



Politecnico
di Torino

ScuDo

Scuola di Dottorato - Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Computer and Control Engineering (XXXVI cycle)

Crowd Monitoring and City Sensing Techniques Supported by Next Generation Mobile Networks

Riccardo Rusca

Supervisor:

Prof. Claudio Ettore Casetti

Doctoral Examination Committee:

Prof. Floriano De Rango, Università della Calabria

Prof. Guido Marchetto, Politecnico di Torino

Prof. Marco Di Felice, Università degli studi di Bologna

Prof. Paolo Giaccone, Politecnico di Torino

Prof. Pietro Manzoni, Universidad Politecnica de Valencia

Politecnico di Torino

2024

Declaration

I affirm that the contents and structure of this thesis represent my unique creation and do not infringe upon the rights of any third parties, including those pertaining to the confidentiality of personal information.

Riccardo Rusca
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I dedicate this Ph.D. thesis with heartfelt gratitude to my parents, whose unwavering support and encouragement have been my pillars of strength throughout this journey.

To Martina, my girlfriend, your love, patience, and understanding have been a constant support for me.

I extend my sincere appreciation to my esteemed advisors, Claudio Casetti and Paolo Giaccone, whose guidance, expertise, and mentorship have shaped my academic and research endeavors. I am grateful to Francesco Restuccia for inviting me to spend some time in Portland, Maine, USA, which provided me with a unique opportunity to expand my knowledge and collaborate with passionate researchers.

I also express my thanks to all the people with whom I had the privilege of collaborating during my Ph.D. Your contributions and collaborative spirit have enriched my academic experience, and I am truly grateful for the collective efforts that have made this research possible.

Abstract

In an era of rapid technological advancement and global challenges, the paradigm of overseeing crowds, ensuring safety, and safeguarding privacy undergoes profound transformation. The COVID-19 pandemic has prompted us to reconsider how we manage large gatherings while respecting personal freedoms and privacy. As we transition beyond the pandemic, it is crucial to strike a balance between enhancing safety measures at events and safeguarding privacy rights.

Traditional crowd surveillance methods often struggle to capture the nuanced dynamics of crowd behavior and movement patterns. With ongoing health concerns, there is a growing need for advanced monitoring systems that provide timely insights and allow for proactive measures. Additionally, the advent of smart cities and new technologies like V2X communication and artificial intelligence presents ethical, legal, and technical challenges, particularly regarding personal data protection. Regulatory efforts like the General Data Protection Regulation (GDPR) [39] aim to address these challenges by establishing clear guidelines for data security and the responsible use of personal information in monitoring initiatives.

This thesis dives deep the intersection of crowd monitoring, public safety, data privacy, and machine learning, accentuated by the challenges and implications ushered in by the post-COVID-19 landscape. A crucial investigation into the complexities of 802.11 Probe Request messages and MAC address randomization unveils nuanced behavioral patterns and privacy concerns, notably concerning MAC addresses. Leveraging this understanding, innovative methodologies are devised to replicate authentic device behaviors, facilitating the generation of realistic data traces able to boost the development of machine learning algorithms for improved people counting, all while reinforcing user privacy through specialized data structures and sophisticated anonymization methods.

As we advance into an era where artificial intelligence and data privacy take center stage, this thesis delves into the transformative concept of federated learning. Embracing the privacy-centric design inherent in federated learning—where clients independently train machine learning models and share only model’s parameters over the network—this ensure robust data protection while also optimizing training efficiency and accelerating convergence rates. Within the realm of vehicular applications, this research shows how federated learning can harness collective intelligence to fine-tune trajectory prediction models. This approach not only delivers scalability and responsiveness but also fortifies privacy safeguards, showcasing the potential of collaborative methodologies in enhancing urban mobility solutions.

This collaborative spirit extends to the development of a innovative data-driven framework tailored for Radio Frequency (RF) mobility scenario generation. By integrating real-world mobility traces with advanced channel modeling techniques, this framework establishes a robust foundation for crafting realistic V2X scenarios for channel emulators, fostering innovation in urban mobility solutions.

As technological advancements continue to reshape our world, striking a harmonious balance between public safety, crowd monitoring, and data privacy remains paramount. This research underscores the significance of adapting innovative methodologies and collaborative approaches, such as artificial intelligence, machine learning, federated learning and advanced RF mobility scenario frameworks, to navigate the complexities of modern urban environments effectively. By prioritizing both safety and privacy, we lay the groundwork for more resilient, secure, and inclusive communities, ensuring that technological progress aligns with the values and rights of individuals.

Contents

List of Figures	xi
List of Tables	xvi
1 Introduction	1
1.1 Publications	3
1.2 Main contributions	5
1.3 Outline	7
2 WiFi Probe Request for Crowd Monitoring	9
2.1 Research motivation and State of the Art	9
2.2 Main contributions	12
2.3 IEEE 802.11 connection procedure	13
2.3.1 Interaction between device and access point	13
2.3.2 IEEE 802.11 Probe Request frame	15
2.3.3 IEEE 802.11 Probe Response frame	17
2.4 MAC address and privacy	18
2.4.1 MAC address	19
2.4.2 MAC address randomization	21
2.5 Methodology	23
2.5.1 Hardware description	24

Contents

2.5.2	Testing location	26
2.5.3	Testing methodology	27
2.5.4	Data analysis	29
2.5.5	Experiment extension	30
2.6	Results	31
2.6.1	Experimental results	32
2.6.2	Analysis and comparison between different devices	38
2.6.3	Results comparison between different years and OS version	46
2.6.4	Experimental findings	49
2.6.5	Limitations	50
2.7	Conclusions and future works	51
3	Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters	52
3.1	Research motivation and State of the Art	53
3.2	Main contributions	54
3.3	Anonymization techniques for storing MAC addresses	55
3.3.1	GDPR	56
3.3.2	Hash function	57
3.3.3	Salted hash	60
3.3.4	Truncated hash	61
3.3.5	Bloom filter	62
3.4	Probe request generator	65
3.4.1	Back-end data	65
3.4.2	Finite-State machine	68
3.4.3	Event-driven generator	72
3.4.4	Messages collision avoidance	74

Contents

3.4.5	Probe request packet generation	74
3.4.6	Validation	76
3.5	Bloom filters for flow analysis	80
3.5.1	Bloom filter sizing	80
3.5.2	Bloom filter privacy properties	84
3.5.3	Anonymization noise	86
3.5.4	Multiple anonymity	87
3.5.5	Counting elements stored in a Bloom filter	88
3.5.6	Intersection of different Bloom filters	90
3.6	Conclusions and future works	93
4	People counting and crowd monitoring in real use cases	95
4.1	Research motivation	95
4.2	Main contributions	97
4.3	WiFi probe request sniffers	98
4.3.1	Meshlium scanner by Libelium	99
4.3.2	Raspberry Pi	100
4.3.3	Meshlium vs Raspberry Pi	101
4.4	WiFi probe requests people counting algorithms	102
4.4.1	Naive algorithms	103
4.4.2	De-randomization algorithms	104
4.5	Real-time presence sensing on a 5G infrastructure	107
4.5.1	Implemented architecture	108
4.5.2	Mobility framework	109
4.5.3	Results	110
4.6	Passive crowd monitoring inside a bus	112
4.6.1	Capturing framework	112

4.6.2	Validation and parameter tuning	115
4.6.3	Performance evaluation	115
4.7	Machine learning-driven privacy-preserving framework for crowd management	118
4.7.1	DBSCAN clustering method	118
4.7.2	DBSCAN clustering algorithm	119
4.7.3	Counting pipeline	121
4.7.4	Counting results	123
4.8	Conclusions and future works	126
5	Federated Learning Empowered Vehicular Networks	128
5.1	Research motivation and State of the Art	129
5.2	Main contributions	131
5.3	Federated Learning	132
5.3.1	Centralized vs. Decentralized	133
5.3.2	Synchronous vs. Asynchronous	134
5.3.3	Federated learning frameworks and implementations	135
5.4	Methodology	136
5.4.1	Implemented architecture	136
5.4.2	Urban Environment simulation	138
5.4.3	Federated learning framework	141
5.5	Results	143
5.5.1	Trajectory dataset and LSTM algorithm	144
5.5.2	Federated learning framework setup	145
5.5.3	Numerical results	148
5.6	Conclusions and future works	152
6	Radio Frequency mobility scenario for wireless channel emulators	154

Contents

6.1	Research motivation and State of the Art	155
6.2	Main contributions	156
6.3	Wireless channel emulators	157
6.4	Methodology	160
6.4.1	Framework	160
6.5	V2X scenario in Colosseum	166
6.5.1	Vehicular dataset	167
6.5.2	Scenario creation	170
6.5.3	Scenario validation	173
6.6	Conclusions and future works	176
7	Conclusions	177
	List of acronyms	180
	Bibliography	185

List of Figures

2.1	Comprehensive overview of the IEEE 802.11 Network Discovery and Association procedures between an access point and a smart device.	14
2.2	Architecture of the IEEE 802.11 Probe Request frame. Reproduced from [41].	15
2.3	Architecture of the IEEE 802.11 Probe Response frame. Reproduced from [41].	18
2.4	Architecture of 48-bit MAC address with highlighted the local/global and unicast/multicast bits.	19
2.5	Sniffer sensor for capturing 802.11 Probe Request messages.	24
2.6	Sniffer wireless network interface configuration.	25
2.7	Location of the test, performed in March 2022.	27
2.8	Example of TShark output file (.pcap) on Wireshark.	28
2.9	Location of the test, performed in March 2023.	30
2.10	Comprehensive analysis of probe request packets sent from an Apple iPhone 11 for different usage phases and for different connection states.	33
2.11	Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 11, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states.	36

List of Figures

2.12 Comprehensive analysis of RSSI power field for an Apple iPhone 11 as a function of time, for different usage phases and for different connection states. 37

2.13 Comprehensive analysis of probe request packets sent from an Apple iPhone 6 for different usage phases and for different connection states. 39

2.14 Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 6, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states. 40

2.15 Comprehensive analysis of RSSI power field for an Apple iPhone 6 as a function of time, for different usage phases and for different connection states. 41

2.16 Comprehensive analysis of probe request packets sent from an Apple iPhone 11, during 2023 experiments, for different usage phases and for different connection states. 47

2.17 Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 11, during 2023 experiments, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states. 48

2.18 Comprehensive analysis of RSSI power field for an Apple iPhone 11, during 2023 experiments, as a function of time, for different usage phases and for different connection states. 49

3.1 Hash function algorithm. 58

3.2 Salted hash function. 61

3.3 Truncated hash function. 62

3.4 Bloom filter algorithm. 63

3.5 Event-driven finite-state machine diagram. 69

3.6 Example of statistics given in output from the probe request generator. 71

3.7 Probability transition values diagram for the device’s state. 73

List of Figures

3.8	False positive probability with different values of k	82
3.9	False positive probability vs. number of inserted MAC addresses into a Bloom filter.	83
3.10	False positive probability vs. number of inserted MAC addresses into different Bloom filters. Each Bloom filter is configured with $m = 10,000$ and a different values of k	83
3.11	Elements v_1, v_2, v_3 belonging to the hiding set of a 10-bit BF, storing x_1, x_2, x_3 . Arrows indicate the bits that are set to one according to the two hash functions H_1 and H_2	84
3.12	γ -deniability value vs. number of inserted MAC addresses into a Bloom filter.	87
3.13	γ -deniability value vs. number of inserted MAC addresses into a Bloom filter, for different values of K	88
3.14	Accuracy evaluation of (3.12) when comparing number of elements inserted in the Bloom filter and counted ones.	89
3.15	Comparison of flow monitoring accuracy between (3.12) denoted as c_1 and (3.13) denoted as c_2	92
3.16	Comparison of flow monitoring relative errors between c_1 and c_2	93
4.1	Meshlium scanner by Libelium [59].	99
4.2	Raspberry Pi 4B [74].	100
4.3	Example of clustering.	105
4.4	WiFi scanner coverage map in the <i>innovation mile</i> in Turin.	108
4.5	Edge cloud architecture in the 5G EVE Eu project.	109
4.6	Heatmap illustrating hourly log-scale detection frequencies for March 2020, 2021, and 2022.	111
4.7	Count of unique MAC addresses identified by each scanner throughout March for 2020, 2021, and 2022.	112
4.8	Sniffer solution featuring a Raspberry Pi 3B, a USB WiFi dongle, and a USB LTE modem.	113

List of Figures

4.9	Mean relative error comparison for two metrics: power filter and MAC occurrence filter.	116
4.10	Performance evaluation with manual counting for two days.	117
4.11	Counting pipeline.	122
4.12	Comparison between ground truth and results obtained from our framework and an implementation of iABACUS one, for Dataset G.	125
5.1	Federated learning architecture overview.	132
5.2	Comparison between centralized and decentralized federated learning.	134
5.3	Implemented architecture.	137
5.4	Considered area in Turin.	139
5.5	Considered area in Turin with seven eNBs.	140
5.6	Urban mobility simulation output file.	141
5.7	Federated learning schema.	142
5.8	Frame of the video footage took by a drone in Cyprus.	145
5.9	Experimental setup scheme.	148
5.10	Comparison of replacement rates in synthetic and real-world scenarios across three distinct client selections.	149
5.11	Comparison of training times in synthetic and real-world scenarios across three distinct client selections.	150
5.12	Comparison of FL cycle counts in synthetic and real-world scenarios needed to reach a set threshold across three distinct client selections.	151
5.13	Comparison of FL round counts in synthetic and real-world scenarios to attain a specified threshold across three distinct client selections.	151
6.1	Overview of the proposed framework for generating and validating RF mobility scenarios.	160
6.2	Visualization of vehicles on a map for a specific timestamp, marking with green dots if the vehicle is inside the coverage area (blue borders) and red dots if outside.	162

List of Figures

6.3	Visualization of the ray-tracing output, depicting the receiver with a blue point and the transmitter with a red point. Each ray represents reflections and diffractions of the signal, with varying colors indicating different power levels.	163
6.4	A sample of the data available within the open dataset [75] leveraged for the creation of our V2X scenario.	167
6.5	SAMARCANDA traces with 1 km ² square area in blue. Reproduced from [75].	169
6.6	Ray-tracing output for a specific timestamp between vehicle 9 and the antenna.	172
6.7	Visualization of the K-means algorithm applied to a collection of rays from the ray-tracing process.	173
6.8	Comparison between RTT and distance from the base station. Data refer to vehicle 9 of SAMARCANDA [75].	174
6.9	Comparison between SNR and the distance from the base station. Data refer to vehicle 9 of SAMARCANDA [75]. The blue line represents the average SNR over 5 different experiments, with the 99% confidence intervals represented by orange dotted lines.	175

List of Tables

2.1	Device list, tested in 2022.	26
2.2	Device list, tested in 2023.	31
2.3	Smartphones behaviour comparison.	43
2.4	Smartphones behaviour comparison.	44
2.5	Tablet and laptops behaviour comparison.	45
3.1	Devices tested for our probe request generator database.	67
3.2	Locked phase results for an Apple iPhone 11 (mean, coefficient of variation).	78
3.3	Awake phase results for an Apple iPhone 11 (mean, coefficient of variation).	78
3.4	Active phase results for an Apple iPhone 11 (mean, coefficient of variation).	79
3.5	Definitions and notations for Bloom filters.	80
4.1	Comparison between Meshlium by Libelium and Raspberry Pi 4B. .	101
4.2	Crowd monitoring results.	124
5.1	Simulation outcomes varying with different sizes of vehicle trajectories.	148
6.1	MATLAB simulation parameters.	171

Chapter 1

Introduction

In an age marked by rapid technological advancements and unprecedented global challenges, the landscape of crowd monitoring, public safety, and data privacy has undergone significant transformations. In the wake of the COVID-19 pandemic, the world has been profoundly reshaped by social distancing measures, relegating the once-thriving realm of public gatherings to the distant past. As we navigate the post-pandemic landscape, restrictions on the size of gatherings continue to loom, necessitating a vigilant approach to crowd control. Nevertheless, the prospect of large-scale public events making a triumphant return in the post-COVID era is on the horizon. This resurgence brings both excitement and formidable challenges to the forefront.

The COVID-19 pandemic serves as a strong reminder of the critical role that crowd monitoring plays in safeguarding public health and ensuring societal resilience. As nations addressed the rapid virus spread, it became evident that traditional crowd monitoring methods had their limitations. Basic headcounts and manual surveillance techniques proved inadequate for capturing the nuanced dynamics of crowd behavior, mobility patterns, and density variations. Consequently, there emerged an urgent need for advanced monitoring systems capable of providing real-time insights, facilitating informed decision-making, and enabling proactive interventions to mitigate health risks and contain the spread of infectious diseases.

However, as organizations and governments raced to deploy sophisticated crowd monitoring technologies, a confluence of ethical, legal, and technical challenges emerged. Foremost among these challenges is the tension between the imperatives of

public safety and the fundamental right to privacy. The proliferation of surveillance technologies, data collection mechanisms, and tracking algorithms has raised profound concerns about individual autonomy, data sovereignty, and civil liberties. In response to these concerns, regulatory frameworks like the General Data Protection Regulation (GDPR) [39] have been implemented to safeguard individual rights, impose stringent data protection standards, and regulate the use of personal data in crowd monitoring applications.

In the context of smart cities, the convergence of people counting, traffic congestion management, public transportation enhancement, pedestrian safety, and vehicle-to-everything (V2X) communication heralds a transformative era of urban mobility and efficiency. As cities grapple with burgeoning populations, increasing vehicular traffic, and evolving mobility patterns, the need for advanced trajectory prediction systems becomes paramount. These systems play a pivotal role in optimizing traffic flow, enhancing public transportation services, mitigating congestion, and ensuring pedestrian safety. By leveraging real-time data from various sources, including vehicular sensors, pedestrian detectors, public transportation networks, and smart infrastructure, trajectory prediction algorithms can anticipate movement patterns, identify potential bottlenecks, and facilitate proactive interventions to optimize urban mobility and enhance public safety.

However, developing robust trajectory prediction systems poses significant computational, logistical, and ethical challenges. Traditional centralized approaches often encounter scalability limitations, data privacy concerns, and regulatory constraints, particularly in open, distributed environments characterized by diverse stakeholders, heterogeneous data sources, and stringent privacy requirements.

In this context, federated learning emerges as a compelling solution to reconcile the competing imperatives of scalability, privacy, and regulatory compliance. By decentralizing the model training process across a network of interconnected devices, vehicles, and infrastructure elements, federated learning enables collaborative learning without compromising sensitive data. In the context of V2X communication, this approach allows vehicles, pedestrians, and smart infrastructure to collaboratively refine trajectory prediction models, thereby enhancing the accuracy, reliability, and responsiveness of urban mobility systems. Moreover, by preserving data privacy and adhering to regulatory requirements, federated learning fosters public trust,

encourages stakeholder collaboration, and promotes responsible innovation in the pursuit of smarter, safer, and more sustainable cities.

Furthermore, the efficacy and reliability of smart city systems are contingent upon the ability to emulate real-world environments accurately. In this context, large-scale channel emulators play a pivotal role, enabling researchers to create reproducible test scenarios, evaluate system performance, and validate algorithms under diverse Radio Frequency (RF) conditions. By simulating the propagation, interference, and attenuation of wireless signals, channel emulators facilitate the development of robust, resilient, and scalable monitoring solutions capable of operating effectively in complex and dynamic environments.

This thesis endeavors to explore, analyze, and innovate at the nexus of crowd monitoring, public safety, data privacy, and machine learning. By conducting a comprehensive review of existing technologies, regulatory frameworks, and ethical considerations, this research aims to identify gaps, address challenges, and propose novel methodologies, algorithms, and frameworks that advance the state of the art in this critical domain. Through rigorous empirical analysis, theoretical insights, and practical applications, this thesis seeks to contribute to the development of more secure, equitable, and resilient societies, ultimately enhancing the safety, well-being, and privacy of individuals around the globe.

1.1 Publications

All the contributions of the three main topics are the following:

Crowd monitoring and people counting leveraging WiFi probe requests

- [82] Riccardo Rusca, Claudio Casetti, and Paolo Giaccone. IoT for real time presence sensing on the 5G EVE infrastructure. In 2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet), Ibiza, Spain, pages 1-8. IEEE, 2021.
Available at <https://ieeexplore.ieee.org/document/9501245>
- [43] Kalkidan Gebru, Marco Rapelli, Riccardo Rusca, Claudio Casetti, Carla Fabiana Chiasserini, and Paolo Giaccone. Edge-based passive crowd monitoring through WiFi Beacons. *Computer Communications*, Volume 192, 2022,

Pages 163-170.

Available at

<https://www.sciencedirect.com/science/article/abs/pii/S0140366422001980>

- [85] Riccardo Rusca, Filippo Sansoldo, Claudio Casetti, and Paolo Giaccone. What WiFi probe requests can tell you. In IEEE Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, pages 1086–1091. IEEE, 2023.

Available at <https://ieeexplore.ieee.org/document/10060447>

- [83] Riccardo Rusca, Alex Carluccio, Diego Gasco, and Paolo 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), Cosenza, Italy, pages 1–6. IEEE, 2023.

Available at <https://ieeexplore.ieee.org/document/10286944>

- [86] Riccardo Rusca, Alex Carluccio, Claudio Casetti, and Paolo Giaccone. Privacy-preserving WiFi-based Crowd Monitoring. *Transactions on Emerging Telecommunications Technologies*, 2023.

