Edge-based Passive Crowd Monitoring Through WiFi Beacons

Edge-based Passive Crowd Monitoring Through WiFi Beacons

Kalkidan Gebru[a], Marco Rapelli[a], Riccardo Rusca[a], Claudio Casetti[a], Carla Fabiana Chiasserini[a], Paolo Giaccone[a]

[a]*Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy*

Abstract

Tracking people's flows has become crucial, not only for safety and security, but also for numerous practical business applications and better management of urban spaces, facilities and services. In this paper, we proposed methodologies that, exploiting IoT technology deployed at the edge of the network, allow for the analysis of people's movement in urban environments, both outdoors and indoors. In particular, leveraging the use of WiFi probe packets sent by smart devices carried by people on the move, we first describe an implementation of our methodology using off-the-shelf hardware to count people boarding public transportation vehicles. We then present an alternate implementation using commercial WiFi scanners connected to the edge and leveraging suitably deployed virtual network functions to process the data collected by a OneM2M IoT platform, proposing also a mobility tracking procedure that can be applied to anonymized data provided by commercial WiFi scanners. Our experimental results show that the proposed approaches to people counting and mobility detection can achieve a good level of accuracy, while overall carrying a low price tag.

Keywords: Mobility, People's flow tracking, Edge computing, Detection, WiFi

1. Introduction

In November 2018, the Italian National Institute of Statistics (ISTAT) published a research on everyday commuters and their means of transportation [1]. Data from the prior year's national census was used in the analysis. According to the findings of this investigation, more than 40% of the Italian population prefers not to use private transportation for daily travels on a regular basis, 19% of the Italian population prefers to go by foot or bicycle, while 23% prefers to use public transportation such as buses, trams, trains, school buses, or subway. Overall, this represents a considerable proportion of national commuters who rely on non-private form of transportation. To go to and from work or school on a regular basis, this section of the urban population relies on adequate infrastructure and services provided by municipal transportation agencies. It is clear that basic data acquired from a census will not be sufficient to achieve this aim. In order to maintain a high level of service, precise monitoring of the number of users of public transportation is required on a regular basis. If the local transportation authority had real-time data on bus passengers collected at each stop, they would be able to expand the service appropriately. In addition, following the global lockdown imposed by governments to stop the spread of the COVID-19 pandemic, capacity restrictions in public places were established to limit the number of individuals who gathered. As an example, in Italy a capacity limit for the maximum number of people that can be transported on public transit vehicles was established. It is therefore clear that designing an efficient system for people's flow monitoring becomes critical. However, fulfilling this need in a trustworthy manner is highly challenging, as the privacy of the data of every individual involved should be preserved in every part of the designed monitoring system.

Importantly, for many safety applications and convenience services designed for mobile users it is essential to detect the pattern taken by people's flows at different times of the day/week. One of the key technologies to achieve this goal is the Internet-of-Things (IoT) [2][3], as IoT devices are becoming pervasive and most of them are equipped with a radio interface, e.g., WiFi, LoRaWan or 5G, that can conveniently connect them with other devices as well as with the communication network infrastructure. Furthermore, IoT devices typically consume little energy, hence they contribute to creating sustainable communication systems, have low cost, and pose fewer privacy issues than other devices like smart city cameras.

In this work, we leverage the IoT technology and tackle the problem of characterizing both people's trajectories and the number of people a flow includes, while *preserving users' privacy*. In particular, we focus on an urban environment and exploit both commercial sensors and simple devices like Raspberry PIs, equipped with a WiFi interface. Such devices can scan the WiFi spectrum for probe requests, i.e., packets transmitted by user hand-handled devices towards nearby access points. Using the logs provided by these spectrum scanners, we develop techniques to increase the privacy level in data collection and processing. Importantly, we aim at developing a solution that can cope with the serious limitations of commercial or Raspberry PI-based scanners, which demands for a new approach with respect to those proposed in prior art.

Unlike existing work, we develop mechanisms that can effectively (i) cope with commercial sensors as well as simple, low-cost ad-hoc designed devices that scan the spectrum for WiFi probes, and (ii) increase the level of users' privacy protection. Further, applying an ML-based scheme, we show how the

data collected through our privacy-preserving technique can be used to characterize people's flows. Our approach is then validated through a proof-of-concept testbed that we developed and a measurement campaign that we performed on public buses run by GTT [4] in the city of Turin, Italy.

The rest of the paper is organized as follows. After discussing some relevant related work in Section 2, in Section 3 we describe some features of the WiFi technology that are relevant for the design of our privacy-preserving solution for people's flows detection. Then Section 4 proposes an approach for people's counting indoors, taking a public bus as an example of indoor public places where a monitoring system could be deployed. The methodology we use for outdoor spaces is instead presented in Section 5. An approach for people's mobility tracking is introduced in Section 6. Finally, Section 7 draws some conclusions.

2. Related Work

In recent years, several different approaches have been used to face the problem of detecting and counting people in an urban area, both in indoor and outdoor scenario. Infrared sensors, cameras, pressure sensors, visible light sensors, RFID, UWB, and audio-processing are some of them, however, the techniques mentioned above do not provide satisfactory results at times in relation to the cost of implementation while at other times they obtain insufficient performance [5].

Some approaches are based on electromagnetic methods that analyze in space and time the received signal, which is affected by the presence of the people in the area. All these methods are suitable for small indoor environments and require specific radio frequency emitters. As an example, the work in [6] proposed to count people exploiting the WiFi signal, assuming that the movements of the human body affect the wireless signal reflections, which results in variations in the CSI (Channel State Information). The method works well in a quasi-static indoor scenario, within the same room (e.g., people in a meeting room or staff in an office), but it is hard to implement in a very dynamic environment, with people moving by car, by bicycle, or on foot. Another work [7] focused on counting people crossing a doorway using off-the-shelf WiFi devices, by exploiting the reflections of the wireless signal on the human body and by installing special receivers that process the reflected signals. This approach would require more than one device to cover all possible accesses, e.g., on a bus, significantly increasing the cost of deployment. Indeed, it cannot be applied in an outdoor urban scenario with the sensors installed (as in our case) on top of traffic lights.

An alternative approach, which however quite expensive, is to count people using cameras and advanced algorithms for video image processing. For example, the state of the art YOLO_V3 library [8] can identify and count people's heads in a 30-FPS video in real-time, but it requires a high-level GPU and a highly equipped server, resulting in a video-based detection platform that is too expensive for large-scale use, especially at the network edge.

Finally, it is worth mentioning that a preliminary version of some parts of this work have appeared in our conference papers [9, 10, 11]. Specifically, [9, 10] sketched our solution for tracking people's flows while accounting for users' privacy. [11], instead, introduced the two Virtual Network Functions (NFV) that we implemented, along with the related scripts, providing some initial results on people's flows detection based on the data collected up to then. In this work, we combine the two approaches, thus yielding an efficient and effective solution and implementation, and we provide extensive, more insightful results, using more than two and a half years of data collected in the area covered by the scanners we deployed.

3. Preliminaries and Main Observations

As mentioned, we use WiFi signal reception and processing to estimate the number of smart devices (e.g., smartphone, tablet, laptop, smartwatch, etc.). Our approach thus consists in scanning the WiFi spectrum, attempting to capture WiFi packets sent by smartphones. When a smartphone's WiFi interface is turned on, it transmits a burst of broadcast messages to discover Access Points (APs) and smartphones nearby. To link a recorded WiFi signal with a smartphone, we examine the MAC (Media Access Control) address information on collected packets.

In the following, we summarize some important features of WiFi that we later leverage to extract useful information from the collection of the transmitted messages, in a privacy-preserving manner.

3.1. WiFi probe requests

Depending on whether the scanning mode is active or passive, smartphones can send Probe Requests on the channel on their own initiative or be triggered by the reception of a Beacon frame transmitted by an AP. Smartphones are usually capable of supporting both scanning modes and of switching back and forth between them. In our experiments, sniffers mainly capture Probe Requests generated by smartphones on their own initiative during an active scan.

3.2. MAC Randomization

Probe Requests are frames of management type, as all other frames transmitted to a WiFi AP during the association operation. They are required in order to create an encrypted channel, which the user will leverage to send data in the future. As a result, any fields included in any of those management frames are broadcast and in clear text. All of the information therein, as well as the sender's MAC address, may be retrieved in plain text by collecting a Probe Request from the WiFi channel. Knowing a device's MAC address, according to [12], is a very sensitive piece of information that might be exploited for a number of cyber attacks as well as for monitoring people's movements.

As a result, practically all smartphone manufacturers randomize the MAC addresses in Probe Request frames. Indeed, software providers have discovered that broadcasting the initial MAC address is unnecessary at this point of the process because

the great majority of Probe Request packets do not result in an actual connection to an AP but are instead used only for scanning reasons. In this case, the universal/local bit is set to one, resulting in a random MAC address.

The MAC randomization methods are proprietary and manufacturer-dependent. Apple was the first to adopt MAC randomization in 2014 for smartphones running iOS 8 [13], followed by Linux with Linux Kernel 3.18 the same year, and Android with Android 6.0 [14] the following year. Despite the fact that it is supported by all operating systems, the MAC randomization option is not enabled by default on many devices (especially those running the Android operating system). During some reverse engineering investigations, it was observed that every time a burst of Probe Requests is issued, all operating systems (OSs) change their MAC addresses. Furthermore, [5] revealed how iOS 10.1.1 devices randomize their MAC address as a result of some user behavior. For smartphones running this software version, every time the device is locked or unlocked, its WiFi interface is enabled or disabled, or a connection with a different AP is attempted, a new randomized MAC address is issued.

MAC randomization solutions, while helpful in reducing the problem of potentially sensitive data being broadcast over the channel, do not completely eradicate it. These implementations contain a variety of problems and drawbacks, including utilizing additional information given in the Probe Request frame or their timing, which may lead to the monitoring of devices even if their MAC addresses are randomized [15]. Indeed, derandomization procedures can been used to greatly reduce the effect of randomization algorithms.

4. Indoors Crowd Monitoring: People Counting on a Bus

In this section, we focus on counting people traveling on a public bus in the city of Turin. Thanks to a very low cost Raspberry Pi 3B, we apply a de-randomization algorithm to count in real-time the number of passengers on-board, using the probe request broadcasted over the WiFi network by the passengers' devices.

4.1. HW description

One of the main goals of our work is to provide a sustainable, low cost solution to people's movements tracking, by creating a cheap sensor-based system that can easily be installed on public buses or other public spaces. For this reason, we aim at keeping the cost of the hardware components we employ low, while still having enough processing power at our disposal to enable on-the-fly analysis. The best solution turned out to be to use Raspberry Pi (RP) hardware. We chose a RP 3 Model B, which has a 1.2 GHz 64-bit quad-core ARM Cortex-A53 CPU, onboard 802.11n WiFi, Bluetooth, 1 GB RAM, an Ethernet connection, and four USB ports, and it runs the Linux OS. Since the RP 3 Model B's WiFi interface does not support monitor mode by default, we purchased a basic USB dongle antenna to capture frames on the WiFi channel when in monitor mode. We used a mobile USB modem that can connect to 3G and 4G/LTE networks using a SIM card to access our RP from different networks. Finally, for both power and Ethernet connectivity, a PoE splitter was used to connect our RP to the PoE cable. Our entire hardware setup is depicted in Fig. 1, which resulted in a total cost of all components of less than 50€.

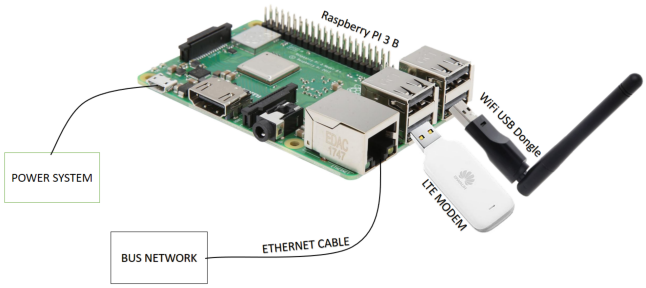

Figure 1: Hardware solution composed of a Raspberry Pi 3 Model B, a USB WiFi dongle, and a USB 3G and 4G/LTE modem.

4.2. Capturing procedure and Probe Request analysis

In order to avoid storing sensitive information, the Probe Requests recorded by our RP are examined in real time, by pipelining the output of the network sniffer on the two interfaces (WiFi showing the Probe Requests packets, and Ethernet showing the bus-internal information on the status of the doors) to the processing scripts used for the analysis of the collected messages. As detailed below, the analysis consists of four main steps, each of which is executed through a separate script file. The scripts are executed one after the other in a pipeline, so that we can perform several operations on the same input instance, in a modular manner.

Time window sampling: Due to the fact that Probe Requests from smartphones are broadcasted in bursts, performing detailed analysis on a single frame of information is impractical. Analyzing a consistent collection of frames is necessary to gain an insight on the number of passengers aboard a bus at any given moment. To this end, a script keeps track of the door status, which is retrieved through UDP from the bus network, as well as of Probe Requests recorded in the file output by the capture procedure. A new Probe Request frame is added to the final position of the time window when the bus doors are closed, and the sample thus created is inspected for further processing. When the door status at a bus stop indicates that a door is opening, the sampled window is cleared and rebuilt from scratch as soon as the doors close. As a consequence, we can be confident that we have a sample that covers the whole time interval between two consecutive door opening, i.e., between two consecutive bus stops.

MAC address analysis: Several occurrences of Probe Request frames with the same MAC address are identified in our sample window since Probe Request frames are broadcasted in bursts. In order to compact the frames from a single MAC source address, a script takes all samples in the time window just elapsed and scans all of the recorded frames. Each individual MAC address that appears inside the sample time window is recorded separately by the application, along with some extra

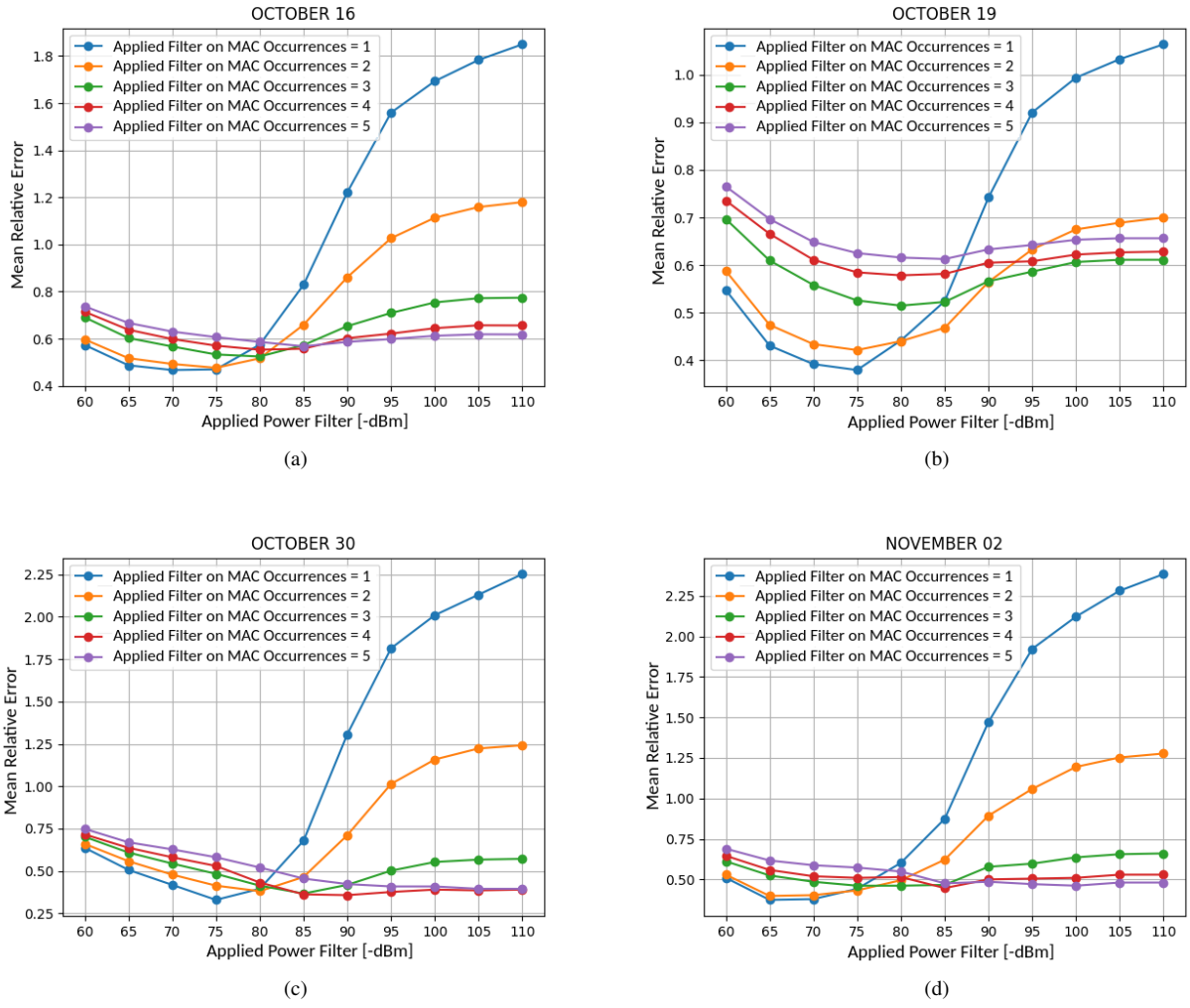

Figure 2: Mean relative error for the power filter and the MAC occurrence filter: (a) October 16, 2020, (b) October 19, 2020, (c) October 30, 2020, (d) November 2, 2020.

information for each record created. At the end of the process, a single record will include all of the data associated with a burst of an identical MAC address. After that, the fresh sample is further processed in the next phase of the analysis.

Probe Requests Filtering: Since we are only interested in detecting the devices of people on board, some filtering is needed to limit the scope of our capture to the bus only. Indeed, it is highly likely that some of the Probe Request frames gathered by our sensor were not sent by devices on board, but by those outside the vehicle. This happens mostly when the bus stops at a traffic light and is surrounded by other vehicles, but it can also occur at any one time. To address this problem, we define a power level threshold, P_l, as well as an occurrence level threshold, O_l, defined as the minimum number of occurrences of a MAC address within a sample window. A shorter number of frames in a burst can indicate that a device is passing near the bus for a brief period of time, while the average power level can be used to infer the distance between the transmitter and the sensor. If the average power of a burst of Probe Requests is less than P_l or the burst includes less than O_l frames, the corresponding input record is deleted. The data that is not removed is assumed to belong to devices on board, and it is sent to the de-randomization algorithm. P_l and O_l are two parameters that must be adequately set and are unique for each environment, i.e., they should be re-calibrated in a different bus. As discussed at the end of this section, to find the most suitable choice of parameters, we conducted extensive tests and compared our results to the actual number of passengers on board, as established by human counting sessions.

MAC de-randomization: The MAC de-randomization is the method at the center of our Probe Request counting procedure. It attempts to determine if two bursts of Probe Requests with different randomized MAC addresses are likely to belong to the same device based on the information received from the pipeline. The input data is a sequence of bursts of Probe Request frames, each with an identical MAC address. The iABACUS method [16], i.e., a recursive mechanism to reverse engineer the MAC randomization approach, is the basis for our de-randomization operation. When the iABACUS method is used, it generates a collection of lists, each of which carries

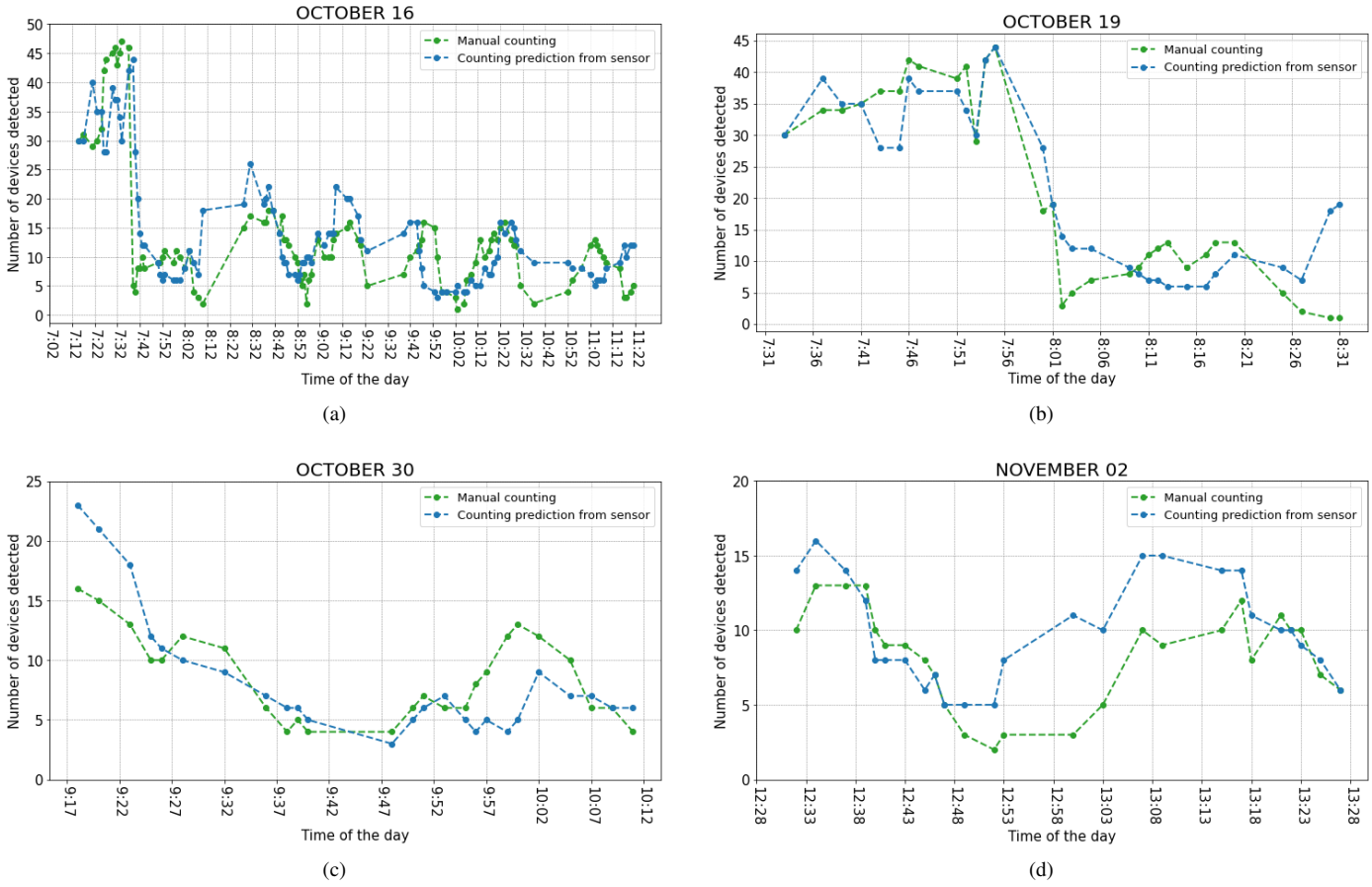

Figure 3: Performance evaluation for the manual counting sessions (all dates in 2020): (a) October 16, (b) October 19, (c) October 30, (d) November 2.

different random MAC addresses that are assumed to be associated with a single device, trying to mask its true MAC address. Finally, we count the number of separate devices identified on board by counting the number of distinct lists formed by the de-randomization process.

As a result, the number of people on board is calculated, together with the detection timestamp, and transmitted as a UDP datagram through our LTE dongle to our own server, where a Grafana dashboard is used for detailed viewing.

4.3. Tuning of system parameters

In order to properly establish all of the criteria that were crucial to the performance of our system, we needed to compare the outcomes of the counting process against the ground truth. In remote sensing systems, the concept of ground truth is used to describe data collected on-site that is used to calibrate acquired data and help in the interpretation and analysis of what has been detected. We thus scheduled several manual counting sessions to determine the exact number of individuals on board of a public bus while our system was in action. Only by comparing the manual counting results to the values detected by the sensor at the same time, one can properly set the system parameters so as to reflect the capturing environment.

We computed the difference between the ground truth numbers at each bus stop and the corresponding values from our software system. By averaging all of the errors over the course of the manual counting session, we were able to compute the mean relative error for a particular set of parameters. The main parameters employed in our methodology were the power level threshold P_l and the occurrence level threshold O_l, and they were both tuned according to the obtained experimental results. Fig. 2 shows the mean relative error for various combinations of power and occurrence thresholds. Using the values of $O_l = 1$ and $P_l = -75$ dBm as thresholds, we can filter out spurious detection occurrences, thus minimizing the mean relative error in virtually all of our comparisons, as one can see from the presented plots.

4.4. Performance evaluation

Thanks to the procedure described above, we can now rather precisely estimate the number of passengers on a bus. To allow for a more accurate evaluation of our system, the number of people on the bus was classified into three categories: “green zone” denotes a small number of passengers on board, namely less than less than 20% of the maximum bus capacity (namely, 16 in our case); “yellow zone” with a number of passengers on board ranging from 20% to almost 40% of the maximum bus capacity (namely, from 16 to 28); “red zone” with more than 40% of overall capacity (namely, 28 passengers or more),

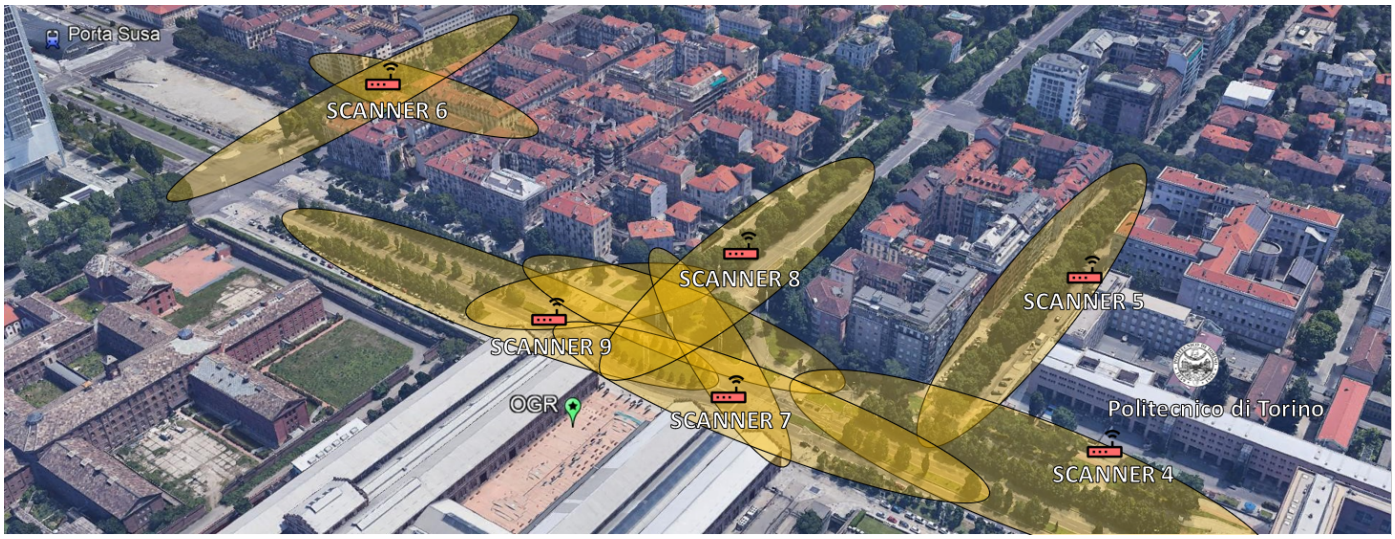

Figure 4: Coverage map of the 5G EVE testbed deployed in the area of the Politecnico di Torino campus.

which implies that safe interpersonal distances on the bus can no longer be guaranteed.

In general, our findings show that we can predict the actual number of people on board with a high degree of accuracy for all scenarios for which a ground truth was available. Fig. 3a depicts the comparison between our counting forecast and the manual counting that took place on October16, 2020. An accuracy level of 85% for the entire day is attained, if we estimate the overall relative error for this scenario. The discrepancy between the ground truth and the manual counting recorded on October 19, 2020 is represented in Fig. 3b. In this case, the overall accuracy results to be quite high, namely, 90%. Lastly, Fig. 3c presents the differences with respect to the ground truth recorded on October 30, 2020, with an overall accuracy of 91%, whereas Fig. 3d depicts the differences with respect to the ground truth recorded on November 2, 2020, with an overall accuracy of 88%.

4.5. Capturing using Bluetooth

The whole analysis provided here for WiFi could be replicated in a similar way for Bluetooth or Bluetooth Low Energy (BLE). Indeed, there are many examples in the literature of similar analysis conducted via Bluetooth interfaces [17, 18], achieving promising results.

At the outset of this project, we configured our sensors to analyse WiFi and BLE in parallel. Similarly to the WiFi capturing method, the BLE procedure featured time-windowed MAC address analysis, filtering and de-randomization. With respect to WiFi, BLE sends broadcast packets with a much smaller size and at a very high frequency. Indeed, a BLE capture can easily reach about 80-90 packets/s, all with a few bytes size. It is clear how performing an analysis on the fly for a high-frequency medium like BLE is very impacting from a CPU load point of view. For this reason, considering the limited computational capabilities of our hardware, we decided to have the whole capturing procedure based on WiFi Probe Requests only.

5. Outdoors Crowd Monitoring

We now tackle an outdoor scenario, focusing on detecting and counting people moving by any means of transportation in a specific area of Turin, Italy. In this testbed, several of commercial scanners were installed along a mile between the Politecnico di Torino campus and the Porta Susa train station, which is the major transit train station of the city of Turin, Italy. In this area, we were able to detect people who commute everyday to our campus on foot, but also a large inflow and outflow of people using different transportation means (e.g., bike, electric scooter, car, motorbike).

With respect to the scenario described in Section 4, here we consider a wider area, all kind of means of transportation, and a very high dynamic context. In this case, we chose to use black-box commercial WiFi scanners (Libelium Meshlium): all the MAC addresses captured were immediately anonymized through a SHA-224 hash function.

5.1. Testbed architecture

Fig. 5 depicts the architecture of the testbed we deployed to address our outdoor use case. It includes an edge cloud where two applications are implemented by combining multiple VNFs. All the WiFi scanners are connected through the Radio Access Network (RAN) to the OneM2M server [19], an open architecture for the provision of IoT services. An MQTT broker allows the connection to the OneM2M server with the edge cloud, hosted in a dedicated data center at Politecnico di Torino. Thanks to the use of the edge paradigm, the proposed architecture can scale horizontally with the number of scanners installed. Furthermore, the use of VNFs to realize the entire mobility application provides high flexibility in terms of required hardware resources.

Through an MQTT connection, all the data stored in the OneM2M server is retrieved by the MOB ("MOBility tracking") VNF and saved in a local MySQL database for better performance during future data analysis. Finally, the

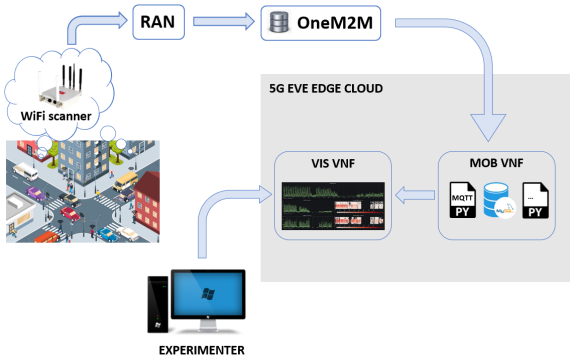

Figure 5: 5G EVE Edge Cloud testbed architecture with the two implemented VNFs.

user/experimenter can see in real-time all the data collected by the scanners through the VIS ("VISualization") VNF where a web-based visualization tool was developed.

As shown in Fig. 4, six Libelium Meshlium WiFi scanners [20] were installed, two of them on campus next to entry gates, and the others on top of traffic light poles. The scanners were configured to passively capture the "Probe Requests" on the 2.4 GHz and 5 GHz ISM bands, periodically sent by mobile devices while searching for known WiFi networks, as explained in Section 3. Each scanner is connected to the platform we developed within the 5G EVE EU project [21], through a cellular connection. Every 51 seconds[1] the scanners group the information about all the devices detected during the last sampling period and every two minutes they upload the collected data to the OneM2M server [19]. Differently from the ad-hoc solution developed in Section 4, the processed data provided by the scanners does not allow to run any de-randomization technique.

5.2. *Mobility application*

We now detail the mobility application we developed, describing each of the VNFs thereof.

5.2.1. *MOBility VNF*

The MOB VNF is in charge of retrieving data from the OneM2M platform through an MQTT (Message Queuing Telemetry Transport) client. MQTT is a very lightweight, open transport protocol, based on the publish-subscribe paradigm. The protocol runs over TCP/IP, thus it is reliable and prevents out-of-order delivery of data.

When the scanners upload a new message to the remote platform, this is saved in the server's local database, and then it is also sent to the MQTT client where the message is parsed, analyzed and stored in the MySQL database of the MOB. Before saving the data on the MOB database, two operations occur in sequence: (i) address digesting, and (ii) stationary device removal. The first operation allows reducing the size of the data on the database, while the second one allows eliminating all those devices which, being fixed objects (e.g., APs, personal computers, etc.) are considered as outliers.

[1]The periodicity is set by default by the Meshlium scanners.

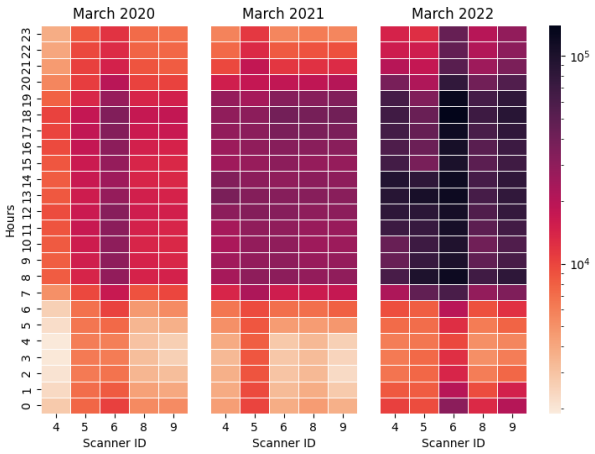

Figure 6: Heatmap showing (in log scale) the frequency of detection at different times of the day, during March 2020, 2021 and 2022.

5.2.2. *VISualization VNF*

The VIS VNF provides an aggregate view of all the data collected and analyzed through a standard web browser by means of both an interactive, real-time visualization dashboard implemented on the Grafana [22] tool, and through a simple dynamic web page linked to a python script.

Grafana is a multi-platform, open-source analytics and interactive visualization web application. It enables the creation of complex monitoring dashboards using interactive query builders on databases and provide interactive charts, graphs and alerts. It is also possible to set threshold values above or below with which to generate alerts.

It is worth mentioning that we deployed the MOB and VIS VNFs in two virtual machines (VMs) that reside on the same local network. This allows the two VMs to communicate with each other by using integrated Grafana API and SQL queries, since the MOB database coincides with the Grafana data source.

5.3. *Data analysis*

We now describe one of the results we obtained using the large amount of data acquired through our test bed. As of October 2019, the total number of detection events has been 97,728,830, corresponding to 51,255,486 distinct MAC addresses. Because of MAC randomization, the latter represents an upper bound on the number of detected devices[2]. More in detail, Figures 6 and 7 show a comparison of the data captured during the month of March of three consecutive years: 2020, 2021, and 2022. Fig. 6 depicts, through a heatmap in a logarithmic scale, the occurrences of detection for each scanner for different hours of the day. We omitted the data from scanner 7 because some data is lacking due to scanner outages. It is possible to see the effect of COVID-19 restrictions over the

[2]Unfortunately, lacking access to the Meshlium software, we cannot implement the derandomization procedure outlined in Section 4.

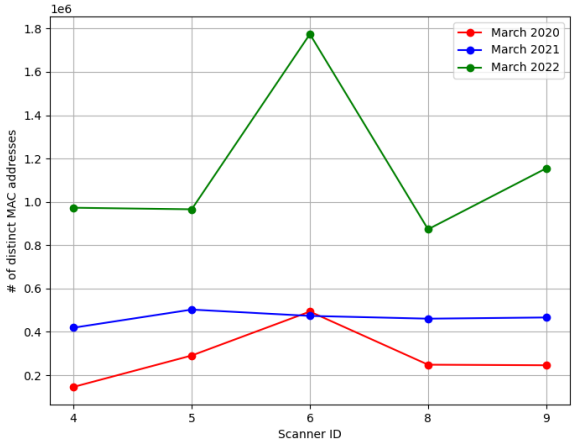

Figure 7: Number of distinct MAC addresses detected by each scanner during the whole month of March 2020, 2021, and 2022.

different years. In March 2020, a strict lockdown prevented people mobility outdoors, and this is reflected in the low numbers of smart devices. In March 2021, restrictions, though still in place, became looser and therefore more devices were detected. Finally, in March 2022 all schools, university and companies were almost back to normal and so the graph shows up to a two-magnitude increase in the presence of people in the testbed area.

Fig. 7 shows the same trend as before but in a more aggregate way, representing in detail the overall detection in the month of March by each scanner, in the three years considered. As expected, the three curves do not overlap, but there is a gap between each other, the only outlier being the data related to scanner 6, i.e., the one near Porta Susa train station, which detected the same amount of devices during 2020 and 2021, and many more compared to the other scanners in 2022.

6. Tracking Mobility

In this section, we introduce a methodology for detecting and also tracking smart devices, that can be applied in the testbed scenario described in Section 5. This methodology is apt to be applied to counting procedures using black-box scanners, as those in the previous section, but it is also amenable to do-it-yourself solutions as the ones described in Section 4.

6.1. Our methodology

Probe request patterns are used in order to pinpoint a temporal sequence, $\mathcal{T}$, to actual walking paths on the streets near the scanners previously described. The aim of our mobility tracking system is thus to associate the probes transmitted by a mobile device and detected by the WiFi scanners to the most likely path, across a given set of predefined paths that have been monitored in the area. The path classification is based on some preliminary experiments to build the ground-truth information, which yields a catalog of fingerprint vectors for each possible

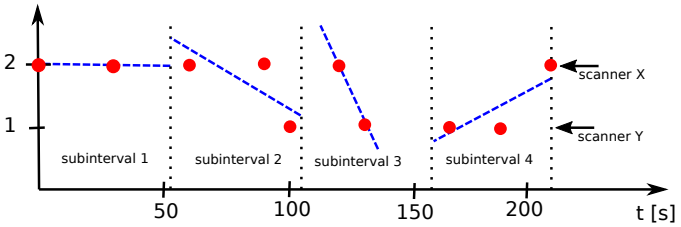

Figure 8: The points of the path map $\gamma(t)$ in the considered toy scenario.

path. Thanks to this catalog, the sequence of probes sent by a new mobile device and detected by the scanners is compared to all known fingerprints, and the path with the most similar fingerprint is selected as output of the mobility tracking, as detailed more formally in the following.

Let $\mathcal{P}$ be the set of predefined paths in the considered area to monitor, and let $p \in \mathcal{P}$ be a generic path. To compute the fingerprint f_p of a path p, we let the scanners collect probe samples by having a person take k walks along p, carrying a device. In the following, we will refer to such a device as "ground-truth device" and to each walk along p as a "run".

Consider the following toy example (assuming all times expressed in seconds):

$$\mathcal{T} = [(0, X), (30, X), (60, X), (90, X), (100, Y),$$

$$(120, X), (130, Y), (160, Y), (190, Y), (220, X)].$$

where $\mathcal{T}$ can be represented as follows: $\mathcal{T} = [(t_i, s_i)]_i$, for increasing values of t_i, $i = 0, 1, 2....$ A generic pair in $\mathcal{T}$ represents the events according to which sensor s_i detected the ground-truth device at time t_i, with $s_i \in \{X, Y\}$. The above expression can be interpreted as follows: the ground-truth device was detected by scanner X at times $0, 30, 60, 90, 120, 210$ and by scanner Y at times $100, 130, 160, 190$. Note that event detection occurs at multiples of 30 s, i.e., periodically as in the considered off-the-shelf scanners, and the sampling events have a 10 s offset between scanners. Now, from $\mathcal{T}$ we set a *path map* defined as follows: $\gamma(t_i) = 2$ if $s_i = X$, and $\gamma(t_i) = 1$ if $s_i = Y$. These two values have been arbitrarily chosen and do not affect at all the final classification result. Fig. 8 shows the path map for the considered toy scenario.

Let δ be the observation period, i.e., the total time interval during which the ground-truth device has been detected, $\delta = \max_i\{t_i\} - \min_i\{t_i\}$. Let us now partition the observation period into N temporal sub-intervals, each of duration δ/N. Notably, N is the only parameter that should be tuned according to the proposed scheme and later we will show that $N = 4$ yields already good results. In the toy example, $\delta = 210$ s and each sub-interval lasts 52.5 s when $N = 4$.

With the above data, we can now compute the fingerprint f_p. We remark that this is just one of the possible fingerprints that can be designed for path identification. The fingerprint we use is represented by a vector of $2N$ real numbers, formally $f_p \in \mathbb{R}^{2N}$. We divide such a vector in two parts.

The first N values of the fingerprint are the *coverage part* and are computed as the average of $\gamma(t)$ over each sub-interval. This

weighs the detection of the device from multiple scanners during the same interval. The remaining N values of the fingerprint are the *direction part* and model the mobility direction between the two scanners. It is computed as the slope of the best fitting linear interpolating function of the samples over the considered path.

In the considered toy example, the sub-intervals would be $[0, 52.5), [52.5, 105), [105, 157.5), [157.5, 210]$ and the corresponding fingerprint would be computed as:

$$f_p = [\underbrace{2, 1.67, 1.5, 1.33}_{\text{coverage}}, \underbrace{0, -0.019, -0.1, 0.018}_{\text{direction}}].$$

Indeed, during the first sub-interval the ground-truth device was detected by scanner X (i.e., 2) only and the corresponding slope is 0. During the second sub-interval, it was detected twice by X (i.e., 2) and once by Y (i.e., 1), thus the average is 1.67 and the corresponding slope is negative, suggesting that the device moved mainly from X to Y. A similar reasoning applies to the following two sub-intervals.

By performing many runs with the ground-truth device, a set of fingerprints is attached to each path. Thus, in order to find a match for a new device, the mobility tracking system computes its fingerprint and looks up the most similar fingerprint, using a simple Euclidean norm to evaluate the distance between vectors. In case many paths show fingerprints at a minimum distance, the path with the maximum number of minimum distance fingerprints is chosen. If still more than one path is found, the device is marked as untraceable.

This is the starting point of our mobile tracking application. We are currently studying how to enhance and optimize our methodology in order to be able, not only to detect and count people under the coverage of our scanners, but also infer some mobility patterns in an aggregated way.

7. Conclusions

Detecting the presence of people in outdoors and indoors areas is useful in several scenarios: from the design of urban spaces, to the planning of public transportation, to the implementation of mobility restriction measures such as the one that many countries put in place during the COVID-19 pandemic. In this paper, we have presented two possible methodologies for people counting and mobility detection based both on off-the-shelf hardware and commercial devices. We have shown that, although these approaches lack the precision provided by solutions based on cameras, they can be implemented in a less expensive, more practical way, which in many cases can be enough to have a rough estimate of the volume of people crowding in area or a public vehicle. These solutions are all amenable to being deployed at the edge of the network, supported by the flexibility of virtual network functions implementing data manipulation, processing, and visualization.

References

[1] Spostamenti Quotidiani e Nuove Forme di Mobilità, Statistical document, ISTAT, Istituto Nazionale di Statistica (2018).

[2] C. Badii, P. Bellini, A. Difino, P. Nesi, Sii-mobility: An IoT/IoE architecture to enhance smart city mobility and transportation services, MDPI Sensors 19 (2019).

[3] M. Uras, R. Cossu, L. Atzori, PmA: a solution for people mobility monitoring and analysis based on WiFi probes, in: IEEE SpliTech, 2019.

[4] GTT - Gruppo Torinese Trasporti, `https://www.gtt.to.it/cms/`.

[5] L. Oliveira, D. Schneider, J. De Souza, W. Shen, Mobile device detection through WiFi probe request analysis, IEEE Access (2019).

[6] F. Wang, F. Zhang, C. Wu, B. Wang, K. J. Ray Liu, Passive people counting using commodity WiFi, in: IEEE WF-IoT, 2020.

[7] Y. Yang, J. Cao, X. Liu, X. Liu, Wi-Count: Passing people counting with COTS WiFi devices, in: IEEE ICCCN, 2018.

[8] YOLOv3, `https://pjreddie.com/darknet/yolo/`.

[9] K. Gebru, C. Casetti, C. F. Chiasserini, P. Giaccone, IoT-based mobility tracking for smart city applications, in: IEEE EuCNC, 2020.

[10] K. Gebru, A privacy-preserving scheme for passive monitoring of people's flows through WiFi beacons, in: IEEE CCNC, 2022.

[11] R. Rusca, C. Casetti, P. Giaccone, IoT for real time presence sensing on the 5G EVE infrastructure, in: IEEE MedComNet, 2021.

[12] M. Cunche, I know your MAC address: targeted tracking of individual using Wi-Fi, Journal of Computer Virology and Hacking Techniques (2014).

[13] 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, Privacy Enhancing Technologies (2017).

[14] C. Matte, Wi-fi tracking: Fingerprinting attacks and counter-measures, Ph.D. thesis, Université de Lyon (2017).

[15] 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: ACM Asia conference on computer and communications security, 2016.

[16] M. Nitti, F. Pinna, L. Pintor, V. Pilloni, B. Barabino, iABACUS: A wi-fi-based automatic bus passenger counting system, MPDI Energies (2020).

[17] E. Longo, A. E. Redondi, M. Cesana, Accurate occupancy estimation with WiFi and bluetooth/BLE packet capture, Computer Networks 163 (2019) 106876. doi:https://doi.org/10.1016/j.comnet.2019.106876.

[18] F. Brockmann, M. Handte, P. J. Marrón, CutiQueue: People Counting in Waiting Lines Using Bluetooth Low Energy Based Passive Presence Detection, in: 2018 14th International Conference on Intelligent Environments (IE), 2018, pp. 1–8. doi:10.1109/IE.2018.00009.

[19] OneM2M, `http://www.onem2m.org`.

[20] Libelium Meshlium, `http://www.libelium.com/products/meshlium/`.

[21] European 5G validation platform for extensive trials, `https://www.5g-eve.eu/`.

[22] Grafana: The open observability platform, `https://grafana.com/`.