Privacy-Preserving WiFi Fingerprint-Based People Counting for Crowd Management

Computer Communications 225 (2024) 339–349

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Privacy-preserving WiFi fingerprint-based people counting for crowd management

Riccardo Rusca *, Diego Gasco, Claudio Casetti, Paolo Giaccone

Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Crowd monitoring
People counting
WiFi
Probe request
Bloom filter
Anonymization noise
DBSCAN
Clustering

ABSTRACT

The practice of people counting serves as an indispensable tool for meticulously monitoring crowd dynamics, enabling informed decision-making in critical situations, and optimizing the management of urban spaces, facilities, and services. Beyond its fundamental role in safety and security, tracking people's flows has evolved into a necessity for diverse business applications and the effective administration of both outdoor and indoor urban environments. In the ongoing exploration of the study, emphasis is placed on employing a passive counting technique. This method leverages WiFi probe request messages emitted by smart devices to assess the number of devices, providing a reliable estimate of the number of people in a specific area. However, it is crucial to acknowledge the dynamic landscape of privacy regulations and the concerted efforts by leading smart-device manufacturers to fortify user privacy, as evidenced by the adoption of MAC address randomization. In response to these considerations, an enhanced iteration of the WiFi traffic generator has been introduced. This upgraded version is designed to generate realistic datasets with ground truth, aligning with the evolving privacy landscape. Additionally, leveraging a profound understanding of probe requests and the capabilities of the designed generator, a novel crowd monitoring solution that incorporates machine learning techniques, named **ARGO**, has been developed. This innovative approach effectively addresses challenges posed by randomized MAC addresses, incorporating Bloom filters to ensure a formal "deniability" that complies with stringent regulations, including the European GDPR (European Parliament, Council of the European Union, Regulation (EU), 2016). The proposed solution adeptly addresses the pivotal task of people counting by harnessing WiFi probe request messages. Significantly, it prioritizes users' privacy, aligning with the foundational principles outlined in regulations such as the European GDPR.

1. Introduction

The shifting dynamics of public gatherings in the wake of the COVID-19 pandemic have prompted a reevaluation of crowd management practices. In the wake of the post-pandemic era, the return of large-scale events introduces challenges related to security threats and potential congestion. Persistent limitations on gathering sizes underscore the critical need for effective crowd management, placing increased responsibility on authorities to ensure public safety. In this evolving context, crowd management analytics have become indispensable tools for adaptive decision-making in response to dynamic circumstances. Effectively assessing crowd dynamics, estimating resource requirements, and optimizing emergency response efforts become pivotal for the successful management of large-scale events. However, counting and tracking individuals within large gatherings or chaotic scenarios persist as challenges, demanding innovative solutions that align with today's privacy regulations and the heightened focus on user privacy.

Traditional crowd monitoring techniques, ranging from ubiquitous surveillance cameras to sophisticated LiDAR and infrared systems, have long served as foundational tools for gathering insights into crowd dynamics. In addition, the utilization of WiFi and Bluetooth fingerprints has provided valuable data for tracking individuals within a given space. However, the effectiveness of these methods is now being challenged by the advent of stringent data protection regulations, exemplified by the European General Data Protection Regulation (GDPR) [1]. The GDPR, enacted to safeguard individuals' privacy rights, imposes rigorous guidelines on collecting, processing, and storing personal data. Furthermore, the increased importance on user privacy, especially targeted by the leading smart device manufacturers, adds a layer of complexity to traditional monitoring methods. Wishing to avoid being blamed for the lack of privacy concerns, these manufacturers have started adopting features like MAC address randomization. This intentional obfuscation of device identifiers poses a significant challenge

* Corresponding author.
E-mail address: riccardo.rusca@polito.it (R. Rusca).

https://doi.org/10.1016/j.comcom.2024.07.010
Received 6 March 2024; Received in revised form 17 May 2024; Accepted 17 July 2024
Available online 25 July 2024

to conventional crowd monitoring systems that rely on consistent and unique identifiers for accurate tracking.

This study proactively addresses this shifting landscape, introducing an advanced WiFi traffic generator crafted to generate realistic datasets and a novel crowd monitoring framework, called ARGO. The proposed crowd counting architecture leverages the capabilities of a Raspberry Pi IoT device, employing it to intercept broadcast WiFi probe request packets, and a machine learning algorithm specifically tailored to mitigate the challenges posed by MAC address randomization. More in detail, the designed methodology incorporates a clustering method, utilizing the DBSCAN algorithm to effectively classify probe requests belonging to a single device. This innovation contributes to the precision and reliability of the crowd counting approach, even in the face of randomized MAC addresses. Furthermore, to ensure the secure transmission of results over the network, ARGO leverage the *Anonymization noise* technique, introduced in [2]. This technique fortifies Bloom filter data structures containing detected MAC addresses, adding an extra layer of privacy protection. Operating under the concept of 1-deniability, the proposed methodology involves the secure transmission of Bloom filters over the network. The resulting server-side intersections allow for an in-depth analysis of people flow within specific time windows, ensuring a robust, privacy-conscious crowd monitoring solution.

The main contributions can be summarized as follows:

- Development of an advanced realistic **WiFi Probe Request Generator**: an upgraded iteration of the WiFi probe request generator has been introduced, capable of generating realistic datasets aligned with evolving privacy regulations.
- Introduction of **ARGO Crowd Monitoring Framework**: a novel crowd monitoring solution, named ARGO, has been developed. This framework leverages machine learning techniques to address challenges posed by randomized MAC addresses, ensuring accurate people counting while prioritizing user privacy.
- Creation of a cost-effective **Counting Architecture**: a novel architecture has been introduced, prioritizing cost-effectiveness without compromising on functionality or accuracy. This architecture encompasses efficient hardware and software components, and it is able to work in every weather condition.
- **Comprehensive Testing and Validation**: the testing and validation phase involved a thorough examination of the developed crowd-monitoring framework. This included the utilization of various datasets, ranging from synthetic data to realistic datasets collected from real-world environments.

The rest of the paper is organized as follows: Section 2 explores pertinent related work, Section 3 delves into the preliminaries and enhanced version of the probe request generator, while Section 4 describe the data handling in a GDPR-compliant way. After that, Section 5 presents the proposed counting framework, that leverages Clustering technique. Section 6 provides an overview of the comprehensive deployed architecture, followed by Section 7, which unveils the results obtained from real-world and synthetic datasets. Finally, Section 8 offers conclusive remarks, summarizing the contributions of the proposed crowd monitoring solution, and the future directions.

2. Related works

In recent years, numerous approaches have been suggested to address the challenge of counting people. Recent research has highlighted the feasibility of utilizing LiDAR sensors to detect individuals based on the distinct temperatures detected in scanned areas. Works such as [3] have demonstrated the tremendous potential of this technology, yet they have also brought attention to limitations such as the involvement of expensive hardware and the complex management of these sensors. Another well-known approach involves the use of multi-camera surveillance systems, combined with advanced deep learning techniques. Specifically, in [4,5], the authors employ a multi-camera surveillance system that integrates partial body detection and extensively utilizes computer vision frameworks like YOLO [6]. While these methods deliver accurate counts of individuals in examined areas, they present challenges related to user-sensitive data and the limited range of environments in which they can be applied.

In recent studies [7], a novel approach to crowd monitoring has been introduced. This technique leverages WiFi communication messages, specifically honing in on the probe request frames transmitted by the WiFi interfaces of devices searching for available access points to establish a connection. This method involves collecting and tracking WiFi fingerprints from mobile devices, both in indoor and outdoor settings. Importantly, compared to other solutions, it requires cheaper equipment and advances new techniques for preserving user privacy. Nowadays, several methodologies are delving into the utilization of artificial intelligence algorithms to categorize probe requests, enabling the identification of individual devices. In [8], the authors focus on implementing a state-machine framework named Sherlock. This framework exploits the analysis and association of probe requests detected in predefined time windows of one minute, providing an estimation of the number of individual devices in each time interval. Conversely, recent literature introduces frameworks that employ both traditional data mining techniques. Examples of these methodologies can be found within [9–11], where authors utilize clustering algorithms to extract groups of messages, each corresponding to a detected device. The authors focus on key probe request fields, such as sniffing timestamp, throughput capabilities, MAC Addresses, and transmission power (RSSI).