Available at <https://onlinelibrary.wiley.com/doi/10.1002/ett.4956>

Federated Learning in vehicular networks

- [49] Giuseppe La Bruna, Carlos Risma Carletti, Riccardo Rusca, Claudio Casetti, Carla Fabiana Chiasserini, Marina Giordanino, and Roberto Tola. Edge-assisted federated learning in vehicular networks. In 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, pages 163–170. IEEE, 2022.

Available at <https://ieeexplore.ieee.org/document/10076689>

Radio Frequency mobility scenario for channel emulators

- [84] Riccardo Rusca, Francesco Raviglione, Claudio Casetti, Paolo Giaccone, and Francesco Restuccia. Mobile RF Scenario Design for Massive-Scale Wireless Channel Emulators. In 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Gothenburg, Sweden,

pages 675–680. IEEE, 2023.

Available at <https://ieeexplore.ieee.org/document/10188319>

1.2 Main contributions

This Section summarizes the main contributions of this thesis:

- **Chapter 2.** Introduction to the 802.11 Probe Request and Probe Response management frames, alongside a detailed explanation of MAC addresses and the concept of MAC address randomization. Additionally, it offers an overview on how the MAC address randomization is implemented across modern operating systems.
- **Chapter 2.** Description of the methodology employed to conduct a thorough analysis of probe request messages across a variety of devices spanning different operating systems, types, and ages. Parameters such as packet frequency, MAC address type (randomized, non-randomized, global, local), burst frequency, inter-packet time, SSID field, VHT capabilities, and others were extracted from captures. These parameters were meticulously compared among the devices to identify correlations and behaviors.
- **Chapter 3.** Introduction to the fundamental functions and data structures crucial for maintaining data privacy when storing sensitive information. It covers a range of techniques including various hash functions employing diverse hashing algorithms, salted hashing, truncated hashing, and Bloom filters.
- **Chapter 3.** Design and specification of a novel probe request generator, capable of swiftly creating realistic probe request traces while maintaining accuracy. With the ability to emulate hundreds of devices within seconds, this generator offers flexibility through customizable input parameters, allowing users to specify the number and types of devices to emulate, as well as the preferred time window. In addition to providing the *.pcap* trace, the output includes invaluable ground truth data, indicating the number of devices that have transmitted at least one probe request, along with other useful statistics.

- **Chapter 3.** Introduction of the concept of γ -deniability and γ - K -anonymity applied to Bloom filters, alongside the novel concept of *anonymization noise*. These concepts are pivotal for achieving GDPR compliance when transmitting and storing sensitive information such as MAC addresses.
- **Chapter 4.** Comprehensive overview of different WiFi probe request sniffers with different sniffing software. Additionally, it introduces a spectrum of people counting algorithms ranging from naive solutions to more complex ML-driven algorithms.
- **Chapter 4.** Design, development and outcomes of various real-world use cases focused on people counting and crowd monitoring. The first scenario encompasses an outdoor setting along the so-called *innovation mile* in the city of Turin. Through the deployment of multiple sniffers, the project collected over 100 million detection events, representing more than 51 million distinct anonymized MAC addresses. In contrast, the second scenario involves an indoor environment, specifically targeting passenger counts onboard buses. During this second scenario we started to apply more sophisticated algorithms for people counting due to the evolving landscape of MAC address randomization implementation on newer devices.
- **Chapter 4.** Introduction of a ML-driven privacy-preserving framework tailored for crowd monitoring applications. Leveraging insights gleaned from our probe request generator and extensive knowledge of probe request behavior, this framework integrates a novel clustering algorithm developed and validated specifically for this purpose. Additionally, the framework incorporates Bloom filter structures fortified with *anonymization noise*, ensuring robust privacy protection while storing MAC addresses. Furthermore, it facilitates the analysis of people flows through straightforward operations such as intersecting multiple Bloom filters.
- **Chapter 5.** Introduction overview of federated learning, elucidating its core principles and methodologies. It delves into the various topologies and architectures prevalent in federated learning, with a particular focus on centralized and decentralized models. Furthermore, it explores the distinctions between synchronous and asynchronous architectures, highlighting their respective advantages and trade-offs.

- **Chapter 5.** Design and specification of a innovative application of federated learning in a vehicular context. The proposed framework involves collaborative training of LSTM models for trajectory prediction using data gathered from different vehicles. Federated learning is employed in both real-world and synthetic scenarios, with the former leveraging actual mobility traces captured around the *innovation mile* in Turin.
- **Chapter 6.** Introduction of wireless channel emulators and description of all their key characteristics. Furthermore, it introduces all the main platforms available for wireless channel emulation.
- **Chapter 6.** Design, development and testing of a novel framework dedicated to crafting radio frequency mobility scenarios for wireless channel emulators. It outlines each step of the process, starting from data collection to the generation of the final matrix output. Following this methodology, we developed a mobility scenario using real-world traces collected in the city of Pinerolo, Italy. To validate the effectiveness and accuracy of our framework and 5G scenario, we conducted rigorous testing using the Colosseum [24] emulator, renowned as the world's largest wireless channel emulator.

1.3 Outline

The remainder of this thesis is organized as follows:

- **Chapter 2** presents a detailed analysis of 802.11 Probe Request messages, encompassing their behavioral patterns, associated parameters across device types, and influencing sending factors. Additionally, it delves into the intricacies of MAC address randomization, highlighting its implementation, changing frequency and implications for privacy and security. Furthermore, Chapter 2 explores correlation analysis among probe request parameters, unveiling underlying patterns and dependencies in device communication behavior.
- **Chapter 3** introduces a groundbreaking probe request generator designed to replicate authentic behaviors of multiple tested devices, enabling the creation of realistic *.pcap* traces paired with accurate ground truth data. Additionally, Chapter 3 unveils a novel application of what we called *anonymization noise* in

Bloom filters, ensuring 1-deniability for stored MAC addresses, thus enhancing privacy and deniability probabilities.

- **Chapter 4** delves into different crowd monitoring applications, showcasing a solution for people counting in Turin’s “innovation mile” through extensive data analysis of over 100 million probe requests. Additionally, Chapter 4 explores innovative methods for passive crowd monitoring inside buses, particularly valuable during the COVID-19 pandemic. Furthermore, it introduces a machine learning-driven privacy-preserving framework that addresses MAC address randomization challenges, incorporating anonymization noise within a Bloom filter data structure to ensure enhanced user privacy in crowd management applications.
- **Chapter 5** presents a pioneering application of federated learning within a vehicular context, where vehicles collaboratively train an LSTM model for trajectory prediction. Additionally, Chapter 5 showcases the practical implementation of this approach by applying federated learning algorithms to real-world mobility traces from Turin’s “innovation mile”, demonstrating its effectiveness when compared to a more static scenario.
- **Chapter 6** unveils a novel data-driven framework for radio frequency mobility scenario creation, leveraging real-world mobility traces. Moreover, Chapter 6 highlights the application of this innovative framework in crafting a comprehensive V2X scenario for Colosseum [24], integrating real-world mobility traces and utilizing advanced ray-tracing algorithms for accurate channel path loss calculations.
- **Chapter 7** presents the conclusions and the future research directions.

Chapter 2

WiFi Probe Request for Crowd Monitoring

Controlling and analyzing large crowds of people has always been important, albeit not always the central focus of research. However, when the COVID-19 pandemic emerged in March 2020, everything changed dramatically. Public and private spaces swiftly implemented stringent requirements regarding maximum occupancy limits, making it imperative to accurately determine the number of people within specific areas and identify gatherings. In this context, one of the primary techniques that have proven invaluable is the analysis of WiFi Probe Request (PR) messages. This Chapter delves into a comprehensive exploration of probe request messages, their behavior, and the key challenges associated with utilizing this technique for crowd monitoring in various environments. In an era where maintaining social distancing and monitoring crowd sizes have become paramount, understanding the intricacies of WiFi probe request messages is instrumental in effectively managing and ensuring public safety.

2.1 Research motivation and State of the Art

In recent years, the significance of crowd estimation has come into sharp focus in the realm of crisis management. Accurate people counting and crowd monitoring have emerged as critical components in crisis management and disaster response scenarios. These capabilities are essential for enabling informed decision-making

and upholding public safety. Indeed, possessing the ability to gauge the number of individuals within a specific environment and understand their movement patterns holds tremendous potential for enhancing situational management. However, the challenge of accurately counting and tracking people in large gatherings or chaotic situations has perennially posed difficulties. Traditional methods, including surveillance cameras, LiDAR and infrared systems, as well as WiFi and Bluetooth fingerprints, have been widely employed in attempting to address this challenge.

In [34], the authors delve into the challenge of people counting within multi-camera surveillance systems. Their proposed method combines partial body detection and person re-identification to achieve precise headcounts in overlapping areas. Similarly, in [62], another group of researchers presents a system for detecting and tallying individuals, employing computer vision techniques, with a particular focus on overhead view-based detection. Both of these prior studies showcased commendable outcomes. However, it is important to acknowledge that techniques relying on video cameras come with their share of issues. Firstly, they entail high hardware costs due to the considerable computational resources demanded. Secondly, outdoor scenarios pose significant challenges, mainly because of fluctuating lighting conditions and the presence of dense crowds. Lastly, the storage and retention of face detection data raise privacy concerns, adding another layer of complexity to the utilization of video-based methods.

Additionally, recent research efforts [44] and [28] have explored the utilization of LiDAR sensors. The proposed systems use a LiDAR sensor to capture coarse human shapes by concatenating multiple two-dimensional range scans. Both the frameworks are designed to work under dynamically illuminated conditions solving the problem of detecting object in indoor and (limited) outdoor scenarios. In contrast to video camera techniques, LiDAR solutions do indeed offer a notable advantage by mitigating privacy concerns. However, they still confront challenges associated with high hardware costs and the need for adaptability across a wide range of applications and environments.

Authors of [36] and [68] take a different approach by investigating the utilization of Passive Infrared (PIR) sensors. These low-cost infrared sensors are employed to detect the presence and movement of individuals within different areas of a building while preserving privacy. In the first paper PIR sensors are used in combination with a multi-camera surveillance systems in order to classify different types of

human motion, specifically entry/exit motions and ordinary activities. While, in the second paper, the sensors capture thermal images and temperature gradients, and the collected data undergoes processing through a specialized pattern recognition algorithm. This algorithm generates outputs based on detected behaviors, such as entry, exit, or occupancy, offering a discreet yet effective means of monitoring spaces. One significant limitation of PIR sensors is their relatively small coverage area. These sensors are typically designed for smaller spaces, making them less suitable for monitoring larger or expansive environments. Managing and analyzing this data can pose challenges, especially when real-time or near-real-time monitoring is required.

Conversely, the studies presented in [77, 25, 63] investigate the use of WiFi probe request messages as an effective approach for crowd monitoring across diverse scenarios. This method involves collecting WiFi fingerprints from mobile devices and offers a promising avenue for crowd analysis. These research endeavors aim to analyze the MAC addresses and other parameters embedded within the PR messages. However, it is worth noting that most of the previously proposed techniques are no longer applicable due to the evolving nature of the information contained in probe requests. This transformation stems from both the passage of time and the deliberate customization of these values by major smart device manufacturers, aimed at bolstering user privacy. As for the other techniques discussed earlier, the WiFi probe request technique stands out as a completely passive approach from the user's perspective. What distinguishes it even further is its versatility, as it can be deployed in virtually any kind of environment and is not susceptible to weather and light conditions. The coverage of this solution primarily depends on the type of antenna used for capturing network data, with the potential to span from approximately 80 meters up to over 300 meters or more when using directional antennas. While the WiFi probe request technique offers several advantages, it is essential to note that it may not be entirely privacy-compliant by default. However, as detailed in Chapter 3, specific ad-hoc techniques can be implemented to ensure full compliance with regulations such as GDPR [39]. These measures can help strike a balance between effective crowd monitoring and safeguarding individuals' privacy rights, making this technique a promising option for various applications and settings.

2.2 Main contributions

The current study, whose contribution can be found in [85], delves into a comprehensive examination of several aspects related to probe request messages. Firstly, it seeks to understand the frequency with which wireless devices transmit network probe requests, including their burst patterns and the content they carry. Additionally, the study investigates how user interactions with smartphones or other smart devices influence the device’s behavior when sending probe requests. It is worth noting that prior attempts to address the “talkativeness” of mobile devices, as explored in [40], were conducted in a different technological landscape. Since then, there have been substantial advancements in technology, operating systems, and the redesign of smart devices. Furthermore, these previous analyses were limited to a small sample of smartphones, without consideration for tablets or laptops.

Our work makes significant contributions to the current literature in the following key areas:

- **Comprehensive Analysis of Probe Request Messages:** We provide an in-depth examination of probe request messages behavior, including insights into the frequency of bursts, the number of packets per burst, and the key parameters associated with various types of devices (such as smartphones, tablets, and laptops). We also consider factors like the device’s vendor, operating system version, and user-specific interaction patterns.
- **MAC Address Randomization Analysis:** We conduct a thorough investigation into MAC address randomization, offering insights into how it is implemented and how frequently MAC addresses are randomized. This analysis sheds light on the privacy and security aspects of wireless device communication.
- **Correlation Analysis of Probe Request Parameters:** We explore the correlation between different parameters within a single device’s probe request messages. This aspect of our study helps uncover patterns and dependencies in the data, contributing to a deeper understanding of device behavior and communication patterns.

In the upcoming Sections, we will conduct a literature review analysis to explore the concepts of probe requests and probe responses. Following that, we will delve

into the subject of MAC address randomization. Subsequently, we will provide a comprehensive and detailed description of the methodology employed for analyzing the behavior of probe request messages. Finally, we will present the results obtained from our analysis and we will conclude this Chapter with our final remarks.

2.3 IEEE 802.11 connection procedure

In this Section, we will provide a comprehensive description of the procedure involved in establishing a connection between a WiFi Access Point (AP) and a device. We will focus our attention on the IEEE 802.11 Probe Request and Probe Response frames, conducting a thorough analysis of these frames by describing all the key fields within their headers and payloads.

2.3.1 Interaction between device and access point

When a WiFi-enabled device is not currently connected to a network, it typically employs one of two primary methods to discover available networks:

- **Passive Scanning:** Involves devices continuously listening for beacons sent by Access Points (APs) to identify known networks.
- **Active Scanning:** Involves devices actively seeking out APs by transmitting a specific type of management frames known as probe requests on IEEE 802.11b/g/n channels.

While passive scanning is a valid approach, it is not particularly efficient. Consequently, proactive scanning, which involves active scanning through probe requests, has become the preferred and more efficient method in contemporary WiFi network discovery.

Figure 2.1 illustrates the entire procedure, from network discovery to the association process. In the initial scanning phase, the device can operate in either passive or active scanning mode. In passive mode, it listens for beacons broadcasted by access points to announce themselves. In active mode, the device sends probe request messages either in broadcast or occasionally in unicast. When a probe request is

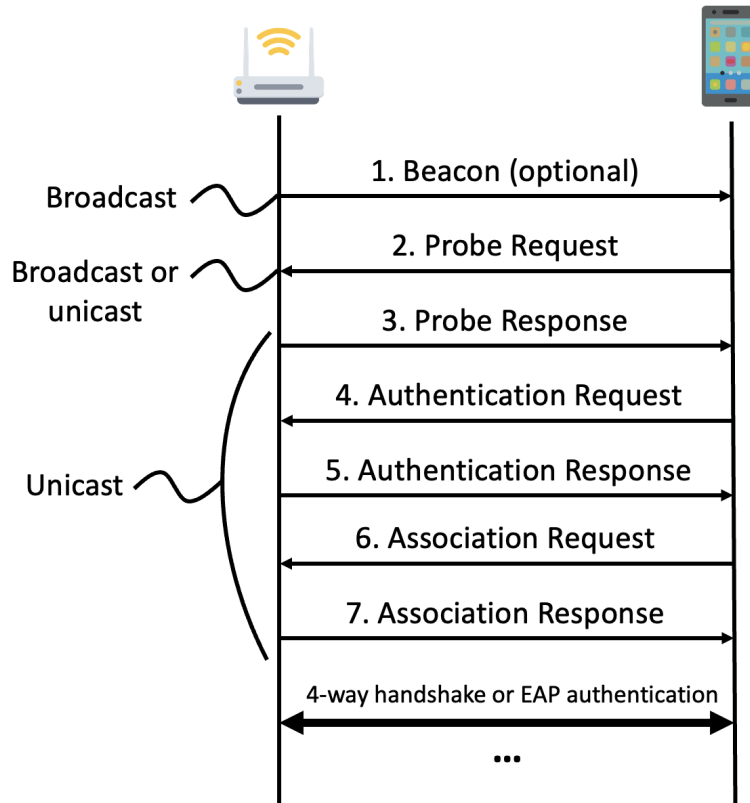


Fig. 2.1 Comprehensive overview of the IEEE 802.11 Network Discovery and Association procedures between an access point and a smart device.

sent, a probe timer is initialized. If no response is received, the device switches to the next channel frequency and repeats the discovery process. However, if a probe response is received, the device initiates the authentication process. This involves exchanging authentication frames with the access point to verify the device’s IEEE 802.11 capabilities and ensure its compatibility with the network. Upon receiving an authentication frame, the access point sends an acknowledgment and then an Authentication Response. If the IEEE 802.11 Authentication phase concludes successfully, indicating a “Success” result, the device proceeds to the Association phase. Here, the device aims to join the network and acquire an Association ID. Once the Association request is acknowledged, the access point carefully examines each field of the request, ensuring they align with its own IEEE 802.11 parameters. If everything matches, the device and the access point commence either the 4-way

handshake or the EAP authentication process, depending on the specific network configuration and security protocols in use.

2.3.2 IEEE 802.11 Probe Request frame

The IEEE 802.11 standard enables smart devices such as smartphones, laptops, tablets, smartwatches, and more to efficiently communicate with nearby access points to expedite the process of connecting to WiFi networks. These communication messages are periodically sent by devices for two primary purposes: to search for known APs to connect to and to seek out APs with stronger signal strength and better performance, even if they are already connected to a WiFi network. These messages are typically organized into groups, referred to as bursts, consisting of one, two, three, or four packets. They are transmitted across a total of 13 channels, which adhere to the IEEE 802.11 standard. These channels allow nearby access points to both receive these messages and respond with probe response frames.

Through the analysis of these probe responses, the device can assess the available networks in its vicinity and determine the optimal access point to connect to, taking into account factors such as signal quality and security. It is important to note that there are scenarios in which the client device may not receive any response, such as when there are no APs nearby or when the APs are in passive mode. For reference, Figure 2.2 provides an example of a probe request frame, according to the IEEE 802.11ac standard [42].

A list of the main fields that can be found in the probe request frames is the following:

- **Frame Control:** It is the initial element of the header and contains information such as the protocol version used, the type and sub-type of the message (i.e., type 0 and sub-type 4 refer to probe request frames).

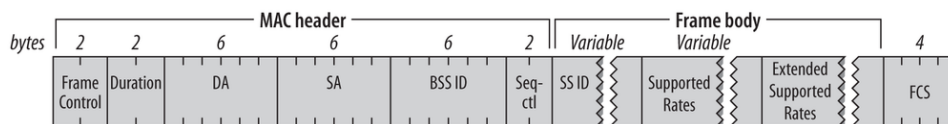


Fig. 2.2 Architecture of the IEEE 802.11 Probe Request frame. Reproduced from [41].

- **Duration:** It indicates how long the frame and its acknowledgment frame are going to occupy the channel. Due to the fact that the AP does not respond to the sender with a dedicated acknowledgment frame but rather with a separate message (i.e., the probe response frame), the length of the duration value in a probe request is always zero.
- **MAC (Media Access Control) Address:** This address serves as a unique identifier assigned to a Network Interface Controller (NIC) and consists of six bytes. It distinguishes a device on the network and is assigned by the vendor, ensuring global uniqueness. In the probe request frame, we can find the Destination Address (DA) and the Source Address (SA). The Destination Address typically takes the form of a broadcast MAC address (i.e., FF:FF:FF:FF:FF:FF) since probe requests are commonly broadcasted. On the other hand, the source address is the MAC address of the WiFi card in the smart device responsible for transmitting the probe request.
- **Sequence Number (Seq-ctl):** Each packet sent over a network connection is assigned a unique sequence number. This value, present in the packet header, facilitates proper packet ordering and reconstruction at the receiving end. Probe request messages use 12-bit sequence numbers, allowing for values ranging from 0 to 4095.
- **SSID (Service Set Identifier):** An alphanumeric string that functions as the name of a wireless network. SSIDs are used to identify and differentiate wireless networks within an area. When devices scan for available networks, they can detect and display the SSIDs of nearby networks. Clients announce their knowledge of specific SSIDs to attempt connections to particular wireless networks.
- **VHT (Very High Throughput) Capabilities:** This field pertains to the capabilities of a WiFi device that supports the IEEE 802.11ac standard, commonly referred to as WiFi 5. It conveys information about the device's support for various features, including supported channel widths, spatial streams, and modulation schemes. These capabilities enable higher data rates and improved throughput in wireless communications. Throughput, in this context, refers to the quantity of data effectively transmitted across a network within a specific time frame.

- **HT (High Throughput) Capabilities:** These capabilities relate to WiFi devices that adhere to the IEEE 802.11n standard, often referred to as WiFi 4. This field provides information about channel bonding, MIMO (Multiple-Input Multiple-Output) configurations, and supported data rates. Such capabilities enhance data transmission speed and overall network performance.
- **Extended Capabilities:** In the context of a WiFi probe request, this field encompasses additional features and capabilities supported by the WiFi device, surpassing the basic functionalities outlined in the VHT and HT fields. Extended capabilities offer insights into advanced features, such as beamforming, spatial reuse, or proprietary extensions specific to particular vendors. They allow devices to communicate their enhanced functionalities to others within the network.
- **WPS (WiFi Protected Setup):** A network security standard designed to simplify the process of connecting devices to a WiFi network securely. WPS enables users to establish a secure connection via methods like PIN entry, push-button configuration, or NFC (Near Field Communication). The WPS field in a probe request message indicates whether the device supports WiFi Protected Setup and provides information about the supported methods for establishing a secure connection.
- **UUID-E (Universally Unique Identifier-Extended):** This field serves to distinguish between different devices. It contains a unique identifier allocated to the device, facilitating its recognition and differentiation from other devices by fellow devices or network systems. The UUID-E may be generated by the device itself or assigned by a network administrator.
- **FCS (Frame Check Sequence):** This field functions as a checksum, verifying that the preceding part of the frame has been received accurately on the receiver's end.

2.3.3 IEEE 802.11 Probe Response frame

When a compatible network is discovered by a probe request, it responds with a probe response frame. This frame carries all the essential parameters found in a Beacon frame, allowing smart devices to match these parameters and initiate the

WiFi Probe Request for Crowd Monitoring

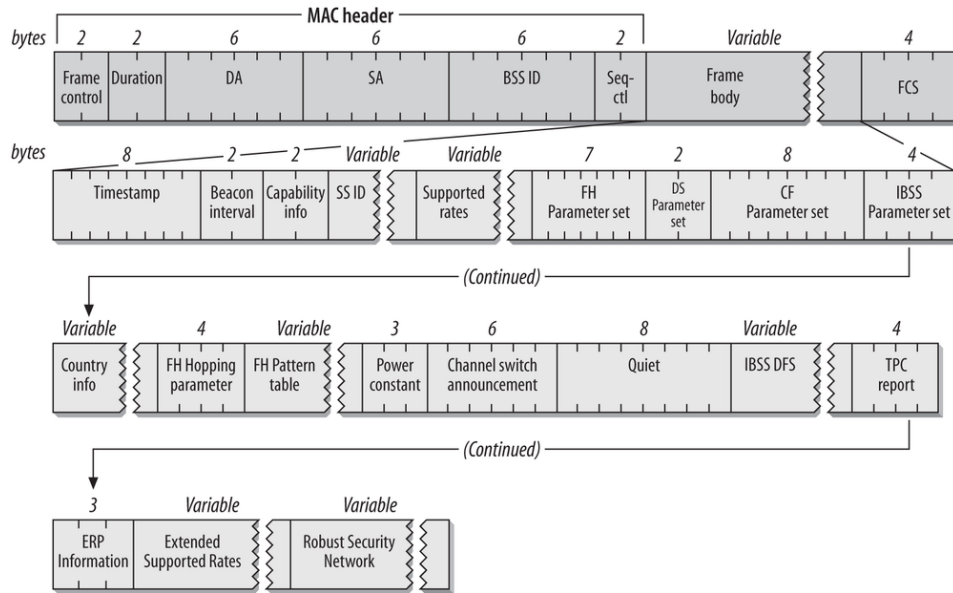


Fig. 2.3 Architecture of the IEEE 802.11 Probe Response frame. Reproduced from [41].

authentication process on the wireless network. In more comprehensive terms, the probe response frame contains a wealth of information about the network. This information encompasses details like the network’s SSID, security protocols (such as WPA2 or WPA3), encryption settings, channel specifics, supported data rates, and other pertinent information. All of these details are provided to the requesting client.

It is important to note that probe response messages are always sent in unicast, addressing the specific client device that initiated the probe request. Figure 2.3 provides an example of a probe response frame, according to the IEEE 802.11ac standard [42].

2.4 MAC address and privacy

In this Section, we will delve into the significance of MAC (Media Access Control) addresses, their importance in networking, and why major smart device vendors have begun implementing techniques to protect user privacy associated with these addresses.

2.4.1 MAC address

A MAC address is a unique identifier assigned to every Network Interface Card (NIC) or network adapter in a computing device. It is a 48-bit (6-byte) value that serves as a hardware-level identifier for devices within a local network. MAC addresses are typically assigned by the manufacturer and remain constant throughout the device's lifetime.

As depicted in Figure 2.4, the MAC address can be visually dissected into two distinct sections. The initial portion houses the Organizationally Unique Identifier (OUI) [45], while the latter part houses the NIC specifier. This division serves to uniquely identify the manufacturer or organization responsible for the device's network interface, followed by a specific identifier for the device itself.

Moreover, MAC addresses are divided into two categories based on the value of the second-least-significant bit of the first byte in the address, known as the *U/L* (Universally/Locally Administered) bit or *GA/LA* (Globally Assigned/Locally

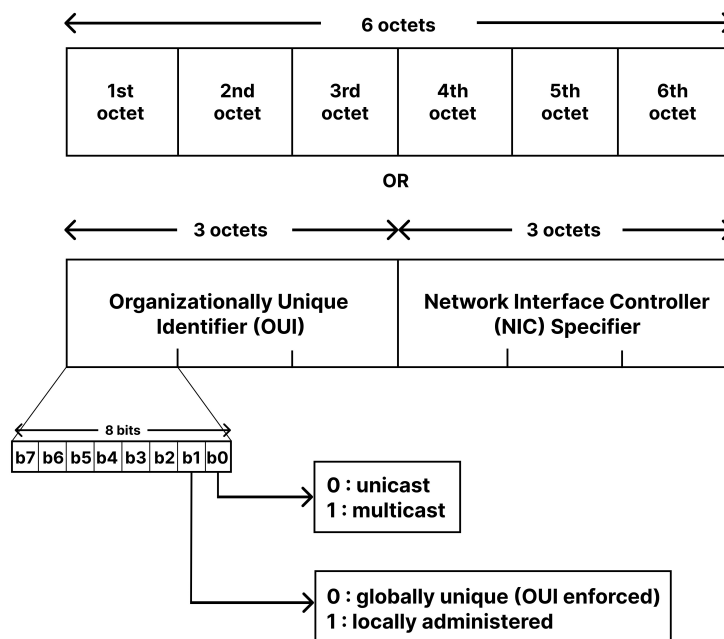


Fig. 2.4 Architecture of 48-bit MAC address with highlighted the local/global and unicast/multicast bits.

Assigned) bit. This bit plays a crucial role in determining the uniqueness and origin of MAC addresses:

- **Globally Administered (GA) Addresses:** MAC addresses with the U/L bit set to 0 are considered globally administered. These addresses are assigned by the IEEE (Institute of Electrical and Electronics Engineers) to device manufacturers. GA addresses are meant to be globally unique and ensure that two devices from different manufacturers have not the same MAC address. The first three bytes of a GA MAC address are the OUI assigned to the manufacturer.
- **Locally Administered (LAA) Addresses:** MAC addresses with the U/L bit set to 1 are locally administered. These addresses can be configured by the device owner or administrator. LAA addresses are typically used when privacy is a concern or when users want to assign their own unique MAC addresses to their devices. They are not guaranteed to be globally unique.

MAC addresses play several critical roles in networking:

- **Device Identification:** MAC addresses are used to uniquely identify devices on a local network. This is essential for routing data packets to the correct destination.
- **Address Resolution:** Devices use the Address Resolution Protocol (ARP) to map IP addresses to MAC addresses, enabling efficient data transmission within a network.
- **Network Security:** MAC filtering is a security measure used in some networks to control device access based on MAC addresses.

While MAC addresses are essential for network functionality, they can raise privacy concerns, particularly in the context of smart devices, because they can be used for tracking purposes, profiling and location inference. Moreover, the GDPR (General Data Protection Regulation) [39] in May 2018 through the document in [10] stated that the MAC addresses are classified as a personal data and for this reason, they must be subject to privacy protection mechanism.

2.4.2 MAC address randomization

Major smart device vendors have indeed recognized the privacy concerns associated with MAC addresses and have taken steps to address these issues while still ensuring network functionality. One prominent technique employed to enhance privacy is known as *MAC address randomization*. This technique leverages specific bits in the MAC address to create randomized addresses, making it more difficult for external parties to track and identify users.

There are two primary modalities of MAC address randomization, each focusing on different aspects of the address:

- **Complete Randomization:** In the case of complete randomization, all six octets (48 bits) of the MAC address are randomized. This means that the entire MAC address, including the Organizationally Unique Identifier (OUI) portion, is replaced with a randomly generated value for each network connection. As a result, the new MAC address bears no resemblance to the device's original hardware address. Complete randomization provides the highest level of privacy because it completely severs the link between the device and its unique identifier.
- **Partial Randomization:** In this modality, the original OUI of the MAC address may be retained, or it can be substituted with a Company Identifier (CID). However, the remaining three octets of the address are randomized for each network connection. This approach balances privacy with some level of network stability. Retaining the OUI or CID allows networks and services to recognize the device manufacturer, which can be useful for network management and device compatibility while still providing a degree of privacy by randomizing the least significant part of the address.

The algorithms used for MAC address randomization are typically proprietary and specific to each device manufacturer. Notably, Apple led the way in implementing MAC address randomization in 2014 for smartphones running iOS 8, setting a significant precedent in the industry [56]. Following Apple's initiative, Linux introduced MAC address randomization in the same year with Linux Kernel 3.18, and Android followed suit the subsequent year with the release of Android 6.0 [57]. Despite the widespread support for MAC address randomization across various operating systems, it is important to note that many devices, particularly those using the

Android operating system, do not have MAC randomization enabled by default [65]. This means that users often need to manually enable this feature if they wish to take advantage of the enhanced privacy protections it offers.

In the remaining part of this Section, we will conduct an analysis of the primary Operating Systems (OS) with regards to the implementation of MAC address randomization.

Android implementation

Starting with Android 6.0, the option for MAC address randomization was introduced [57]. However, this feature was not enabled by default and was left to the discretion of manufacturers to implement. With the release of Android 8.0, Android devices began using randomized MAC addresses for probing new networks when not connected to any network. In Android 9, a developer option (disabled by default) allowed users to enable randomized MAC addresses when connecting to WiFi networks. From Android 10 onwards, MAC randomization is enabled by default in client mode [70]. It is important to note that the implementation of MAC address randomization on Android devices can vary depending on the manufacturer, despite Android's open-source nature.

iOS implementation

Apple adopted random MAC addresses starting with iOS 8, and it is now the default behavior for all devices, although users have the option to deactivate it [57]. In iOS 14 and later, devices use a unique random MAC address per network when connecting to WiFi. Users have control over this feature through device settings, and under certain conditions, the device may revert to its actual MAC address [46].

Windows implementation

MAC address randomization was introduced with Windows 10, contingent upon hardware and driver support. Similar to Android and iOS, Windows assigns a distinct random address to each network it connects to, following a specific algorithm outlined in [57].

Linux implementation

Linux introduced support for MAC address randomization during network scans in kernel version 3.18. During each scan iteration, the MAC address is randomized. On the software side, randomization is facilitated by `wpa_supplicant`, a widely-used WiFi software on Linux, with support introduced in 2015 (v2.4).

While MAC address randomization techniques are beneficial for mitigating the transmission of potentially sensitive information over the channel, it is important to note that they do not provide complete elimination of this risk. Numerous flaws and vulnerabilities in these implementations have been demonstrated in [101, 56]. These weaknesses can involve exploiting other elements present in the probe request frame or the timing of these frames, thereby enabling device tracking even when MAC addresses are randomized. In response to the development of MAC address randomization algorithms, countermeasures in the form of de-randomization processes have been devised to negate the effects of randomization. Chapter 4 will provide an overview of various de-randomization techniques documented in the literature. Additionally, it will introduce our own approach, which utilizes a Machine Learning (ML) algorithm to effectively address the challenges posed by randomization.

2.5 Methodology

In this Section, our focus is on the hardware and methodology employed for conducting the experiments and the subsequent data analysis. In detail, Section 2.5.1 outlines the hardware used for the experiments, including the sniffer and the devices under test. Section 2.5.2 provides an introduction to the location where the experiments were conducted. Section 2.5.3 elaborates on the steps followed for each device to ensure consistent and replicable results across all devices. In Section 2.5.4, we delve into the analysis process, describing how the captured files were analyzed, the applied filters, and the key parameters of the probe request messages that received our primary focus. Lastly, in Section 2.5.5 we describe an extension of our experiments, pointing out the newer test location and the newer set of devices under analysis.

2.5.1 Hardware description

In this Section we present an overview of the hardware utilized, which includes the sniffer and the devices under test.

Sniffer device

The tests were conducted using a custom ad-hoc sniffer, purpose-built by the Dropper company [6], using off-the-shelf components. This sniffer, showed in Figure 2.5 consists of a Raspberry Pi model Zero 2 (Figure 2.5a) integrated with an external wireless interface (Atheros AR927 chipset) connected via a micro USB cable. Additionally, it includes a cellular connection interface (SIM7000E NB-IoT HAT) for SSH communication with a control center laptop. The fully assembled sensor is illustrated in Figure 2.5b. The Raspberry Pi runs Raspbian GNU/Linux 11 (bullseye) as its operating system, and the capture software used is TShark version 3.4.10, which captures data packets from a live network, applies capture filters (e.g., for probe requests), and stores data in *.pcap* files.

Figure 2.6 shows the wireless network interface of the sensor. Unfortunately, we are unable to use the built-in WiFi interface of the Raspberry Pi since it does not support monitor mode. Consequently, the wlan0 interface (the Raspberry's integrated wireless interface) is disabled for our purposes. Instead, we have configured the wlan1 interface (the external wireless interface) to operate in monitor mode, and it is responsible for packet capture. Notably, the monitor mode interface is fixed to a

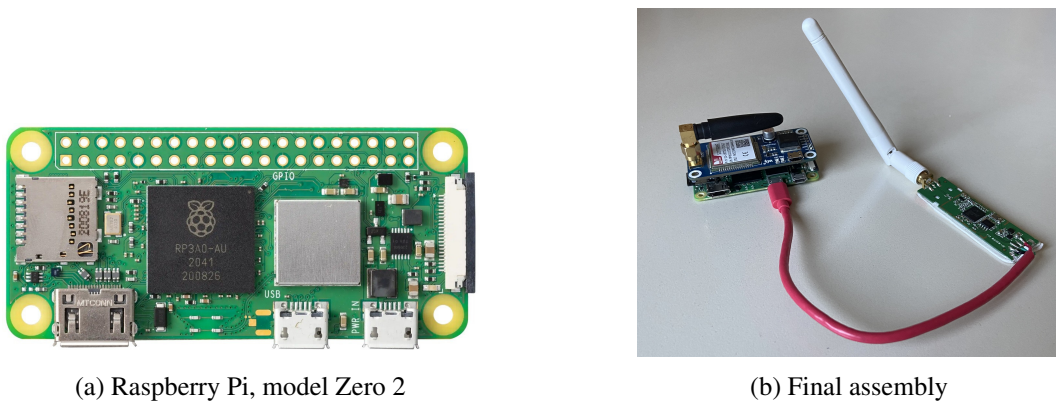


Fig. 2.5 Sniffer sensor for capturing 802.11 Probe Request messages.

```
pi@raspberrypi:~ $ iwconfig
lo          no wireless extensions.

wlan0      IEEE 802.11  ESSID:off/any
           Mode:Managed  Access Point: Not-Associated  Tx-Power=31 dBm
           Retry short limit:7  RTS thr:off  Fragment thr:off
           Power Management:on

wlan1      IEEE 802.11  Mode:Monitor  Frequency:2.412 GHz  Tx-Power=20 dBm
           Retry short limit:7  RTS thr:off  Fragment thr:off
           Power Management:off

ppp0       no wireless extensions.

tun0       no wireless extensions.
```

Fig. 2.6 Sniffer wireless network interface configuration.

single 2.4 GHz band channel, specifically channel 1 (2.412 GHz). This choice of a single channel was made due to the availability of antennas and sensors.

Tested devices

The selection of devices under test was meticulously carried out, taking into account various factors to ensure maximum diversity in terms of manufacturer, OS version, release year, and representation of typical devices used today. Table 2.1 provides an overview of each device, including information on the vendor, model, release year, and the OS version it was running. The inclusion of both tablets and laptops was driven by the need to discern potential differences in probe request transmission and randomization algorithms across these device types.

Our objective was to create a comprehensive understanding of MAC address randomization, encompassing as many scenarios as possible and considering potential implementation changes over the years. Furthermore, this study aimed to shed light on the intricate relationship between hardware and MAC address randomization, emphasizing how randomization patterns could evolve with changes in hardware configurations.

It is worth noting that, for completeness, the *private Wi-Fi address* option was enabled on the iPhone devices, as per the default settings.

Type	Vendor	Model	OS	Year
SmartPhone	OnePlus	Nord 5G	Android 11.0	2021
SmartPhone	Samsung	Note 20 Ultra	Android 12.0	2020
SmartPhone	Apple	iPhone 11	iOS 15.0.1	2019
SmartPhone	Xiaomi	Redmi Note 8T	Android 10.0	2019
SmartPhone	Huawei	P9 Lite	Android 7.0	2016
SmartPhone	Apple	iPhone 6	iOS 12.5.5	2014
Tablet	Apple	iPad 8	iPadOS 14.8.1	2020
Laptop	Lenovo	ThinkPad X13 Gen1	Windows 11	2021
Laptop	Apple	MacBookAir M1	macOS 12.1	2020

Table 2.1 Device list, tested in 2022.

2.5.2 Testing location

The ideal setting for conducting these tests would have been an anechoic chamber. However, due to the high costs associated with building and maintaining such facilities, not all universities have access to them. Consequently, we decided to carry out our experiments in a remote area, away from any potential interference.

Our chosen location for the experiments was La Mandria Regional Park, situated on the outskirts of Turin, Italy, surrounding the famous Royal Palace of Venaria Reale. It is the perfect spot to execute tests in a controlled environment, thanks to its vast open spaces, and the low turnout of people. The timing of the experiments, on a cold morning in early March 2022, further reduced the likelihood of encountering other individuals in the area.

Figure 2.7 provides two perspectives of the selected location within the park. In the area there are 3 gazebos, used respectively as (1) control center, (2) location of a



Fig. 2.7 Location of the test, performed in March 2022.

device creating a hotspot for some test, and (3) location of the sniffer. The distance between (1) and (3) is about 90m, while gazebo (2) is located halfway between the other two.

2.5.3 Testing methodology

At the beginning of the experiment, we conducted two capture sessions: the first with all the tested devices running, serving as a check to confirm that both the sniffer and TShark were functioning correctly; the second one, lasting for 10 minutes, took place after turning off all the devices. Its purpose was to verify that the wireless channel remained free of any transmissions from nearby devices, ensuring no interference with the data we intended to capture. As anticipated, the remote location guaranteed a lack of interference.

For the TShark command, we used the configuration specified in Listing 2.1. This setup included setting the capture interface to `wlan1`, applying a capture filter to isolate probe requests, specifying the output format as `.pcap`, and designating the output file name.

```
sudo tshark -i wlan1
           -f 'subtype probe-req'
           -w filename.pcap -F pcap -V
```

Listing 2.1 tshark command

WiFi Probe Request for Crowd Monitoring

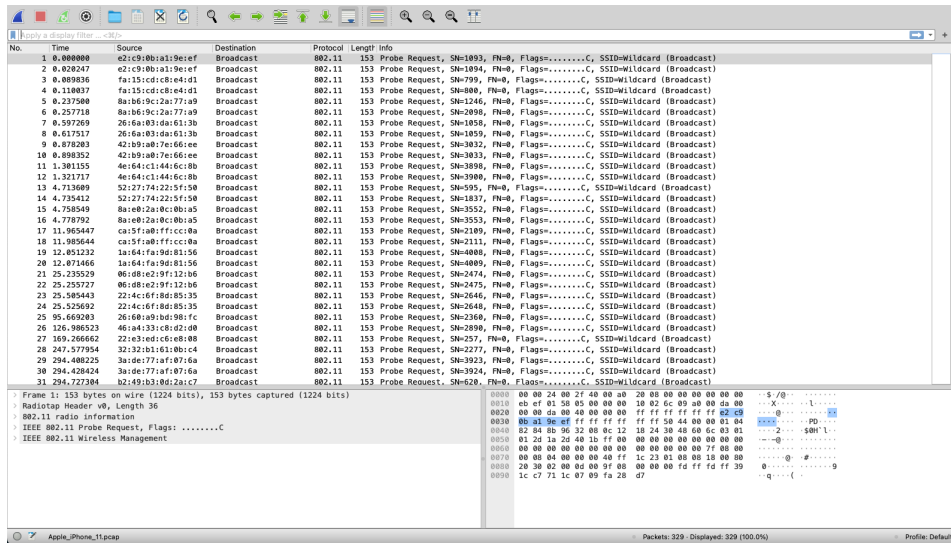


Fig. 2.8 Example of TShark output file (.pcap) on Wireshark.

An illustrative instance of TShark’s output is depicted in Figure 2.8. By employing Wireshark [14], we can observe the packets within the generated *.pcap* file and effectively explore their contents.

Each device experiment is subdivided into four distinct phases: Locked, Awake, Active, and Changing AP. These experiments are conducted in two primary scenarios: one where the device is not connected to any access point (AP), and the other where it is connected to an AP created through a third smartphone, with an active hotspot and the device’s WiFi interface disabled. The “Changing AP” phase represents the period between the device’s disconnection from the first AP, referred to as ‘OPEN’, and its subsequent connection to a second AP, known as ‘OPEN2’. Here is a detailed description of each phase:

- **Locked:** In this phase, the device is locked with WiFi enabled. This phase typically lasts between 3 to 8 minutes, depending on the time required to detect a minimum number of probe request messages.
- **Awake:** During the Awake phase, the device’s screen is tapped at intervals of approximately 30 seconds, with WiFi enabled.
- **Active:** In the Active phase, the device is unlocked, and the user actively interacts with various apps and settings on the device.

- **Changing AP:** This phase occurs when the device is disconnected from the initial AP and subsequently connected to a new one.

Tablets and PCs were tested under active usage conditions in both scenarios (i.e., connected and not connected to the AP).

2.5.4 Data analysis

We utilized the PyShark [48] Python library to process the captured data and generate the results presented later in Section 2.6. PyShark serves as a wrapper for TShark, enabling Python to parse packets using Wireshark’s built-in dissectors.

Our primary motivation was to develop a method for distinguishing a specific device among others within the captured data. Inside the probe request messages, certain fields can be used to uniquely identify a device, effectively creating a kind of fingerprint. Notably, fields within the frame body, such as SSID, HT capabilities, and extended capabilities, lend themselves to device identification. The SSID, in particular, can be either a wildcard that allows the device to connect to any available nearby access point (AP) or a specific network name. Additionally, we examined the length of probe requests, specifically in terms of management frames within IEEE 802.11, which includes only relevant fields while excluding the IEEE 802.11 MAC header fields. The core of our analysis began with packet filtering to isolate the traffic relevant to controlling the sniffer’s behavior. We also applied the filter in Listing 2.2, that is based on a minimum Received Signal Strength Indicator (RSSI) to further ensure that we captured data only from devices within the experiment’s capture range.

```
wlan_radio.signal_dbm > -70
```

Listing 2.2 Wireshark filter

Subsequently, our attention shifted to parsing each field within the packets and constructing a data structure that facilitated the understanding of various metrics, such as burst length, burst inter-arrival time, message length, and RSSI. Lastly, we focused on examining additional parameters that appeared to characterize specific devices, referred to as Information Elements (IEs), a concept also considered by the authors of [63]. These parameters included HT capabilities, the desired AP’s SSID for device connection, extended capabilities, and the vendor ID.

This approach allowed us to comprehensively analyze the probe request data and extract valuable insights for device identification.

2.5.5 Experiment extension

To expand upon the investigations carried out at La Mandria Regional Park in 2022, we replicated the experiments employing a mix of similar and diverse devices. The primary objective was to widen the range of tested devices, thereby enriching the dataset and fine-tuning the performance and capabilities of the probe request generator discussed in Section 3.4.

In line with the considerations outlined in Section 2.5.2, our decision to conduct these extended tests brought us to Parco della Pellerina, a large park in Turin. The testing took place once more on a crisp morning in early March 2023. Figure 2.9 provides a visual representation of the specific area within Parco della Pellerina where the sniffer was deployed, and various experiments were undertaken.

Throughout this experimental series, the devices subjected to testing are detailed in Table 2.2. Each device is accompanied by a thorough overview, including essential



Fig. 2.9 Location of the test, performed in March 2023.

Type	Vendor	Model	OS	Year
SmartPhone	Apple	iPhone 14 Pro	iOS 16.4	2022
SmartPhone	Apple	iPhone 13 Pro	iOS 16.3	2021
SmartPhone	Xiaomi	Mi9 Lite	Android 10.0	2020
SmartPhone	Apple	iPhone 11	iOS 16.3.1	2019
SmartPhone	Apple	iPhone 7	iOS 15.2	2016
Laptop	Apple	MacBookPro M1	macOS 11.6.2	2015

Table 2.2 Device list, tested in 2023.

information such as the vendor, model, release year, and the operating system version utilized during the testing phase.

In contrast to the first testing cycle, the second set of experiments focused exclusively on conducting tests when the devices were not connected to an access point. Our emphasis was on the connection phase of the devices, a critical and challenging aspect. This is particularly noteworthy because during this phase, devices tend to consistently randomize their MAC addresses instead of maintaining a fixed address as observed when they are associated with a WiFi network.

The testing methodology and data analysis procedures applied in this second cycle remained consistent with those detailed in Sections 2.5.3 and 2.5.4.

2.6 Results

In this Section, we present the outcomes of our on-field experiments, offering a comprehensive analysis of the data collected. Section 2.6.1 showcases the results through a series of informative graphs, providing insights into the analyzed probe requests emitted by the various devices, as detailed in Table 2.1. For a deeper understanding of the diverse characteristics and features, Section 2.6.2 offers an overview, comparing and contrasting the tested devices. Section 2.6.3 provides a

comprehensive comparison between the outcomes derived from the tests conducted in 2023 and those conducted in 2022. Finally, in Sections 2.6.4 and 2.6.5, we provide the key findings and takeaways derived from our experiments, and the main limitations.

Prior to proceeding further, it is essential to note that all the raw data and graphical representations for every device under consideration have been made publicly accessible through a GitHub repository [81].

2.6.1 Experimental results

The *.pcap* files for each device were subjected to comprehensive analysis, resulting in the generation of three distinct types of graphs, akin to those presented in Figures 2.10, 2.11, and 2.12. These figures respectively illustrate the analysis of received packets, vendor-related data with burst analysis, as well as power-related insights.

Figures 2.10 and 2.11 provide a detailed portrayal of the outcomes derived from an analysis of probe requests captured from the Apple iPhone 11 at La Mandria Regional Park in 2022. Each histogram within these figures corresponds to a 30-second observation window. The graphs employ color-coded bands to demarcate distinct test phases. Specifically, the yellow band denotes the Locked phase, the green band signifies the Awake phase, and the purple band designates the Active phase. Additionally, the pink band marks the instance when the device transitions to a different access point (AP). A dotted vertical green line serves as a clear division, indicating whether the device is connected to an AP (on the right) or not (on the left).

Analysis on received packets

In Figure 2.10, there are six distinct plots. Starting from the top, the first plot depicts the number of packets received during various time intervals. As expected, the Active phase displays a higher packet count than both the Locked and Awake phases. Even when the device is connected to an AP, it continues to send probe requests, which suggests that it persistently seeks a potentially better connection from another AP, but as we can see the frequency of sending probe request decreases a lot. Furthermore, it is worth highlighting the differences in probe request behavior during the various

WiFi Probe Request for Crowd Monitoring

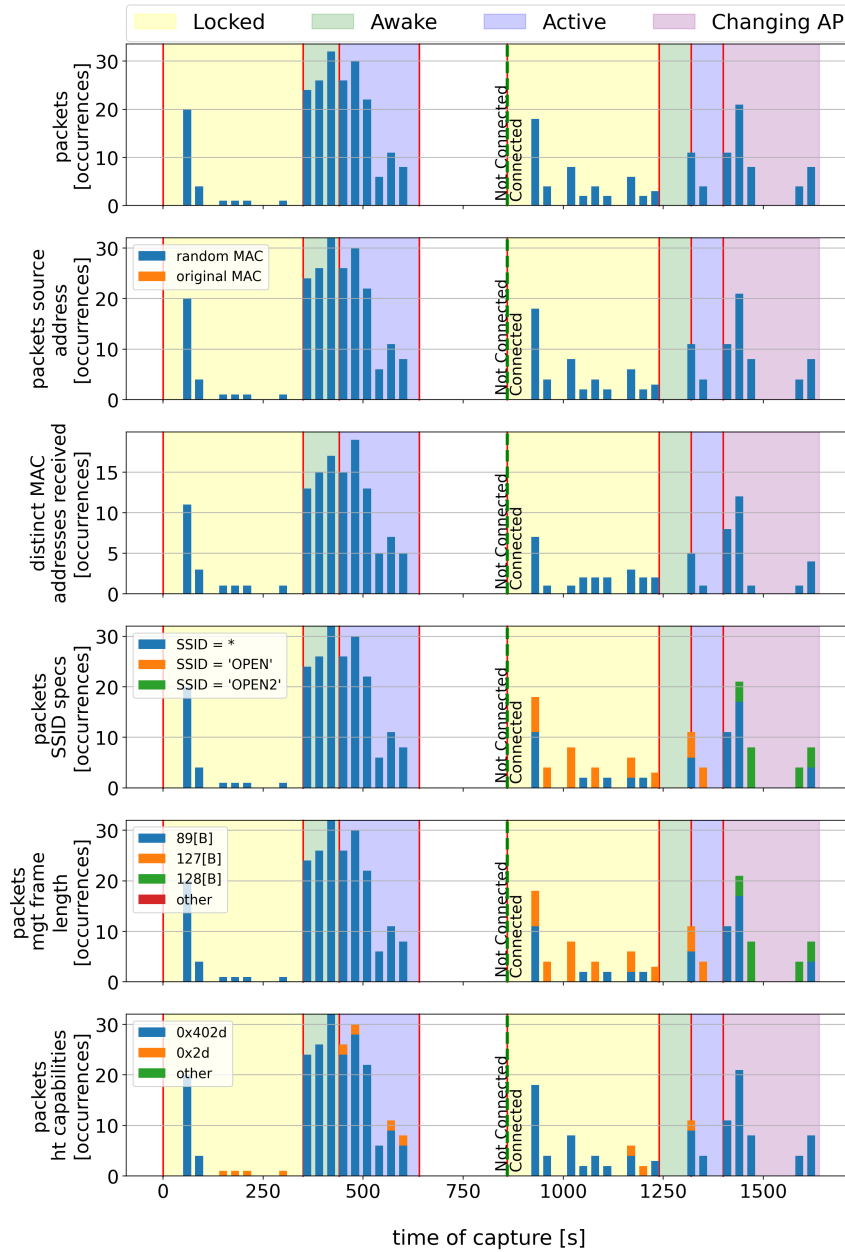


Fig. 2.10 Comprehensive analysis of probe request packets sent from an Apple iPhone 11 for different usage phases and for different connection states.

phases when the device is not connected to an access point. In the Locked phase, probe requests are initially sent and then stop. However, during the Awake and Active phases, probe requests occur more frequently, even though these phases are shorter in duration compared to the initial Locked phase. This indicates increased activity in sending probe requests during the later phases, possibly driven by the device's search for network access or performance enhancement.

The second plot showcases occurrences of original and randomly generated source MAC addresses, offering insights into MAC address randomization behavior. Remarkably, the iPhone 11 consistently employs randomized MAC addresses, even while connected to an AP.

The third plot illustrates the overall number of distinct MAC addresses observed throughout the entire experiment duration. This graph bears a connection to the last graph in Figure 2.11, indicating that MAC address changes occur with each new burst of packets. For example, when not connected to an AP, 117 distinct MAC addresses are observed out of a total of 213 packets, corresponding to approximately half of the packets. This reflects that MAC addresses change every two packets, in alignment with the subsequent revelation that packets are transmitted in bursts of two, with each burst employing a unique randomized MAC address. Interestingly, when connected to the AP, the number of distinct MAC addresses within each burst diminishes.

The fourth plot demonstrates occurrences of each access point SSID as observed in the probe requests. Three distinct values are evident: the wildcard SSID and the two preset SSIDs. Following the device's connection to an AP, it tends to actively search for the specific AP to which it is connected and includes the SSID in plaintext within the probe request. However, when unconnected, it frequently employs the wildcard value.

The fifth plot portrays the length, in bytes, of probe request messages over time. Notably, when the phone connects to an AP, the probe request frame length increases as it now includes the SSID and additional tags for vendor specification.

The final diagram in Figure 2.10 illustrates the frequency of HT (High Throughput) capabilities values in the transmitted packets. It is evident that the value "0x402d" predominates most of the time, even when the device is connected to the AP.

Analysis on vendor-related data and burst characterization

Figure 2.11 delves deeper into additional insights gleaned from the analysis. The first plot mirrors the one found in Figure 2.10 and serves as a temporal reference. The second plot, however, focuses on packet occurrences where the vendor ID (OUI=00:17:F2) is explicitly specified. Notably, it illustrates that the vendor ID starts to emerge after the device has connected to an access point, even when the device's MAC address continues to be randomized. Indeed, the OUI 00:17:F2 is associated with Apple Inc. as per the OUI (Organizationally Unique Identifier) list [45]. This signifies that when the device establishes a connection with an access point, it maintains the first half of its MAC address fixed with one of the OUIs owned by Apple, while continuing to randomize the second half of the MAC address. This behavior reflects Apple's strategy of preserving a consistent organizational identifier while introducing variability to enhance user privacy and security.

The last three plots provide further details:

- **inter-packet time in burst:** This represents the time between packets within the same burst. When the device is not connected to an access point, the inter-packet time is consistently 20ms.
- **burst length:** This indicates the duration of a burst in milliseconds.
- **packets per burst:** It measures the burst's length in terms of the number of packets it contains.

From these observations, it becomes apparent that when a device is connected to an access point, probe request bursts tend to be of longer duration. Upon scrutinizing Wireshark [14] traces, it was discovered that this lengthening is due to the inclusion of additional tags containing information about the vendor ID.

Prior to connecting to an access point, all bursts consist of 1 or 2 packets, each with a distinct MAC address. This implies that MAC randomization operates at the burst level rather than at the packet level.

Additionally, a study of the temporal progression of sequence numbers revealed that they increment by one or a few units within the same burst (with the same random MAC address). In contrast, the initial sequence number for a new burst is chosen randomly or follows an unpredictable pattern. This observation underscores

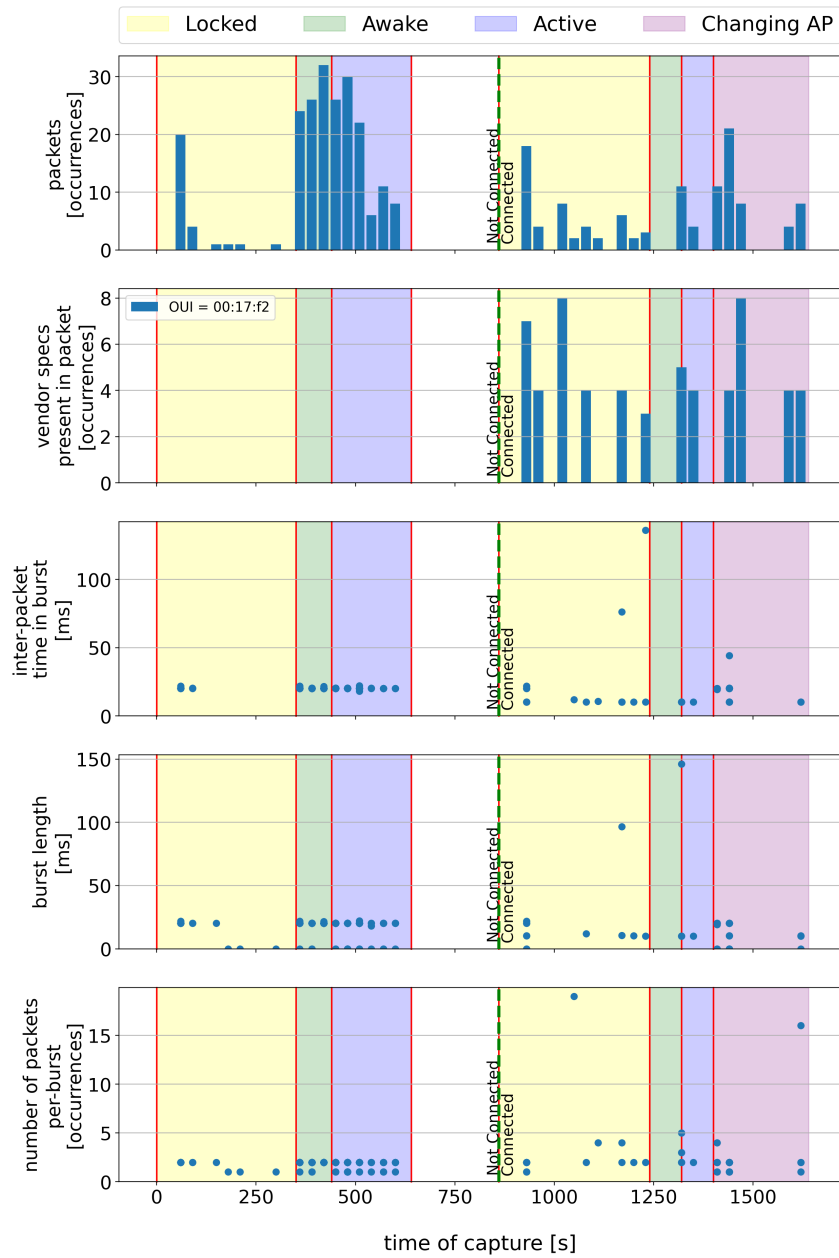


Fig. 2.11 Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 11, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states.

the deliberate countermeasures adopted by iPhone devices to mitigate fingerprinting attempts.

Analysis on RSSI power field

Regarding the power at which packets are received by the sniffer, it can be concluded that there is no distinct correlation with the different phases of the test. As illustrated in Figure 2.12, the Received Signal Strength Indicator (RSSI) fluctuates over time. However, these variations are primarily a result of minor movements made by the experimenter when the phone is in an active state. During the Locked phase, the device remains stationary in the same location. In contrast, during the Active phase, the user interacts with the device's screen using their fingers, which introduces changes in the propagation conditions. Based on our experimental findings, it is evident that RSSI values are highly erratic, even when the smartphone is positioned in a fixed location. Therefore, RSSI alone cannot reliably serve as a unique identifier for a device or determine the specific phase it is in.

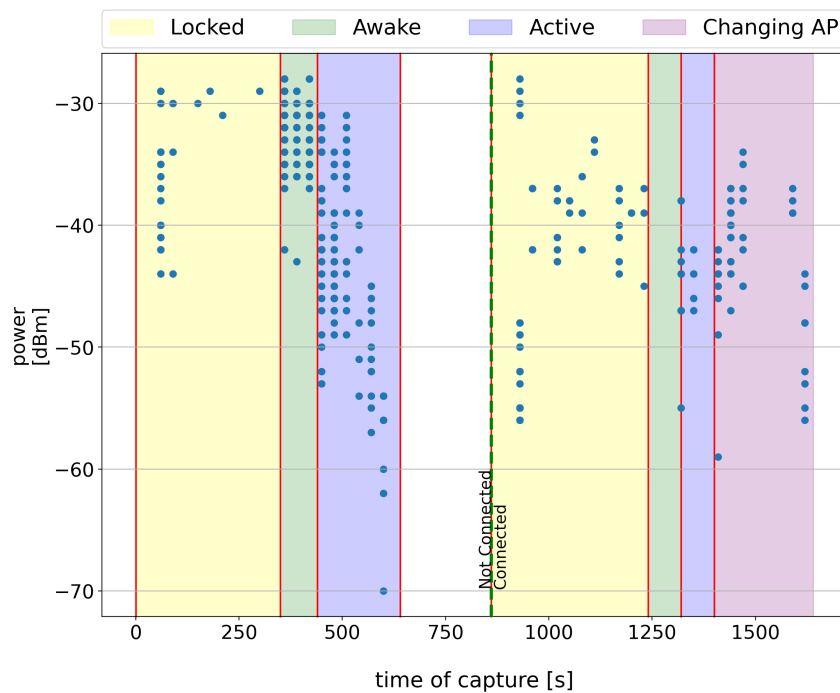


Fig. 2.12 Comprehensive analysis of RSSI power field for an Apple iPhone 11 as a function of time, for different usage phases and for different connection states.

2.6.2 Analysis and comparison between different devices

Throughout our analysis, we also made efforts to compare the behaviors of different devices, whether they belong to the same family or category (e.g., smartphones, tablets, or laptops).

Figures 2.13, 2.14, and 2.15 present the results obtained with the Apple iPhone 6. It is worth noting that, in contrast to the captured data from the Apple iPhone 11, we intentionally left the WiFi interface enabled during the configuration of the hotspot for the second part of the experiment. This allowed us to observe the effects at the boundaries during the connection to an AP.

It is intriguing to observe how the behavior of devices from the same manufacturer has evolved over the years. A striking example of this evolution can be seen when comparing the Apple iPhone 6 (released in 2014) and the Apple iPhone 11 (released in 2019). Notable changes are evident in the second graph of Figure 2.13, where occurrences of the original MAC address can be found. This suggests that in 2014, there was no randomization for Apple devices when they were connected to access points (APs); randomization was active only when they were not connected. This same behavior was also observed for Android 7.0, specifically when analyzing the Huawei P9 Lite. Furthermore, the number of distinct MAC addresses aligns with the results discussed in the previous Section. In terms of the SSID field and frame length, the behavior remains consistent with the previous analysis. However, a more substantial difference emerges in the last graph of Figure 2.13, specifically related to the HT (High Throughput) capabilities. In this case, the HT capabilities appear to remain constant throughout the experiment. This observation may be connected to the fact that newer generations of iPhones possess more complex wireless interfaces, enabling them to establish a broader range of connection types, while older generations lack this capability. Consequently, older devices may continue to send probes with the same HT capabilities value due to their more limited wireless capabilities.

Regarding the burst analysis of the iPhone 6, we observe that there are not significant differences compared to the iPhone 11. Figure 2.14 illustrates this similarity, with only a few traits that distinguish it from the previous device. Notably, the vendor-specific data appears more frequently in the iPhone 6 analysis compared to the iPhone 11. Additionally, the time duration of the bursts and inter-packet times appear to be higher for the iPhone 6 in comparison to its newer counterpart.

WiFi Probe Request for Crowd Monitoring

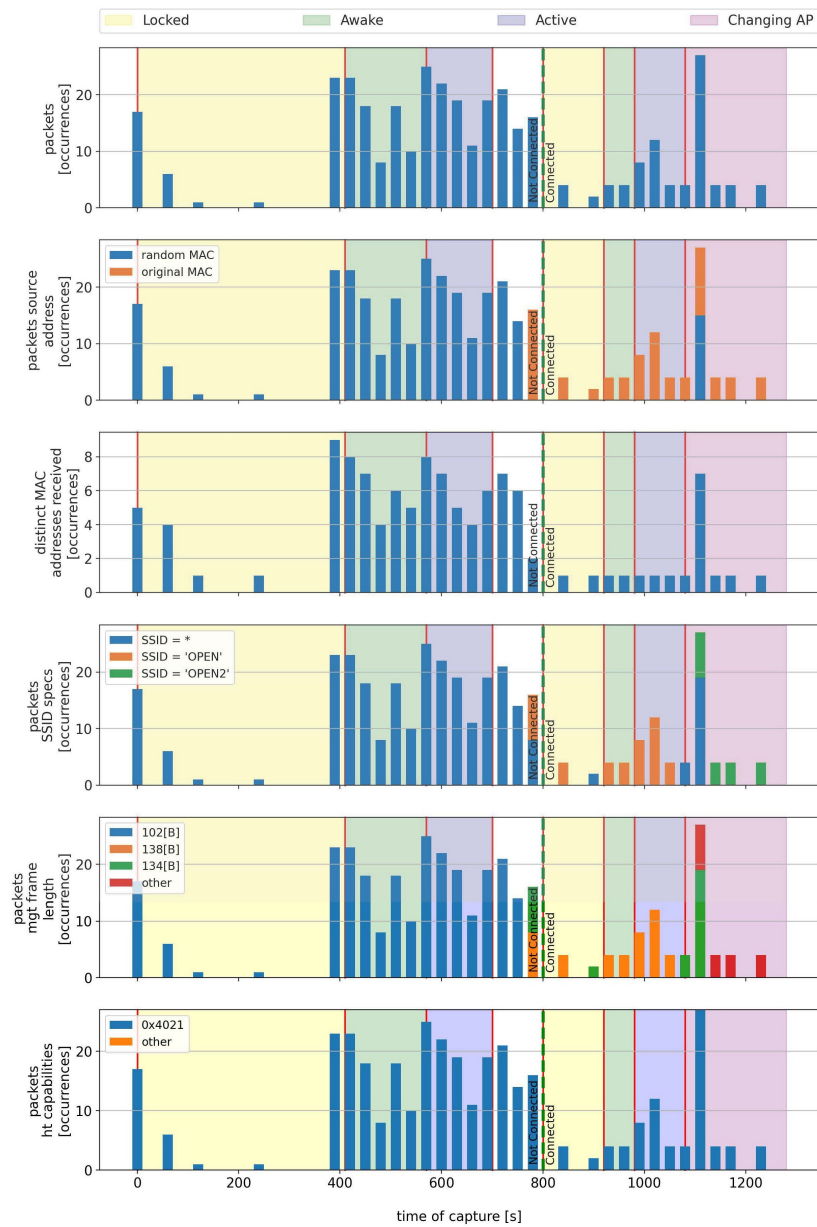


Fig. 2.13 Comprehensive analysis of probe request packets sent from an Apple iPhone 6 for different usage phases and for different connection states.

WiFi Probe Request for Crowd Monitoring

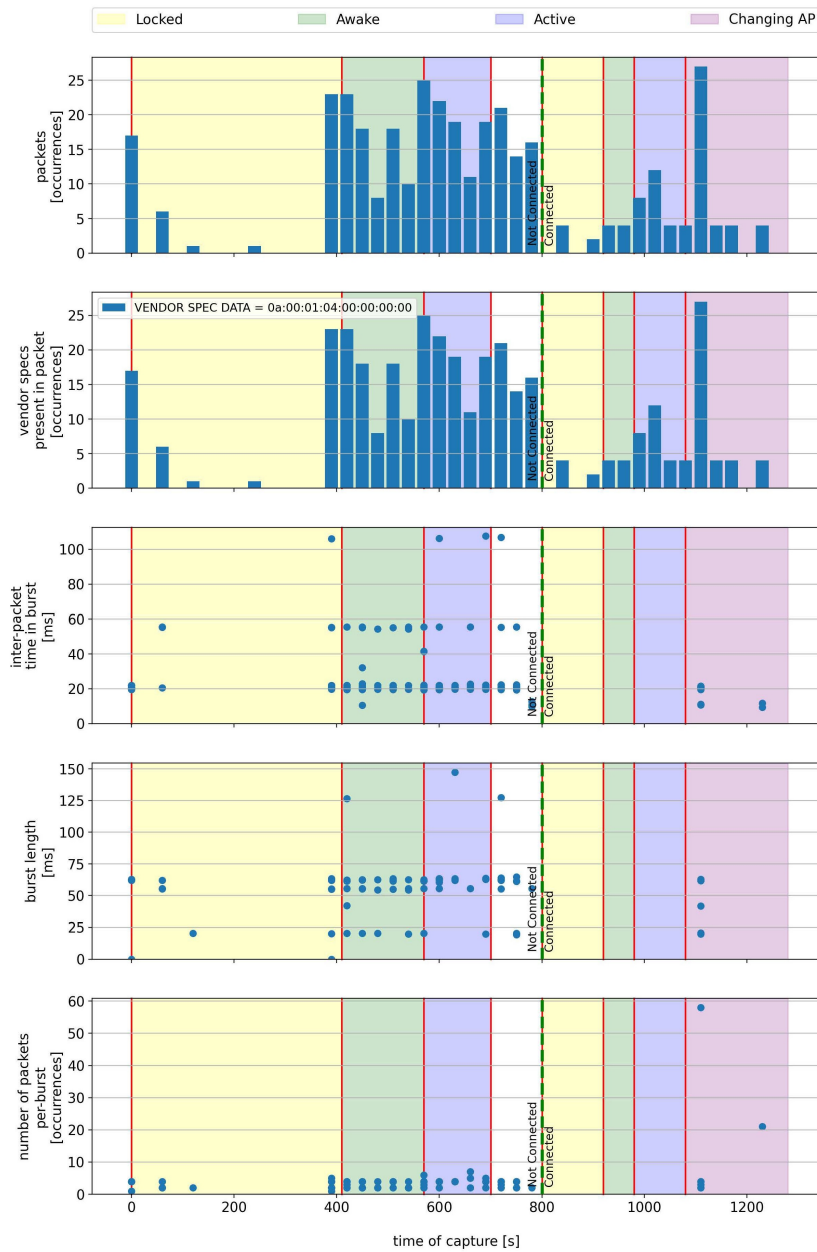


Fig. 2.14 Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 6, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states.

WiFi Probe Request for Crowd Monitoring

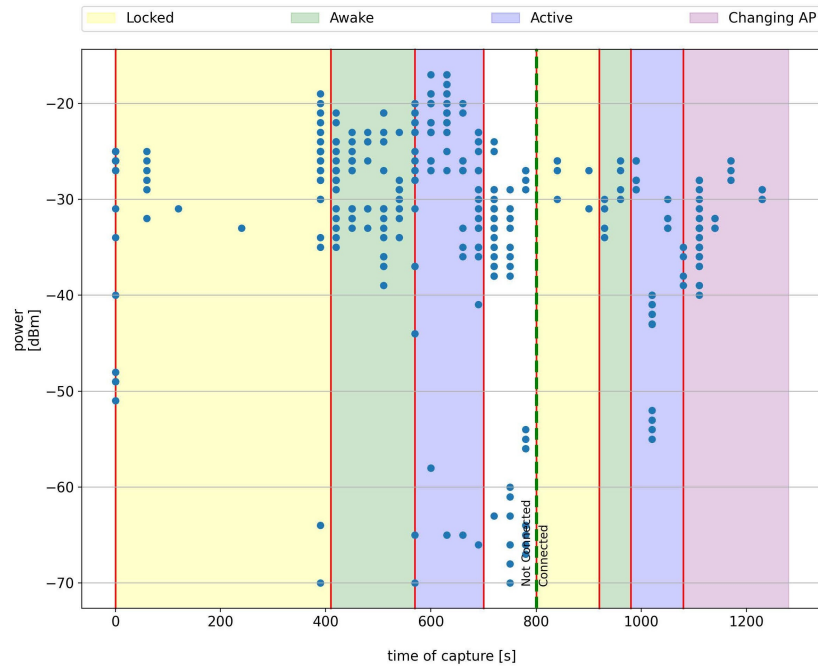


Fig. 2.15 Comprehensive analysis of RSSI power field for an Apple iPhone 6 as a function of time, for different usage phases and for different connection states.

The power analysis of the iPhone 6, as depicted in Figure 2.15, reveals no significant differences when compared to the iPhone 11. There are a few packets between the Active and the Awake phases that arrived with slightly higher power, but these observations do not provide substantial grounds for making assumptions or drawing conclusions.

Finally, we have undertaken the task of summarizing and comparing the experimental results obtained for all the devices. The key characteristics have been consolidated and presented in Tables 2.3, 2.4, and 2.5. This comprehensive comparison encompasses the following features:

- **MAC address randomization;**
- **Burst characteristics;**
- **Sequence number;**
- **High Throughput (HT) Capabilities;**
- **Management (MGT) Frame;**
- **SSID Field;**
- **Vendor Field.**

In Tables 2.3, 2.4, and 2.5 the “connected/unconnected” label indicates the device’s connection status to the AP.

As evident, virtually every device displays distinctive behaviors within each of the aforementioned categories, thereby adding complexity to the challenge of establishing guidelines for quantifying the number of unique devices, and consequently, the individuals carrying them, within a given area. To sum up, in Section 2.6.4, we will provide a summary of our research findings.

WiFi Probe Request for Crowd Monitoring

	OnePlus Nord 5G (2021)	Samsung Note 20 Ultra (2020)	Apple iPhone 11 (2019)
MAC address Randomization	randomized at each burst	unconnected: changes at each burst; connected: constant random MAC address	unconnected: changes at each burst; connected: constant random MAC address
Burst length	about 3/5 packets	unconnected: about 3/5 packets; connected: large bursts with same MAC address	unconnected: 2/3 packets; connected: longer
Sequence Number	increases inside the burst, random starting number in different bursts	unconnected: changes randomly; connected: increases constantly	unconnected: changes randomly; connected: increases (almost) constantly
HT Capabilities	same text in each packet	same text in each packet	mostly same text in each packet
Management (MGT) Frame	unconnected: variable frame lengths; connected: constant length	unconnected: variable frame lengths; connected: constant length	unconnected: variable frame lengths; connected: length depends with AP
SSID Field	never specified	SSID in plaintext in some packets; in others, strings not traceable to any SSID name used in the experiment	SSID in plaintext when connected; wildcard used to look for other APs
Vendor Field	Vendor ID present, but sometimes undefined	Vendor ID always specified	Vendor ID in clear only when connected

Table 2.3 Smartphones behaviour comparison.

WiFi Probe Request for Crowd Monitoring

	Xiaomi Redmi Note 8T (2019)	Huawei P9 Lite (2016)	Apple iPhone 6 (2014)
MAC address Randomization	unconnected: changes at each burst; connected: same random address for each AP	always original, not randomized MAC address	unconnected: changes at each burst; connected: original, not randomized MAC address
Burst length	unconnected: 3/4 packets; connected: longer	1 packet	about 3/5 packets
Sequence Number	unconnected: changes randomly; connected: increases (almost) constantly	increases constantly	increases constantly
HT Capabilities	same text in each packet	changes depending on AP	same text in each packet
Management (MGT) Frame	unconnected: variable frame lengths; connected: length depends on associated AP	frame length constant except for some packets when connecting	unconnected: variable frame lengths; connected: length depends on associated AP
SSID Field	unconnected: wildcard; connected: SSID in plaintext; wildcard used to look for other APs	connected: SSID in plaintext; unconnected: not in plaintext	SSID in plaintext when connected
Vendor Field	Vendor ID always specified	Vendor ID always specified	Vendor ID always specified

Table 2.4 Smartphones behaviour comparison.

WiFi Probe Request for Crowd Monitoring

	Apple iPad 8 (2020)	Lenovo ThinkPad X13 Gen1 (2021)	Apple Mac-BookAir M1 (2020)
MAC address randomization	randomized at each burst	none	unconnected: changes at each burst; connected: original MAC address
Burst length	2/4 packets	1 packet	2/4 packets
Sequence Number	always changes randomly	always changes randomly	always changes randomly
HT Capabilities	same text in each packet	same text in each packet	same text in each packet
Management (MGT) Frame	constant length, which changes only when connecting to AP	constant length when looking for APs, then changes when SSID field is specified	constant length when looking for APs, then changes when SSID field is specified
SSID Field	wildcard, except when connecting to AP	SSID specified a few times also when connected to AP	wildcard, except when connecting to AP
Vendor Field	Vendor ID specified in clear only at the moment of connection to the AP	Vendor ID never specified	Vendor ID specified in clear only at the moment of connection to the AP

Table 2.5 Tablet and laptops behaviour comparison.

2.6.3 Results comparison between different years and OS version

Utilizing the findings from the second set of experiments conducted in 2023, we can now compare the performance of the Apple iPhone 11 over the span of a year and with an updated operating system, transitioning from iOS 15.0.1 to iOS 16.3.1.

Figures 2.16, 2.17, and 2.18 present analyses of received packets, vendor-related data with burst analysis, and power-related insights for the 2023 experiments. These figures can be juxtaposed with their counterparts from the 2022 experiments, showcased in Figures 2.10, 2.11, and 2.12. It is important to note that the results extracted in 2023 should be compared specifically with the “Non Connected” part of the graphs, approximately the first half of each graph before the vertical green dotted line, in the 2022 counterpart.

Examining the number of packets sent during the experiments, the newer version of the operating system appears to send fewer packets compared to its predecessor. Instead, the consistent use of randomized MAC addresses and the inclusion of a Wildcard value in the SSID list are confirmed. Additionally, the length of the management frame and the HT capabilities value remains unchanged.

Shifting the focus to vendor-related data, a similar packet count per burst is observed, but higher burst length and inter-packet time in bursts are detected compared to the older experiments. This suggests that the newer OS of the Apple iPhone 11 tends to send probe request packets less frequently, with a longer time gap between consecutive packets within the same burst.

Regarding power analysis, the notion that RSSI values are highly erratic is confirmed. Figure 2.18 illustrates that the RSSI set of values ranges between -36 dBm and -69 dBm, even when the smartphone is positioned in a fixed location in a large park without any obstacles around.

WiFi Probe Request for Crowd Monitoring

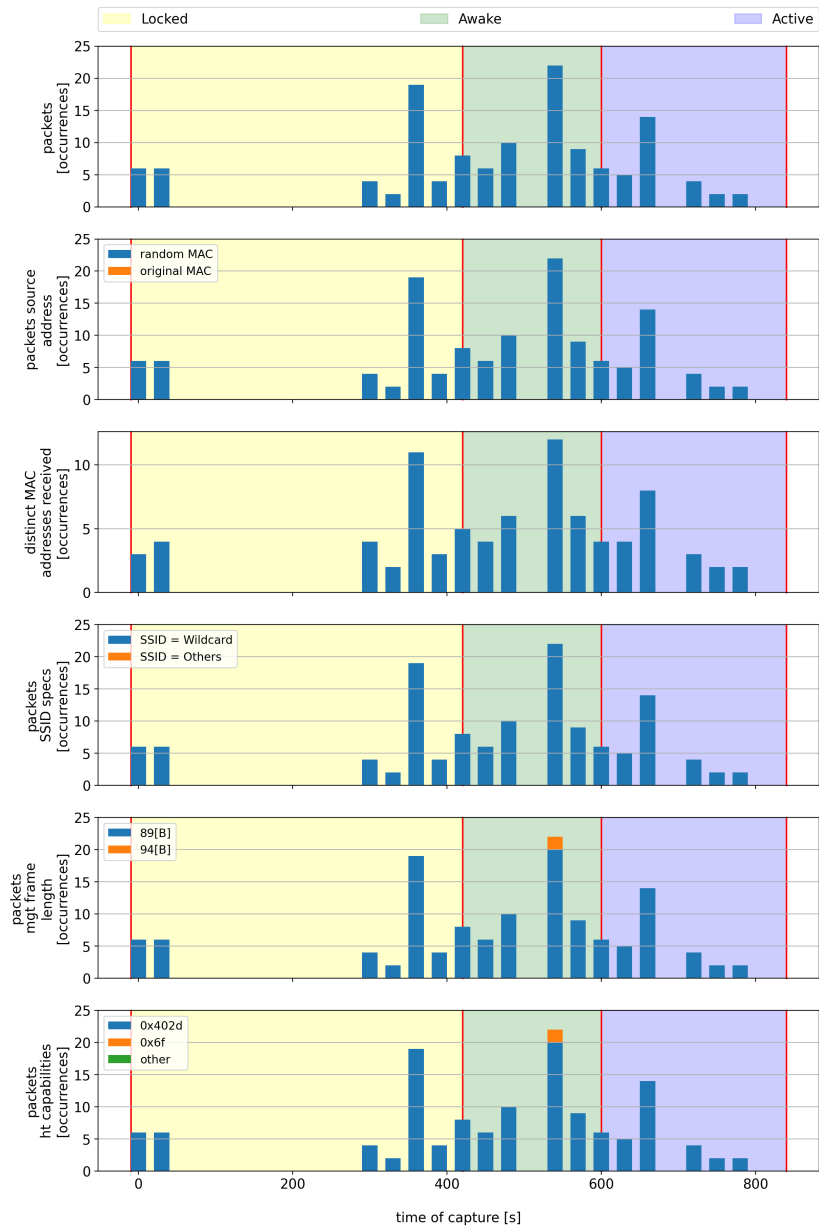


Fig. 2.16 Comprehensive analysis of probe request packets sent from an Apple iPhone 11, during 2023 experiments, for different usage phases and for different connection states.

WiFi Probe Request for Crowd Monitoring

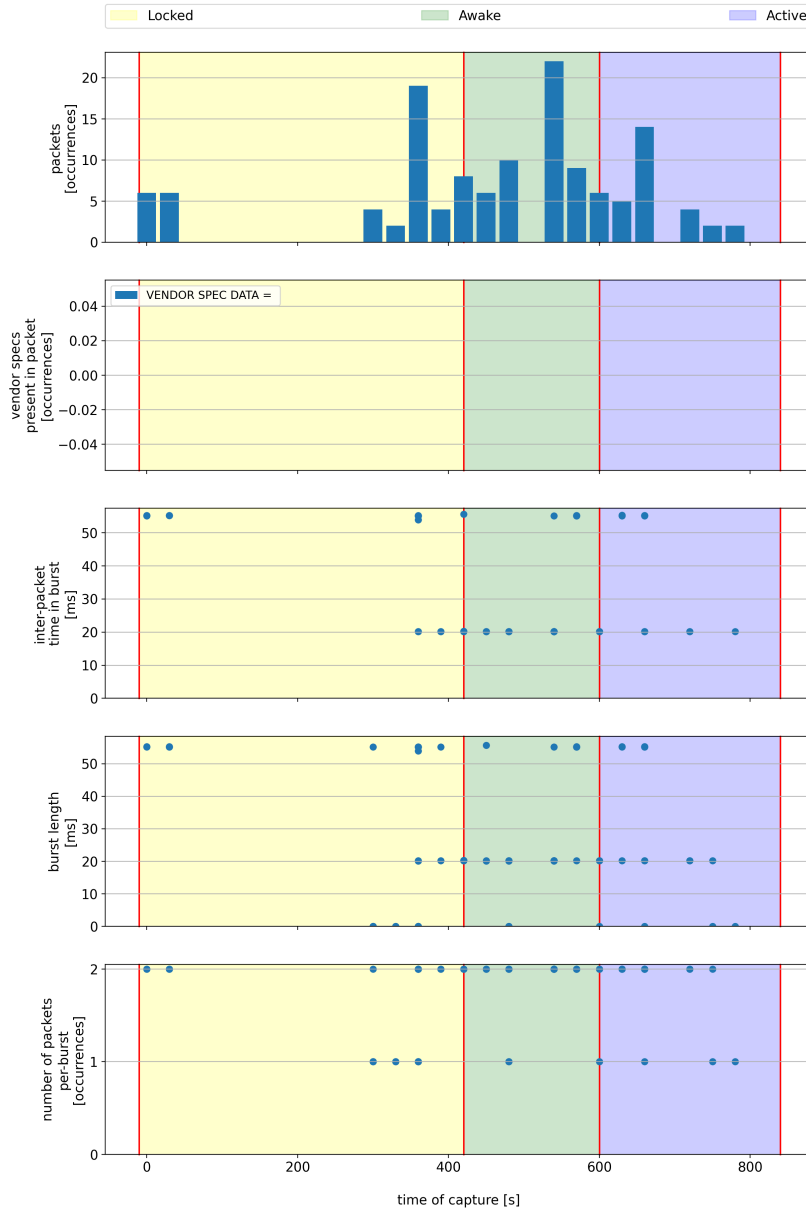


Fig. 2.17 Comprehensive analysis of vendor-related data and probe requests packets burst characterization sent from an Apple iPhone 11, during 2023 experiments, highlighting the bursts and the vendor field as a function of time, for different usage phases and for different connection states.

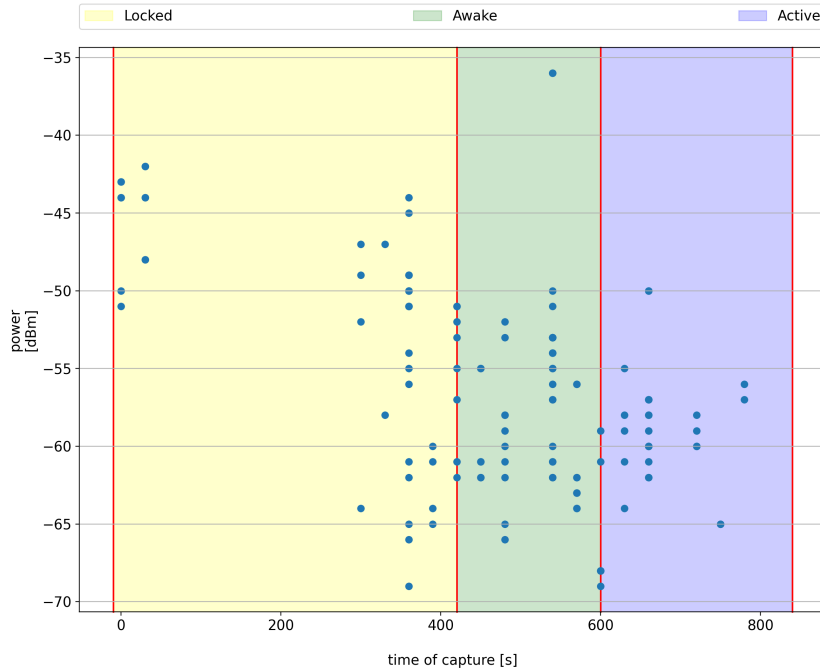


Fig. 2.18 Comprehensive analysis of RSSI power field for an Apple iPhone 11, during 2023 experiments, as a function of time, for different usage phases and for different connection states.

2.6.4 Experimental findings

In summary, the main takeaway from our analysis is that each device has a distinct behavior when it comes to sending probe request messages. There are notable differences among the tested devices.

The number of probe requests varies significantly, i.e., Xiaomi sends only a few packets, while OnePlus sends the highest number.

MAC randomization processes differ substantially across devices. Randomization is typically applied on a per-burst basis, and burst length increases slightly when transitioning from iOS to Android. Notably, MAC randomization is not consistent within models from the same manufacturer, as seen in the transition from Apple iPhone 6 to iPhone 11. In some instances, like the Huawei P9 Lite and Lenovo ThinkPad, MAC addresses are not randomized at all, and the real MAC address appears in plaintext in every probe request.

Moreover, we found that the SSID field is an unreliable unique identifier, as devices do not transmit it regularly, and for example, OnePlus never transmits it.

Similar behavior for the sequence numbers in packet bursts that vary a lot. Older phones display linear increases, while newer ones exhibit random increments, making unique device identification challenging.

Some features, such as HT capabilities, remain relatively consistent in many devices, with the exception of the iPhone 11, which rarely changes them. In contrast, the Huawei P9 changes its HT capabilities whenever connected to different access points (APs). Finally, when a device is connected to an AP, its probe request includes information about the connected AP, such as Vendor ID, resulting in a larger probe request size than when not connected.

2.6.5 Limitations

One notable limitation to our study is the rapidly evolving landscape of the smartphone market. New devices are continually being introduced, each with unique broadcasting strategies and message formats. The devices tested in this study represent only a snapshot of the smartphone ecosystem at the time of experimentation. Therefore, it is reasonable to expect that newer devices, not included in this study, may exhibit different behaviors, potentially impacting the generalizability of our findings. Furthermore, the dynamics of mobile operating systems play a significant role in shaping device behavior. Updates and revisions to operating systems can introduce changes in how devices handle probe requests, affecting their broadcasting strategies.

It worth mentioning that while efforts were made to select a diverse set of devices, it is impossible to test every device model available in the market comprehensively. Despite these limitations, it is essential to recognize the valuable insights gained from this study. Understanding how device usage influences the frequency of probe request transmissions provides valuable knowledge for optimizing the estimation of device counts. Moreover, the calibration of parameters based on observed behaviors enables more accurate device counting by defining appropriate capturing windows.

2.7 Conclusions and future works

In light of the widespread adoption of mobile phones and various smart devices, the capacity to track their presence through data messages they broadcast, such as WiFi probe requests, carries significant implications. This capability allows for the estimation of the number of individuals within a particular area and holds promise for a range of applications, including security enhancements and the optimization of public services. However, achieving this goal is challenging due to the diverse broadcasting strategies and message formats employed by these devices, despite standardization efforts.

In this study, we have undertaken an extensive analysis of the behavior of various device types and operating systems. Our findings reveal that numerous strategies, including MAC address randomization, pose obstacles to accurate device quantification. Nevertheless, by considering additional parameters, such as the duration of beacon bursts or specific fields within the beacons themselves, it becomes feasible to devise more comprehensive approaches for approximating individual device broadcasts.

This research represents the initial phase in the ongoing journey towards implementing a system capable of accurately counting the number of devices within its coverage area and gaining valuable insights into the flow of people across various access points. The motivation behind this study was rooted in the necessity to gain a deeper understanding of probe request message behavior to enhance counting algorithms. This investigation serves as the foundational groundwork for the activities detailed in Chapter 3. It provides the essential knowledge required to develop and deploy our custom solution, along with a tailored counting framework, as elucidated in Chapter 4. The culmination of these efforts aims to offer an advanced and effective solution for device counting and provide invaluable insights into user behavior in a variety of scenarios.

Chapter 3

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters

Accurate estimation of the number of individuals within a specific area plays a vital role in the fields of crisis management and disaster response. This capability enables the precise monitoring of crowd dynamics and supports effective decision-making processes. However, when we leverage the WiFi fingerprint technique, which relies on the MAC addresses of mobile devices as a proxy for people counting, it introduces privacy concerns. The stringent European General Data Protection Regulation (GDPR) [39] and the proactive measures taken by leading smart device vendors, such as MAC randomization, led to a reevaluation and reconfiguration of most techniques explored in the past. In this Chapter, we present a specialized WiFi traffic generator designed to mimic the authentic behavior of WiFi cards. This generator serves as a crucial tool for establishing the ground truth against which counting algorithms can be evaluated. Additionally, we introduce a novel crowd monitoring technique that utilizes Bloom filters to ensure a formal *deniability* property, thereby safeguarding users' privacy.

3.1 Research motivation and State of the Art

In recent years, companies and organizations have undergone a significant shift in their approach towards safeguarding users' privacy. This transformation has been particularly pronounced within the European Union, where a crucial milestone was achieved in 2018 with the implementation of the General Data Protection Regulation, commonly known as GDPR [39]. This legislative initiative has empowered individuals with greater control over their personal data and has placed stringent ethical and responsible data management obligations upon organizations. The GDPR framework comprises meticulously crafted rules that encompass all facets of data-related activities, spanning from data collection to utilization and sharing. In response to the privacy-enhancing measures introduced by vendors, Machine Learning (ML) techniques have emerged as a viable solution to enhance counting accuracy by attempting to infer whether different WiFi probe requests belong to the same device. This approach necessitates a substantial volume of data with ground truth information to train these models effectively. Furthermore, in compliance with GDPR's stringent regulations, there is a mandatory shift in the way MAC addresses are stored and managed.

In the document [10], the Italian Data Protection Officer (DPO) underscores the classification of MAC addresses as personal data under GDPR, necessitating the implementation of robust privacy protection mechanisms.

Numerous solutions have been put forth in the existing literature to tackle this issue, including references [91–93]. These solutions collectively address privacy concerns by harnessing Bloom filters for the storage of MAC address information and by employing an asymmetric homomorphic encryption system, which is applied to the data within the Bloom filter. Homomorphic encryption (HE) [78] represents a specific encryption paradigm that permits mathematical operations to be executed directly on encrypted data without necessitating decryption. The outcomes of these operations, also encrypted, are identical to the results obtained from performing the same operations on unencrypted data. Homomorphic encryption not only upholds the security of sensitive information but also retains the ability to compute intersections between distinct Bloom filters. This capability is pivotal in the computation of crowd flow trajectories, maintaining a robust level of privacy protection throughout the process.

As demonstrated in [21], Bloom filters offer a means of preserving the privacy of stored data. However, to achieve the formal level of anonymity mandated by GDPR, it is imperative to have a minimum number of data insertions. In other words, a small quantity of inserted data does not guarantee the desired level of anonymity. It is important to note that this observation is not addressed in [91–93], rendering their proposed solutions applicable primarily for sufficiently large crowds. Furthermore, in cases where the Bloom filters are only encrypted when transmitted over the network, there is a risk of exposing the data contained within the sniffer device to potential privacy breaches.

Moreover, the work presented in [21] introduces two crucial concepts pertaining to anonymity protection: γ -deniability and γ - K -anonymity. The first concept posits that an element stored in the Bloom filter is considered “deniable” if it can be substituted with other elements not initially inserted into the filter, all without altering the Bloom filter bitmap. The second concept extends the first by asserting that an element in a Bloom filter attains γ - K -anonymity if it can be “covered”, with a probability of γ , by $K - 1$ other elements that were not originally included in the filter. These principles are fundamental for achieving the desired level of privacy and anonymity in data storage and analysis.

3.2 Main contributions

The current study, the details of which are outlined in [83, 86], undertakes a comprehensive exploration of two primary facets. Firstly, it focuses on the development of a WiFi probe request generator capable of producing numerous datasets in the form of *.pcap* traces, all equipped with invaluable ground truth data and various accompanying statistics. Building upon the insights provided in [85], this generator enables the emulation of the probe request behaviors of the analyzed devices, as well as those analyzed subsequently. Furthermore, the study delves into the effective storage of MAC addresses, ensuring compliance with the regulatory constraints imposed by GDPR [39]. It introduces the novel concept of anonymization noise as a means to fulfill these regulatory requirements.

Our work makes significant contributions to the current literature in the following key areas:

- **Probe Request Generator:** We have developed a unique and pioneering probe request generator, capable of accurately emulating the real behavior of multiple devices that were subject to testing. This tool enables the creation of diverse scenarios, facilitating the generation of realistic *.pcap* traces along with their corresponding ground truth data.
- **Anonymization noise applied to Bloom Filters:** We introduce the novel concept of anonymization noise, a crucial addition to Bloom filters. This innovative approach ensures that all the MAC addresses stored into the Bloom filter are 1-deniable, signifying that they can be deniable with probability equal to one.

In the following Sections, our approach will involve a thorough examination of prevalent anonymization techniques currently in use. Following this, we will furnish an extensive and in-depth account of our newly developed probe request generator, drawing from insights obtained in [85]. Subsequently, we will unveil our privacy-conscious solution for the analysis of people flows, harnessing the power of Bloom filters and the γ -deniability property. In closing, we will present the outcomes derived from both of these contributions, offering a comprehensive overview of the results and some final remarks.

3.3 Anonymization techniques for storing MAC addresses

Anonymization techniques are a set of strategies and processes employed to eliminate or modify identifying information within personal data. Their primary purpose is to render data either fully anonymous or pseudo-anonymous, thereby diminishing the risk of individual recognition. These techniques aim to strike a delicate balance between facilitating the use of data for legitimate purposes such as research and analysis while simultaneously upholding the rights and privacy of individuals.

In this Section, we embark on an exploration of the General Data Protection Regulation (GDPR) [39] by elucidating its core principles and identifying the key stakeholders within its framework. Subsequently, we delve into an exhaustive analysis of diverse anonymization techniques. Our focus will be on describing

various types of hash functions and the versatile Bloom filter data structure. By doing so, we aim to provide an in-depth understanding of these techniques and their applications in preserving data privacy and security.

3.3.1 GDPR

In the digital age, the General Data Protection Regulation (GDPR) [39], or RGPD in Italian, stands as a cornerstone of data protection. This comprehensive European regulation seeks to harmonize and fortify rules governing the collection and processing of personal data, with the overarching goal of safeguarding individual privacy

GDPR's core mission revolves around ensuring the privacy and security of personal data. The regulation divides data into two categories: sensitive data and identifying data. Sensitive data encompasses deeply personal information, such as an individual's race, religious beliefs, political affiliations, sexual orientation, health status, and economic and social standing. Identifying data includes personal details, residential addresses, and digital identifiers like email addresses, cookies, IP addresses, and geolocation data. These distinctions are pivotal, as they provide a foundation for different levels of protection based on data sensitivity.

One of GDPR's remarkable features is its broad applicability. It extends its protective umbrella over a wide range of entities, encompassing not only individuals but also professionals and companies, regardless of their location. GDPR governs the processing of personal data belonging to European citizens, whether it occurs inside or outside the European Union and in both online and offline contexts.

To fulfill its mission effectively, GDPR identifies four key actors in data protection:

- **Data Subjects:** These individuals are the rightful owners of the data being processed, empowered with certain rights to access, rectify, or erase their data. This hands control back to the individuals, aligning with GDPR's philosophy of data privacy as a fundamental human right.
- **Data Controllers:** These are typically companies to whom users willingly entrust their data. Data Controllers determine the objectives and procedures for data processing and are responsible for implementing measures to ensure

compliance with privacy regulations, emphasizing the principle of privacy by design.

- **Data Processors:** Individuals appointed by Data Controllers who work collaboratively on implementing technical and organizational measures to ensure data security. They play a pivotal role in responsible data handling.
- **Data Protection Officers (DPOs):** Responsible for ensuring adherence to GDPR, DPOs must possess specialized knowledge of data protection regulations.

In the context of employing WiFi probe requests for people counting, even when MAC addresses are partially randomized, concerns about privacy persist. While measures such as de-identification, encryption, or pseudonymization are often used to protect personal data, it is important to recognize that data that can still be used to re-identify an individual falls within the scope of the GDPR. In a document referenced as [10], the Italian Data Protection Officer (DPO) has underlined this classification of MAC addresses as personal data. As a result, the GDPR necessitates the implementation of robust privacy protection mechanisms when dealing with such data.

3.3.2 Hash function

Hash functions are cryptographic algorithms utilized to transform data of varying sizes, called *message*, into a consistent, fixed-length value known as *hash*. These algorithms are engineered to operate in a one-way fashion, implying that it should, in theory, be infeasible to reverse the function and deduce the original plaintext data. In Figure 3.1, we can observe a practical example of how these hash functions operate.

The process begins by segmenting the input data into uniform, fixed-sized blocks, aptly named *data blocks*. If the data blocks are not sufficiently large, padding may be introduced to ensure they reach the required size. Subsequently, the hash function is iterated for as many times as there are data blocks, and with each iteration, the output of the preceding data block serves as the input for the subsequent block. Consequently, the final output represents the cumulative value of all the processed blocks. It is noteworthy that even a single alteration in one bit anywhere within the message will result in a completely different hash value, highlighting the sensitivity

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters

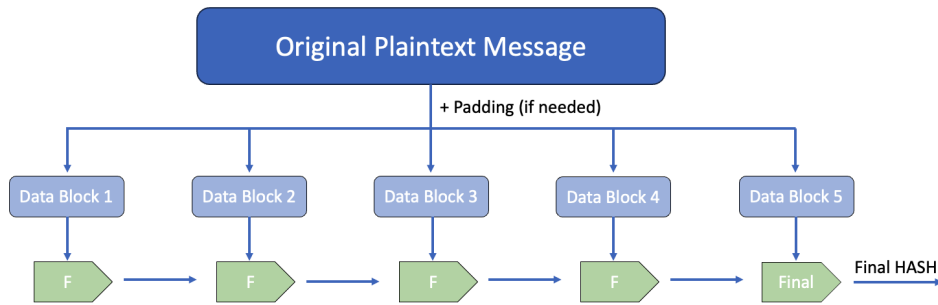


Fig. 3.1 Hash function algorithm.

and effectiveness of hash functions in detecting even the slightest modifications in the input data.

Hash functions possess several crucial characteristics, each contributing to their effectiveness and utility:

- **Uniqueness:** When the same hash function is applied to a specific input, it will unfailingly produce the same output. However, two distinct inputs should result in entirely dissimilar hash values. This characteristic is essential for the integrity and consistency of the hash function.
- **Speed:** Hash functions must exhibit efficiency and rapidity in calculating the digest. This attribute is pivotal when dealing with large volumes of data, ensuring that the processing of information remains expeditious.
- **Collision Resistance:** Collisions occur when two distinct inputs yield the same hash output. Hash functions are meticulously engineered to minimize the likelihood of collisions as much as possible. This robust collision resistance makes it arduous, if not practically impossible, to find two different input values that generate the same hash value.
- **Data Sensitivity:** Even minor alterations in the input data should result in a significantly distinct hash value. This property, often referred to as *diffusion*, bolsters the security of the hash function, making it highly responsive to changes in the input data.

Ideally, hash functions are designed to be irreversible, implying that while it is relatively quick and straightforward to compute the hash when the input message

is known, it should be exceedingly challenging to reverse the process and deduce the original input message when only the hash value is available. The process of computing the input message from the output hash value, although feasible, requires substantial computational power and is typically accomplished through a “brute force” search. This method involves employing trial and error to identify a message that matches the given hash value, which is an intricate and resource-intensive process.

Some common hashing algorithms include MD5, SHA-1, SHA-2, LANMAN, and NTLM.

MD5

MD5 is the fifth version of the Message Digest algorithm, producing 128-bit hash outputs. Initially, MD5 was a widely adopted hashing algorithm, but it fell out of favor due to the emergence of vulnerabilities, notably collisions. Consequently, it was gradually phased out.

SHA-1

SHA-1, the second version of the Secure Hash Algorithm standard (following SHA-0), generates 160-bit hash outputs. In response to the vulnerabilities discovered in MD5, SHA-1 gained prominence as a replacement.

SHA-2

SHA-2 represents a suite of hashing algorithms that introduces significant improvements over its predecessor, SHA-1. The SHA-2 family comprises six distinct hash functions, each producing digests (hash values) of varying lengths, including 224, 256, 384, or 512 bits. These specific members of the SHA-2 family are SHA-224, SHA-256, SHA-384, and SHA-512.

LANMAN

Microsoft LANMAN (LAN Manager) served as a hashing algorithm for storing passwords on legacy Windows systems. It employed DES (Data Encryption Standard) algorithms for hashing. Unfortunately, the DES implementation in LANMAN

is not very secure, making it susceptible to brute force attacks. Password hashes in LANMAN can be cracked in just a few hours. Microsoft no longer defaults to LANMAN as the storage mechanism, although it remains available but is turned off by default.

NTLM

NTLM, which stands for NT LAN Manager, is an algorithm used for password hashing during authentication. It succeeded LANMAN and was subsequently followed by NTLMv2, which employs an HMAC-MD5 algorithm for hashing.

3.3.3 Salted hash

One of the challenges posed by hash functions is their deterministic nature, where the same input consistently yields the same output. This predictability becomes a security concern when multiple users select the same password because their hashed passwords will be identical. This predictability can aid attackers in their efforts to discover the original plaintext corresponding to a particular hash. Once the password is uncovered, it can potentially be used to gain access to all accounts using that particular hash.

One effective solution to counter this issue is the utilization of a technique called *salted hashing*. Salted hashing enhances security by introducing an extra layer of randomness into the hashing process, as depicted in Figure 3.2. This involves the inclusion of a randomly generated string known as the *salt* into the input before performing the hash operation. This addition of salt makes it considerably more challenging for an attacker to deduce the original plaintext without access to both the salt and the hashed value.

Moreover, the length of the salt plays a critical role in fortifying the security of the hashing process. A longer salt substantially increases the computational complexity involved in attacking the hash, thereby increasing the candidate set exponentially. Furthermore, a longer salt also augments the storage space required for hash tables, diminishing the likelihood that such tables exist in the wild. This combination of factors significantly bolsters the security of the salted hashing approach.

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters





				
Password	qwerty	qwerty	qwerty	qwerty
Salt	-	-	5ucc355	r35ul7
Hash	f3cc12a	f3cc12a	11f4a12	76a12f1

Fig. 3.2 Salted hash function.

The practice of salting offers several crucial advantages for data security. Primarily, it serves as an effective defense against common attacks like rainbow table ones. In these attacks, pre-computed tables of hashed values are employed to accelerate the process of reversing hashes. The introduction of a salt disrupts the predictability of these tables, making them significantly less effective in compromising hashed data.

Additionally, salting provides a robust defense against brute-force attacks, where attackers attempt to guess the plaintext by exhaustively trying various potential inputs. The presence of a salt introduces an additional layer of complexity, making the hash function appear non-deterministic. This is a desirable characteristic as it helps prevent the disclosure of duplicates through the hashing process, thereby enhancing the security of sensitive data.

3.3.4 Truncated hash

Another variation of hash functions is known as truncated hashing. This type of hashing involves the process of shortening the output of a hash function to a specific number of bits. Instead of producing the complete hash value, only a portion of it is retained, as illustrated in Figure 3.3. It is essential to recognize that when a hash is truncated, its theoretical collision-resistance is reduced. In other words, there is a higher probability that different inputs will produce the same truncated hash.

It is noteworthy to mention that within the SHA-2 family of hash functions, there exist simplified truncated variants of their full counterparts. These include SHA-256/224, SHA-512/224, SHA-512/256, and SHA-512/384. The notation, for instance, SHA- X/Y , signifies a full-length SHA- X hash truncated to Y bits. These

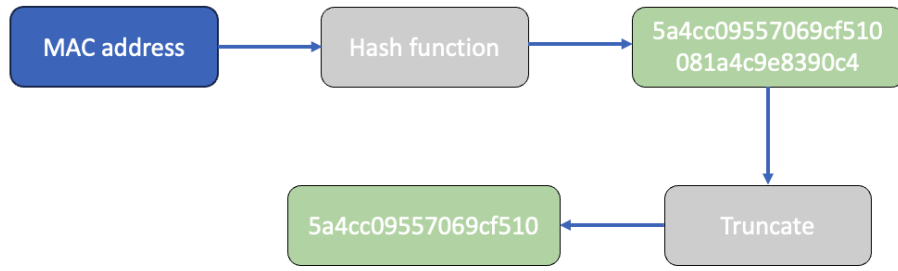


Fig. 3.3 Truncated hash function.

truncated variants provide a way to balance computational efficiency with the desired level of security.

3.3.5 Bloom filter

A Bloom filter is a probabilistic data structure, well known in the literature [95, 27], employed to represent a set of elements. It is implemented using an array of bits, denoted as $BF \in 0, 1^m$, where m is the length of the array, and k independent hash functions, labeled as H_1, H_2, \dots, H_k . These hash functions map an input element x to one of the m bits within the bit array. We denote the i th bit of BF as $BF[i]$. Initially, all bits are set to 0. When inserting an element x into the Bloom filter (a graphical representation can be found in Figure 3.4), the k hash functions are applied to x , and the bits in BF corresponding to the positions generated by the hash functions are set to 1:

$$BF[H_i(x)] = 1 \quad \forall i = 1, \dots, k \quad (3.1)$$

To verify if an element is present in a Bloom filter, the element is hashed through the same set of k hash functions, and the output is compared to the current values of the corresponding bits in the Bloom filter (BF). If all the 1s in the output match the corresponding bits in BF (i.e., both are set to 1), the element is considered *probably present* in the Bloom filter. However, if even a single bit in the match is set to 0, the element is considered definitely not present in the Bloom filter.

One notable limitation of Bloom filters is their inability to delete elements from the filter, as there is no guarantee that the element is present in the filter. However, in

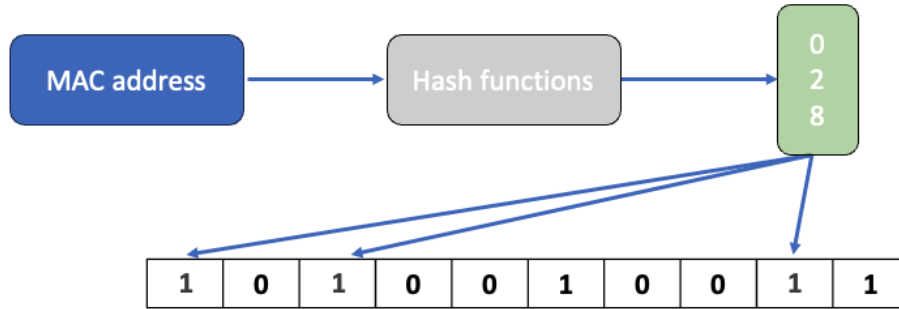


Fig. 3.4 Bloom filter algorithm.

our specific use case, this limitation is not a concern as our primary need is to add MAC addresses to the filter without the requirement to remove any of them.

False Positive

It is important to note that a Bloom filter can potentially provide false positives, meaning that it may mistakenly indicate that an element is present when it is not. However, it does not provide false negatives, meaning that it cannot mistakenly indicate that an element is not present when it actually is.

By mapping elements from a larger universe to a smaller universe represented by a bit-vector, there is a calculable probability that an element x not belonging to set S experiences collisions for each output of the k hash functions. When such collisions occur, it is referred to as a “false positive”. (3.2) provides a way to compute this probability.

$$\Pr(\text{false positive}) = (1 - p)^k = \left(1 - \left(1 - \frac{1}{n}\right)^{mk}\right)^k \simeq \left(1 - e^{-\frac{km}{n}}\right)^k \quad (3.2)$$

where n is the number of bits set to 1 in the Bloom filter.

Privacy

The evaluation of the privacy protection offered by Bloom filters, as discussed in work [94], involves an analysis of the extent to which an adversary can gather information in different scenarios. This assessment encompasses three distinct attack scenarios, each varying in terms of the adversary's level of knowledge. Importantly, in all of these scenarios, the adversary does not possess specific knowledge about the elements contained in the stored dataset. The three attack scenarios are the following:

- **Agnostic Outsider:** In this scenario, the adversary has absolutely no knowledge of the algorithms, data structures, data format, or any concealed secrets used in the scheme. This situation may occur if data is unintentionally disclosed by an insider, and an external adversary captures the data without understanding its content. In such cases, it is critical that the data does not reveal any information about its inherent composition. However, because data often comes with contextual information and metadata, such as data origin or file names, an adversary might attempt to gather more information about the scheme using publicly available data or through reverse engineering. Consequently, the feasibility of this scenario is limited due to practical constraints.
- **Outsider:** In this scenario, the attacker understands the algorithms, data structures, and data format implemented within the scheme but lacks knowledge of any concealed secrets. This scenario commonly arises when an external adversary targets the system's networks, potentially extracting data from an employee's laptop. In such situations, the adversary might gain access to documentation, applications, or even metadata associated with the data structure and algorithms if they are accessible within the system.
- **Insider:** In this scenario, the adversary is aware of the algorithms, data structures, data format, and the shared secrets employed in the scheme. However, they remain unaware of the actual entries within the data set. This situation is frequently encountered when dealing with an internal adversary, such as a curious or disgruntled employee, or when a powerful external adversary targets various systems and networks.

3.4 Probe request generator

In this Section, our primary focus is on the development and validation of a sophisticated probe request generator. This generator has the capability to faithfully replicate the behavior of various devices that we have thoroughly analyzed. It is important to highlight that the primary goal of the probe request generator lies in its ability to produce as many as we need dataset comprising not only probe request messages in the form of *.pcap* traces but also their corresponding ground truth. This capability enables us to create datasets that can be used, together with real ones, to train sophisticated machine learning models. Once these models are trained, they have the potential to significantly improve performance in tasks like people counting, effectively overcoming challenges associated with MAC address randomization.

In the subsequent parts of this Section, we will delve into different aspects of the probe request generator. Section 3.4.1 outlines the crucial back-end data that the generator utilizes to construct and emulate the real behavior of the devices under examination with exceptional precision. Moreover, it catalogues the array of devices we have examined and incorporated into our unique generator. Sections 3.4.2 and 3.4.3 offer an overview of the event-driven finite-state machine generator we have developed, with a particular focus on the various states and transitions within the generator. Furthermore, Section 3.4.4 highlights one of the prominent challenges encountered when emulating the behavior of actual packets: the issue of collisions between multiple packets arriving simultaneously. Section 3.4.5 describe the process of generating a probe request packet in Python. Finally, in Section 3.4.6, we thoroughly explore the validation aspect of the generator, demonstrating its exceptional ability to faithfully emulate the real behavior of probe request messages.

3.4.1 Back-end data

In this Section, we provide an introduction to the key components of our investigation. Specifically, we first present the devices that have undergone testing, then we delve into the analysis of probe request characteristics for each device in its three primary states: locked, awake, and active. The comprehensive insights presented in this Section collectively contribute to the back-end data used by our probe request generator. This data is instrumental in enabling the generator to emulate the realistic

behavior of devices when transmitting probe request messages with a high level of accuracy.

Devices

Our investigation commenced with the devices analyzed in [85]. Then we further expanded our dataset by including newer devices and meticulously replicating all the experiments outlined in Chapter 2. To accomplish this, we followed the methodology elucidated in Section 2.5.

Table 3.1 provides a comprehensive overview of all the devices that were subjected to testing and analysis. These devices collectively contribute to the population of our probe request generator database. Each entry in Table 3.1 includes information such as the device type (e.g., smartphone, tablet, or laptop), vendor, model, operating system, and year of production. It is important to note that our approach was not limited to a single family of devices or a specific type. Instead, we sought to examine a diverse array of combinations, encompassing various device types, production years, ranging from the latest high-end models to older or more affordable devices. Furthermore, the database can easily be extended by incorporating newer devices, thereby enhancing the capabilities of our generator.

Analyzed characteristics

For each device present in Table 3.1 we characterize the following metrics:

- whether a randomized MAC address is used;
- number of packets inside a burst (burst length);
- 802.11 VHT capabilities;
- 802.11 Extended capabilities;
- 802.11 HT capabilities;
- sequence number;
- WiFi Protected Setup (WPS);

Probe Request Generator and Privacy-Aware People Flow Monitoring through
Bloom Filters

- Universally Unique Identifier-Extended (UUID-E);
- time between packets inside the same burst (inter-packet time);
- time between different bursts (inter-burst time).

Type	Vendor	Model	OS	Year
SmartPhone	Apple	iPhone 14 Pro	iOS 16.4	2022
SmartPhone	Apple	iPhone 13 Pro	iOS 16.3	2021
SmartPhone	OnePlus	Nord 5G	Android 11.0	2021
SmartPhone	Samsung	Note 20 Ultra	Android 12.0	2020
SmartPhone	Xiaomi	Mi9 Lite	Android 10.0	2020
SmartPhone	Apple	iPhone 11	iOS 15.0.1	2019
SmartPhone	Apple	iPhone 11	iOS 16.3.1	2019
SmartPhone	Xiaomi	Redmi Note 8T	Android 10.0	2019
SmartPhone	Apple	iPhone 7	iOS 15.2	2016
SmartPhone	Huawei	P9 Lite	Android 7.0	2016
SmartPhone	Apple	iPhone 6	iOS 12.5.5	2014
Tablet	Apple	iPad 8	iPadOS 14.8.1	2020
Laptop	Lenovo	ThinkPad X13 Gen1	Windows 11	2021
Laptop	Apple	MacBookAir M1	macOS 12.1	2020
Laptop	Apple	MacBookPro	macOS 11.6.2	2015

Table 3.1 Devices tested for our probe request generator database.

Each time and burst feature, including inter-packet time, inter-burst time, and the number of packets per burst, is not represented as a single value, but rather as a series of values along with associated probabilities. This approach facilitates the assignment of specific weights to individual values during the selection process inside the generator. Notably, we leverage the capabilities of the Python library called Numpy [64], which allows us to make random selections from a set of values, associating each value with its respective weight. In our specific case, the weights correspond to the probabilities associated with the values obtained from real probe request data. Through this method, the system can accurately replicate the time and burst behaviors of the device when sending probe requests. For instance, let us consider the inter-packet time for the Apple iPhone 6 during the locked phase. It can be represented as a pair comprising a value and its associated weight, as shown below:

$$0.02:0.833 - 0.06:0.167$$

In this example, the device is expected to exhibit an inter-packet time of 0.02 seconds with an approximately 83% probability, while there remains a 17% probability of experiencing an inter-packet time of 0.06 seconds.

3.4.2 Finite-State machine

The structure of the probe request generator follows a state machine, also referred to as a Finite-State Machine (FSM). In the realm of computer science and engineering, an FSM serves as a computational model to represent the behavior of systems that can exist in a finite number of states. It operates as an abstract machine that transitions from one state to another in response to external inputs, internal conditions, or a combination of both.

Usually, a state machine is characterized by four key components:

- **States:** These are distinct conditions or configurations that a system can assume at any given moment. Each state encapsulates a specific behavior or operational mode, defining how the system responds and functions under particular circumstances.

- **Transitions:** Transitions define the conditions under which the state machine shifts from one state to another. These transitions are triggered by events or input signals and determine the next state of the system.
- **Events:** Events are external inputs or signals that induce the state machine to transition from one state to another. They can encompass user inputs, sensor readings, or any other triggers that influence the system's behavior.
- **Actions:** Actions represent the activities carried out when transitioning from one state to another and are closely linked with transitions. They detail the changes that occur in the system's state, or in the environment, during a transition.

Building upon this computational model, the probe request generator was developed to align with environmental conditions and system characteristics related to probe request transmission and monitoring.

A visual representation of the finite-state machine used to construct the generator is presented in Figure 3.5. The complete implementation of the generator was carried out in Python [71], with the use of Scapy [87] for the generation of probe request packets.

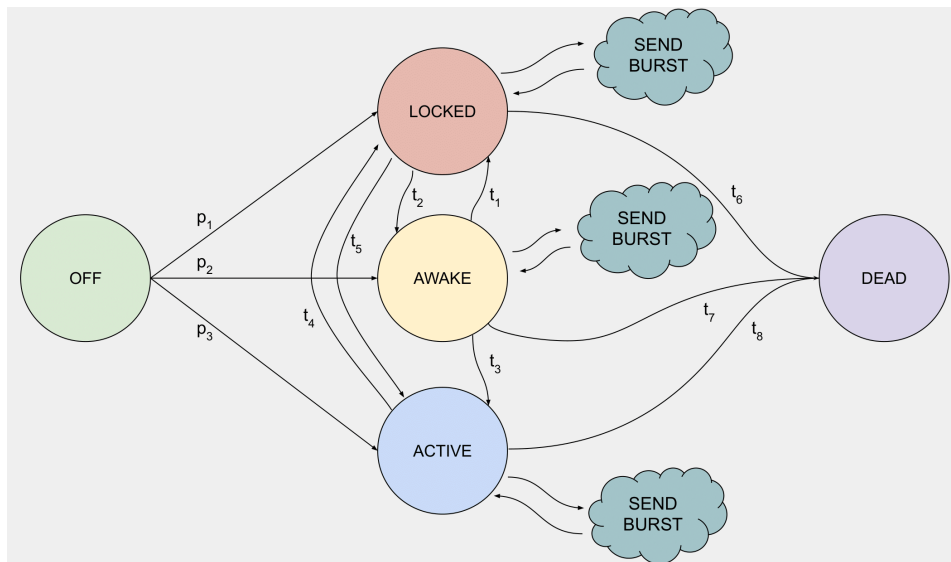


Fig. 3.5 Event-driven finite-state machine diagram.

Inputs

Our probe request generator provides users with a considerable degree of flexibility when it comes to input parameters. Specifically, users have the ability to customize the following parameters:

- **Script time:** This corresponds to the capturing window of the final *.pcap* trace.
- **Average device number:** This parameter allows users to specify the average number of active devices at any given time.
- **Average permanence time:** This parameter signifies the average duration during which a device is considered to be within the coverage of the antenna. We model this feature as it reflects the state of a device being reachable by the scanner when within range, or powered off when out of range.

Furthermore, with minor modifications to the generator code, users have the option to create traces with specific subsets of devices. For instance, it is possible to generate traces containing only specific types of devices, such as smartphones, exclusively Apple/Samsung/etc. devices, or even a single device, among other possibilities.

Outputs

In the generated output, our probe request generator comprises several important elements. Firstly, it produces a *.pcap* trace file, which encompasses all the probe request messages transmitted by the devices within the emulated time window. Furthermore, the output includes the crucial *Ground Truth* dataset that precisely indicates the number of devices that have dispatched at least one probe request. This dataset serves as a reference for analysis.

Additionally, the generator furnishes supplementary statistics. An illustrative example is shown in Figure 3.6. These statistics provide insights into each emulated device, encompassing information about the device itself, the number of MAC addresses employed during its stay within the generator, and various statistical details concerning the volume of packets sent.

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters

```
+++++++Simulation start+++++++

Device 0 information:
Vendor: Apple
Model: iphone6
Use randomization: 1
MAC address: ['9e:48:a4:da:12:85', 'fe:41:c4:3d:4d:1f']
SSID: []
Number of different MAC address: 2
Number of packets sent: 9
Number of different bursts sent: 2

...

Device 5 information:
Vendor: Huawei
Model: p9lite
Use randomization: 0
MAC address: ['00:22:a1:f4:aa:35']
SSID: ['5u9gE9DHA0EbPvCoDSH0Kp0L1ts41XhH', 'RbdoImKj5Ude3Cmjscgq3dXxg7eT6mrz',
'M1PxQMxgRH79Yzg7Wj3rTRS9K6iMLdA0', 'v7VvU0szYzVwmyZcy5fs22s7Pa0158rP',
'qdqtaZ3iAYTXUs3savgSUA9qN7sTmliM', 'VZd1XGS0YSMX5v4PoHd7BwuBAXBwmgNR']
Number of different MAC address: 1
Number of packets sent: 5
Number of different bursts sent: 5

...

Device 22 information:
Vendor: Apple
Model: iphone11
Use randomization: 1
MAC address: ['b2:af:fd:1b:5f:c1', 'e6:61:58:d9:8b:39', '5a:b7:2c:59:21:ac',
'3e:7d:55:5a:d5:a7', 'd2:c0:3c:0d:a8:a1', '5a:96:a0:11:c5:70', '62:7e:e9:ed:6b:72',
'0a:d2:f3:60:a7:38', 'b6:20:05:e0:6b:2c', '8e:67:00:b4:d8:a6', '42:2a:51:0b:9e:1b',
'66:a8:25:6a:ba:de', 'd6:f4:68:7c:94:2f', '52:c2:0b:9d:0a:4e', 'ce:72:e2:94:42:01',
'3a:74:1a:3e:40:4c']
SSID: []
Number of different MAC address: 16
Number of packets sent: 29
Number of different bursts sent: 16

...

Device 37 information:
Vendor: Samsung
Model: note20ultra
Use randomization: 1
MAC address: ['ce:3b:a4:df:f5:65', 'ee:59:2b:52:15:a8', '62:89:82:37:d6:5c',
'be:ab:66:bb:67:53', '5e:40:8b:fa:bb:69', 'f2:95:8a:30:90:8c']
SSID: []
Number of different MAC address: 6
Number of packets sent: 11
Number of different bursts sent: 6

+++++++Simulation end+++++++
Total number of different MAC addresses: 247
Total number of packets sent: 1599
```

Fig. 3.6 Example of statistics given in output from the probe request generator.

3.4.3 Event-driven generator

Each device can assume various states and perform distinct actions, some of which may transition it to a new state. The orchestration of these actions is managed through a priority queue. In computer science, a priority queue is a specialized data structure that ensures quick access and removal of elements with the highest priority, which may vary depending on the specific task. This data structure is particularly useful in scenarios where tasks or items must be processed in order of their significance. In our case, the priority key is the execution time of events. Each action is associated with a specific time for its execution, a timing parameter determined within the generator based on the prevailing circumstances and information extracted from the generator's database.

The possible events available inside the queue are:

- **CreateDevice:** This event initiates the creation of a new device. The initial phase (e.g., locked, awake, or active) of the device is determined using a carefully chosen probability distribution, as depicted in Figure 3.7. This event is also responsible for generating other related events, such as *DeleteDevice*, *ChangePhase*, *CreateBurst*, and *CreateDevice*. To determine the vendor and model of the device to be generated, a specific reference was employed [61], which provides statistics on device sales and usage.
- **DeleteDevice:** This event removes the device from the emulation environment.
- **ChangePhase:** When triggered, this event selects the next device phase, based on the probabilistic distribution and the current phase. Subsequently, it schedules the next *ChangePhase* and *CreateBurst* events for the device. The primary goal is to emulate human behavior, and as such, the duration assigned to each phase is not a constant value, but instead adheres to a probability distribution. In particular, a negative exponential distribution is utilized to model these variable durations.
- **CreateBurst:** This event generates a series of *SendPacket* events in accordance with the device's specified number of packets per burst, as retrieved from the generator's database. It then creates a new *CreateBurst* event.

- **SendPacket:** This event contains the built probe request packet and saves it to the output *.pcap* file.

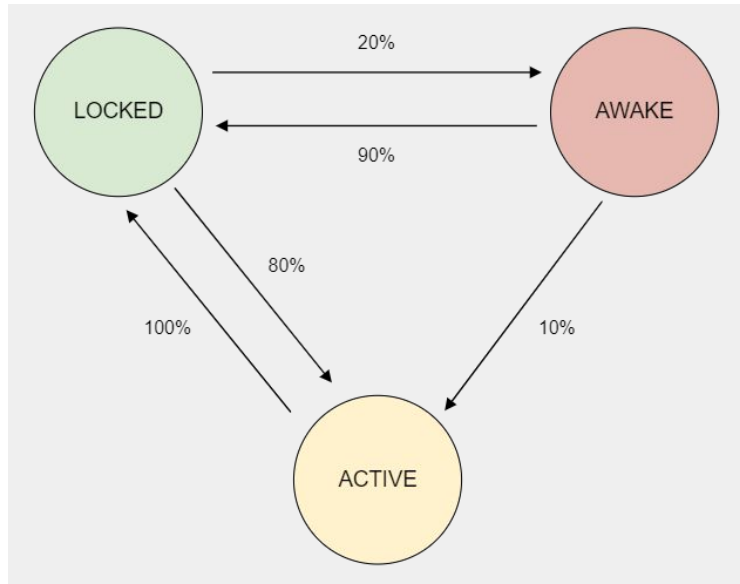


Fig. 3.7 Probability transition values diagram for the device's state.

The timing of each event is determined by the specific task at hand. For example, consecutive *SendPacket* events are spaced according to the device inter-packet time, while two *CreateBurst* events are spaced by the device inter-burst time. When a new event is created, it is added to the priority queue. Subsequently, the event list is sorted based on the execution times of the events, placing the event with the nearest execution time at the head of the queue.

It is important to highlight that the execution time for the next *CreateDevice* event is not a fixed value but is calculated dynamically when a device creation event is required. The mathematical principle utilized for this calculation is Little's Law [52]:

$$L = \lambda W \quad (3.3)$$

where:

L signifies the average number of customers or devices within the system at any given moment.

λ stands for the average rate at which customers or devices enter the system (measured in customers or devices per unit of time).

W represents the average time a customer or device spends within the system, also referred to as the average permanence time.

In essence, Little's Law articulates that the average number of entities present in the system is equivalent to the product of the average arrival rate and the average permanence time.

3.4.4 Messages collision avoidance

One of the primary challenges in packet trace generation is effectively managing the timing of message arrivals. This challenge becomes especially pertinent in densely populated environments, where the possibility of multiple probe requests from different devices reaching the sniffer simultaneously cannot be dismissed. Such occurrences, often referred to as *collisions* in real-world systems, require specialized recovery mechanisms or network segmentation to resolve.

Within the probe request generator, every packet is written directly to the *.pcap* file upon creation. To mitigate the potential for collisions and prevent the loss of packets, an additional feature has been incorporated into the system. Specifically, when a new *SendPacket* event is about to be added to the event list, a check is performed on other events of the same type to detect any possible time overlaps. If no collision is detected, the event is added to the list as usual. However, if a collision is identified, a mechanism is triggered to adjust their timing.

In this adjustment process, the execution time of the new event is advanced by a small increment of 0.001 seconds. This little adjustment is designed to exert minimal influence on the originally scheduled arrival time of the packet. Following this adjustment, the new event is inserted into the event list. This adjustment process ensures that subsequent packets within the ongoing burst are synchronized with the revised arrival time established for the preceding packet. Once these essential timing adjustments are made, the event is added to the generator's list of events.

3.4.5 Probe request packet generation

The *CreateBurst* event serves as a pivotal component, responsible for generating two distinct types of events:

- A specified number of *SendPacket* events, each separated by time intervals equal to the device's inter-packet time specific to the current phase;
- a new *CreateBurst* event, scheduled for execution within a time interval equivalent to the device's inter-burst time in the specific phase.

The number of *SendPacket* events created aligns with the chosen quantity of packets that compose the burst. This number may vary based on the probability distribution governing the number of packets per burst, which is device-specific. Each *SendPacket* event is tasked with constructing a probe request packet with predefined characteristics. It is noteworthy that the first *SendPacket* event in the sequence shares the same execution time as the *CreateBurst* event, effectively synchronizing the first packet's arrival time with the *CreateBurst* execution time. To facilitate the creation of network packets, a valuable Python library named Scapy [87] is employed. Scapy provides the essential tools to ensure compliance with the 802.11 standard, simplifying the process of constructing network messages.

MAC address

Each device within the generator's database is characterized by a flag that determines whether it implements MAC address randomization. When a device does not employ randomization, its MAC address is constructed using the specific vendor's Organizationally Unique Identifier (OUI) for the initial 24 bytes, while the remaining bytes are filled with random values. For models that utilize randomization, a new random MAC address is generated for each burst, encompassing all packets within that burst. This random MAC address consistently has its Globally/Locally bit set to 1, designating it as a locally administered address. Furthermore, each packet is transmitted with a destination MAC address of FF:FF:FF:FF:FF:FF, hence it is sent in broadcast.

802.11 probe request fields

Probe request headers encompass numerous fields, some of which vary between device models, while others depend on environmental factors. Yet, certain fields maintain relatively consistent values. Variations across different device models are documented and stored in the generator's database. These include VHT capabilities,

Extended capabilities, and HT capabilities. Fields influenced by environmental conditions, such as signal power, have their values randomly assigned within the packets. In contrast, for fields that remain constant across different probe requests, a default value is consistently used.

3.4.6 Validation

This Section will elucidate the methods utilized to validate the probe request generator described in the preceding Sections. It is noteworthy to highlight that we have provided an open-source implementation of the entire probe request generator to the research community [79], simplifying the creation of new datasets, making it a seamless and practical process.

The experiments conducted to validate the probe request generator were carried out within a simulated environment designed to emulate real-world conditions. These experiments were characterized by several key features:

- A fixed duration in terms of simulated time;
- a controlled environment with only one active device operating within the generator;
- a comparison between real traces and simulated ones to assess accuracy;
- separate analyses for each distinct phase, namely, locked, awake, and active phases;
- a 30 seconds observation window for all statistical measurements.

The main characteristic of the generator is to faithfully reproduce authentic traces by emulating device behavior in a manner consistent with real-world traces. To evaluate its performance, we have extracted various metrics from both real traces, sourced from [85], and simulated ones. These metrics include:

- The quantity of packets generated within the observation window;
- the time interval between packets within the same burst, referred to as inter-packet time;

- the time difference between the arrival of the first and last packet within the same burst, refereed to as burst duration;
- the count of packets per burst;
- the time elapsed between capturing the last packet of one burst and capturing the first packet of the subsequent burst, refereed to as inter-burst time;

To compare real traces with simulated ones, we assess the mean (computed using (3.4)) and the coefficient of variation (determined through (3.6)) for the five aforementioned metrics.

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (3.4)$$

More in detail, the coefficient of variation (CV) represents the ratio of the standard deviation (as defined in (3.5)) to the mean of a dataset.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (3.5)$$

$$CV = \frac{\sigma}{\bar{x}} \quad (3.6)$$

For every device within the generator’s dataset, we calculated the mean and coefficient of variation for each of the five metrics mentioned earlier. We repeated the emulation four times, each lasting three hours, with one hour dedicated to each phase. The results for the real trace metrics and the simulated metrics for all three phases are documented in Tables 3.2, 3.3, and 3.4, where we focus on the Apple iPhone 11 results. The entries in these tables are annotated using a set of acronyms:

- **PO:** Packet Occurrences.
- **IPT:** Inter Packet Time.
- **BL:** Burst Length.
- **PPB:** Packets Per Burst.
- **IBT:** Inter Burst Time.

Probe Request Generator and Privacy-Aware People Flow Monitoring through
Bloom Filters

Metric	Real
PO [occurrences]	(3.11, 2.52)
IPT [ms]	(20.38, 0.02)
BL [ms]	(16.3, 0.52)
PPB [occurrences]	(1.8, 0.23)
IBT [s]	(16.49, 1.62)

Metric	Simulation 1	Simulation 2	Simulation 3	Simulation 4
PO [occurrences]	(3.42, 1.33)	(3.32, 1.36)	(3.29, 1.29)	(3.25, 1.34)
IPT [ms]	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
BL [ms]	(16.73, 0.44)	(16.5, 0.46)	(16.76, 0.44)	(16.85, 0.43)
PPB [occurrences]	(1.84, 0.2)	(1.83, 0.21)	(1.84, 0.2)	(1.84, 0.2)
IBT [s]	(16.07, 1.59)	(16.48, 1.57)	(16.74, 1.56)	(17.01, 1.55)

Table 3.2 Locked phase results for an Apple iPhone 11 (mean, coefficient of variation).

Metric	Real
PO [occurrences]	(7.54, 1.46)
IPT [ms]	(20.33, 0.02)
BL [ms]	(17.94, 0.37)
PPB [occurrences]	(1.88, 0.17)
IBT [s]	(7.2, 2.32)

Metric	Simulation 1	Simulation 2	Simulation 3	Simulation 4
PO [occurrences]	(7.46, 1.15)	(7.74, 1.18)	(7.79, 1.13)	(7.42, 1.11)
IPT [ms]	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
BL [ms]	(16.68, 0.45)	(16.55, 0.46)	(16.66, 0.45)	(16.5, 0.46)
PPB [occurrences]	(1.83, 0.2)	(1.83, 0.21)	(1.83, 0.2)	(1.83, 0.21)
IBT [s]	(7.35, 2.26)	(7.06, 2.33)	(7.04, 2.3)	(7.36, 2.27)

Table 3.3 Awake phase results for an Apple iPhone 11 (mean, coefficient of variation).

Based on the obtained results, it is evident that the traces generated by the probe request generator closely resemble the real traces in terms of mean and coefficient of variation for the metrics. This indicates that our probe request generator is able to produce realistic traces with a very high accuracy.

The probe request generator aims to create a varied dataset of *.pcap* traces with associated ground truth for training and testing ML algorithms for enhancing the accuracy of people counting. Initial tests involved training a three-layer neural

Probe Request Generator and Privacy-Aware People Flow Monitoring through
Bloom Filters

Metric	Real
PO [occurrences]	(11.16, 1.06)
IPT [ms]	(20.27, 0.02)
BL [ms]	(16.6, 0.47)
PPB [occurrences]	(1.82, 0.21)
IBT [s]	(4.81, 2.41)

Metric	Simulation 1	Simulation 2	Simulation 3	Simulation 4
PO [occurrences]	(10.47, 0.89)	(10.85, 0.91)	(10.87, 0.89)	(10.71, 0.85)
IPT [ms]	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
BL [ms]	(16.4, 0.47)	(16.85, 0.43)	(16.62, 0.45)	(16.61, 0.45)
PPB [occurrences]	(1.82, 0.21)	(1.84, 0.20)	(1.83, 0.2)	(1.83, 0.21)
IBT [s]	(5.2, 2.39)	(5.08, 2.42)	(5.03, 2.38)	(5.11, 2.27)

Table 3.4 Active phase results for an Apple iPhone 11 (mean, coefficient of variation).

network using synthetic and real-world data. While the synthetic data yielded promising results with an MSE range of 0.5 to 2, testing with real traces showed increased MSE, indicating limitations as discussed in Section 2.6.5. To address this, future works will focus on diversifying the dataset with more real-world traces and refining the generator to better emulate real-world complexities, aiming to enhance the reliability of ML algorithms for people counting.

It is worth mentioning that extending the dataset of devices available within the probe request generator is relatively straightforward. This is why we have publicly released the code of the generator, the dataset, and the methodology used to collect the data on GitHub [79]. This release enables future extensions by adding new devices with different operating systems, fostering community contributions and improvements to the tool. This collaborative approach ensures the continual enrichment and relevance of the generator, aligning with the evolving landscape of mobile device technologies.

3.5 Bloom filters for flow analysis

In this Section, our primary focus revolves around the Bloom filter and the introduction of the *anonymization noise*. This addition ensures the privacy of stored MAC addresses while still allowing for the intersection of different Bloom filters for flow analysis. We cover various aspects of this topic in the following subsections. Section 3.5.1 outlines the proper sizing of a Bloom filter, taking into account the specific application use case. Section 3.5.2 provides an overview of the privacy properties that can be applied to Bloom filters. Section 3.5.3 and Section 3.5.4 elaborates on our proposed solution for guaranteeing the privacy of elements stored in the Bloom filter, aligning with the principles of GDPR (General Data Protection Regulation). Lastly, in Section 3.5.5 and 3.5.6 we delve into the counting process, elucidating how we can precisely determine the number of elements inserted in a Bloom filter and efficiently compute the intersection between multiple Bloom filters.

Table 3.5 provides a comprehensive summary of the notations and definitions that will be utilized throughout the entire Section.

Notation	Definition
U	Set of elements in the universe
S	Set of elements stored in the Bloom filter
$BF(S)$	Bloom filter storing a set S
$n = S $	Number of elements stored in the Bloom filter
m	Size in bit of the Bloom filter
k	Number of hash functions
t_i	Number of bits set to 1 in the Bloom filter BF_i
V	Hiding Set

Table 3.5 Definitions and notations for Bloom filters.

3.5.1 Bloom filter sizing

To size a Bloom filter effectively, an analysis of the usage context is essential. This analysis involves making appropriate assessments to determine the optimal number of bits and hash functions to use, all while ensuring that the probability of false positives remains below an acceptable threshold. Failing to do so can lead to one of these two scenarios:

- A Bloom filter that is too small will result in an elevated probability of false positives, rendering it ineffective.
- A Bloom filter that is too large will lead to space wastage, a critical concern in resource-constrained contexts.

To minimize the occurrence of false positives, it is possible to compute the optimal value of k based on the available memory m . This computation is done in a way that achieves a desired probability of false positives. The selection of the optimal k depends on factors such as the number of elements to be stored in the Bloom filter and the acceptable false positive rate. The equation for determining the optimal value of k is presented in (3.7).

$$k_{opt} = \frac{m}{n} \log(2) \quad (3.7)$$

It is important to note that:

- A small value of k increases the fraction of 0 bits in the array, making them available for elements that are not part of the set S .
- A large value of k enhances the probability of finding at least one 0 bit for an element that is not a member of S .

Let us consider an illustrative use case: we aim to capture WiFi probe requests within a 120-second time window, with the goal of accommodating roughly 1,000 insertions into the Bloom filter. This implies that we anticipate detecting a maximum of 1,000 distinct MAC addresses within the defined time frame. Additionally, we set the size of the Bloom filter to 10,000 bits. Once we have fixed the values of n and m , through (3.7), we can ascertain that the optimal value for k , which denotes the number of hash functions, is 7.

To confirm the result, we showed in Figure 3.8 the variation of the false positive probability in relation to the change of parameter k . For the creation of this graph (3.2) was utilized, where parameters m remained fixed at 10,000, and n was maintained at 1,000. As it can be observed, the point of minimum is obtained for $k = 7$.

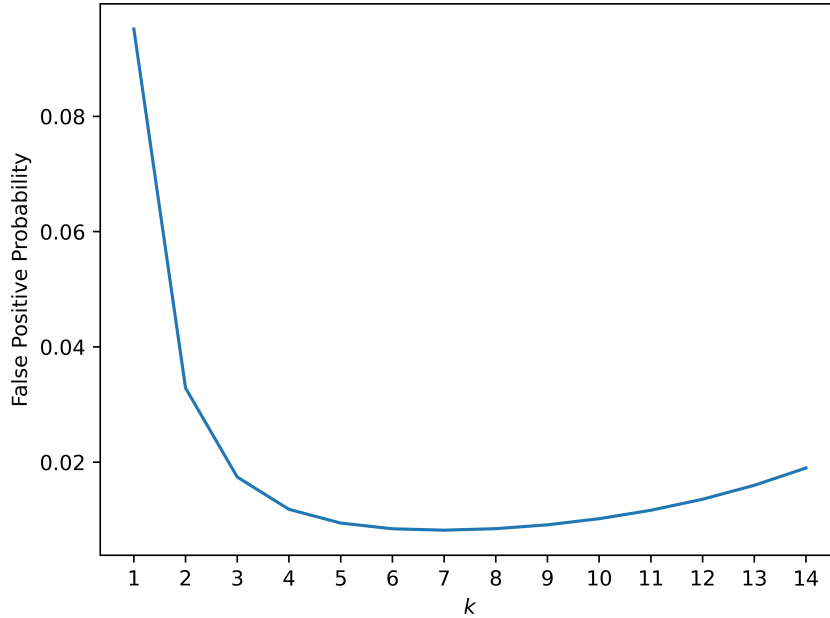


Fig. 3.8 False positive probability with different values of k .

Figure 3.9 illustrates the evolution of the false positive probability in a Bloom filter configured with $k = 7$ and $m = 10,000$ during the insertion of elements. Notably, the false positive probability remains consistent up to 1,000 insertions, after which it enters a phase of increasing consistency.

In Figure 3.10, we present the results of an analysis that explores the impact of the parameter k on the probability of false positives. Using a Bloom filter with m set to 10,000, we chart the trend of the false positive probability in relation to the number of MAC addresses while varying the value of k . Notably, it is evident that higher values of k necessitate fewer insertions to attain an equivalent probability of false positives.

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters

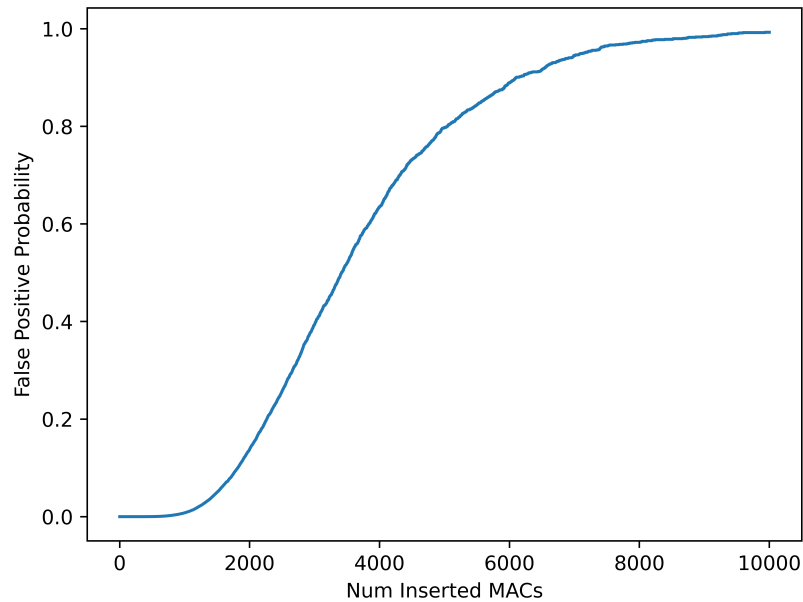


Fig. 3.9 False positive probability vs. number of inserted MAC addresses into a Bloom filter.

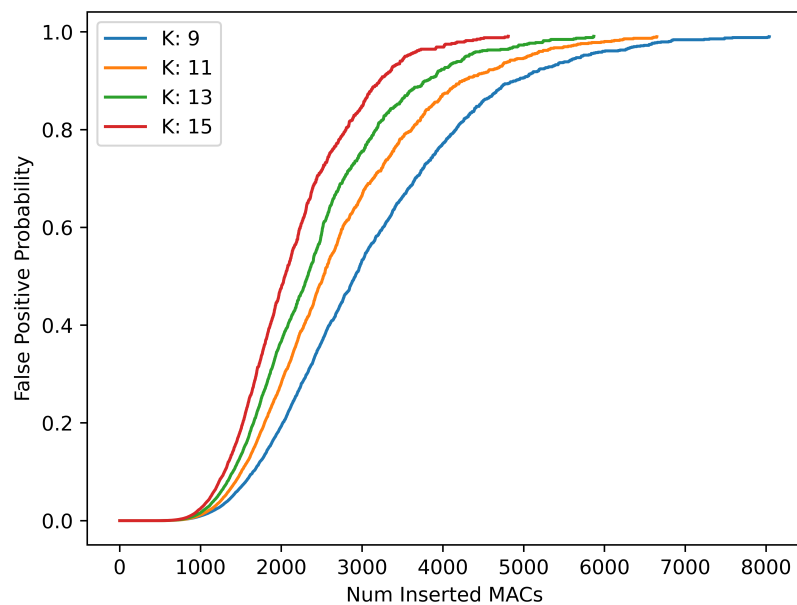


Fig. 3.10 False positive probability vs. number of inserted MAC addresses into different Bloom filters. Each Bloom filter is configured with $m = 10,000$ and a different values of k .

3.5.2 Bloom filter privacy properties

In this Section, we introduce three primary privacy definitions that are applied to Bloom filters. These definitions, proposed by [21], serve as the foundation for the *anonymization noise* discussed in Section 3.5.3.

Definition 1

A set V is called *Hiding Set* for a Bloom filter $BF(S)$ if V contains all the elements $v_i \in U$ s.t. $v_i \notin S$ and a query for v_i in the Bloom filter returns 1. In other words, a hiding set is a set of elements not present in the filter’s stored set S that, when queried, falsely indicate their presence in the filter, which is often referred to as a “false positive”.

Equation 3.8, proposed in [21], allows the calculation of the cardinality of the hiding set V , represented by the random variable N_v , using a binomial probability distribution

$$P\{N_v = v\} = \binom{|U| - n}{v} \psi(m, k, n)^v (1 - \psi(m, k, n))^{|U| - n - v} \quad (3.8)$$

and mean value $E[N_v] = (|U| - n)\psi(m, k, n)$.

Figure 3.11 illustrates an example of a “Hiding Set” where a 10-bit Bloom filter, employing 2 hash functions, has been used to insert 3 elements x_1, x_2, x_3 . