB/6G applications in various contexts and situations.

3.1.1 Referenced Scenario

This thesis is developed in conjunction with the goals of the mentioned European Project, to contribute to one of its use cases. The considered use case is "UC5: Control Room in Metaverse".

The goal of *UC5* is to improve the efficacy of first responders to emergency events by using the characteristics of advanced technologies such as Extended Reality, Metaverse, and IoT. In particular, it has been observed that Italian law enforcement and emergency services have their communication systems. However, there is a lack of fast and efficient connections among the various groups. UC5 aims to address this gap as one of its primary objectives.

In this particular scenario, law enforcement agencies are provided of a virtual room within the Metaverse. Officers can access within the virtual hub from any location and, from there, they can share information in their possession, i.e., they could visualize data, maps, and other resources, and coordinate their decision. In this way, with more information at their disposal, they could enhance their awareness about any emergency while improving the decision-making process.

This thesis aims to describe and develop a sensor whose data can be easily transmitted in the virtual environment. This projected system is a crowd-monitoring framework able to estimate the size of the crowd within a monitored area. This would allow effective crowd management and planning in various contexts. A sensor used in this application have to satisfy a specific set of requirements:

- **Precision Accuracy**: To guarantee the correct evaluation of the situation by first responders, the output provided by the framework must be as exact as possible
- **Real-Time Processing**: The operations performed by the framework must require low computational time as they must provide information exactly when it is needed.
- **Privacy Compliance**: Confidentiality of personal data must be guaranteed, according to the privacy regulations.
- **Cost-Effectiveness**: To monitor an extensive region, multiple sensors need to be employed; to ensure appropriate deployment and maintenance, the system ideally

ought to be affordable in terms of cost.

- **Energy Efficiency**: A low power consumption is required to reduce expenses related to device management and power supply.

3.2 Proposed Framework

3.2.1 Hardware

One of the key aims of the proposed framework is to minimize hardware expenses. The solution that proved to be the most suitable for the purpose is the one implemented in [69].

The application uses a Raspberry Pi 4B as hardware. This solution has been chosen for its affordable price and relatively high-quality components. This device is a Single-Board Computer (*SBC*), i.e., a microcomputer built on a single circuit board, that has all the necessary components for its proper functioning. Besides the initial setup, shown in Figure 3.2, other components must be integrated to guarantee that the system functions correctly.

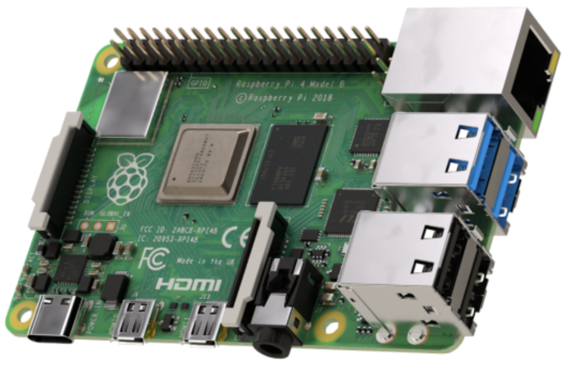


Figure 3.2: A Raspberry Pi 4B. Reproduced from [69]

To capture the probe request messages, the Raspberry Pi must be set in "Monitor mode", an operational mode that enables to monitor the wireless traffic within a certain range. "Monitor mode" is not supported by the built-in WiFi interface of the Raspberry Pi 4B and, for this, the implementation of a WiFi USB dongle is required to collect all the necessary data.

In addition, a LTE modem is necessary for post-implementation accessibility, to verify if the system is functioning properly, and to address any potential issue. The modem enables the system to connect to 3G and 4G/LTE networks using a SIM card. Nevertheless, the LTE modem is not sufficient to guarantee post-implementation access to the Raspberry Pi (RP) as it lacks of a static IP address. To allow remote accessing, a tunnel SSH is established between the RP and an external server with a static IP address. In this way, it is possible to connect remotely to the Raspberry Pi using the server as a relay.

Finally, since the final framework must be implemented outdoor, a waterproof case is used. The image of the complete hardware of the system is shown in Figure 3.3.

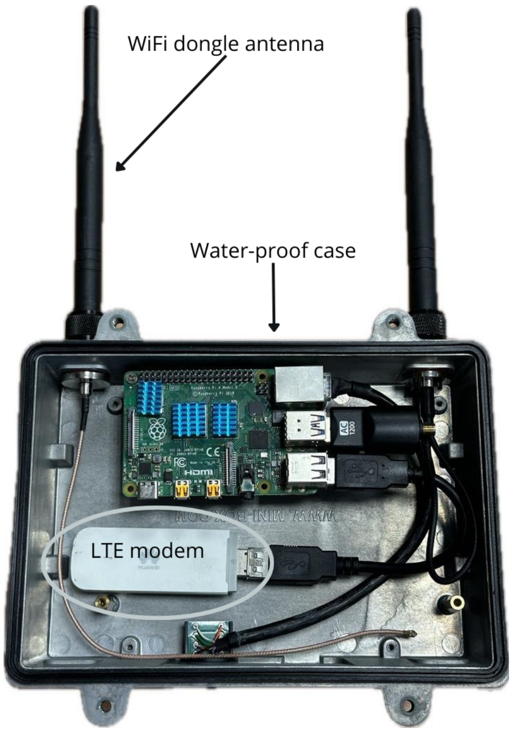


Figure 3.3: Hardware of the proposed crowd-monitoring system

3.2.2 Sniffing Pipeline

The crowd-monitoring system's overall pipeline is shown in Figure 3.4.

The system initialization begins with the configuration of the WiFi interface, executed during system boot. The initialization process is performed using a script executed via the "rc.local" file, where are specified all the operations the Raspberry must perform when it is turned on. After this process, the default WiFi interface is disabled and the new USB WiFi interface can be used in monitor mode.

After the configuration phase, the system is able to correctly receive probe request messages broadcasted by other devices. They are monitored using Tshark software [70], a

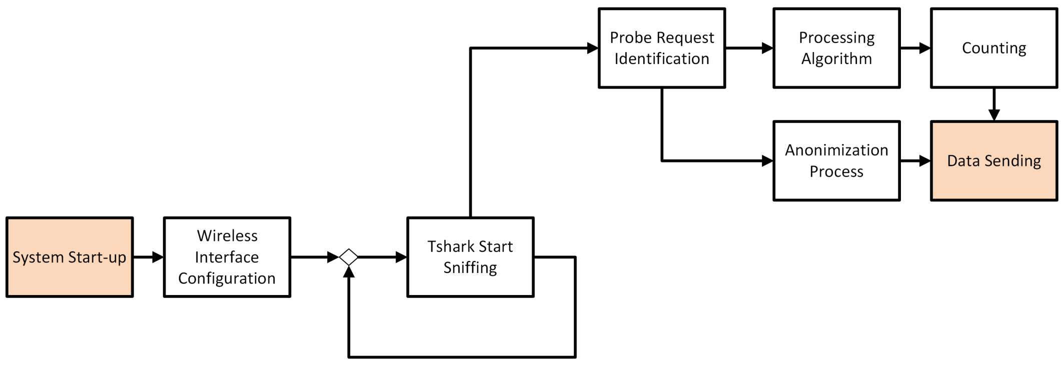


Figure 3.4: Sniffing Pipeline for Crowd-Counting

packet analyzer integrated into the Wireshark network analysis collection. Unlike Wireshark, which provides a graphical user interface (GUI) for packet analysis; Tshark functions solely through command line. For this reason, it is suited for an application like the one in our use case, where a graphical interface is unavailable. Tshark is used to capture 802.11 traffic within a time window of 4 minutes; after each capture, a new *.pcap* file is generated and sent to the next stage of the pipeline. Meanwhile, a new capture process can start once again.

When a new *.pcap* file is created, the probe requests within it are analyzed. These messages fall into two different groups, treated differently by the algorithm:

- **Probes with a globally unique MAC address**: They undergo an anonymization process as their addresses is considered sensitive information.
- **Probes with a locally-administrated MAC address**: An identifier is computed for each probe request. Identifiers are clustered together and an estimate of the number of devices in the area is computed.

Finally, the outputs of the branches are combined and the results are sent to a database using UDP protocol. The database stores all the collected data for possible future analysis. To guarantee a more manageable data visualization, we used Grafana [71] to create an interactive dashboard. It is an open-source platform able to collect the data from the database displaying them in real-time. This solution can provide many insights about system performances and facilitate the visualization of the crowd-size over time.

3.3 Device Counting

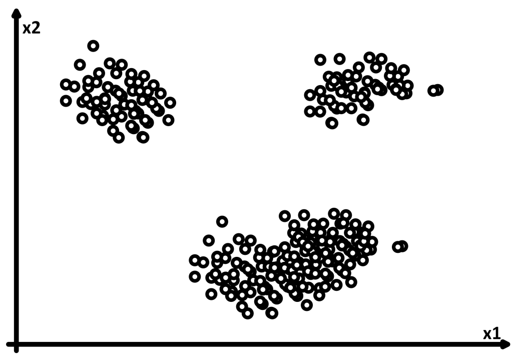
The counting process employs information from the Probe Requests to infer the number of devices in the area.

To relate the number of devices to the number of people, an initial assumption is made: each person carries a single device with them. This assumption appears to be the most suitable in a scenario like our case study. Information is obtained from the IE fields and then clustered together. The clustering approach seems particularly advantageous for a WiFi-based crowd-monitoring system as it does not require any training phase. Furthermore, since only the similarities across different frames are considered, the system can easily adapt to future modification in the structure of Probe Requests frame.

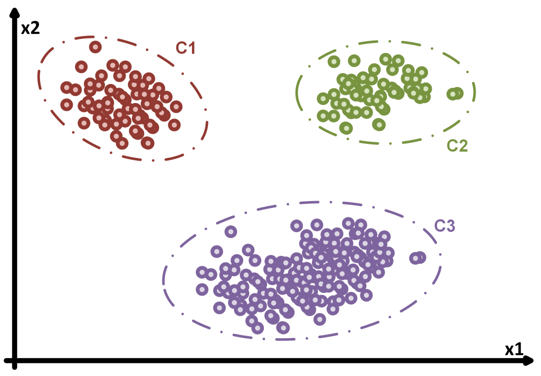
Chapter 4 contains further information about feature extraction, while the remaining part of this Section offers a comprehensive overview of the clustering algorithms used.

3.3.1 Clustering algorithms

Each object, whose state can be described by a finite-dimensional set of values, can be represented by a point in the state-space. Clustering is a type of unsupervised machine learning technique that groups the points in the state space according to how much similar they are. Each group of similar elements, formed through this process, is called *cluster*.



(a) Unprocessed data



(b) Output of the clustering process

Figure 3.5

As Figure 3.5 shows, after clustering the machine is able to recognize points with similar characteristics as being part of the same group. For data processing applications, a wide variety of clustering techniques can be employed. Nevertheless, for this framework we have selected two specific algorithms: Density-Based Spatial Clustering of Applications with Noise (*DBSCAN*) and Ordering Points to Identify the Clustering Structure (*OPTICS*). These algorithms were selected due to their ability in handling cluster of varying densities and their robustness against noise.

3.3.2 DBSCAN

DBSCAN is a widespread clustering algorithm able to group points together according to how close they are each other. Unlike conventional clustering processes, DBSCAN allows the identification of clusters of any shape without requiring the user to specify the number of clusters beforehand. Clusters are simply defined as high density regions separated by areas of lower density.

To fully understand the mechanisms at the basis of this clustering process, we need to introduce some basic definitions. DBSCAN algorithm divides points according to three different classes:

- **Core Points**: they are the points that, inside their ϵ-radius, have at least $N =$ Min_points close points; they define the dense regions in the dataset.
- **Border points**: they are the points that lie within the ϵ-radius of a core-point but, at the same time, do not have enough near points to be considered as core points. They represents the boundaries of a cluster.
- **Noise points**: they are points that cannot be classified either as border or as core points. They are then discarded as effect of noise.

Each point in the data is examined to verify if it meets the requirements to be considered as a core point. If so, it is labeled as a core point, and the cluster expansion process begins.

A point p is directly density-reachable from another point q if p is within the ϵ-radius of q and q is a core point. Additionally, p is density-reachable from q if there is a chain of points connecting them, where each point in the chain is directly density-reachable from the previous one.

The expansion implicates considering, as part of the cluster, all points reachable from the core points through density connections. The cluster expansion occurs recursively until all reachable points have been included.

After expansion, DBSCAN algorithm identifies points that do not meet the core point requirements but lie within the ε-radius of a core point. These points are classified as border points and are assigned to the cluster of the nearest core point.

One of the main problems of this algorithm is that it struggles to identify clusters of highly different densities. For this limitation, OPTICS is also employed in order to compare the results of the two different clustering algorithms.

3.3.3 OPTICS

OPTICS is an algorithm developed to overcome the limitations of DBSCAN, specifically its inability to manage datasets with clusters of variable densities. It examines how data aggregates at different density scales, permitting the observation of both dense and less dense clusters. Also in this case, it is important to have a good understanding of the key concepts of the algorithm to understand how the OPTICS algorithm works with data provided as input.

The notions of core, border, and noise points are the same as those introduced with DBSCAN. Additionally, the concepts of core distance and reachability distance must be introduced.

- **Core Distance**: Given ϵ and $N = Min\_point$; the core distance of a point is defined as the minimum distance from its N-th nearest neighbor within the ϵ-radius. Core distance is used to determine the local density around the considered point p. If p does not have at least N points within the ϵ radius, the core distance is undefined or infinite.

- **Reachability distance**: The reachability distance of a point p from a point q is the maximum between the core distance of q and the Euclidean distance between q and p.

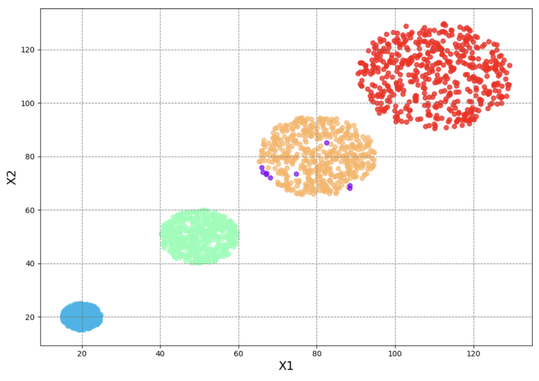
  $$\text{reachability distance}(p, q) = \max(\text{core distance}(q), \text{distance}(p, q)) \tag{3.1}$$

  It measures how reachable a point is from another densely populated region.

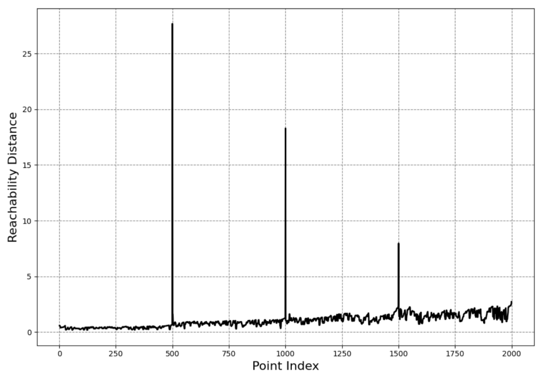
The OPTICS algorithm starts by selecting a randomly chosen point from the dataset as the starting point. This point is classified as a "core point" and its reachability distance is set to zero. After that, the algorithm examines all the points within its ϵ-radius. Each neighbor that can be classified as a core point is called a "directly-reachable point". All directly reachable points are organized into a data structure on the basis of their reachability distances. In the next steps of the process the algorithm examines all the points in the data structure according to the same procedure, updating the reachability distances stored in the data structure.

This process continues up to the moment in which all points in the dataset have been examined, and the complete reachability structure is built. The final structure encapsulates all the information by recording the reachability distances and the order in which points are processed. Figure 3.6 shows the results of a clustering process executed through OPTICS.

To extract clusters, the reachability plot can be examined. A reachability plot, shown in Figure 3.6-B, is a graphical representation of the data structure computed during the



(a) OPTICS -Cluster identification



(b) OPTICS - Reachability plot

Figure 3.6

execution of the algorithm. In this graph, the x-axis represents the indices of all the processed points, while the y-axis indicates their reachability distances. As can be clearly seen, the plot is characterized by peaks and valleys. Valleys identifies the clusters. In valleys, points have low reachability distances, indicating that they lie within a densely populated area.

3.4 Anonymization Process for MAC addresses

3.4.1 Bloom-Filter

A Bloom filter is a compact data structure that offers an effective way to store a wide set of elements avoiding excessive wastes in terms of memory [72]. The key to this efficiency lies in how elements are inserted into the filter.

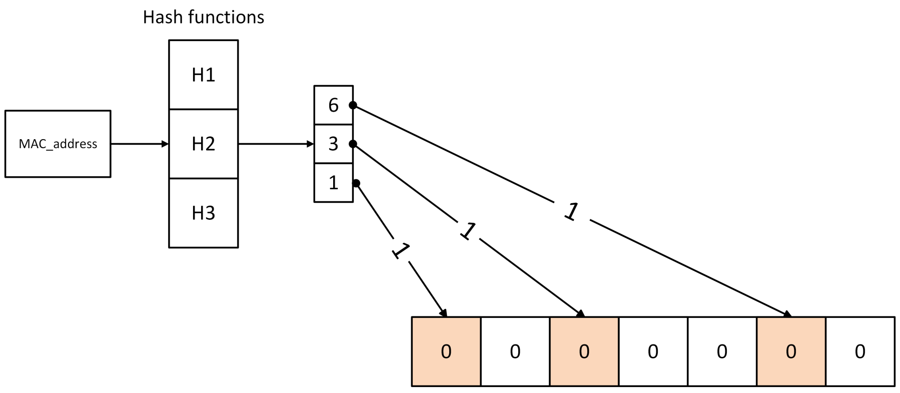


Figure 3.7: Insertion of a new element in a Bloom Filter

The Bloom filter, denoted as $BF \in \{0,1\}^m$, is composed of a m-length array of bits and k hash functions. When a new element x_j has to be stored into the Bloom filter, each hash function maps it to an element i in the range $\{1, \ldots, m\}$ [73] and the bit at position i is then set to 1. Through this insertion process, displayed in Figure 3.7, each m-lenght element is converted in an element of smaller size.

One of the main drawbacks of this strategy is that any element added to the filter loses its individuality, because it exists only as a component of the set. Nonetheless, a trivial validation test can be used to determine whether an element is present in the set or not. The element under examination is s processed by means of the hash functions composing the Bloom Filter and the resulting bit positions are then checked out.

- If even a single bit in the computed position is set 0 the element is definitely absent in the Bloom filter.
- Otherwise, the element is presumed to be present.

The uncertainty in the latter case is a consequence of the compacteness of representation of the stored elements. Hash collisions or overlapping positions among different elements may cause an element to be erroneously indicated as present, hence producing to a false positive result. The likelihood of obtaining a false positive can be estimated through (3.2), derived from [74].

$$p \approx \left(1 - e^{-\frac{n \cdot k}{m}}\right)^k \tag{3.2}$$

In this case n represents the number of element inserted within the Bloom Filter, and as it increases the false positive rate increases too, according to the Equation above. For this reason the Bloom Filter must be reset when it is near to saturation. Furthermore, the length of the bit array and the number of hash functions are critical factors able to heavily affect the performance of Bloom filters. Correctly choosing their values plays a significant role in balancing the trade-off between space efficiency and detection accuracy. To manage this trade-off some considerations must be made [75]:

- m must be big enough to store the desired number of elements. For this application $m = 10000$ has been chosen.
- Considering a capture window of 4 minutes and the environment under consideration, we do not expect more than 1000 devices per capture window. Therefore, we set $n = 1000$ accordingly.
- Small values of k leads to a higher number of $0s$ in the BF. More elements can then be stored inside the array.

- Higher values of k increase the probability of finding at least one 0 bit for an element not present in the filter, reducing *False positive rate.*

Given these considerations, the optimal value of k can then be computed via (3.3).

$$k_{\text{opt}} = \frac{n}{m}\log(2) \approx 7 \tag{3.3}$$

3.4.2 Bloom Filters for privacy protection

While in many applications the inherent presence of errors in querying is considered as a drawback, authors of [76] have shown that this characteristic can be exploited to address privacy-issues.

Before describing the role of Bloom Filter in privacy protection some key concepts must be introduced.

- A "Hiding Set" for a Bloom filter $BF(S)$ is a set V that contains all the elements v_i that are not stored in the filter, but when queried they return a false positive result.

- An object $x \in S$, stored in $BF(S)$, is said to be deniable if it can be confused with an element v_i part of the *Hiding set.* In particular it must exist an element $v_i \in V$ such that:
$$\forall j \in \{1 \dots k\}, \exists z \in \{1 \dots k\} \text{ s.t. } H_j(x) = H_z(v).$$

Starting from these definitions a Bloom Filter is γ-deniable if a randomly chosen element inserted is deniable with a probability of γ [76, 77]. In particular, using γ equal to 1 means that the presence of any element inserted in the Bloom Filter cannot be sure due to the presence of possible false positives [3].

This solution implies that the anonymity of sensitive information is preserved, satisfying the requirements imposed by privacy regulations [4, 38].

γ can be estimated through (3.4), where m is the number of bit in the filter, n is the number of elements inserted within, k is the number of hash functions and h is the cardinality of the hiding set.

$$\gamma(\text{BF}(S)) = \left(1 - \exp\left(-\frac{hk}{m\left(1 - e^{-\frac{nk}{m}}\right)}\right)\right)^k \tag{3.4}$$

In particular, for this application values of $m = 10000$ and $k = 7$ have been chosen and h is estimated through (3.5).

$$\begin{cases} h = (|U| - n)\left(1 - e^{-\frac{nk}{m}}\right)^k \\ |U| \approx 2^{48} \end{cases} \tag{3.5}$$

Starting from (3.4) and the application values previously discussed the value of n required to obtain $\gamma = 1$ computed and, as can be seen from the plot represented in Figure 3.8, it is obtained for $n_{\min} = 30$.

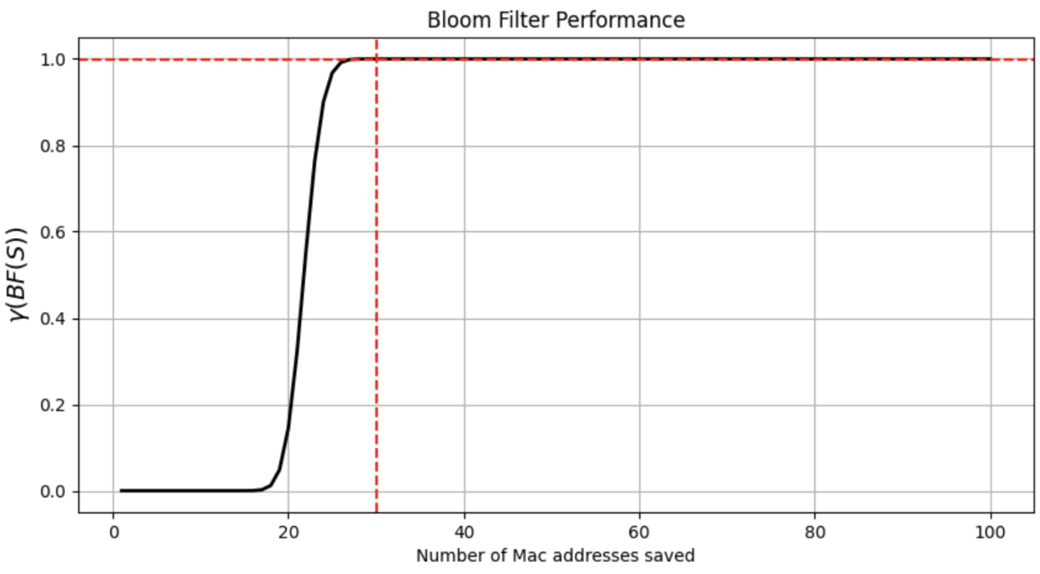


Figure 3.8: γ-deniability in function of n

In order to preserve privacy then, 30 fake MAC addresses are inserted in the Bloom Filters. The inserted MAC addresses represent the *anonymization noise*, which is used to preserve users' privacy as soon the Bloom Filter is created [3].

3.4.3 Bloom Filter operations

After the implementation of Bloom Filters, two main operations can be performed on the stored elements [3]:

- **Counting**: it is possible to count the number of people passed through the monitored area by counting the number of stored elements. The number of elements inserted in the Bloom Filter can be simply obtained via (3.6).

$$c_1 = -\frac{m}{k}\ln\left(1 - \frac{t}{m}\right) \tag{3.6}$$

- **Intersection**: Considering two different Bloom Filters, BF_1 and BF_2; the movement of one elements from one area to another is identified by performing intersection between the two filters. The intersection BF_3 is obtained by performing a bitwise AND operation between the two vectors. Once BF_3 is obtained, the number of elements contained within it can be calculated through (3.7), where t_k are the number of ones in the k-th filter.

$$c_2 = \frac{\ln\left(m - \frac{t_3 \times m - t_1 \times t_2}{m - t_1 - t_2 + t_3}\right) - \ln(m)}{k \times \ln\left(1 - \frac{1}{m}\right)} \tag{3.7}$$

3.5 System's power consumption

As previously said, one of the key requirement for the system is to have a low power consumption. Reducing the power required for the system's functioning helps in reducing the costs related to operation and to have a system more sustainable for the environment. To assess the amount of power required by this system during its operation the same smart plug shown in Figure 3.9 is used.

Figure 3.9: Meross Smart Plug MSS310R IT. Reproduced from [78]

This kind of smart plug allows to track the usage history of the connected device, monitoring the data associated to the real-time energy consumption. During regular operation, we connected the Raspberry Pi to the smart plug; collecting the data about voltage, current and power consumption over one hour. The obtained results are shown in Figure 3.10

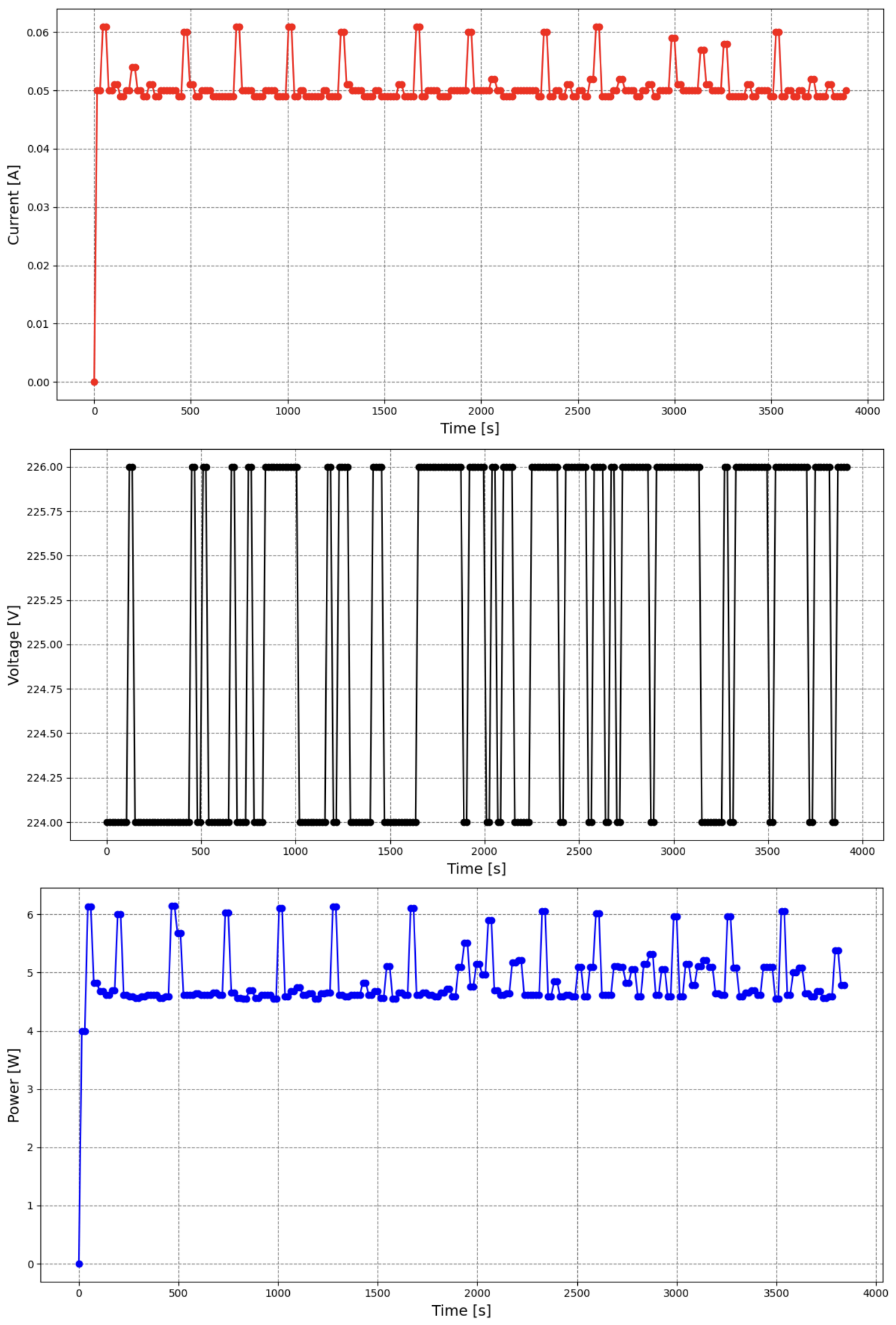


Figure 3.10: Current, Voltage and Power consumption of the proposed framework

In particular, the framework shows very low power-consumption during its functioning. The required power ranges between 5 watts, during the sniffing phase, and 6 watts during the clustering process. This energy efficiency significantly contributes to the system's operational efficiency, minimizing operational costs and environmental impact.

3.6 Location

To verify whether the system can satisfy all requirements imposed by [68], an experimental phase is planned. The experimental activities are scheduled to take place in the events area of the Valentino Park; where the sensors were implemented on October 10th 2024. We have chosen to employ two sensors within the target environment. The first sensor is housed in a box, while the second is on a pole. We have positioned the sensors approximately 90 meters apart, allowing them to capture and relay data over a significant area. Figure 3.11 depicts the moment when the sensors were installed.

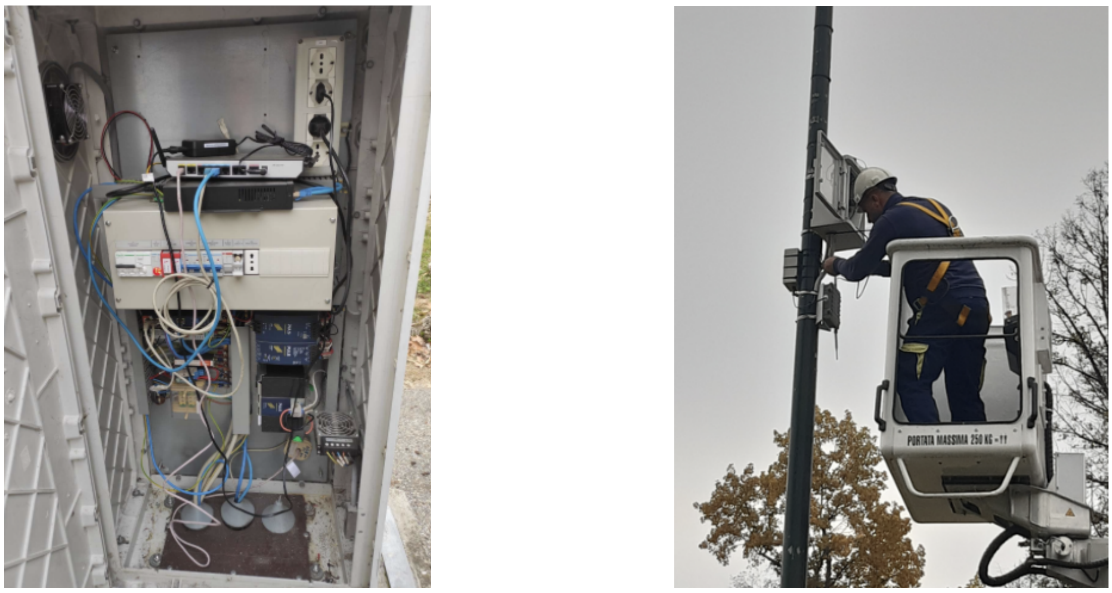


Figure 3.11: Installation of sensors in the target area

Chapter 4

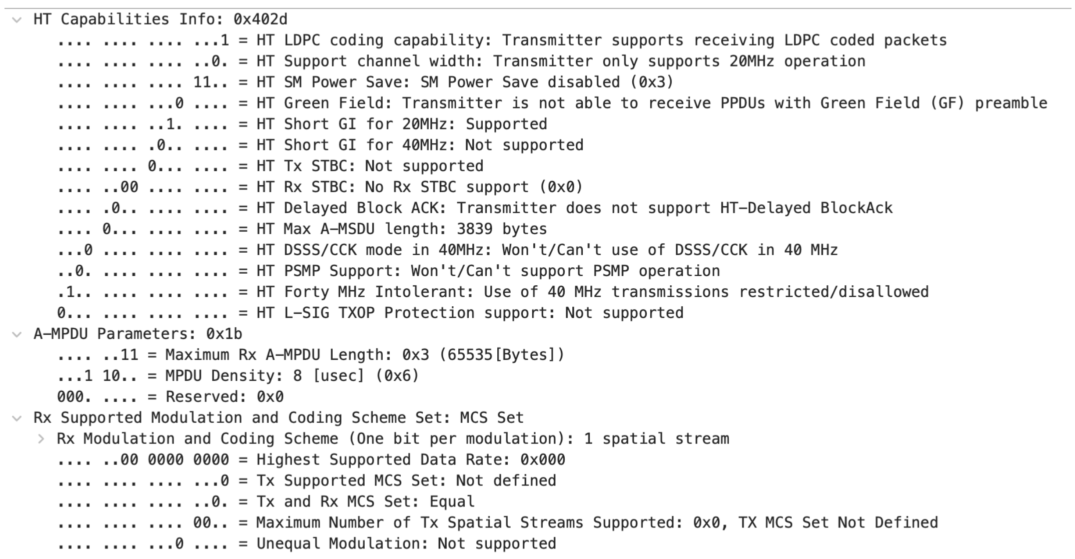
ARGO - Ai-driven frameworRk for countinG peOple

In this Chapter, we analyze the counting system proposed by [77]. In the first two Sections we study how the algorithm extrapolates the identifier of the devices starting by the Probe Requests. After that the counting process is examined. Finally, the test phase is the main topic of Section 4.3; here performances are evaluated in different scenarios to assess the algorithm's pros and cons.

4.1 Feature extraction

The developed algorithm leverages Information Elements (IEs) fields within probe request messages. These messages are part of the WiFi communication protocol, and they are sent by devices aiming to connect to a wireless network. The core function of the algorithm is to extract and analyze data from specific IE fields within probe request messages to construct a unique signature for each message. This signature enables the algorithm to differentiate devices and potentially identify them. The algorithm begins by extracting the relevant IE fields from each probe request message. The focus is on the HT Capabilities, VHT Capabilities, and Extended Capabilities fields as they can be considered as a comprehensive representation of the throughput capabilities of the device. They are series of bits that represent specific features/flags or capabilities of the device. Figure 4.1 shows the HT capabilities field of an iPhone 11, here the value of each bit represents the presence or absence of one specific feature.

The algorithm examines all the packets in the *.pcap* file. Each probe request is converted into a vector of three entries, each representing a specific field. The conversion is performed through a simple algorithm, shown in Algorithm 2.



```
⌄ HT Capabilities Info: 0x402d
    .... .... .... ...1 = HT LDPC coding capability: Transmitter supports receiving LDPC coded packets
    .... .... .... ..0. = HT Support channel width: Transmitter only supports 20MHz operation
    .... .... .... 11.. = HT SM Power Save: SM Power Save disabled (0x3)
    .... .... ...0 .... = HT Green Field: Transmitter is not able to receive PPDUs with Green Field (GF) preamble
    .... .... ..1. .... = HT Short GI for 20MHz: Supported
    .... .... .0.. .... = HT Short GI for 40MHz: Not supported
    .... .... 0... .... = HT Tx STBC: Not supported
    .... ..00 .... .... = HT Rx STBC: No Rx STBC support (0x0)
    .... .0.. .... .... = HT Delayed Block ACK: Transmitter does not support HT-Delayed BlockAck
    .... 0... .... .... = HT Max A-MSDU length: 3839 bytes
    ...0 .... .... .... = HT DSSS/CCK mode in 40MHz: Won't/Can't use of DSSS/CCK in 40 MHz
    ..0. .... .... .... = HT PSMP Support: Won't/Can't support PSMP operation
    .1.. .... .... .... = HT Forty MHz Intolerant: Use of 40 MHz transmissions restricted/disallowed
    0... .... .... .... = HT L-SIG TXOP Protection support: Not supported
⌄ A-MPDU Parameters: 0x1b
    .... ..11 = Maximum Rx A-MPDU Length: 0x3 (65535[Bytes])
    ...1 10.. = MPDU Density: 8 [usec] (0x6)
    000. .... = Reserved: 0x0
⌄ Rx Supported Modulation and Coding Scheme Set: MCS Set
  › Rx Modulation and Coding Scheme (One bit per modulation): 1 spatial stream
    .... ..00 0000 0000 = Highest Supported Data Rate: 0x000
    .... .... .... ...0 = Tx Supported MCS Set: Not defined
    .... .... .... ..0. = Tx and Rx MCS Set: Equal
    .... .... .... 00.. = Maximum Number of Tx Spatial Streams Supported: 0x0, TX MCS Set Not Defined
    .... .... ...0 .... = Unequal Modulation: Not supported
```

Figure 4.1: HT capabilities field of iPhone 11

For each frame, ARGO initializes three counters to zero. They store information related to the content of the fields we are interested in. The algorithm analyzes all the fields in the examined frame, searching for the VHT, HT, and Extended Capabilities fields. For each of them, the respective value is updated by adding a value which depends on data. The added value is equal to the number of bits set to one in the considered field.

Algorithm 2 Algorithm to compute the probe requests model identifiers.

```
Require: .pcap file with the capture
 1: $ids\_list \leftarrow empty\_list$
 2: for $packet$ in $capture$ do
 3:     $HT\_counter \leftarrow 0$                              ▷ HT parameter initialization
 4:     $VHT\_counter \leftarrow 0$                             ▷ VHT parameter initialization
 5:     $Extended\_counter \leftarrow 0$                        ▷ Ext parameter initialization
 6:     for $field$ in $packet$ do                               ▷ Parsing packet's field
 7:         if $field$ contains HT info then
 8:             $HT\_counter \leftarrow HT\_counter + value$             ▷ Updating the HT value
 9:         else if $field$ contains VHT info then
10:             $VHT\_counter \leftarrow VHT\_counter + value$           ▷ Updating VHT value
11:         else if $field$ contains Extended info then
12:             $Extended\_counter \leftarrow Extended\_counter + value$ ▷ Updating Ext value
13:         end if
14:     end for
15:     $ids\_list \leftarrow insert(HT\_counter, VHT\_counter, Extended\_counter)$ ▷ ID creation
16: end for
```

After the packet has been processed, the three counters are combined to form a unique vector which is the frame's identifier. All vectors so obtained are stored in "ids_list" that, by the end of the algorithm's execution, contains the set of all the computed identifiers.

4.2 Counting algorithm

The approach implemented by ARGO, described for the first time in [77], has been driven by the results of the analysis of [61, 62]. These studies have highlighted how *IE*-fields can be useful to distinguish different devices in an area. In particular, [62] talks about *stability* and *entropy* concepts, related to these fields.

- **Stability**: It refers to the consistency of these *IEs* over time. Authors observe that *IEs* remain stable for over 90% of devices, reflecting their intrinsic capabilities.
- **Entropy**: It measures the amount of identifying information provided by each element. Authors show how these fields are characterized by high entropy too.

The flowchart shown in Figure 4.2 summarizes the process for identifying and clustering devices based on their MAC addresses and probe request messages.

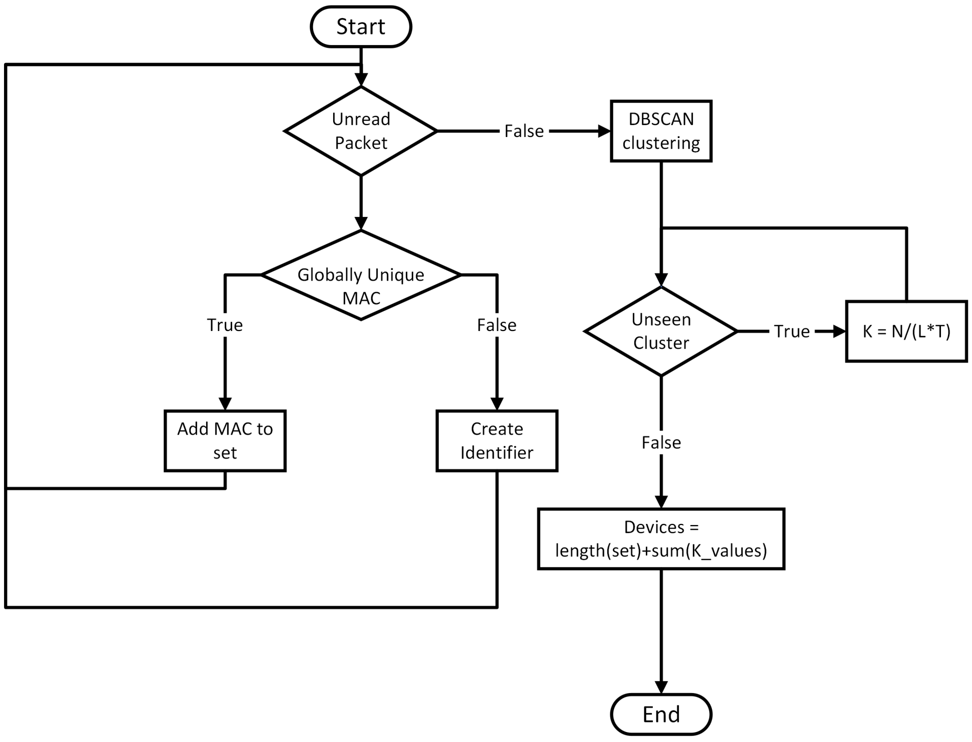


Figure 4.2: Counting Algorithm

The procedure examines all unread packets; if there are none, the process goes to the DBSCAN clustering step. However, if unread packets are detected, the algorithm evaluates whether the MAC address of the frame is a globally unique one or not.

For packets with globally unique MAC addresses, the algorithm adds the MAC address to the Bloom Filter, described in Chapter 3, and examines the next unread packet. Contrariwise, if the MAC address is not globally unique, an identifier is created using the conversion rule previously described in Algorithm 2. Once all packets have been processed, the algorithm can employ DBSCAN to cluster the collected identifiers. In this way we obtain a number of clusters which reflects the number of device model in the area.

The clustering phase is avoided in case of a poor number of frames with locally administrated MAC addresses. In fact, since clustering may be computationally expensive, it is verified that at least 2% of frames have locally unique MAC addresses. If this is not verified just the count of globally unique probes is provided as output.

The approach implemented through *ARGO* is similar to the one presented in [61]. However, authors of [61] do not consider cases in which multiple devices of the same model are present simultaneously. In this case, only counting the number of clusters is not enough to infer the number of people, resulting in underestimating the crowd-size. For this reason, such an approach is unfeasible for highly crowded environments.

The main contribution of ARGO is to provide a way to solve this issue. In particular, assuming to know the probe requests' sending rate and that it is constant over time, the number of devices within the cluster can be computed through (4.1).

$$N = \frac{K}{L \cdot T} \tag{4.1}$$

Where:

- K is the total number of messages sent. This value represents the overall quantity of messages transmitted by all devices in the cluster during the specified time period.
- N is the number of devices. This indicates how many devices are sending messages.
- L is the sending rate (messages per unit of time) for each device. This represents the frequency at which each individual device sends messages.
- T is the time interval considered. This is the period over which the total number of messages sent is measured, hence the capturing window.

The main problem with this strategy is that in many cases, the sending rate of probes is not known. In this case a further assumption is done: *IEs* carry information about devices' throughput capabilities, hence devices with similar signatures must share similar sending rate [77].

A *model.json* file is created. The file stores information about the devices analyzed in [55]. This file includes details about each device's signature and the sending rates of the probe request, as shown in Table 4.1. According to [55], the sending rate is influenced by the state of activity of the device. Therefore, the file contains multiple rates to adapt to any possible scenario, such as when the devices are in locked phase, awake or active.

	Apple iPhone14Pro	**Apple iPhone13Pro**	**OnePlus Nord5G**	**Xiaomi Mi9Lite**
Locked Rate	0.08	0.11	0.22	-
Awake Rate	0.44	0.67	3.90	-
Active Rate	0.38	0.84	0.54	0.24
Mean Rate	0.22	0.39	0.43	0.26
Model ID	[0, 149, 16981]	[0, 147, 16981]	[4052911066, 142, 972]	[2442297858, 141, 843]

Table 4.1: Device Information

For each cluster identified by DBSCAN, the most similar device in "model.json" is identified, and its sending rate L is used. Through (4.1), the number of devices N can then be computed. The default rate used is the average rate, calculated as the mean of the active and locked rates. Nevertheless, anyone of the other rates can be used to obtain a more precise estimate of the number of devices if additional information about the scenario are available.

At the end, the number of total devices is computed as the sum of the elements with a globally unique MAC address and the number of devices derived from the clustering algorithm.

4.2.1 Fine-Tuning DBSCAN parameters

DBSCAN requires the tuning of different parameters to function effectively. These parameters significantly influence the algorithm's performance, carefully selecting them is crucial for a precise crowd-size estimation. The parameters that need to be tuned are:

- **ϵ-radius**: it defines the maximum distance within which two data points are considered neighbors.
- **Min_points**: it is the minimum number of points required to form a dense region.
- **Metric**: it is the distance metric used to calculate the distance between points.

Since the probe requests are represented as points in a three dimensional state space, the Euclidean distance is a suitable metric to measure the distance (and the dissimilarity) between points. In this case, it would reflect how much two probe requests are different based on their content. A smaller Euclidean distance would indicate that two probe requests have similar IEs values, suggesting they may have originated from the same source. For what concerns ϵ-radius and Min_points parameters, an example of their influence on the output is shown in Figure 4.3. Here, the clustering algorithm is applied to the same data using three different couples of parameters, showing results very different from one to the other.

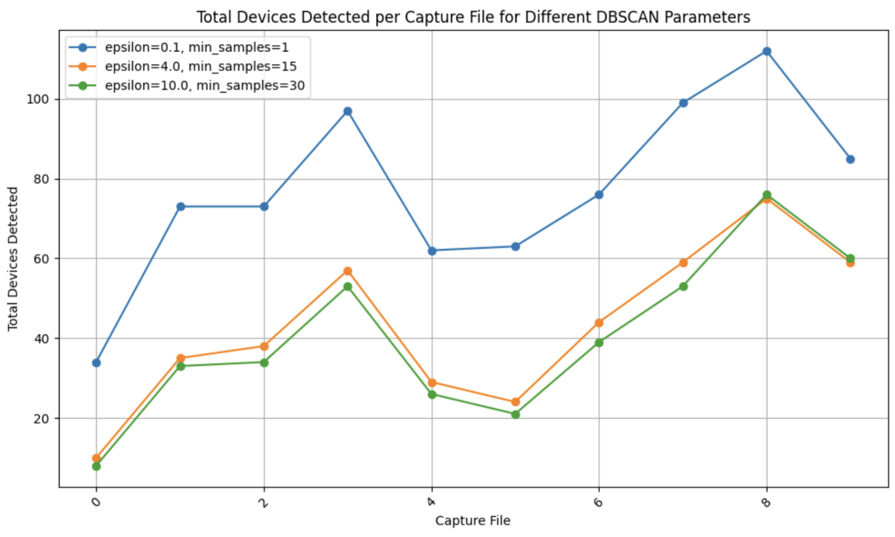


Figure 4.3: Number of devices detected for different Parameters

Considering the high stability and entropy values of the *IE* fields, the ϵ-radius is chosen to be small to achieve a finer grouping between devices.

To select the *Min_points* parameter different consideration have been done:

- Analyzing the captures in [55] revealed that some devices can send probes with varying Information Elements (IEs). Setting a higher value for the Min_points parameter can help mark these irregular probes as outliers. Contrariwise, Using a too-low Min_points value may lead to overestimating the number of devices by counting the presence of the aforementioned outliers.

- Setting the Min_points parameter too high can lead to miscalculating the number of devices too. Since at least $N = Min\_Points$ probes requests with similar IEs are required to form a cluster; devices with a low sending rate may not satisfy this threshold, leading to underestimating the total number of devices present.

Author of [79] conducted some experiments in a controlled environment to fine-tune the parameters values.

Precisely, they employed the sensor described in Chapter 2 to obtain a sequence of sniffing of probe requests in a room at the Politecnico di Torino. They used the sniffer for twenty minutes, obtaining ten captures, each lasting two minutes. The ground truth established by manually counting the number of people in the room.

The most suited ϵ-radius and the *Min_points* parameter are chosen to maximize the accuracy with respect to ground truth. The best results are obtained for ϵ-radius $= 4$ and $Min\_samples = 15$ and they can be seen in Figure 4.4 [79].

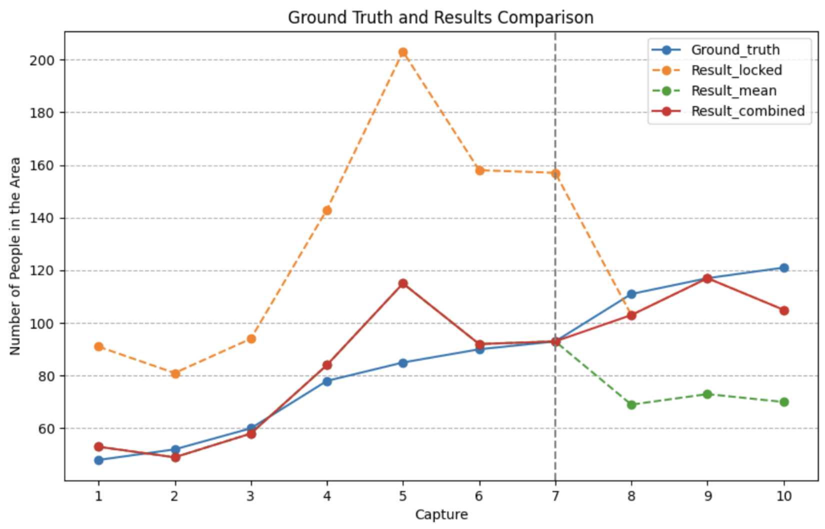


Figure 4.4: Performance of the Algorithm after parameter selection

These results underline the significance of having extra context information. At the beginning of the sniffing phase, the lesson was not started yet. In this context, using the mean rate was a good approximation since some people were using their phone while other were not. After Capture 7, when the lesson in the room started, using the mean_rate was no more a good choice as students turned off their devices. In fact, by using the locked_rate, performances increased. The overall accuracy obtained by the framework after parameter selection was 91%.

4.3 Performance evaluation

To evaluate the performance of the algorithm, we used several testing methods across different contexts and scenarios:

- **Simulated environment**: a Probe Request Generator [3, 77] is used to produce a set of *.pcap* file. The advantage of creating traces with a simulator is that the ground truth is known, although the temporal behavior of the simulated devices does not reflect the real pattern, as the standardized channel access mechanism is not emulated yet.

- **Real environment**: In this case two scenarios are considered for the test: an indoor test and a open door scenario. The indoor location has been chosen in order to be as isolated as possible to limit the possibility of capturing messages coming from the outside. For the outdoor tests, the results obtained by the sensors in Valentino park are examined.
- **Controlled environment**: authors of [80] recorded a set of captures in anechoic chamber, which have been used as a basis for the task in "CONFRONT" challenge [81]. We used the challenge to test the algorithm and to obtain a clear and precise idea of its performances.

By combining diverse testing environments, the evaluation provides a grasp of the algorithm's strengths, weaknesses, and possible areas for improvement.

4.3.1 Performance in Simulated Environment

Probe Request Generator

The Probe Request Generator is a software tool designed to simulate WiFi traffic generated by mobile devices in real-world environments [77]. Its primary function is to create probe request packets, simulating realistic frames that comply with the IEEE 802.11 standard and emulate different device throughput capabilities.

The generator operates based on a state-machine model. It is designed as a scheduling queue where device creation and deletion, phase changes and packet transmission are scheduled according to their starting times. The model uses probability distributions derived from real-world data to define the initial state of a device, transitions between states (i.e., from locked to awake to active), and the sending rate of probes. This approach allows the generation of realistic probe request traces that accurately reflect the behavior of actual devices in various usage scenarios.

The main advantage of this tool is that it has at its disposal a set of devices with known characteristics i.e., the same studied by authors in [55]. It can then simulate an environment where an arbitrarily chosen number of these devices is present.

Performance evaluation

The PR simulator creates an environment with an arbitrarily chosen number of devices, reproducing the presence of all the devices examined by the authors of [55]. A set of .pcap traces is generated, each including probes sent by different devices. This approach permits the evaluation of the algorithm's performance in various contexts, such as highly crowded or sparsely populated environments. Table 4.2 shows the description of all the datasets used for evaluation.

Dataset	**Description**
A	Contains only one device.
B	60 devices all belonging to one single model.
C	6 devices with different models.
D	70 devices of various models.
E	120 devices of various models.

Table 4.2: Description of Datasets

We run *ARGO* on all the presented datasets. Table 4.3 summarizes the results achieved. As can be seen from the detection accuracy, the results are promising across all considered scenarios. However, it is crucial to consider the several limitations of this evaluation method before extrapolating any consideration about performances. Firstly, while the detection accuracy is good, it is essential to note that in real-world scenarios, devices may change the value of some fields in their Information Elements (IEs). The simulator does not implement this behavior, which assumes more static and predictable patterns instead: while the initial results are encouraging, they represent a controlled and idealized environment.

To fully grasp the actual performance of the algorithm, further testing and performance evaluations are required. Conducting more experiments in real environments is needed: here, devices display natural variations in their IEs, providing a more accurate representation of the capabilities and limitations of ARGO.

capture_file	**ground truth**	**total_devices**	**detection accuracy**
A	1	1	1.00
B	60	52	0.87
C	6	7	0.83
D	70	58	0.83
E	120	110	0.92

Table 4.3: Results table with detection accuracy

4.3.2 Performance in Real Environment

To extensively evaluate the performance of ARGO in realistic settings, two types of real-world applications are considered: a closed environment within a room at the Politecnico di Torino and an open environment in a public park. Using this dual approach, we can verify the algorithm's robustness and accuracy under different conditions that mimic actual usage scenarios.

Indoor Environment

We employed the sensors described in Chapter 2 within a room at Politecnico di Torino to obtain a set of captures with the probe requests sent by the devices in the target area: 30 captures were collected, each 2 minutes long. To verify if the algorithm can follow the increasing trend of the ground truth, we commenced the test 10 minutes before the beginning of the lesson. We have deployed one sensor for the experiment, positioned centrally within the room; manual counting of people within the area furnished the ground truth. It is worth noting that the ground truth only considers people within the room, while the sniffers can also receive some undesired messages from outside the desired area. A power threshold on the received message was used to filter out packets with a too low received signal strength in capture analysis. In this specific use case, we set the power threshold to −100 dBm to cover the whole room, excluding the outside. Figure 4.5 shows the performances achieved by *ARGO* in this scenario.

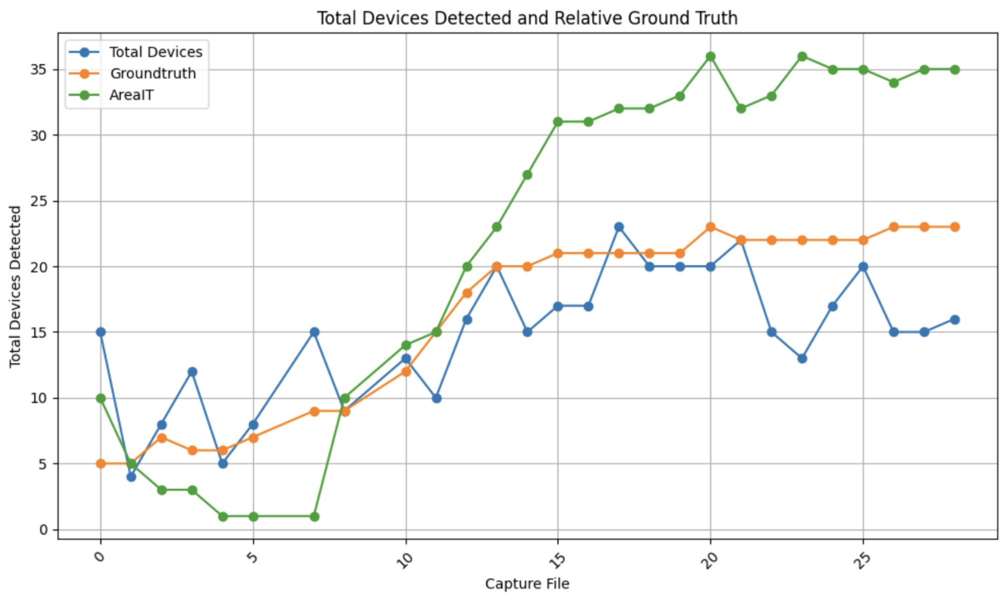


Figure 4.5: Estimated vs Actual Device Count in a Room 14

Despite of some peaks, the algorithm follows quite well the trend of the ground truth, with an accuracy higher than 70%. The peaks in figure can be due to a not optimal isolation of the area of interest and to a instability of the *IEs* sent by some devices. For this reasons, it is reasonable to have a not precise clustering results. Finally, Figure 4.5

depicts the results supplied by the Information Technology (IT) Department, referenced as *AreaIT*. The provided data show the number of devices within the room connected to WiFi. A remarkable observation is the difference between the framework's output and the number of devices effectively present. The challenge lies in distinguishing and identifying one particular device between a group of similar ones. This task becomes increasingly complex when multiple devices show comparable or common characteristics, potentially leading to an underestimated result.

Outdoor Environment

To gauge information about the algorithm's reliability, we have analyzed the captures from one of the sensors installed in Valentino Park, in Turin. We have considered captures obtained for a week, from May 30th to June 5th. we chose that week because the weather was nice and it also included a public holiday. We hope to observe a peak in attendance during that day, along with a reasonable trend in results.

Figure 4.6 shows the obtained output. Each point in the graph represents the number of devices identified in 10 minutes by the ARGO algorithm. The computed results demonstrate a noteworthy compliance with the day-night trends, with notable peaks during afternoon hours, compatible with expectations. Additionally, there is higher attendance on the public holidays of June 1st and 2nd compared to weekdays. Considering both observations, these statements validate the plausibility of the algorithm's results in an outdoor environment such as a public park.

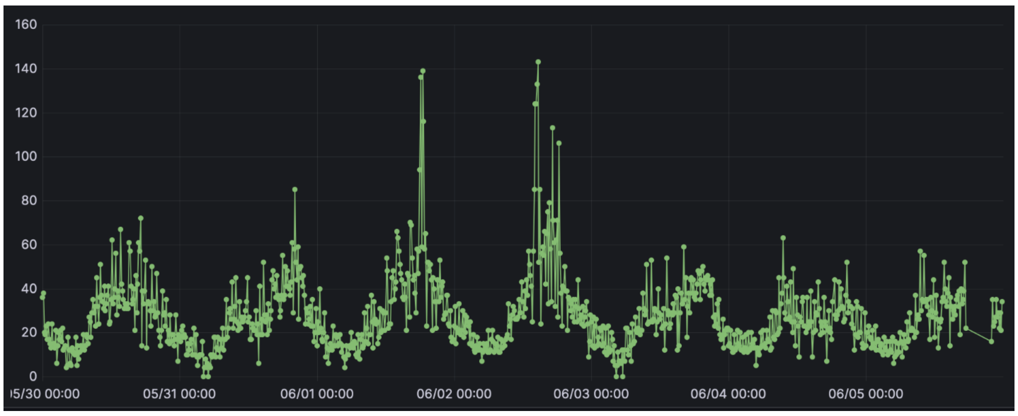


Figure 4.6: Captures in Real life, from May 30th to June 5th

4.3.3 Performances in Controlled Environment

The CONFRONT challenge [81] (Challenge ON wifi FRame fingerprinting for people cOunting aNd Tracking) is a competition aimed at comparing and improving people counting and tracking techniques using WiFi frame data. The challenge is structured into three tasks (A, B, and C) that assess participants' ability to identify and count devices present in a WiFi network, even in complex situations where devices randomize their MAC address.

Task	Description	Output
A	In task A, challengers use a file derived from amalgamating individual device captures executed by [61]. The precise count of devices is known.	The required outcome is a CSV file composed by two columns: the IDs of the samples and their respective labels. Discarded samples must be labeled with -1.
B	In task B, challengers use a file derived from amalgamating individual device captures executed by [61]. In this case the number of devices in capture is not known.	The expected result is a CSV file with two columns: the IDs of the samples and their corresponding labels. Discarded samples must be marked with label -1.
C	In task C, challengers use a file derived after sniffing a collection of devices within an anechoic chamber.	The expected output is the count of devices detected along with the number of discarded samples.

Table 4.4: Description of Tasks

Table 4.4 provides a description of the tasks along with the output the challengers must provide. The challengers are then classified on the basis of a score, ranging from 0 to 300. The final score is given as:

$$s = s_A + s_B + s_C \tag{4.2}$$

Where s_A, s_B, s_C are the scores achieved for each task, computed in the following manner:

- **Task A and Task B**: for these tasks the score is computed as:

  $$s_i = V_i \times d_i\% \quad \forall\, i \in \{A, B\} \tag{4.3}$$

  Where $d_i\%$ is the percentage of discarded samples and V is the V-measure, a metric used to evaluate the quality of clustering by assessing both the homogeneity and completeness of the clusters. It is computed as:

  $$V = \frac{(1+\beta) \times \text{homogeneity} \times \text{completeness}}{(\beta \times \text{homogeneity}) + \text{completeness}} \tag{4.4}$$

  The parameter β represents the trade-off between the importance given to homogeneity versus completeness and, in this case, it is set to 1.

- **Task C**: it is computed by the product of the percentage error in counting devices ($e_c\%$) and the percentage of discarded samples ($d_c\%$).

  $$s_c = e\% \times d_c\% \tag{4.5}$$

Each score ranges between 0 and 100 and assesses the ability of challengers in device recognition.

Table 4.5 shows the results achieved by the algorithm along with the ground truth provided by the challenge organizers.

Dataset	**Ground truth**	**Score ARGO**
A	9	99.37
B	15	16.96
C	22	70.58

Table 4.5: Scores for different datasets

While results in Task A and C are good in terms of performance; task B shows how this algorithm need some improvement yet. In particular both completeness and homogeneity values are quite low for task B (around 17%). This result give many insight about the weakness of the algorithm and the way of solving them. The chosen *IEs* are not sufficient, it is necessary to chose further identification elements.

Chapter 5

Exploiting Device Fingerprinting

In this Chapter, we analyse the limitations of the ARGO algorithm. Along with each main drawback, we present a possible solution. In this way, we can enhance the algorithm's accuracy in detecting devices through 802.11 Probe Requests. Aiming this, we have included additional features for clustering, refined the signature generation, and employed a new clustering algorithm (*OPTICS*). Finally, we test ARGO 2.0 and the applied modification in Section 5.4, to verify their effectiveness in real and controlled scenarios.

5.1 Limitations

The test phase presented in the previous Chapter has highlighted some lacks in the algorithm that we have to address to improve the performance of our framework. The main problems are:

- **Low-resolution clustering**: The features employed in ARGO, derived from specific Information Elements (IEs) within probe request messages, are not always sufficient to distinguish between different devices. This leads to a not precise device count.

- **Signature collision**: the trivial signature generation algorithm used by ARGO is susceptible to misclassification errors. If two devices have similar bit configurations in their IE fields, they might be incorrectly identified as the same device, leading to inaccurate results.

- **Label's assignment**: when multiple devices of the same model are in the monitored area, ARGO can only estimate the total device count for each cluster; no individual label can be assigned to each probe, impeding the tracking of specific devices within the group.

In the following Sections, we examine the first two problems while we postpone the discussion related to the last problem in the following Chapter.

5.2 Low-resolution Clustering

Table 5.1 shows the comprehensive results achieved by ARGO in the CONFRONT challenge. In particular:

- **Completeness** measures how precisely the algorithm can cluster together the data points of a single ground truth class.
- **Homogeneity** evaluates whether each cluster contains data points from a single ground truth class.

	Task A	**Task B**	**Task C**
Completeness	1.000000	0.174659	N/A
Homogeneity	0.987631	0.164925	N/A
Score	99.377718	16.965223	70.588235
Number of Devices Identified	8	14	17
Number of Real Devices	9	15	22

Table 5.1: Comparison of trials A, B, and C

As seen in Table 5.1, ARGO has achieved low scores in both metrics in Task B. This suggests that ARGO's clustering is not aligned with the ground truth. In this case, a low completeness value indicates that the algorithm divides elements of the same ground truth class across multiple clusters. Additionally, a low homogeneity value means that clusters contain mixed data points from different classes.

After having analyzed the *.pcap* file for these tests, the reason for these low scores becomes clear: despite many frames having identical field content for the IEs used, the probe messages differ in other field contents. This analysis shows that additional fields must be considered to grasp a better understanding of the device's identity.

5.2.1 New Features

To enhance the cluster quality produced by Argo, we examined a set of captures obtained by our sensors in Valentino Park. We considered a month of captures starting from November 1st to December 1st. For any *.pcap* file, we analyzed all the probes with a locally unique MAC address. The other Probe Requests are discarded for this analysis, as ARGO does not consider them in the clustering phase. We analyzed over 30000 probe requests and, for each, we store the list of the present *IEs*.

Figure 5.1 shows the results of this analysis and reports the found *IEs*, in the format IE_Counter. The counter variables takes into account that in some probes, multiple fields of the same type can be present. The y-axis show the occurrence rate of each specific IE.

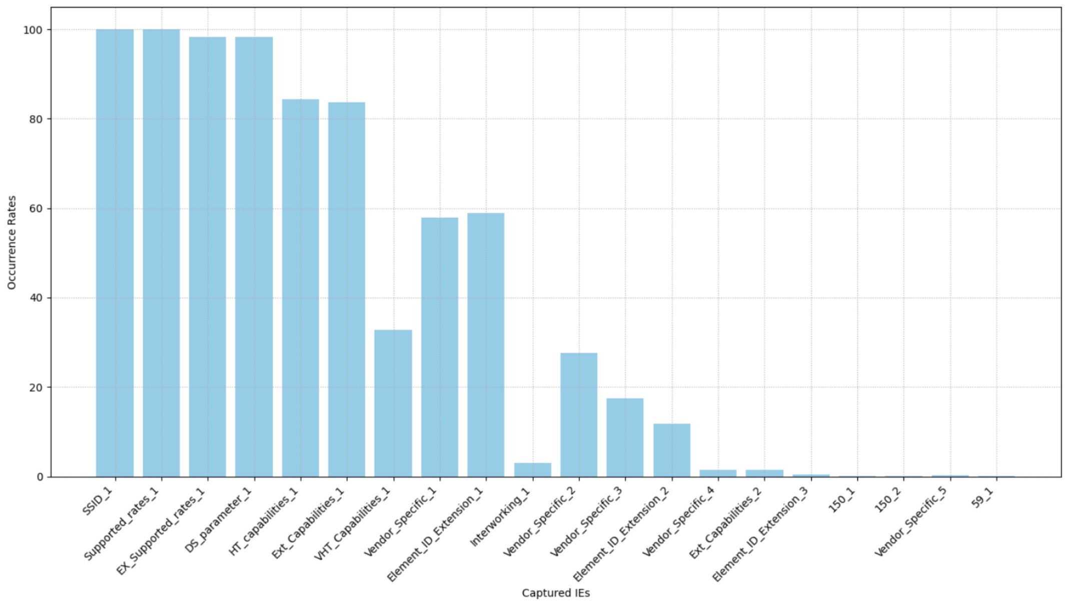


Figure 5.1: Occurrence rates of different Information Elements

All the found *IEs* can provide further information to be applied in clustering phase. However some of them need to be excluded due to the following considerations:

- **SSID**: It contains the list of the device's preferred networks. Due to the potential privacy issues, in modern devices the list is often empty, with no information to be used.

- **Supported Rates and Extended Supported Rates**: We decided to exclude these fields as many devices share the same supported rates. Their use could damage the clustering functioning, lowering the clusters' quality.

- **DS parameter Set**: It contains only the current channel used by the Basic Service Set (BSS), no information on the source device can be extrapolated.

- **Interworking**: It contains various information that help Wi-Fi devices understand and interact with external networks. No information on the source device are contained.

- **Other minor IEs**: there are several other *IEs* with low occurrence rates. We decided to exclude them as their rate of occurrence is too low to make them useful in categorize devices.

We decided to use the information provided by the Vendor Specific Information Element. This element contains information about the device vendor and can be useful to distinguish between devices built by different manufacturers.

These considerations are compatible with the results of the analysis conducted by [61]. They have analyzed probe requests sent by different devices operating in diverse conditions: with the screen turned on (Mode A) and turned off (Mode S). This study aims to find the Information Elements (IEs) carrying the most significant amount of information. A Random Forest algorithm is employed to measure the Gini importance of each IE.

The Gini importance is a metric used in decision tree-based machine learning algorithms. It measures the feature's capability to guess the outcome of a random input. For each node in the forest, the feature that splits the node is the one that reduces the most Gini impurity, i.e., the probability of incorrectly classifying a randomly chosen element. The Gini importance for a variable is the sum of the impurity reductions at all nodes where that variable is used for splitting, normalized by the total impurity reductions of all variables. A variable with high Gini importance is frequently used to create effective splits in the tree and significantly contribute to reducing impurity.

Table 5.2 shows the Gini Importance of the different IEs in the different phase for the source device, i.e., mode A and mode B.

IE ID	Name	Gini Importance (A)	Gini Importance(S)
0	SSID	0.088	0.004
1	Supported Rates	0.056	0.124
3	DS Parameter Set	-	-
45	HT Capabilities	0.175	0.247
50	Extended Supported Rates	0.024	0.024
107	Interworking	0.001	0.019
127	Extended Capabilities	0.340	0.251
191	VHT Capabilities	0.073	0.014
221	Vendor Specific	0.162	0.197
255	Element ID Extension	0.081	0.120

Table 5.2: Information Elements in the Dataset with A and S Importance Values

We decided to continue using the VHT capabilities field despite of its low Gini Importance. This because it contains the information related to IEEE 802.11ac features of the source device. We expect the importance of this feature to grow as the use of this standard becomes more widespread. The effectiveness of our choice can be seen by the increase of the occurrence rate of this field over the years: in 2022 this field was present only in the 11.1% of the probes [61] while nowadays, according to our analysis, its occurrence rate increases to over 30%.

The IEs used by ARGO are not the optimal set in terms of possible performance. However, with the newly included information, we aim to improve the categorization of devices based on the sent frames.

5.3 Signature collision

The signature generation algorithm employed by ARGO is prone to misclassification errors. Specifically, the algorithm just counts the number of bits set to 1 in each field. This can cause some problem: even a minimal difference in field content, such the one shown in Figure 5.2 where a single bit is in a different location, could lead two devices with similar field content to be erroneously recognized as the same. This can lead to an inaccuracy in counting and result in a non-optimal performance.

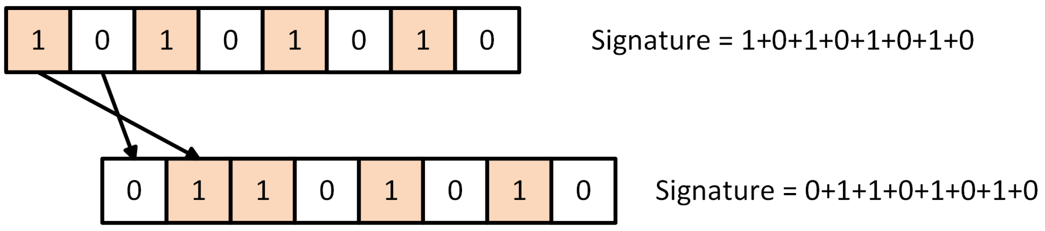


Figure 5.2: Feature misclassification problem

In the newly-presented version, we have employed a more robust conversion process: rather than simply using the count of non-null bits, ARGO 2.0 now considers the values of the bytes within the IE fields. For two randomly chosen 24-bit sequences, the new conversion process reduces the signature collision probability from 11.24% to approximately 0.21%. The more robust behavior with respect to minor variations in IE fields could enhance the accuracy of device clustering, particularly in scenarios where devices have similar but not identical IE field values.

5.3.1 Improved Algorithm

The new signature generation process employed by ARGO 2.0, shown in Algorithm 3, improves algorithm's performance in terms of accuracy and resilience in probe request identification.

Algorithm 3 Algorithm to compute the probe requests device model identifiers

```
Require: .pcap file with the capture
Ensure: $identifiers\_list$ source model identifiers list
 1: $identifiers\_list \leftarrow []$
 2: for $packet$ in $capture$ do
 3:     $rate \leftarrow frame\_rate(packet)$                    ▷ Get the frame rate
 4:     if $rate! = 1.0$ then
 5:         $Continue$
 6:     end if
 7:     $HT \leftarrow 0$                                        ▷ HT parameter initialization
 8:     $VHT \leftarrow 0$                                       ▷ VHT parameter initialization
 9:     $Extended \leftarrow 0$                                  ▷ Ext parameter initialization
10:     $Vendor \leftarrow 0$                                    ▷ Vendor data initialization
11:     $packet\_fields \leftarrow extract\_all\_fields(packet)$
                                                ▷ Extract all the Information Elements
12:     for $key, value$ in $packet\_fields$ do
13:         if $key == HT\ info$ then
14:             $HT \leftarrow process\_value(value)$            ▷ HT value
15:         else if $key == VHT\ info$ then
16:             $VHT \leftarrow process\_value(value)$           ▷ VHT value
17:         else if $key == Extended\ info$ then
18:             $Ext \leftarrow process\_value(value)$           ▷ Ext value
19:         else if $key == Vendor\ info$ then
20:             $Vendor \leftarrow process\_value(value)$
                                                             ▷ Vendor value
21:         end if
22:     end for
23:     $identifier \leftarrow (HT, VHT, Extended, Vendor)$
24:     $identifiers\_list \leftarrow insert(identifier)$
                                                ▷ Identifiers list update
25: end for
26: return $identifiers\_list$
```

The conversion algorithm begins by verifying whether each packet's frame rate is equal 1.0 Mb/s, discarding the frames which do not meet this requirement. This initial check is crucial because our observations revealed that some devices, such as Apple iPhones, occasionally transmit probe requests at varying throughputs, specifically 6.0 Mb/s for 3% of the time and 1.0 Mb/s for the remaining. These frames differ not only in the bit rate but also in the presence or absence of some capabilities information. We opted to exclusively consider the probe requests sent at the standard throughput, thus enhancing the outliers-invariance of our counting algorithm.

After the first check, the algorithm proceeds by examining each probe request packet.

For each of them, it initializes to 0 a set of counters that stores the converted content of the used IEs: the HT capabilities, VHT capabilities, Extended Capabilities, and Vendor-specific fields of the packet.

Each IE is recognized within the frame by an identifier. Starting from Line 11, the algorithm extracts all the fields within the probe request, searching for the desired IDs, i.e., $ID \in \{45,127,191,221\}$. For each field, if it corresponds to HT information, ARGO 2.0 converts the field content with the procedure described in Section 5.3 and stores the output in the *HT* variable. The conversion process continues for all the fields extracted, updating the values of VHT, Extended, and Vendor variables.

The identifiers are then clustered together. Initially, ARGO utilized the DBSCAN clustering algorithm in its first version; however, in this version, an attempt was made using OPTICS. This decision was pivoted by the results achieved in [60]. Here, the clustering executed with OPTICS shows better performance when compared to DBSCAN. This is probably due to its better capabilities in handling clusters with varying densities.

Finally, after the clustering phase, the counting process proceeds similarly to Chapter 4. For each cluster, we can find the model with the most similar identifier whose probe requests' sending rate is known. Knowing the sending rate and the number of probe requests, it is possible to use (4.1) to infer the number of devices in the cluster. The final count of devices is obtained by summing the number of devices identified and the ones with global MAC addresses.

5.3.2 Parameter setting

For this new version, we need to fine-tune the clustering algorithm parameter. For the OPTICS algorithm, the ϵ-radius does not need to be explicitly specified; it is estimated instead during the algorithm's execution based on data density and point distances. This adaptability allows OPTICS to obtain better knowledge about the data structure without explicit user input. Thus, the only parameter to be tuned is the *Min_points* parameter. The same captures used for fine-tuning DBSCAN are employed here, allowing a performance comparison between the two algorithms in optimal conditions. After tuning, the best performance are obtained for $Min\_points = 10$

Figure 5.3 shows the outputs provided by both counting algorithm versions. The first image shows the outputs of the algorithms, while the second shows their accuracy. The results provided by the ARGO 2.0 are slightly better than the ones achieved by its predecessor, with a consistently higher accuracy. The following Sections show the new algorithm tested on different scenarios, to gauge information about the performance that new version achieves in diverse contexts.

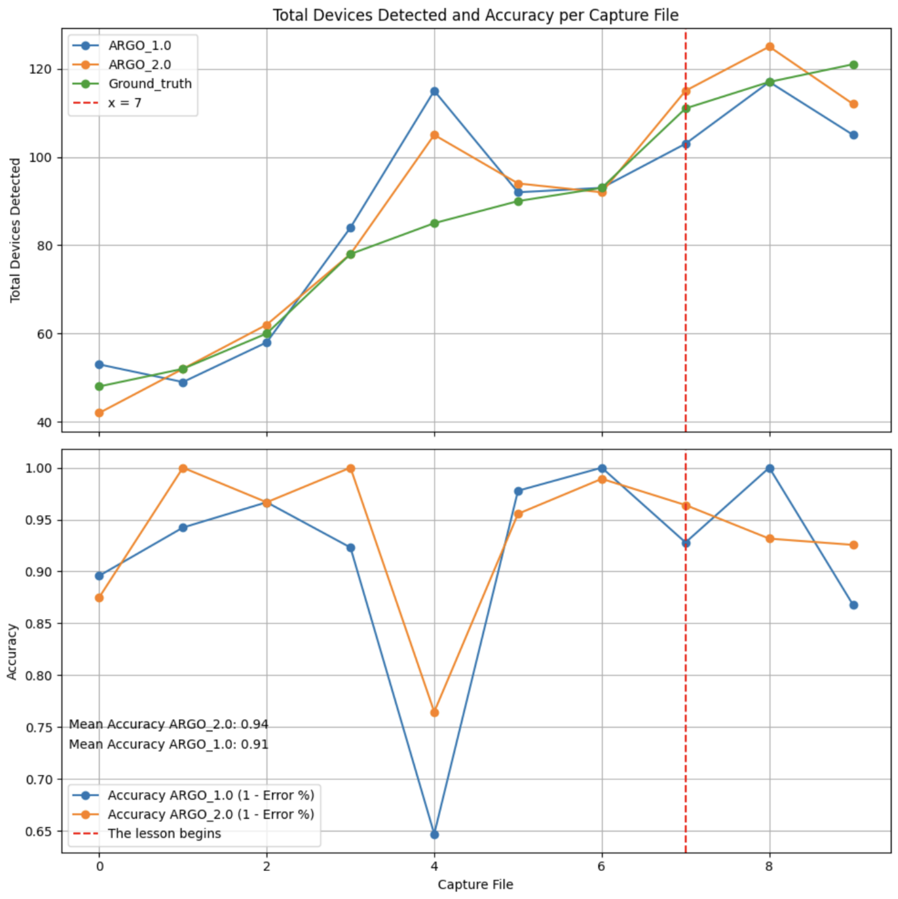


Figure 5.3: Parameter selection and accuracy comparison

5.4 Performance Evaluation

The evaluation process follows the same methodology previously described in the Chapter 4. In particular Real-word environment and Controlled environment are considered. Notably, simulated environment testing was not feasible due to limitations of the Probe Request Generator, which does not includes Vendor Specific *IE* in the probes generation.

5.4.1 Performances in Controlled Environment

We uses the dataset and evaluation method provided by CONFRONT challenge [81] to test the second version of the algorithm. In particular the reliability of their ground truth is guaranteed by the use of an anechoic chamber. Using these datasets we can give a precise evaluation about the quality of the cluster produced by the algorithm.

Table 5.3 shows a comprehensive summary of the outcomes of Tasks A, B, and C achieved by both the versions of ARGO. Key evaluation metrics, i.e., completeness, homogeneity, overall score, and the number of identified versus real devices, are detailed for each task.

ARGO	**Task A**	**Task B**	**Task C**
Completeness	1.000.000	0.174659	
Homogeneity	0.987631	0.164925	
Number of Devices Identified	8	14	17
Number of Real Devices	9	15	22
Score	99.3777	16.9652	70.5882
ARGO 2.0	**Task A**	**Task B**	**Task C**
Completeness	0.987631	0.942033	
Homogeneity	0.966179	0.916066	
Number of Devices Identified	9	15	22
Number of Real Devices	9	15	25
Score	97.6787	92.8868	87.9138

Table 5.3: Summary of tasks A, B, and C for ARGO versions 1 and 2.

As reported in Chapter 4, the output provided by ARGO is not aligned with the ground truth; showing low values for homogeneity and completeness. This is due to the characteristics of density-based clustering algorithms: they focus on finding groups of points closely packed together and greatly separated from other groups by less dense areas. In case of partially overlapping clusters this type of algorithm can lead to misclassification problems showing then poor performance.

On the opposite side, the results achieved by ARGO 2.0 are very good in terms of these metrics. The introduction of the new feature and the new signature generation process has allowed to capture additional aspects of the data; this has lead to an higher distance among points in the state-space as a consequence. These changes in the distribution of data points have given a series of benefits with respect to the previous version of the algorithm:

- **Better separation**: The improved methodology grant a more accurate data segregation into clusters. These clusters almost perfectly reflect the original classes, ensuring each represents a distinct device model.
- **More effective clustering:** Since each cluster is far from the others, there is less overlap and misunderstanding among various classes. The higher distance between different groups leads to a higher completeness score, i.e., the points of each ground truth class are frequently assigned to a single cluster
- **Lower dispersion:** The new approach reduces the scattering of points of the same

ground truth class across multiple clusters. Data points of the same class are more frequently grouped, leading to more homogeneous clusters.

5.4.2 Performances in Real environment

To assure a possible comparison among the two versions in real environment, we use the same captures used for testing the first version of algorithm. The two graphs in Figure 5.4 represents the people count provided and the accuracy achieved by each version of ARGO.

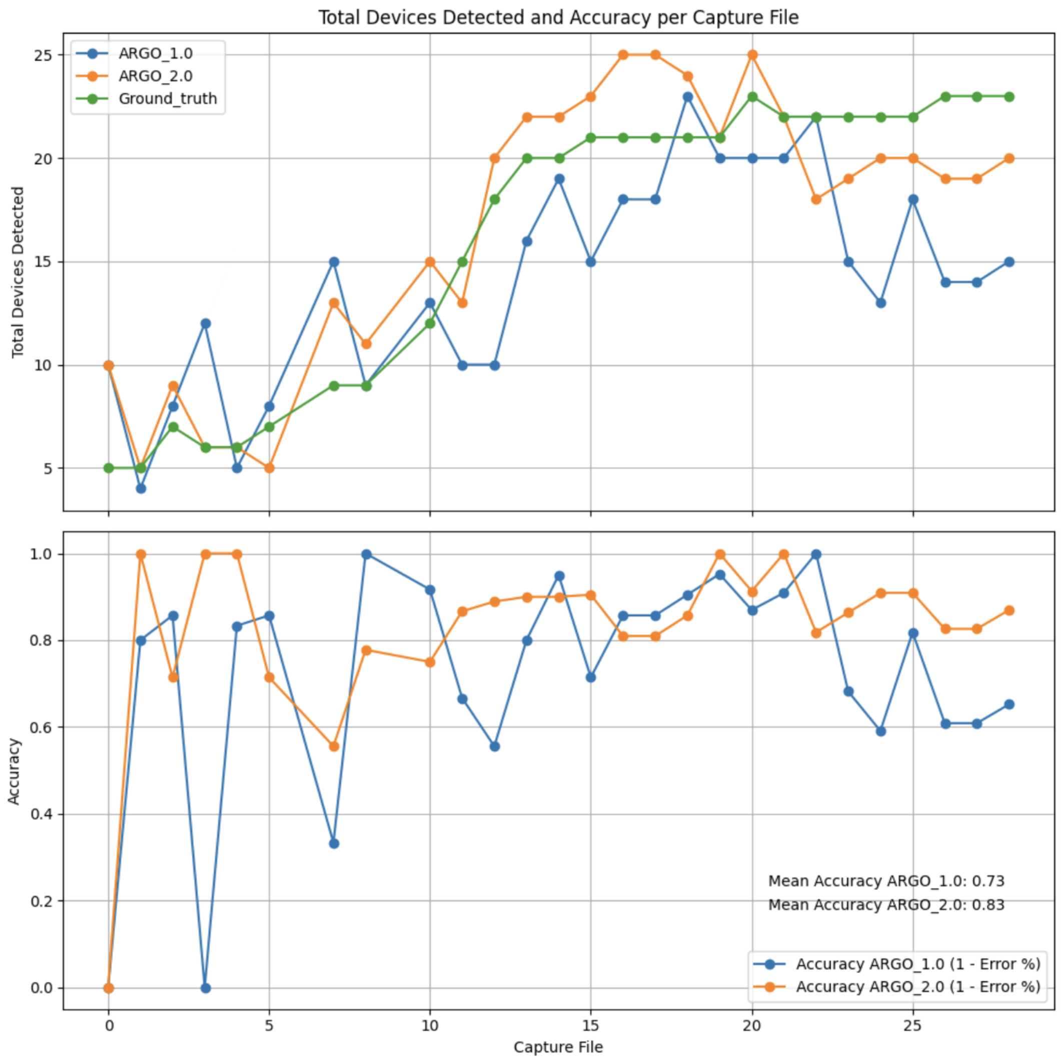


Figure 5.4: Comparison between vers._1 and vers._2 of ARGO

Figure 5.4 effectively shows how the performance of the newly presented version are way better with respect to ARGO. In particular, the older version shows very high peaks in the output, with a quite unstable accuracy. Differently ARGO 2.0 has shown a more stable trend, with a reduced number of peaks. This lead to a better tracking of the ground truth's behavior and to an accuracy constantly higher.

Each one of the used tests, has shown that WiFi-based crowd-counting could provide good performance while preserving users' privacy. The proposed framework can estimate the number of devices within the monitored area with high accuracy, especially compared to its low costs and power consumption.

To obtain further information to enhance the performance of these systems, the next Chapter presents a study of the temporal behavior of probe requests.

Chapter 6

Probe Timing Analysis

In this Chapter a study on the time behavior of probe requests is performed in order to further enhance the performance of WiFi-based crowd-monitoring systems. In particular, Section 6.1 examines the Δt time interval between two groups of probe requests. This analysis aims to verify whether we can employ the IBT as a temporal signature for the clustering algorithm. Section 6.2 talks about the influence that the state of the channel has on the effective probe request throughput. Moreover, in this Chapter we highlight one of the main limitation of probe request when used for crowd-monitoring system, i.e., the lack of ground truth, as no further research is possible without having a large and reliable dataset to be analyzed.

6.1 Inter-Burst Time

According to the research developed by [55], devices send probe requests in groups. Each of them is called "burst" and all the probes within the same one share the same MAC address. Thanks to this behaviour, bursts are easy to identify.

While authors were more interested in how the content of the probe request varies according to the state of the analyzed device, we focus on how the device's state influences the sending rate of probe requests. By analyzing the dataset of singular devices provided by [55], we created histograms to represent the distribution of IBT in each condition.

More in detail, we used bins of 0.5 seconds each, allowing a detailed view of the time distribution within each phase. Figure 6.1 shows the results of this analysis for an Apple iPhone 14. We performed the same analysis for all the other devices under test. Figure 6.2 shows the collection of the results for all analyzed devices. Different colors show the various rates of occurrences of the bins, while we represent on the x-axis all examined devices and phases in the format phase_device.

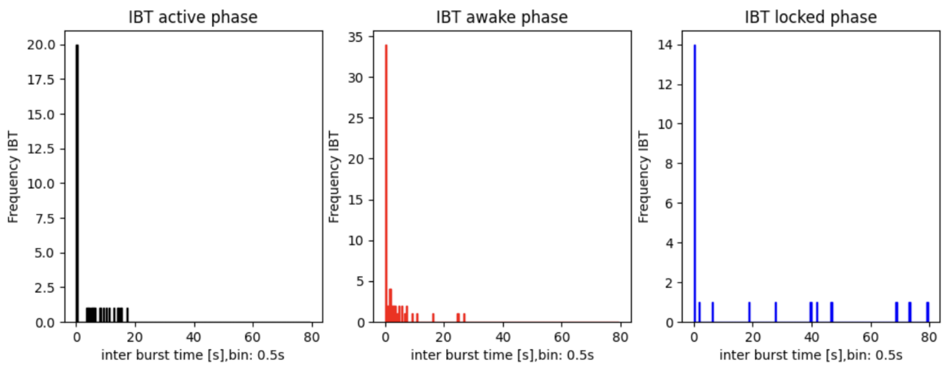


Figure 6.1: Histogram of IBTs of Apple iPhone 14 Pro

The minimal time interval between two burst does not appear to be related to the device vendor; while the maximum Inter-Burst Time depends on the actual state of the device, i.e., devices in locked state are more likely to wait longer times than devices in an active state to send a new group of Probe Requests.

We decided to use a different binning approach to explore low-scale time intervals more deeply: we divided the period from 0 to 5 seconds using 100 points on a logarithmic

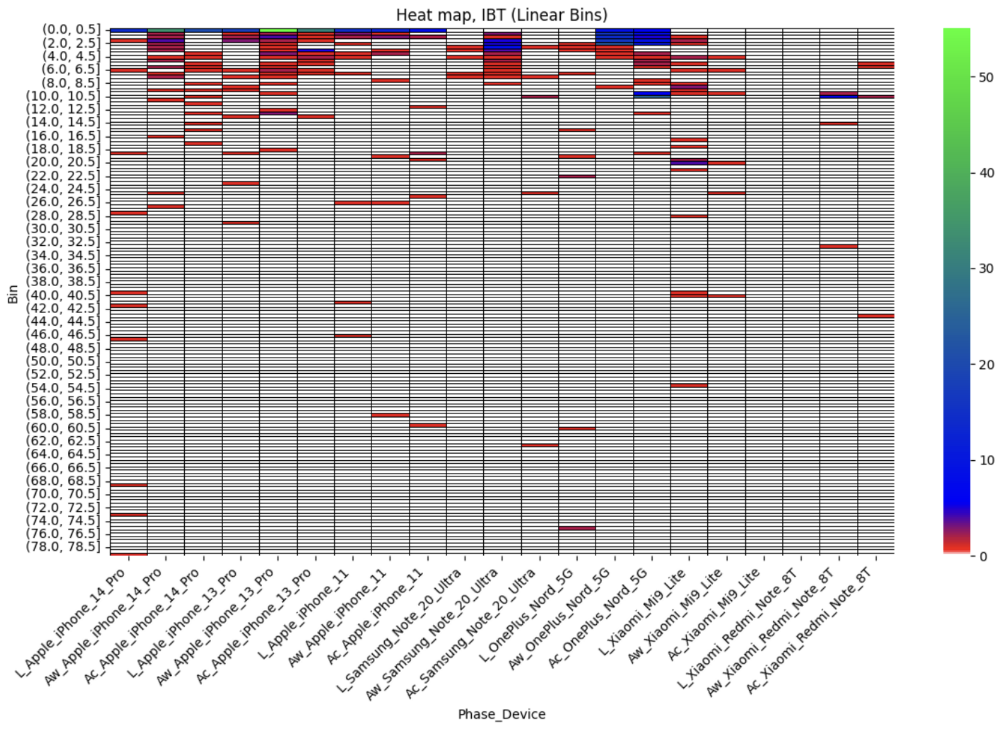


Figure 6.2: Inter-Burst Time for different devices, Linear Binning

scale, performing the same analysis using these new bins. Figure 6.3 shows the heat map obtained.

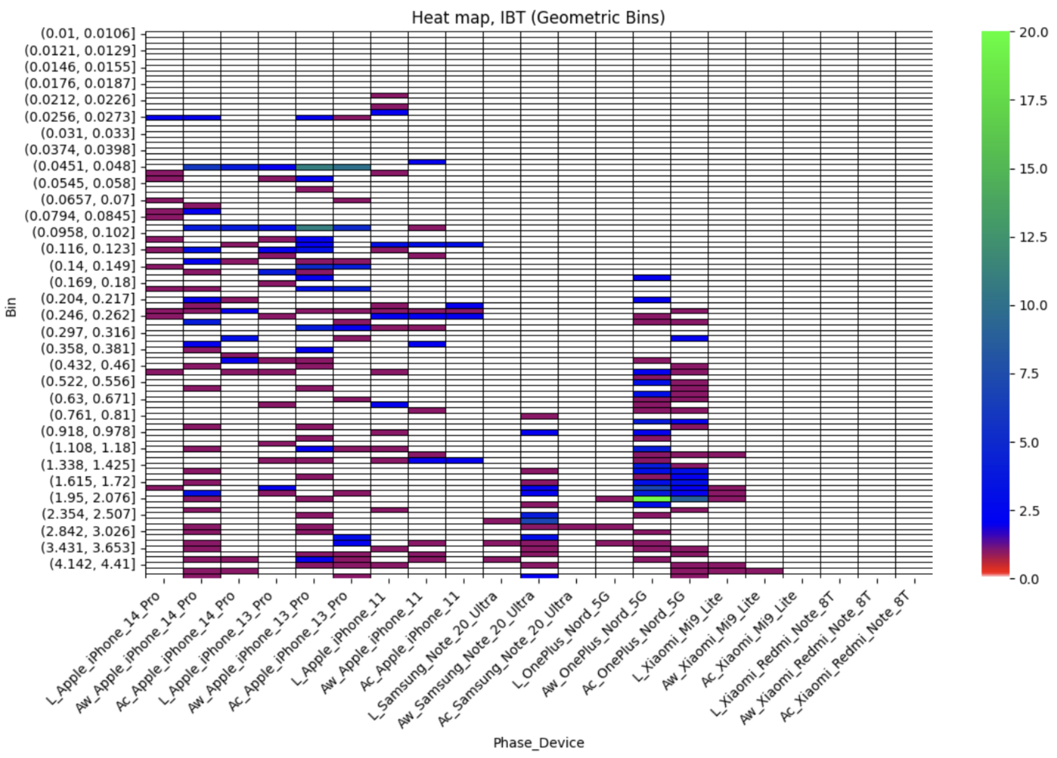


Figure 6.3: Inter-Burst Time for different devices, Logarithmic Binning

This analysis has emphasized that the minimum Δt that devices have to wait before sending a new burst of probes is related to the specific device. At first, we thought to use this information to differentiate bursts based on the source device. If two bursts have a too-low intercurrent time interval between them, they are presumably sent by different source devices. However, we no longer proceed this way due to the lack of ground truth: in a real-world context, we cannot verify whether a single MAC address is being used by one device or another. The impossibility of such verification makes further researches in this direction unfeasible.

Considerations drawn by [82] pivoted a further analysis of probe requests' time behavior. The authors show how active scanning requires a higher level of hardware engagement with respect to passive scanning, leading to a higher power consumption for the devices. It is reasonable to assume that there should be a balance between the number of probe requests within a burst and the burst's sending rates. In particular, we expected to see the IBT increase as the number of frames within each burst increases. In this way, devices could reduce the power consumption due to active scanning.

To verify this, we used the dataset provided by [55], and for each device, we computed the mean IBT, along with the number of frames within each burst. Moreover, we computed the Pearson correlation coefficient between the two variables using (6.1).

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.1}$$

Where:

- x_i, y_i are the values of the two variables in the observation i.
- $\bar{x}, \bar{y}$ represents the means of the two variables.
- n is the number of the observation, in this case the number of devices.

The computed coefficient, denoted as r_{xy}, is a measure of the linear relationship between two continuous variables, i.e., the Inter-Burst Time and the number of frames per burst. Pearson coefficient can have values ranging from -1 to +1, where higher values identify a stronger positive correlation among variables. The analysis shows a positive correlation among the two variables, with a correlation factor of 0.4.

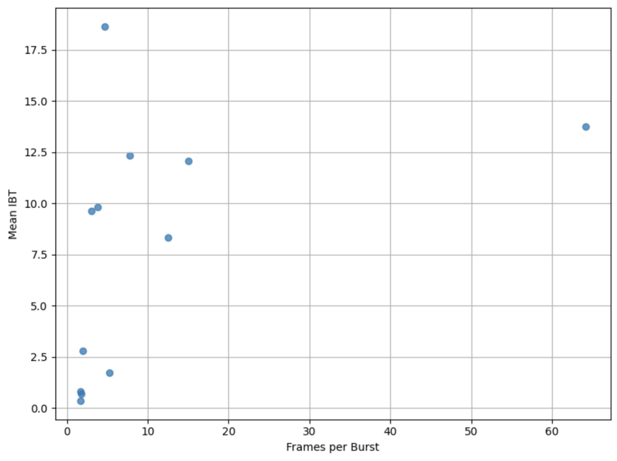


Figure 6.4: Average inter-frame time versus frame count per message

Figure 6.4 shows the graphical representations of data used in this analysis. Each point represents a singular device, with its mean IBT and the number of frames sent for each burst. The relationship is non-linear, and the variance among data is very high. Therefore, the computed Pearson coefficient can only provide us with an idea of the correlation without any chance of drawing any definitive conclusion.

6.2 Channel Occupancy and Throughput Reduction

All analyses conducted so far focus on devices in isolated conditions. However, it may be valuable to verify whether the channel state influences the actual throughput of probe requests. We expect that the actual rate of probes decrease as the channel occupancy increases, due to the increased packet collision rate.

6.2.1 Experimental set-up

The congestion of the WiFi channel can occur in overcrowded environments, thus we emulated this scenario through the use of a flooding technique.

A flooding attack is a type of cyber attack where a system, a network, or a server is inundated with an excessive amount of traffic or requests, aiming to overload the available resources and make them inaccessible to legitimate users. We used a Python script, modified from [83], that performs DoS attacks on 802.11 networks with customized flooding packets, in this case probe requests. To execute the analysis, we employed two Raspberry Pi 4B: the first one was used in monitor mode to sniff all the probe requests in the area, meanwhile the other one executed the flooding attack.

To understand the influence of the congestion of the channel, we sniffed the 802.11 WiFi probe requests traffic in a room 3 times to have a general baseline, which could give us an idea of the nominative operation characteristics. An increasing quantity of traffic has then been generated through the second Raspberry Pi 4B, aiming to occupy the channel for the longest time possible. Table 6.1 summarizes the details about the generated flood traffic in each test.

Test	**Flood messages**	**Frame lenght [B]**	**Channel Occupancy [s]**
1	97	48	0.06
2	720	48	0.27
3	2385	48	0.92
4	8958	48	3.4
5	15685	48	6
6	23482	48	9
7	47674	48	18
8	61643	48	23
9	64744	48	25
10	21 000	285	69
11	34086	285	82

Table 6.1: Information about the generated flood messages

The variation that affects the baseline in the number of probe requests received can give us an idea of the effect of the channel congestion.

6.2.2 Analysis results

Figure 6.5 shows the results of this analysis. On the x-axis, we printed the percentage of time that the channel was occupied by our flood messages, while the y-axis shows the number of effective probe requests detected by the sniffing Raspberry Pi.

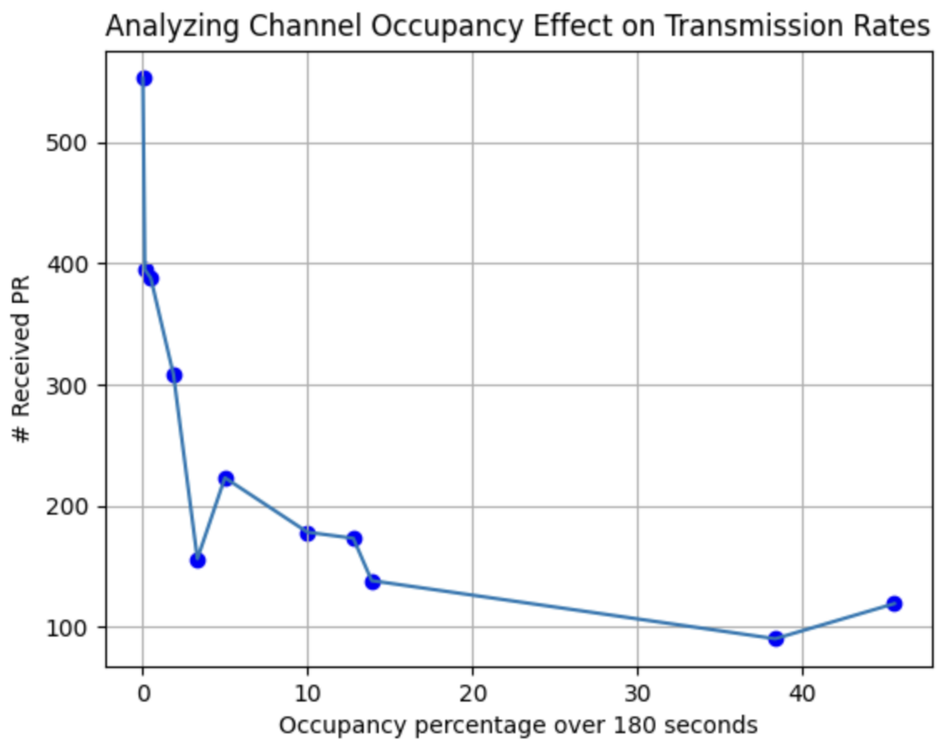


Figure 6.5: Channel occupancy effect on transmission rates

As it can be seen from the Figure, the actual throughput exponentially depends on the occupancy percentage of the channel. The maximum probe request throughput was obtained in a non-congested situation, while it constantly decreases as the congestion increases. The congestion seems to no longer affect the throughput after a certain occupation threshold, around 20%.

This could be due to the channel access standards imposed by IEEE 802.11. In the IEEE 802.11 standard, a back-off mechanism is used to ensure fair access to the channel among all devices. Before accessing the channel, any device first listens if it is free [84]. If the channel is busy, the device waits a random back-off time before attempting to transmit again. This process helps to reduce collisions, leading to more stable throughput levels once the occupancy reaches a certain threshold.

The current clustering-based approaches are not feasible in an overcrowded environment. The rates that relate the number of probes with the number of devices, obtained in isolated conditions, are not significant in channel highly congested, as only a 1% occupancy rate leads to a reduction of the effective throughput by 50%.

6.3 Limitations of the ARGO Algorithm

The approach implemented with ARGO is very high-performing in a poorly congested environment: the number of clusters identified is a good approximation for the number of people in the area. Furthermore, as we do not employ sensitive information for its functioning, these systems can be very effective for daily monitoring applications.

In the case of an overcrowded environment, however, the number of clusters is no longer a good approximation because of the presence of multiple models of the same type. In this case, ARGO obtained the number of devices by dividing the number of probes by some reference rates. This idea is valuable when no other information is available. However, it is based on two non-verified assumptions:

- The Probe Requests' sending rate is constant for each device.
- As the probe request shares information about the device throughput characteristics, devices with similar content in IEs share comparable throughput.

The first assumption can be presumed true for transit environments where channel congestion does not occur frequently. To test this theory, we used one of the sensors employed in Valentino's Park.

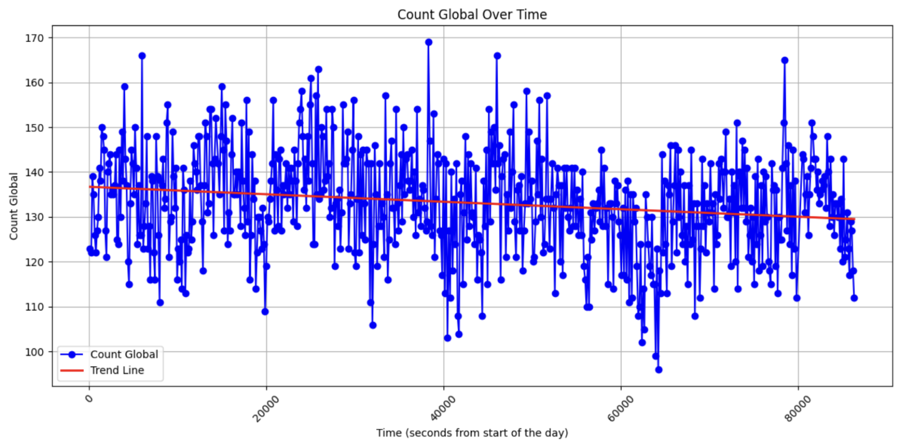


Figure 6.6: Traffic generated by an IoT device over time

In particular, we analyzed the rate of an IoT device in the area, which does not employ MAC address randomization. We monitored the amount of probe request traffic generated by this device over one day. Figure 6.6 shows the result of the analysis. Despite some oscillations, the trend line of this graph shows an almost flat behavior. We can observe a minor diminution of the mean rate in the afternoon hour, but it can be considered negligible with respect to the mean rate. In this case, using reference rates seems to be a good approximation in cases where no other information is available.

The second hypothesis instead revealed a false assumption: analyzing the rate of the devices studied by [55], we noticed that smartphones with similar characteristics, i.e., Apple iPhones, show very different mean sending rates. Considering this, the framework could provide an estimate for the crowd-size very different from the ground-truth data.

Furthermore, the used mean rate cannot reflect the devices' behavior in highly congested environments. As reported in Section 6.2, the probes' sending rate is not constant over time, but it is a nonlinear function of the occupancy of the channel. When numerous devices are present, the sending rate of probes may decrease exponentially. Using the standard approach in this context may be dangerous as it may lead to an underestimation of the crowd size, generating errors in the decision process of stakeholders like first aiders.

To tackle this issue we proposed using an adaptive, situation-dependent rate that varies according to the number of devices in the monitored area. Initially, a tuning period would be necessary. During the frameworks' tuning, the actual number of people present is known and the optimal rate can be computed. The computation can be performed by minimizing the difference with respect to the ground truth. The error can be quantified by any chosen error metrics, such as the MSE (Mean Square Error). Once this value is determined, the system will become autonomous in counting devices.

Our proposed solution loses its flexibility in an overcrowded environment as it can be used only for the situations for which it is tuned. However, it could work even in conditions of high congestion, leading to a more precise crowd-counting in more complex scenarios.

Chapter 7

Conclusion

With this Thesis, we aimed to develop a WiFi-based crowd-monitoring system. The developed system must infer the crowd-size in the monitored area by clustering together 802.11 Probe Request messages. In the first phase of the research, we examined the content of Probe Request messages using a large number of probe requests collected by sensors deployed in Valentino's Park, in Turin. We applied these findings to ARGO [79], a network-based counting system. The analysis led to a new crowd-counting algorithm that utilizes additional fields of probe requests and the OPTICS clustering algorithm to classify and group requests based on the presumed source. The modifications improved crowd counting accuracy, which ranges from 83% to 93%, and enhanced clustering quality.

In the second phase, we studied the temporal behavior of Probe Requests, examining how their sending rate varies according channel congestion and other variables, such as the vendor and the phase of the device. These observations led to a proposal for adapting performance to specific contexts, potentially resulting in a more flexible algorithm. However, this is feasible only with ground truth data, as the rate is tuned by minimizing the error relative to the actual ground truth.

To sum up, this Thesis shows how WiFi-based crowd-monitoring techniques offer a valuable approach to crowd-counting applications. Using Probe Request messages allows to build not only a privacy-preserving system that does not use sensitive information, but also, operates efficiently without requiring costly hardware and consuming minimal power.

7.1 Future Works

The principal limitations encountered in this work are related to the lack of ground truth. Although it is possible to count the number of people within a room using a sniffing system, it is not possible to perfectly isolate the monitored area. Using thresholds on the admissible signal strength can help up to a certain point, but perfect isolation cannot be achieved. The main focus for future research should be to obtain a ground truth as accurate as possible. Without precise reference, further development could be considered only as a speculation without any possible validation test. Furthermore, in creating labeled datasets, different contexts should be considered, ranging from highly populated scenarios to sparsely populated ones.

An extensive study on the probe requests should be done for what concerns their time behavior: from our analysis, their behavior depends on many variables, i.e., their source device, the channel congestion, and the state of the device itself. However, due to the limited number of devices at our disposal, we could not guess any general behavior for what regards temporal patterns.

At the end of Chapter 6 we also suggested a new method for adapting the algorithm's performance in overcrowded environments. This strategy needs to be developed and verified. Future enhancements should implement this mechanism and test its efficacy in real-world scenarios. By solving these issues, WiFi-based crowd monitoring systems can be further enhanced as they can lead to more accurate, reliable solutions for crowd estimation and management.

Bibliography

[1] V. Sivaraman, H. H. Gharakheili, C. Fernandes, N. Clark, T. Karliychuk, Smart iot devices in the home: Security and privacy implications, IEEE Technology and Society Magazine 37 (2) (2018) 71–79.

[2] V. Upadrista, V. Upadrista, The iot standards reference model, IoT Standards with Blockchain: Enterprise Methodology for Internet of Things (2021) 61–86.

[3] R. Rusca, A. Carluccio, C. Casetti, P. Giaccone, Privacy-preserving wifi-based crowd monitoring, Transactions on Emerging Telecommunications Technologies 35 (3) (2024) e4956.

[4] European Parliament, Council of the European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council.
URL `https://data.europa.eu/eli/reg/2016/679/oj`

[5] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, A review of video surveillance systems, Journal of Visual Communication and Image Representation (2021).

[6] Z. Zhang, M. Wang, X. Geng, Crowd counting in public video surveillance by label distribution learning, Neurocomputing (2015).

[7] Z. Fan, H. Zhang, Z. Zhang, G. Lu, Y. Zhang, Y. Wang, A survey of crowd counting and density estimation based on convolutional neural network, Neurocomputing 472 (2022) 224–251.

[8] N. Paragios, V. Ramesh, A mrf-based approach for real-time subway monitoring, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1, IEEE, 2001, pp. I–I.

[9] J. Wen, Z. Zhong, Z. Zhang, L. Fei, Z. Lai, R. Chen, Adaptive locality preserving regression, IEEE Transactions on Circuits and Systems for Video Technology 30 (1) (2020) 75–88.

[10] K. Chen, S. Gong, T. Xiang, C. Change Loy, Cumulative attribute space for age and crowd density estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 2467–2474.

[11] D. Ryan, S. Denman, S. Sridharan, C. Fookes, An evaluation of crowd counting

methods, features and regression models, Computer Vision and Image Understanding 130 (2015) 1–17.

[12] A. B. Chan, N. Vasconcelos, Counting people with low-level features and bayesian regression, IEEE Transactions on image processing 21 (4) (2011) 2160–2177.

[13] H. Idrees, I. Saleemi, C. Seibert, M. Shah, Multi-source multi-scale counting in extremely dense crowd images, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 2547–2554.

[14] Z. Zhao, H. Li, R. Zhao, X. Wang, Crossing-line crowd counting with two-phase deep neural networks, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14, Springer, 2016, pp. 712–726.

[15] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE computer society conference on computer vision and pattern recognition, Ieee, 2005.

[16] P. Sabzmeydani, G. Mori, Detecting pedestrians by learning shapelet features, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.

[17] P. Viola, M. J. Jones, Robust real-time face detection, International journal of computer vision 57 (2004) 137–154.

[18] V. Rabaud, S. Belongie, Counting crowded moving objects, in: 2006 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2006.

[19] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, 1981.

[20] G. J. Brostow, R. Cipolla, Unsupervised bayesian detection of independent motion in crowds, in: 2006 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2006.

[21] A. C. Davies, J. H. Yin, S. A. Velastin, Crowd monitoring using image processing, Electronics & Communication Engineering Journal (1995).

[22] W. Ma, L. Huang, C. Liu, Crowd density analysis using co-occurrence texture features, in: 5th International Conference on Computer Sciences and Convergence Information Technology, IEEE, 2010, pp. 170–175.

[23] A. Albiol, A. Albiol, J. Silla, Statistical video analysis for crowds counting, in: 2009 16th IEEE International Conference on Image Processing (ICIP), IEEE, 2009, pp. 2569–2572.

[24] A. B. Chan, N. Vasconcelos, Counting people with low-level features and bayesian regression, IEEE Transactions on image processing (2011).

[25] A. B. Chan, N. Vasconcelos, Bayesian poisson regression for crowd counting, in: 2009

IEEE 12th International Conference on Computer Vision, 2009.

[26] Y. Cong, H. Gong, S.-C. Zhu, Y. Tang, Flow mosaicking: Real-time pedestrian counting without scene-specific learning, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[27] K. Chen, C. C. Loy, S. Gong, T. Xiang, Feature mining for localised crowd counting., in: Bmvc, 2012.

[28] M. A. Khan, H. Menouar, R. Hamila, Revisiting crowd counting: State-of-the-art, trends, and future perspectives, Image and Vision Computing 129 (2023) 104597.

[29] S. Mostafa, F.-X. Wu, Chapter 3 - diagnosis of autism spectrum disorder with convolutional autoencoder and structural mri images, in: A. S. El-Baz, J. S. Suri (Eds.), Neural Engineering Techniques for Autism Spectrum Disorder, Academic Press, 2021, pp. 23–38.

[30] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, F. Boussaid, Chapter 5 - computer vision for humanâmachine interaction, in: M. Leo, G. M. Farinella (Eds.), Computer Vision for Assistive Healthcare, Computer Vision and Pattern Recognition, Academic Press, 2018, pp. 127–145.

[31] K. Santosh, N. Das, S. Ghosh, Chapter 2 - deep learning: a review, in: K. Santosh, N. Das, S. Ghosh (Eds.), Deep Learning Models for Medical Imaging, Primers in Biomedical Imaging Devices and Systems, Academic Press, 2022, pp. 29–63.

[32] C. Wang, H. Zhang, L. Yang, S. Liu, X. Cao, Deep people counting in extremely dense crowds, in: Proceedings of the 23rd ACM international conference on Multimedia, 2015, pp. 1299–1302.

[33] L. Deng, Q. Zhou, S. Wang, J. M. Górriz, Y. Zhang, Deep learning in crowd counting: A survey, CAAI Transactions on Intelligence Technology (2023).

[34] K. Nakamura, H. Zhao, X. Shao, R. Shibasaki, Human sensing in crowd using laser scanners, Laser Scanner Technology (2012) 15–32.

[35] V. Tsakanikas, T. Dagiuklas, Video surveillance systems-current status and future trends, Computers & Electrical Engineering (2018).

[36] V. K. Sharma, R. N. Mir, C. Singh, Scale-aware cnn for crowd density estimation and crowd behavior analysis, Computers and Electrical Engineering 106 (2023) 108569.

[37] F. Nilsson, Intelligent network video: Understanding modern video surveillance systems, crc Press, 2008.

[38] Preliminary verification. collection, analysis and processing of data, through the installation of equipment, for marketing and market research purposes.
URL `http://www.garanteprivacy.it/home/docweb/-/docweb-display/docweb/9022068`

[39] Il pir motion detector â un sensore di movimento per arduino e raspberry pi.

URL https://www.meccanismocomplesso.org/pir-motion-detector/

[40] A. Shokrollahi, J. A. Persson, R. Malekian, A. Sarkheyli-Hägele, F. Karlsson, Passive infrared sensor-based occupancy monitoring in smart buildings: A review of methodologies and machine learning approaches, Sensors 24 (5) (2024) 1533.

[41] I. Udrea, N. D. Simion, C. G. Alionte, V. Ionut, V. F. K. Gheorghe, S. Petrache, Counting versus detection in an fm application that deals with rooms reservation, Journal of Eastern Europe Research in Business and Economics, Norristown, PA, USA (2022).

[42] P.-R. Tsou, C.-E. Wu, Y.-R. Chen, Y.-T. Ho, J.-K. Chang, H.-P. Tsai, Counting people by using convolutional neural network and a pir array, in: 2020 21st IEEE International Conference on Mobile Data Management (MDM), IEEE, 2020, pp. 342–347.

[43] M. Kuki, H. Nakajima, N. Tsuchiya, Y. Hata, Multi-human locating in real environment by thermal sensor, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, IEEE.

[44] J. Yun, D. Kim, D. M. Kim, T. Song, J. Woo, Gan-based sensor data augmentation: Application for counting moving people and detecting directions using pir sensors, Engineering Applications of Artificial Intelligence 117 (2023) 105508.

[45] Optical encoders and lidar scanning.
URL https://www.renishaw.com/it/optical-encoders-and-lidar-scanning--39244

[46] Goetting ist partner von velodyne lidar.
URL https://www.goetting.de/news/2018/goetting-partner-von-velodyne#main

[47] S. T. Kouyoumdjieva, P. Danielis, G. Karlsson, Survey of non-image-based approaches for counting people, IEEE Communications Surveys & Tutorials 22 (2) (2019) 1305–1336.

[48] J.-S. Yoon, S.-H. Bae, T.-y. Kuc, Human recognition and tracking in narrow indoor environment using 3d lidar sensor, in: 2020 20th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2020, pp. 978–981.

[49] M. Bouazizi, C. Ye, T. Ohtsuki, 2-d lidar-based approach for activity identification and fall detection, IEEE Internet of Things Journal 9 (13) (2021) 10872–10890.

[50] F. Luo, S. Poslad, E. Bodanese, Temporal convolutional networks for multiperson activity recognition using a 2-d lidar, IEEE Internet of Things Journal 7 (8) (2020) 7432–7442.

[51] J. Bray, C. F. Sturman, Bluetooth 1.1: connect without cables, pearson Education, 2001.

[52] V. Kostakos, T. Camacho, C. Mantero, Wireless detection of end-to-end passenger trips on public transport buses, in: 13th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2010, pp. 1795–1800.

[53] M. Versichele, T. Neutens, M. Delafontaine, N. Van de Weghe, The use of bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the ghent festivities, Applied Geography 32 (2) (2012) 208–220.

[54] N. Abedi, A. Bhaskar, E. Chung, Bluetooth and wi-fi mac address based crowd collection and monitoring: Benefits, challenges and enhancement, in: Australasian Transport Research Forum 2013 Proceedings, Australasian Transport Research Forum, 2013, pp. 1–17.

[55] R. Rusca, F. Sansoldo, C. Casetti, P. Giaccone, What WiFi probe requests can tell you (2023).

[56] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) (2012) 1–2793.

[57] Chapter 4. 802.11 framing in detail.
URL `https://www.oreilly.com/library/view/80211-wireless-networks/0596100523/ch04.html`

[58] M. Cunche, I know your mac address: targeted tracking of individual using wi-fi, Journal of Computer Virology and Hacking Techniques 10 (2014) 219–227.

[59] C. Matte, M. Cunche, Spread of mac address randomization studied using locally administered mac addresses use historic, Ph.D. thesis, Inria Grenoble Rhône-Alpes (2018).

[60] M. Uras, R. Cossu, E. Ferrara, O. Bagdasar, A. Liotta, L. Atzori, Wifi probes sniffing: an artificial intelligence based approach for mac addresses de-randomization, in: 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), IEEE, 2020, pp. 1–6.

[61] L. Pintor, L. Atzori, Analysis of wi-fi probe requests towards information element fingerprinting, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 3857–3862.

[62] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, 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, 2016, pp. 413–424.

[63] A. K. Mishra, A. C. Viana, N. Achir, Bleach: From wifi probe-request signatures to mac association, Available at SSRN 4673079.

[64] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, J. Stefa, Signals from the crowd: uncovering social relationships through smartphone probes, in: Proceedings of the 2013 conference on Internet measurement conference, 2013, pp. 265–276.

[65] T. Bravenec, J. Torres-Sospedra, M. Gould, T. Fryza, Uji probes revisited: Deeper dive into the dataset of wi-fi probe requests, IEEE Journal of Indoor and Seamless Positioning and Navigation (2023).

[66] M. Nitti, F. Pinna, L. Pintor, V. Pilloni, B. Barabino, iabacus: A wi-fi-based automatic bus passenger counting system, Energies 13 (6) (2020) 1446.

[67] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, D. Brown, A study of mac address randomization in mobile devices and when it fails, arXiv preprint arXiv:1703.02874 (2017).

[68] Trialsnet EU project.
URL `https://trialsnet.eu/`

[69] K. Gebru, M. Rapelli, R. Rusca, C. Casetti, C. F. Chiasserini, P. Giaccone, Edge-based passive crowd monitoring through wifi beacons, Computer Communications 192 (2022) 163–170.

[70] D.2. tshark: Terminal-based wireshark.
URL `https://www.wireshark.org/docs/wsug_html_chunked/AppToolstshark.html`

[71] Grafana.
URL `https://grafana.com`

[72] D. Guo, J. Wu, H. Chen, Y. Yuan, X. Luo, The dynamic bloom filters, IEEE Transactions on knowledge and data engineering 22 (1) (2009) 120–133.

[73] A. Broder, M. Mitzenmacher, Network applications of bloom filters: A survey, Internet mathematics 1 (4) (2004) 485–509.

[74] A. Broder, M. Mitzenmacher, Network applications of bloom filters: A survey, Internet mathematics 1 (4) (2004) 485–509.

[75] A. Carluccio, Privacy-preserving people flow monitoring with bloom filters, available at `https://webthesis.biblio.polito.it/28442/` (2023).

[76] G. Bianchi, L. Bracciale, P. Loreti, "better than nothing": Privacy with bloom filters: To what extent?, in: International Conference on Privacy in Statistical Databases, Springer, 2012, pp. 348–363.

[77] R. Rusca, A. Carluccio, D. Gasco, P. Giaccone, Privacy-aware crowd monitoring and wifi traffic emulation for effective crisis management, in: 2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), IEEE, 2023, pp. 1–6.

[78] Meross presa smart wifi, presa intelligente italiana, smart plug con monitoraggio energia 16a 3840w, funzione timer, compatibile con amazon alexa, smartthings, google assistant, app controllo remoto.
URL https://www.amazon.it/intelligente-controllo-compatibile-Assistant-meross/dp/B07GSVQBMY/ref=asc_df_B07GSVQBMY/?tag=googshopit-21&linkCode=df0&hvadid=700878162198&hvpos=&hvnetw=g&hvrand=15913274524022041585&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1008141&hvtargid=pla-540207773169&mcid=7d75eaf95a1e3d39b769efb0950fb3ce&gad_source=1&th=1

[79] D. Gasco, Enhancing crowd-monitoring through wifi fingerprint analysis, available at https://webthesis.biblio.polito.it/28445/ (2023).

[80] L. Pintor, L. Atzori, A dataset of labelled device wi-fi probe requests for mac address de-randomization, Computer Networks 205 (2022) 108783.

[81] Confront â challenge on wifi frame fingerprinting for people counting and tracking.
URL https://sites.unica.it/net4u/confront-challenge-on-wifi-frame-fingerprinting-for-people-counting-and-trackingconfront/

[82] Y. Choi, S. Choi, Energy-aware wlan scanning in integrated ieee 802.16 e/802.11 networks, Computer Communications 32 (15) (2009) 1588–1599.

[83] dos-tester-802.11.
URL https://github.com/oz9un/dos-tester-802.11

[84] M. Natkaniec, A. R. Pach, An analysis of the backoff mechanism used in ieee 802.11 networks, in: Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications, IEEE, 2000, pp. 444–449.