Clustering-based techniques possess the capability to group probe requests based on predefined metrics, offering a flexible approach to pattern recognition. A distinctive characteristic of these techniques is that the number of clusters formed is not predetermined before the process but emerges organically as the outcome. Consequently, specifying the number of groups becomes a parameter that cannot be predefined for the algorithm. In light of this consideration, traditional clustering methods like K-Means, which depend on a predetermined number of clusters, are avoided. Instead, ARGO is designed to favor density-based clustering algorithms. These density-based methods, exemplified by algorithms such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [12] and OPTICS (Ordering Points To Identify the Clustering Structure) [13], are particularly well-suited for identifying clusters of arbitrary shapes and sizes. Unlike K-Means, which assumes a pre-established number of clusters and relies on distance metrics, density-based algorithms discern clusters based on the density of data points. This characteristic renders them more robust in handling intricate patterns and mitigating noise within the dataset provided as input. The adaptability of density-based algorithms is especially valuable when dealing with real-world scenarios where the number and shapes of clusters may vary, offering a more versatile and reliable solution for the developed clustering-based crowd counting methodology.

Modern machine learning approaches have also been investigated, for example, in [14], authors focus on overcoming the MAC address randomization through a neural network that has the objective of associating different probe requests to one device.

The utilization of WiFi probe requests presents significant challenges, particularly in the fine-tuning of machine learning models and the validation of systems. These challenges stem from the absence of collections of probe requests with associated information about the number of devices that generated them, commonly referred to as ground truth. Acquiring this crucial data involves employing approximated methods, which are inherently complex, particularly in densely populated environments. Illustrating the essential nature of this ground truth data and, simultaneously, the formidable challenge of obtaining it with precision, is exemplified in the work outlined in [15]. In this study, a meticulously crafted dataset was curated, comprising

probe requests from 21 devices. Each device underwent thorough analysis within an anechoic chamber, where packet transmissions were measured across various phases of activity. The significance of this endeavor lay in the complete isolation of the chamber from other electronic devices equipped with WiFi interfaces, ensuring measurements could be conducted without interference.

Recent regulations, notably the General Data Protection Regulation (GDPR) [1], impose significant challenges on the storage and management of sensitive information. A specific concern arises from the continuous emission of probe request messages by mobile devices, containing crucial information such as MAC addresses, essential for device identification and monitoring. It is noteworthy that the GDPR categorizes MAC addresses as personal data [16]. Even with the widespread adoption of privacy measures like MAC address randomization by most devices, as detailed in [17], compliance with GDPR regulations remains a paramount consideration.

To address the privacy concerns inherent in handling such data, ARGO incorporates a specific data structure designed to store collected MAC addresses while adhering to GDPR rules — the Bloom Filter [18]. This particular data structure serves as a space-efficient probabilistic tool for membership queries. Studies in the literature [19,20] have demonstrated that, through the application of symmetric homomorphic encryption, these structures allow for the anonymization of stored MAC addresses.

Finally, it is worth mentioning that a preliminary version of some parts of this work has appeared in [2,21]. Specifically, the introduction of the first version of the probe request generator capable of producing realistic traces with both probe request messages and the essential ground truth, and the concept of *Anonymization noise* applied to Bloom filters. In this current study, an improved iteration of the probe request generator is unveiled, featuring additional enhancements to render generated traces more faithful to real-world scenarios. Moreover, a novel framework that capitalizes on advanced clustering techniques and incorporates new features is introduced, specifically it focuses on the frequency of message delivery and information gleaned from individual device models about their behavior under diverse network conditions. Importantly, ARGO is designed in strict adherence to the robust guidelines set forth by GDPR.

3. Probe request generator

As previously mentioned, the proposed approach relies on WiFi signal reception and processing to estimate the presence of smart devices such as smartphones, tablets, laptops, and smartwatches. This method involves scanning the WiFi spectrum in order to capture WiFi packets transmitted by smart devices. When the WiFi interface is activated, it emits a burst of broadcast messages, called WiFi probe requests, to locate nearby Access Points (APs). In the following, an overview of key WiFi features is provided, which are exploited to extract valuable information from the transmitted messages. Then, the Probe Request Generator is described in detail.

3.1. Probe request frames

The identification and the subsequent connection to a network, pass through a series of WiFi management frames called probe requests. If a device has its WiFi interface activated, it automatically starts a discovery phase for available Access Points (APs) by broadcasting this type of message. Even after the device connection to an AP, the probe request sending process still continues with the aim of identifying other available APs. Until 2014, it was common to find the original MAC address as the device identifier within probe requests, and this posed significant privacy issues for users, considering its globally unique nature. In the past years, vendors have started to adopt MAC address randomization as a privacy-enhancing measure, as underlined in a study [22] and analyzed in the research [17]. While some devices completely randomize the 64-byte MAC address, others choose to randomize only the second half while retaining the first 3 bytes, known as the Organizationally Unique Identifier (OUI). This new method aims to advance user protection against possible malicious attacks since it would be easy to use the original MAC address of a device to commit various crimes, such as device tracking or spoofing. It is important to underline that when a device establishes an association, it continues to use the last MAC address employed for communication with the Access Point (AP) with which it is connected. However, with other access points, the device still adopts random MAC addresses. For crowd monitoring research, the adoption of this protection measure poses substantial challenges, since the collected traces now may present thousands of different MAC addresses even if the actual devices present in the sniffing area are many fewer.

3.2. Probe request generator

To address the challenge of MAC address randomization, various machine learning techniques were employed. The goal was to discern whether different MAC addresses could be attributed to the same device. Achieving this, required access to datasets of varying sizes to train and test the machine learning algorithms effectively. Getting large datasets proved relatively straightforward, however obtaining accurate ground truth data to accompany these datasets presented a significant hurdle. In light of this challenge, a unique realistic probe request generator has been created, an initial version has been introduced in [2]. This generator enables the production of an abundant supply of realistic traces while also furnishing the crucial ground truth data required. Following the initial public release, potential for further improvement was identified. Consequently, the generator was enhanced, incorporating new features to refine its capabilities and further advance research in this domain.

3.2.1. Enhanced frame generation

First of all, the construction of the probe request frames was delegated to the Python library called Scapy [23]. This solution facilitates adherence to the 802.11 standard, simplifying the process of crafting network messages.

Additionally, after conducting some analysis, a precise pattern for the management of frame sequence numbers was identified. Specifically, it was observed that the initial frame in each burst displays a randomly chosen sequence number that is independent of previous frames, while subsequent frames within the same burst have sequence numbers that increment sequentially by one. This behavior was incorporated into the new frame creation process.

In the generator's database, each device is marked with a flag indicating whether it employs MAC address randomization. When a device does not employ randomization, its MAC address is constructed using the vendor's specific Organizationally Unique Identifier (OUI) for the initial 24 bytes, with random bytes used for the remaining portion. It is noteworthy that all packets sent by these kinds of device share the same MAC address.

3.2.2. Throughput capabilities

Each device model encapsulates a set of capabilities within its probe requests, pertaining to throughput, such as: VHT (Very High Throughput), HT (High Throughput), and Extended. These capabilities provide insight into how the specific model operates and its potential data transfer speeds. VHT indicates support for the highest throughput, typically associated with 802.11ac and newer standards, while HT denotes support for high throughput, commonly found in 802.11n devices. The "Extended" capability implies additional features or enhancements beyond the standard throughput specifications. These capabilities are crucial for network management and optimization, allowing efficient utilization of available bandwidth and resources.

3.2.3. Added special fields: WPS and UUID-E

Probe requests can sometimes contain the WPS with the UUID-E in the header and the SSID in the payload. According to [24], it is evident that only a small number of messages include all three of these fields. Specifically, the combination of WPS and UUID-E is found in only 11% of the cases analyzed, and there is no precise statistical information available for the SSID field.

As a result, when a new device is powered on in the emulated environment, an extra option is now available: there is an 11% chance that the device will be assigned randomly generated WPS and UUID-E values. If these values are assigned to a particular device, they are subsequently included in the packet creation process, and placed inside the headers of the packets.

3.2.4. Probe collision avoidance

One of the primary challenges in generating packet traces is managing the arrival times of messages. Particularly in heavily congested environments, the possibility of two probe requests, sent from different devices, reaching the sniffer simultaneously cannot be ignored. In the generated *.pcap* file, this results in two frames having the same timestamp. In real systems, these occurrences are referred to as *collisions* and are typically resolved through backoff-based recovery procedures in case of unicast traffic, while they may go unnoticed for broadcast data.

To address the potential for collisions and ensure that synthetic packets are not lost, an additional feature has been incorporated into the system. Specifically, when a new event of type *SendPacket* is about to be added to the event list, it checks for possible time overlaps with existing events of the same type. If no collision is detected, the event is added as usual. However, if a collision is identified, a mechanism is activated to adjust the timing.

In these cases, the execution time of the new event is shifted forward by a very small increment, specifically, *0.001* s. This minimal adjustment is chosen to have the least possible impact on the originally scheduled arrival time of the packet. Moreover, the order of magnitude around milliseconds was driven by the typical timing between different packets within the same burst, which tends to fall within the millisecond range, between values of 20 to 50 ms. After this adjustment, the new event is inserted into the event list as usual. The cumulative time by which the new event has been changed is recorded as the return value. This aids in synchronizing subsequent packets within the ongoing burst to the revised arrival time established for the preceding packet. Once these necessary time adjustments are made, the event is added to the simulator's list of events.

However, when dealing with thousands of simulated devices, it is important to address a potential issue related to time shifting. There is a risk of encountering an endless chain of collision events, disrupting the perception of time with a significant accumulation of delays. To mitigate this, the generator implements a safeguard mechanism. A specific threshold is established: if the number of shifts surpasses it, the packet is dropped. With a time factor of 0.001 s, the threshold is set to 10, ensuring that the variance between the originally scheduled time and the adjusted time, caused by collisions, remains within one hundredth of a second. This approach aims to balance collision resolution while mitigating the risk of excessive adjustments disrupting packet timing integrity.

Fig. 1 illustrates the collision management inside the Event Queue of the probe request generator. Whenever a new probe request (PR3) is generated by a device, the generator attempts to insert the event at the head of the Event Queue. However, if its associated time coincides with that of another probe request, such as PR1 in this example, a collision is detected. In response, the time of PR3 is adjusted by a time shift of 0.001 s, iterating this adjustment process until a vacant time slot is identified, thereby enabling the insertion of the new probe request into the queue.

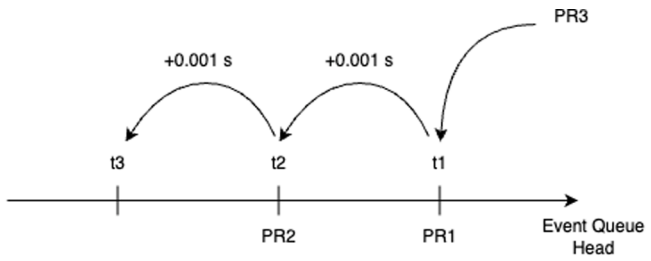

Fig. 1. Collision management for a probe request PR3, when trying to insert it in the Event Queue.

4. GDPR-compliant data handling and storage

To store and analyze the flow of people efficiently, it is essential to consolidate data collected from multiple access points into a central server for information extraction. However, this process must adhere to GDPR-compliant data handling. To address this challenge, inspiration was drawn from the work of the authors in [25], where they introduced a pivotal concept for safeguarding anonymity: γ-deniability.

In previous works, presented in [2], and explored in depth in [21], the concept of γ-deniability applied specifically to Bloom filters was harnessed, with a focus on utilizing $\gamma = 1$. As a reminder, Definitions 1 and 2 delineate the fundamental notions of the hiding set and γ-deniability. These concepts serve as crucial pillars in the understanding and implementation of the privacy-preserving techniques, called **anonymization noise**.

Definition 1 (*taken from [25]* **Hiding Set).** A set V is called *Hiding Set* for a Bloom filter $BF(S)$ if V contains all the elements $v_i \in U$ such that $v_i \notin S$ and a query for v_i in the Bloom filter returns 1. Where $|U|$ represents a large set, approximately equal to $2^{48} \approx 2.8\ 10^{14}$.

Let $BF(S)$ be a Bloom filter storing a set S of elements, and k the number of independent hash functions. The definition of γ-Deniability is the following:

Definition 2 (*taken from [25]* γ**-Deniability).**: An element $x \in S$ inserted in $BF(S)$ is defined *deniable* if $\forall i \in \{1..k\}$ there exists at least one element $v \in V$ such that $\exists j \in \{1..k\}$ such that $H_i(x) = H_j(v)$. A $BF(S)$ is γ*-deniable* whenever a randomly chosen element $x \in S$ is deniable with probability γ.

Additionally, the novel notion "anonymization noise" was introduced, consisting of $n_{\min}$ randomly generated elements. These elements are inserted into the Bloom filter as soon as it is instantiated, thereby ensuring 1-deniability, and preserving the anonymity of data from the very first insertion of a MAC address into the Bloom filter. This innovative approach enables the maintenance of privacy and security of the collected data throughout the analysis process. Thanks to the 1-deniability property, the Bloom filter effectively safeguards the privacy of inserted MAC addresses. Even if an attacker gains access to the Bloom filter, there remains no discernible evidence regarding the number of hash functions employed for data insertion. Furthermore, if the attacker possess knowledge of the hash function count, attempts at a guessing attack are inherently limited. While the attacker may observe the presence or absence of various MAC addresses, the conclusive determination is restricted to the probability of a MAC address's presence, particularly when all corresponding bits are set to 1. This limitation arises from the preservation of 1-deniability, ensuring the existence of at least one uninserted element within the Bloom filter's structure. Thus, any attempt to insert such an element would leave the bitmap unchanged, effectively concealing the actual MAC address from detection.

Given a Bloom filter it is possible to compute γ for the level of γ-Deniability desired according to:

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

where:

- m represents the length of the bits array, which is the number of bits used to construct the Bloom filter.
- k denotes the number of independent hash functions employed in the filter. These functions map an input x to one of the m bits in the bit array.
- n stands for the count of stored elements within the Bloom filter.
- h corresponds to the average cardinality of the hiding set, given by the formula $h = (|U| - n)(1 - e^{-nk/m})^k$.
- $|U|$ represents a large set, approximately equal to $2^{48} \approx 2.8\ 10^{14}$.

Fig. 2 illustrates the evolution of deniability following the insertion of n MAC addresses, as determined by (1), considering a Bloom filter with $m = 10{,}000$ bits and $k = 7$ hash functions. It can be demonstrated that the selection of k represents the optimal choice for minimizing the probability of false positives when storing $n = 1{,}000$ elements. The graph indicates that after approximately 30 insertions into the Bloom filter, the deniability value reaches 1. This signifies the assurance that there will consistently exist at least one element within the hiding set capable of refuting the insertion of any given element into the Bloom filter.

5. Counting framework

This Section, delve into the details of the innovative privacy-preserving people counting framework, named ARGO. This framework has been designed to address the critical challenges associated with preserving user privacy while harnessing the power of IoT and WiFi technologies for crowd monitoring and device characterization. It is important to emphasize that in every stage of development and testing of the framework, the probe request generator played a central role. The datasets it provided, alongside their corresponding ground truths, were instrumental in shaping both the design and implementation of this framework.

5.1. Clustering method

Distinguishing between probe requests based on device features is challenging because header fields often contain information shared among devices of the same model. To maximize the potential benefits, a clustering for model-based differentiation was employed. The analysis conducted in [11] has already underlined a set of fields within each frame that are crucial for discerning the message source. For this reason, this framework leverages the presence of VHT, HT, Extended Capabilities and Vendor Specific Data. In this context, probe requests can be seen as data points in an N-dimensional space, with N representing the number of these extracted features. To facilitate the clustering process, it is essential to generate unique model identifiers for each probe request. These identifiers serve as coordinates within the space where the clustering method will be subsequently applied. For other probe requests originating from devices using randomization techniques, a model identifier is computed as detailed in Algorithm 1.

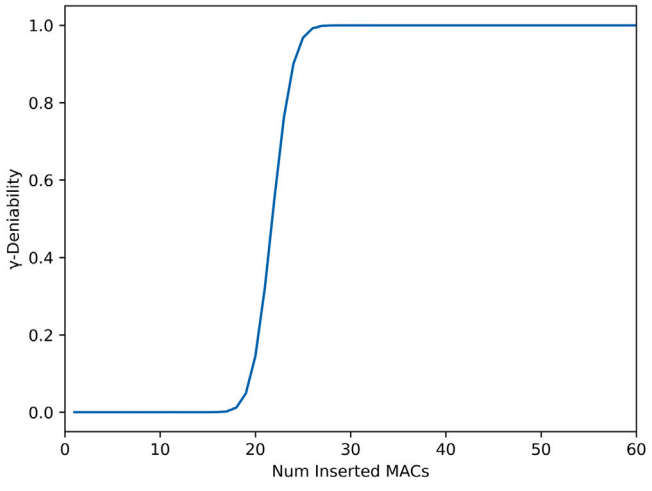

Fig. 2. γ-deniability value in relation to the number of inserted MAC addresses.

Algorithm 1 Algorithm to compute the probe requests model identifiers

Input: *.pcap* file with the capture
Output: $identifiers\_list$ source model identifiers list
Output: $bloom\_filter \leftarrow$ structure to anonymize the MAC addresses

1: $identifiers\_list \leftarrow []$
2: $bloom\_filter \leftarrow initialize\_with\_noise()$ ▷ Noise insertion
3: $bloom\_filter \leftarrow insert(global\_MAC\_addresses)$ ▷ Global MAC addresses insertion inside the Bloom filter
4: **for** $packet$ in $capture$ **do**
5: $rate \leftarrow frame\_rate(packet)$ ▷ Get the frame rate
6: **if** $rate! = 1.0$ **then**
7: $Continue$
8: **end if**
9: $HT \leftarrow 0$ ▷ HT parameter initialization
10: $VHT \leftarrow 0$ ▷ VHT parameter initialization
11: $Extended \leftarrow 0$ ▷ Ext parameter initialization
12: $Vendor \leftarrow 0$ ▷ Vendor data initialization
13: $packet\_fields \leftarrow extract\_all\_fields(packet)$ ▷ Extract all the Information Elements
14: **for** $key, value$ in $packet\_fields$ **do**
15: **if** $key == HT\ info$ **then**
16: $HT \leftarrow process\_value(value)$ ▷ HT value
17: **else if** $key == VHT\ info$ **then**
18: $VHT \leftarrow process\_value(value)$ ▷ VHT value
19: **else if** $key == Extended\ info$ **then**
20: $Ext \leftarrow process\_value(value)$ ▷ Ext value
21: **else if** $key == Vendor\ info$ **then**
22: $Vendor \leftarrow process\_value(value)$ ▷ Vendor value
23: **end if**
24: **end for**
25: $identifier \leftarrow (HT, VHT, Extended, Vendor)$
26: $identifiers\_list \leftarrow insert(identifier)$ ▷ Identifiers list update
27: **end for**
28: **return** $identifiers\_list$

More in detail, the packet processing procedure unfolds as follows: initially, the Bloom filter is initialized with n_{min} random MAC addresses to ensure 1-deniability (line 2), after that, in line 3, all the global MAC addresses are inserted in the Bloom filter. Subsequently, the frame rate is extracted for each packet (line 5), and if the packet is not transmitted at the basic rate, it is promptly discarded (line 7); conversely, if it adheres to the basic rate, the analysis continues. The initialization of the four considered identifiers variables occurs within lines 9 to 12. Leveraging the capabilities of Scapy [23], all packet fields are extracted (line 13). Then, specific packet fields, including HT, VHT, Extended Capabilities, and Vendor data, are extracted and stored in their corresponding variables (lines 15 to 22). Upon the completion of field extraction, a unique device identifier is generated in line 25, as a

tuple, and subsequently appended to the identifier list in line 26. This iterative process continues until all captured packets undergo thorough analysis.

These model identifiers serve as the data points that represent the probe requests, and they are input into a clustering algorithm, specifically DBSCAN [12]. The primary aim is to group probe requests based on the device models from which they originated. Before applying the clustering process, a check is performed to ensure that at least a certain percentage of probe requests (defaulted to 2% of the entire capture) have locally administered MAC addresses. This threshold is crucial because it ensures that DBSCAN is applied to a substantial portion of the received probe requests. If the threshold is not met, only messages with globally unique MAC addresses are considered for counting purposes. This approach helps ensure the robustness and reliability of the clustering result.

To determine the number of devices for each model, two primary approaches can be employed, each contingent on a thorough understanding of the capture area. The first method is applicable when it is assumed that the environment boasts a diverse array of device models, assuming each device corresponds to a unique model. In this scenario, the approach involves simply tallying the number of clusters identified by the algorithm. The second approach involves a more in-depth analysis of the clusters' content. Specifically, probe request rates within the formed groups are scrutinized. Each device model typically emits probe requests at a specific frequency, influenced by the user's interaction patterns. Rather than segregating clusters based on user interaction phases, ARGO focuses on the average sending rate for each model. However, in instances where crowd-monitoring occurs in environments where a particular phase predominates, analyzing message rates for each phase individually may prove beneficial.

Considering just the devices within the same cluster, i.e., of the same model. It holds that $N = K \cdot L \cdot T$, where N is the total number of probe requests sent by devices of a certain model, K is the number of devices belonging to the model, L is the average probe request rate for the model, and T is the capturing time window. Now K can be estimated as:

$$K = \frac{N}{L \cdot T} \tag{2}$$

While parameters N and T are readily available, the rate L must be determined from a source. It is essential to find a specific L value for each formed cluster, as each model exhibits significantly different rates during the experiments. In [2,17] analyses of numerous devices revealed distinct rates, influenced in part by user interactions with the device.

5.2. Counting algorithm

Each probe request corresponds to a model identifier, serving as a data point within the DBSCAN algorithm. These identifiers, based on VHT, HT, Extended, and Vendor data, convey information about system throughput, indicating the data transmission capacity of a network within a specific timeframe. These model identifiers serve a dual purpose: they aid in clustering probe requests by device models and gauge the similarity of these clusters to models in the generator database. The underlying idea is that when two models possess similar identifiers, there is a higher likelihood that their probe request rates are also similar. The generator database contains multiple models, each with its identifiers and rates. This approach enables the comparison of each generated cluster with all models in the database using their identifiers. This facilitates the identification of the most similar model, from which the rate can be extracted as the L parameter.

To gather these values, the probe request generator allows simulations with individual devices from the database, making it possible to calculate the message rate associated with a specific model.

For each model in the generator database, two types of information are computed: the quadruplets used for model identification and its probe request rates. With this updated database of models, it becomes possible to find and utilize probe request rates to assign a parameter L to each formed cluster. However, before searching for a model in the database, each cluster undergoes a normalization process. Not all probe requests collected within a group may have identical identifiers; they are expected to be similar due to the use of a density-based clustering algorithm, but perfect identity is not guaranteed. Therefore, within each group, all identifiers are averaged to determine a unique cluster identifier for pairing with the models in the database.

The pseudo-code for the framework counting procedure using model rates is reported in Algorithm 2.

Algorithm 2 Algorithm to count the number of devices inside a .pcap trace

Input: $identifiers\_list$ identifiers from Algorithm 1
Input: $models\_rates$ probe request rates of device models
Input: $local\_count$ locally administrated unique MAC addresses counter
Input: $global\_count$ globally unique MAC addresses counter
Input: $packets\_count$ number of packets inside the capture
Output: $device\_count$ number of detected devices

1: **if** $\frac{local\_count}{packets\_count} < 2\%$ **then**
2: $total\_devices \leftarrow global\_count$
3: **return** $total\_devices$ ▷ If clustering can not be done
4: **end if**
5: $labels \leftarrow DBSCAN(identifiers\_list)$ ▷ DBSCAN clustering
6: $labels \leftarrow remove\_noise(labels)$ ▷ Remove -1 labels
7: $labels\_unique \leftarrow set(labels)$ ▷ Unique labels
8: $T \leftarrow get\_time\_window()$ ▷ Capture time window
9: $count\_devices \leftarrow 0$
10: **for** l in $labels\_unique$ **do**
11: $identifier \leftarrow get\_identifier(identifiers\_list, l)$
12: $L \leftarrow closest\_rate(models\_rates, identifier)$ ▷ Rates match
13: $N \leftarrow count\_PRs(labels, l)$
14: $K \leftarrow \frac{N}{L \cdot T}$ ▷ Apply Eq. (2)
15: $count\_devices \leftarrow count\_devices + K$ ▷ Count update
16: **end for**
17: $total\_devices \leftarrow global\_count + count\_devices$
18: **return** $total\_devices$ ▷ Return the number of devices

Analyzing Algorithm 2 in detail, the process begins in line 1 by evaluating the percentage of packets with locally administrated MAC addresses among the total number of packets. If this percentage falls below 2%, indicating insufficient data for clustering, the total number of devices is set equal to the number of distinct global MAC addresses (line 2). Conversely, if the percentage exceeds 2%, clustering is performed in line 5, followed by noise removal, eliminating all labels with "-1". Subsequently, in line 7, the remaining labels are inserted into a set to obtain a list of distinct labels. Then, in lines 11 and 12, for each label, its identifier is extracted from the identifiers_list, generated by Algorithm 1, and the closest rate is retrieved from the model rates database created by the probe request generator. Finally, in line 14, Eq. (2) is applied to calculate the value of K, representing the number of devices within the considered cluster. Ultimately, the total number of devices is determined by the sum of the global count and the count derived from the clustering algorithm.

An open-source repository for this framework is available to the research community[1], fostering collaboration and advancing the development of this innovative crowd monitoring strategy. Beyond the comprehensive codebase, the repository includes detailed documentation, facilitating a deeper understanding and seamless utilization of the framework for researchers and developers alike.

[1] https://github.com/Diegomangasco/ARGO

6. Architecture

In this Section, the main attention is directed towards showing the architecture developed for the purpose of people counting. Insights will be provided into the hardware components utilized, as well as the counting pipeline developed for this study, including details on the total cost of the hardware.

6.1. Hardware description

The primary objective is to develop a cost-effective, sustainable solution for tracking people's movements across various environments and conditions. To achieve this, a strong emphasis is placed on minimizing the cost of hardware components while ensuring sufficient processing power for real-time analysis. After careful consideration, the ideal hardware solution was found in the form of a Raspberry Pi (RP). In contrast to alternative solutions like Meshlium by Libellium [26], the proposed solution boasts a hardware cost that is an order of magnitude lower. Despite this cost reduction, the solution maintains comparable performance and detection range, making the Raspberry Pi the perfect solution for this use case. Specifically, the proposed architecture relies on a RP 4 Model B, which boasts a 1.8 GHz 64-bit quad-core ARM Cortex-A72 CPU, onboard 802.11ac WiFi interface, Bluetooth capabilities, up to 8 GB of RAM, an Ethernet connection, and plenty of USB ports. It runs the Linux OS, providing a versatile platform for the use case considered. To enable monitor mode for WiFi frame capture, the RP was supplemented with a basic USB WiFi dongle equipped with an external antenna. Additionally, the RP offers flexibility by supporting both traditional power source and Power over Ethernet (PoE), simplifying deployment considerations. Furthermore, to facilitate interaction with the Raspberry Pi once it is installed, a USB LTE antenna with a SIM card was included. This configuration not only enables the Raspberry Pi to access the network and send data to the server but also grants the capability to SSH into the Raspberry Pi. This SSH access allows for data collection, modification of running scripts, and perform other necessary tasks when needed.

The hardware solution setup is illustrated in Fig. 3, showcasing the components essential for the crowd-monitoring system.

6.2. Capturing procedure and probe request analysis

Fig. 4 presents the architecture deployed as part of the participation in the TrialsNet EU project [27]. Specifically, the setup consists in two

Fig. 3. Hardware solution composed of a Raspberry Pi 4 Model B, a USB WiFi dongle with an external antenna, a USB LTE antenna, all inside an IP67 waterproof box.

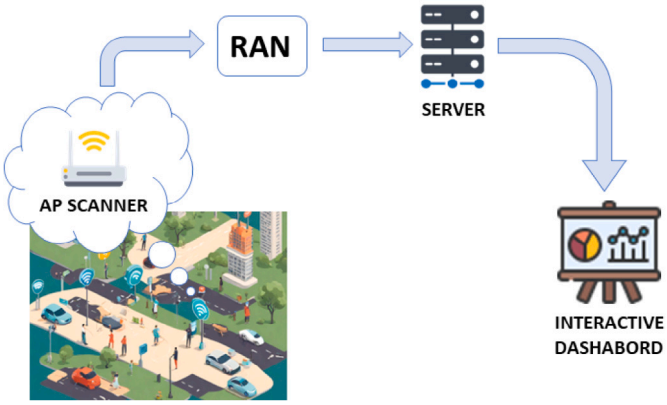

Fig. 4. Architecture for crowd-monitoring analysis. Each AP scanner sends periodically its counting and the Bloom filter to the server, where all the data are stored and visualized in an interactive dashboard.

Raspberry Pi 4B (RPs) devices, as the one illustrated in Fig. 3, deployed in a large park in Turin, Italy. These RPs are equipped with USB WiFi dongles configured in monitor mode, allowing them to capture all data transmitted by smart devices in the covered area. Once the captured data is processed through the onboard sniffer pipeline, it is transmitted over the network using the UDP protocol and received by a central server situated at the network's edge. The primary role of the server is twofold: first, if the incoming data originate from a single scanner, it stores the data in a database for future visualization. Secondly, if the data includes Bloom filters, it stores and processes them as needed. Furthermore, the server hosts an interactive web visualization platform that facilitates the presentation of time-series data in an engaging, interactive fashion. To achieve this, the Grafana visualization tool [28] is employed.

The counting pipeline is depicted in Fig. 5. To delve into the details, the system initialization process begins by configuring all the necessary environment variables and executing a wireless interface configuration script via the *rc.local* file. This script serves the purpose of deactivating the onboard WiFi interface of the Raspberry Pi and activating the one associated with the USB WiFi dongle. This is essential because the embedded WiFi interface cannot be placed in monitor mode. Once all configurations are in place, the main sniffing script is initiated. Utilizing tshark [29], probe request messages are captured for a two-minute time window. Subsequently, the resulting *.pcap* file is passed to the next stage of the processing pipeline, while the tshark script restarts to initiate another capturing cycle.

Every time a new *.pcap* file is generated a script begins analyzing all the probe requests detected. All messages are divided into two groups based on the value of the second-least-significant bit of the first octet of their MAC addresses. Subsequently, for each probe request with a locally administered MAC address, the creation of model identifiers takes place. In particular, the throughput capabilities are leveraged to generate four-dimensional data points that are input into the DBSCAN clustering algorithm, as explained in Section 5.1. Following this clustering process, two operations are performed.

In the first operation, the counting algorithm, outlined in Section 5.2, is applied. In the second operation, the resulting MAC addresses are stored within a Bloom filter. This filter is specifically initialized with n_{min} random MAC addresses to satisfy the 1-deniability property, a requirement for ensuring GDPR compliance. As a result, the first timestamp of the time window, a numerical value representing the counting within that window, and the populated Bloom filter are obtained.

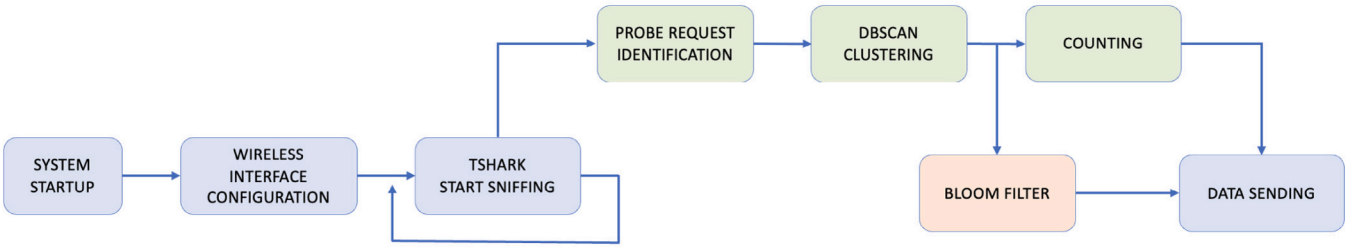

Fig. 5. Counting pipeline onboard the scanner.

7. Counting results

This Section provides an in-depth account of the tests carried out to validate the developed crowd-monitoring framework. To commence, the experimental setup is described, followed by the datasets utilized in the experiments, with descriptions of the specific environments under consideration. Following this, the effectiveness of the entire system pipeline in accurately counting the number of devices within a designated area is demonstrated. Finally, insights into the data collected in the newer real-world deployment are provided.

7.1. Experimental setup

Every test executed using ARGO meticulously followed a specific set of parameters concerning both the capturing area, the packet processing, and the DBSCAN clustering algorithm.

Table 1 reports a comprehensive overview of the primary parameters along with their respective values.

Table 1
Experimental setup.

Parameter	Value
Frequency	2.4 GHz
Channel	1
Time window	2 min
Minimum samples (DBSCAN)	3 samples
Epsilon (DBSCAN)	0.001
Distance metric (DBSCAN)	Euclidean
Sniffing Software	TShark
Packet Processing Library	Scapy [23]

7.2. Datasets used

The primary objective of these studies is to conduct a comprehensive analysis of the performance exhibited by crowd-monitoring systems within different environmental situations. Through the utilization of the probe request generator, a list of scenarios can be generated, encompassing situations that emphasize the presence of a limited number of devices but with a wide variety of distinct models, as well as scenarios that exhibit the inverse condition, and additionally, scenarios characterized by mixed situations.

Furthermore, the algorithms have undergone rigorous testing utilizing authentic *.pcap* traces. However, in these instances, a distinct challenge emerges: the inherent difficulty in establishing a precise ground truth. This particular challenge was addressed by deriving an estimate, formulated by the person responsible for data collection, of the rough number of individuals or devices believed to be present. Below is an enumeration of the datasets adopted during the experimentation:

- Simulated traces:
  - *Dataset A*: 60 devices, all of which exclusively pertain to a single vendor-model.
  - *Dataset B*: 6 devices, each representing a distinct vendor-model.
  - *Dataset C*: contains only one device.
  - *Dataset D*: medium-crowded situation, with 70 devices of various vendor-models.
  - *Dataset E*: large-crowded situation, with 120 devices of various vendor-models.
- Real Traces:
  - *Dataset F*: contains the captures conducted within an anechoic chamber as part of the research detailed in [15].
  - *Dataset G*: comprises a collection of 10 two-minute captures collected in a lecture hall at Politecnico di Torino, during a class of the 2023/2024 academic year.

As previously hinted at, each dataset is the result of a unique environment, characterized by its own distinctive attributes.

7.3. Crowd-monitoring results

In the clustering part of the framework, the DBSCAN [12] algorithm is deployed to process probe requests with locally administered MAC addresses. The algorithm is configured with the following parameters:

- *Metric*: the metric to use when calculating the distance between instances in a feature array.
- *Epsilon*: the maximum distance at which two samples can be considered each other's neighbors.
- *Min samples*: the number of samples in a neighborhood such that a point can be considered as a cluster centroid.

After a thorough analysis of probe requests and their behaviors, the parameter settings listed above were fine-tuned. Specifically, the Euclidean distance is employed as the *metric* because the identifiers computed for each message are treated as data points within a 4-dimensional space. The *epsilon* parameter was consistently set to a constant value of 0.001, a choice made after observing the relationship between probe requests originating from the same device and those from different devices. Lastly, it was determined that the *min sample* parameter is influenced by environmental conditions and the duration of data capture. In scenarios with short time windows and devices generating a limited number of messages, clusters may contain fewer data points. Therefore, it is important to customize this parameter to align with the specific requirements of the use case, ensuring optimal cluster identification.

The results of the experiments have been tabulated in Table 2, where each row represents a specific set of tests conducted using a *.pcap* file that simulates an environment. The table columns provide various details, including the dataset identifier, ground truth, the count of detected devices, and the accuracy. To calculate the accuracy, a method that considers the difference between the result and the ground truth is adopted. Specifically, the counting error is determined by taking the absolute difference between the ground truth and the obtained result,

Table 2
Crowd-monitoring results.

Dataset	Ground truth	ARGO	iABACUS [30]
A	60	52 (86.66%)	36 (60%)
B	6	7 (83.33%)	88 (−1266.70%)
C	1	1 (100%)	45 (-4400%)
D	70	58 (82.85%)	NA
E	120	110 (91.66%)	NA
F	22	23 (95.45%)	NA
G	up to 121	overall 91.48%	overall 27.63%

and then normalizing it relative to the ground truth. Accuracy can be easily obtained by subtracting this error from 1. The mathematical expressions for these calculations are the following:

$$Err = \frac{|GT - R|}{GT} \tag{3}$$

$$Acc = 1 - Err \tag{4}$$

In these equations, *Err* represents the error, *GT* stands for the ground truth, *R* is the obtained result, and *Acc* denotes the accuracy. The accuracy value is indicated alongside the number of detected devices in parentheses.

Table 2 presents the results obtained from ARGO and compare them to those obtained using an implementation of the framework proposed in [30]. It is worth noting that the authors of [30] introduced a de-randomization framework known for its high accuracy in their experiments. This framework uses multiple if-then-else constructs and several levels of recursion to assess the likelihood that multiple MAC addresses belong to the same device.

For the experiments, an implementation of the iABACUS framework was executed, starting from the flow chart available in [30], on an Apple MacBook Pro laptop equipped with the M1 Pro CPU. However, despite the considerable computational power available, difficulties were encountered when processing datasets D, E, and F. The script failed to complete its execution due to recursion errors, exceeding the maximum recursion depth.

While Eq. (4) provides a viable method for assessing the framework performance, it fails to consider a critical factor: the variability in the number of devices present in different environments. Some datasets contain numerous devices, while others have very few. For instance, *Dataset E* comprises over 100 simulated elements, whereas *Dataset C* includes only a single device. It becomes evident that an error of 1 device has a more significant impact in sparsely populated environments but may be deemed insignificant in densely populated ones. This underscores the importance of not solely relying on accuracy as an isolated metric but rather evaluating it within the context of the simulation.

Thanks to the combination of clustering and time-related features, ARGO has exhibited remarkable effectiveness in crowd monitoring. The number of devices estimation reached a high level of precision with respect to the ground truth, as demonstrated by the high accuracy obtained.

7.3.1. Real-life indoor scenario

To evaluate ARGO within a real-life indoor scenario, *Dataset G* has been considered. This dataset is composed by a collection of traces comprises a twenty-minute recording, with capturing windows spanning two minutes each. The recording took place in Room 14 at the University of Politecnico di Torino, Italy, during the first day of the academic year 2023/2024.

What distinguishes this dataset is its ever-changing character. Instead of a "static" use case, the traces have been collected to capture the scene a few minutes before the start of a lesson and continued recording for a few minutes afterward. Consequently, at the beginning of the scenario, a few individuals were already present in the room.

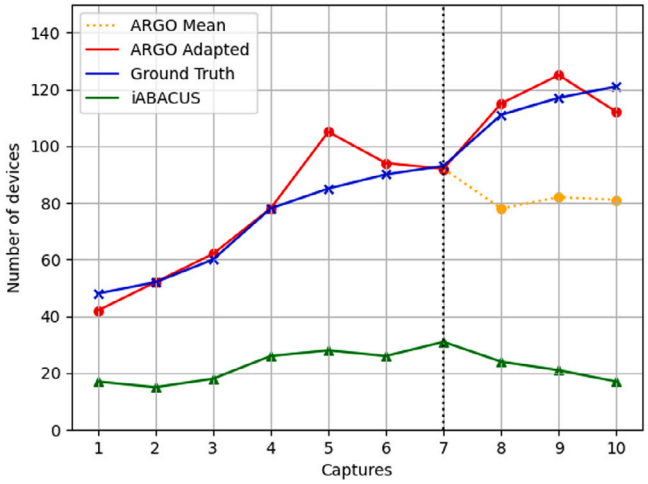

Fig. 6. Dataset G counting results between the ground truth and the results obtained from ARGO and an implementation of iABACUS.

Over the next twelve minutes, more people entered the room, seeking seats. In the final three capturing windows, the lesson began, with only a handful of additional individuals arriving. This dynamic recording scenario allows for the simulation of the gradual entry of people into the room, reflecting a real-world situation.

An overall accuracy is provided in Table 2 for both the frameworks considered. While Fig. 6 shows a detailed version of the results obtained.

The incremental presence of people in the room is readily apparent when examining the ground truth line in blue in Fig. 6. The results obtained with ARGO are represented in red, while those obtained with the implemented version of the iABACUS framework are shown in green. On the x-axis, various captured windows are displayed, while the y-axis indicates the number of detected people/devices. In particular, ARGO demonstrates superior performance when compared to the iABACUS framework. The latter tends to significantly underestimate the number of detected devices. ARGO demonstrates great capacity in following the growing flow of people who gradually entered the room. On the other hand, iABACUS tends to underestimate the presence of people and it does not even recognize the increasing trend of the ground truth, resulting in a substantially flat line.

It is important to highlight that during the initial seven capture windows, people were often in motion, using their phones, or engaged in conversations while their smartphones were in a locked state. In these cases, the average rate is considered the most suitable to factor both the locked, awake, and active phases. However, starting from the eighth capture, which corresponds to the beginning of a lesson, using the average rate is no longer an accurate choice, as it tends to underestimate the counting. This trend is underlined by the dotted orange line that, starting from the eight capture, leads Argo's predictions to deviate significantly from the ground truth. Instead, if the framework is set to focus solely on the locked rate, a greater accuracy is achieved, and the trend of the framework continues to strictly follow the ground truth line. This emphasizes another important point emerged from this analysis: the significance of considering the context in which captures are conducted, as it is of paramount importance to get even better results. Knowing in advance the characteristics of the environment in which ARGO is working is crucial to set correctly the rate type that could lead to a better accuracy in crowd estimation.

ARGO consistently achieves over 90% accuracy for nearly every time window, resulting in an impressive overall accuracy of 91.48%. This demonstrates the effectiveness of this innovative approach in accurately detecting and tracking the presence of people/devices in the room, especially during dynamic scenarios like the one presented in *Dataset G*.

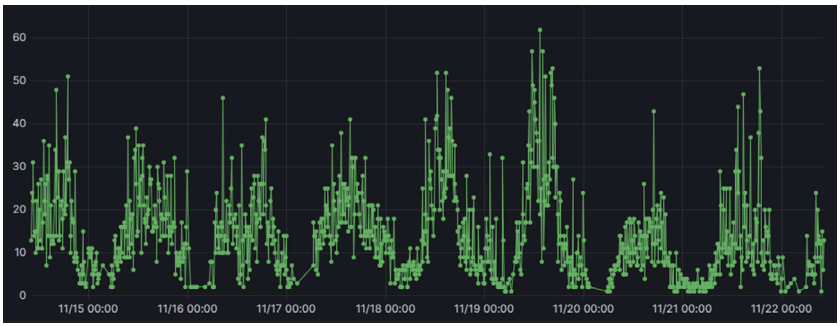

Fig. 7. Counting values with an aggregation window of 10 min, data spanning the week from 14/11/23 to 22/11/23.

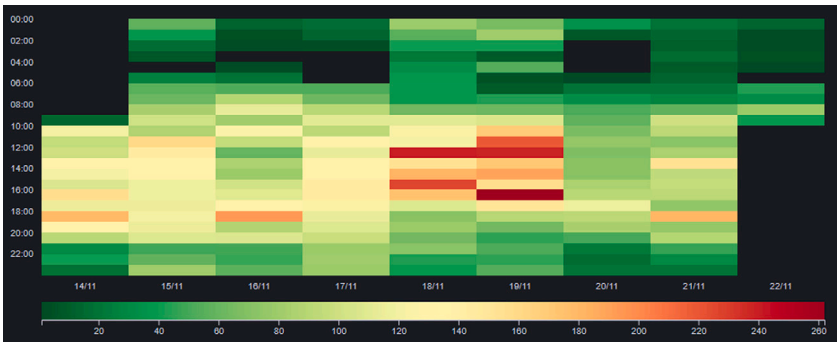

Fig. 8. Hourly heatmap of counted devices, data spanning the week from 14/11/23 to 22/11/23.

7.3.2. Real-life outdoor scenario

To assess the efficacy of ARGO in a real-world outdoor setting, the entire pipeline detailed in Section 6 has been implemented, at a public park located in the heart of Turin. Two sensors were strategically placed near one of the park entrances, known for hosting concerts and events. Captures were scheduled every two minutes, providing valuable data for analysis. The framework analysis yielded essential information, including the capture timestamp, the count of detected devices, and the MAC addresses stored in the bloom filter. These outputs were stored in a database on a server, facilitating easy access and monitoring.

To visualize and comprehend the trends, an interactive dashboard was created using Grafana [28], a tool enabling the extraction of raw data from a database and the creation of insightful graphs and views. Figs. 7 and 8 showcase the results of the analysis spanning a sample week, i.e., from 14/11/2023 to 22/11/2023. In Fig. 7, a graph illustrates the temporal progression of device counts, with timestamps plotted on the x-axis. The results reveal distinct patterns at various times of the day and week. Notably, park visits steadily rise from early morning to an afternoon peak, followed by a gradual decline towards the evening. Weekends exhibit increased counts, as one would expect.

In addition to the temporal trends, a heatmap is provided in Fig. 8, representing the distribution of data points across specified time intervals. Each cell in the grid corresponds to a specific combination of day and time intervals, with color gradients indicating the magnitude of the counts. The color scale ranges from deep green for lower numbers to a red scale for higher counts. The hours from mid-morning to mid-afternoon consistently show increased counts compared to other periods. Further analysis of cells highlighted in dark red on November 18, 2023, and November 19, 2023 (Saturday and Sunday) corresponds to peaks in park visits during weekends.

8. Conclusions

This study introduced an innovative framework designed for precise people/device detection and tracking across various scenarios. The developed framework consistently delivered an outstanding performance, achieving accuracy rates exceeding 90% for the real traces analyzed and consistently surpassing 80% for all other cases. In direct comparison with the iABACUS framework [30], ARGO demonstrated superior performance by effectively avoiding significant underestimations. This highlights the challenge that older frameworks face in adapting to the dynamic nature of probe request behavior and the continuous updates of operating systems.

One notable advantage of ARGO lies in the flexibility of the probe request generator, which enables easy adaptation to future changes in probe request behavior. This adaptability is achieved because the generator is easily expandable, accommodating the addition of new devices and updates to existing ones. This facilitates the development of newer iterations of the machine learning model to maintain accuracy over time. It is worth to point out that ARGO places a strong emphasis on privacy compliance. No data is stored on the sniffer device, and after the processing pipeline is completed, all data is promptly discarded. Only the Bloom filters and counting values are transmitted over the network, ensuring data security and privacy.

In future endeavors, there exists considerable potential to enhance the performance of the counting algorithm by integrating temporal analysis and considering network congestion dynamics. By integrating temporal analysis, the system could account for variations in probe request patterns over time. While, conducting an analysis of network congestion could serve to further enhance the algorithm's accuracy. By dynamically adjusting counting parameters based on prevailing network traffic conditions, the algorithm can effectively adapt to varying levels of crowd congestion, thereby refining its accuracy.

CRediT authorship contribution statement

Riccardo Rusca: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Diego Gasco:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project

administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Claudio Casetti:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Paolo Giaccone:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments and disclaimer

This work is funded by the European Union's Horizon-JU-SNS-2022 Research and Innovation Programme through the TrialsNet project (Grant Agreement No. 101095871).

This manuscript reflects only the authors' views and opinions and does not necessarily reflect the view of the European Union neither the European Commission can be considered responsible for them.

This publication is part of the project PNRR-NGEU which has received funding from the MUR - DM 117/2023.

References

[1] European Parliament, Council of the European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016-05-04, URL https://data.europa.eu/eli/reg/2016/679/oj.

[2] 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, 2023, pp. 1–6, http://dx.doi.org/10.1109/ICT-DM58371.2023.10286944.

[3] M. Hasan, J. Hanawa, R. Goto, R. Suzuki, H. Fukuda, Y. Kuno, Y. Kobayashi, LiDAR-based detection, tracking, and property estimation: A contemporary review, Neurocomputing 506 (2022) 393–405, http://dx.doi.org/10.1016/j.neucom.2022.07.087, URL https://www.sciencedirect.com/science/article/pii/S0925231222009365.

[4] M. Al-Sa'd, S. Kiranyaz, I. Ahmad, C. Sundell, M. Vakkuri, M. Gabbouj, A social distance estimation and crowd monitoring system for surveillance cameras, Sensors 22 (2) (2022) http://dx.doi.org/10.3390/s22020418, URL https://www.mdpi.com/1424-8220/22/2/418.

[5] M.H.K. Khel, K.A. Kadir, S. Khan, M. Noor, H. Nasir, N. Waqas, A. Khan, Realtime crowd monitoring—Estimating count, speed and direction of people using hybridized YOLOv4, IEEE Access 11 (2023) 56368–56379, http://dx.doi.org/10.1109/ACCESS.2023.3272481.

[6] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, P. Rai, ultralytics/yolov5: v3.1 - Bug fixes and performance improvements, 2020, http://dx.doi.org/10.5281/zenodo.4154370.

[7] A.E. Redondi, M. Cesana, Building up knowledge through passive WiFi probes, Comput. Commun. 117 (2018) 1–12, http://dx.doi.org/10.1016/j.comcom.2017.12.012.

[8] L. Oliveira, D. Schneider, J. De Souza, W. Shen, Mobile device detection through WiFi probe request analysis, IEEE Access 7 (2019) 98579–98588, http://dx.doi.org/10.1109/ACCESS.2019.2925406.

[9] A. Simončič, M. Mohorčič, M. Mohorčič, A. Hrovat, Non-intrusive privacy-preserving approach for presence monitoring based on WiFi probe requests, Sensors 23 (5) (2023) http://dx.doi.org/10.3390/s23052588.

[10] M. Uras, E. Ferrara, R. Cossu, A. Liotta, L. Atzori, MAC address de-randomization for WiFi device counting: Combining temporal- and content-based fingerprints, Comput. Netw. 218 (2022) 109393, http://dx.doi.org/10.1016/j.comnet.2022.109393.

[11] L. Pintor, L. Atzori, Analysis of Wi-Fi probe requests towards information element fingerprinting, in: GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 3857–3862, http://dx.doi.org/10.1109/GLOBECOM48099.2022.10001618.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, KDD '96, AAAI Press, 1996, pp. 226–231.

[13] M. Ankerst, M. Breunig, P. Kröger, J. Sander, OPTICS: Ordering points to identify the clustering structure, 28, 1999, pp. 49–60, http://dx.doi.org/10.1145/304182.304187,

[14] T. He, J. Tan, S.-H.G. Chan, Self-supervised association of Wi-Fi probe requests under MAC address randomization, IEEE Trans. Mob. Comput. 22 (12) (2023) 7044–7056, http://dx.doi.org/10.1109/TMC.2022.3205924.

[15] L. Pintor, L. Atzori, A dataset of labelled device Wi-Fi probe requests for MAC address de-randomization, Comput. Netw. 205 (2022) 108783, http://dx.doi.org/10.1016/j.comnet.2022.108783, URL https://www.sciencedirect.com/science/article/pii/S1389128622000196.

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

[17] R. Rusca, F. Sansoldo, C. Casetti, P. Giaccone, What WiFi probe requests can tell you, in: IEEE Consumer Communications & Networking Conference, CCNC, 2023, pp. 1086–1091, http://dx.doi.org/10.1109/CCNC51644.2023.10060447.

[18] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM 13 (1970) 422–426, URL https://api.semanticscholar.org/CorpusID:7931252.

[19] V.-D. Stanciu, M.v. Steen, C. Dobre, A. Peter, Privacy-preserving crowd-monitoring using Bloom filters and homomorphic encryption, in: International Workshop on Edge Systems, Analytics and Networking, EdgeSys, ACM, 2021, pp. 37–42.

[20] V.-D. Stanciu, M. van Steen, C. Dobre, A. Peter, Anonymized counting of non-stationary Wi-Fi devices when monitoring crowds, in: International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM, ACM, 2022, pp. 213–222.

[21] R. Rusca, A. Carluccio, C. Casetti, P. Giaccone, Privacy-preserving WiFi-based crowd monitoring, Trans. Emerg. Telecommun. Technol. 35 (3) (2024) e4956, http://dx.doi.org/10.1002/ett.4956, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.4956, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4956.

[22] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. Rye, D. Brown, A study of MAC address randomization in mobile devices and when it fails, in: Proceedings on Privacy Enhancing Technologies, 2017, http://dx.doi.org/10.1515/popets-2017-0054.

[23] Scapy library, [Online]. URL https://scapy.net/.

[24] T. Bravenec, J. Torres-Sospedra, M. Gould, T. Fryza, Exploration of user privacy in 802.11 probe requests with MAC address randomization using temporal pattern analysis, 2022, arXiv:2206.10927.

[25] G. Bianchi, L. Bracciale, P. Loreti, "Better than nothing" privacy with bloom filters: To what extent? in: Privacy in Statistical Databases, Springer Berlin Heidelberg, 2012.

[26] Libelium Meshlium, URL http://www.libelium.com/products/meshlium/.

[27] TrialsNet EU project, URL https://trialsnet.eu/.

[28] Grafana Labs, Grafana documentation, 2018, URL https://grafana.com/docs/.

[29] tshark, URL https://www.wireshark.org/docs/man-pages/tshark.html.

[30] 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.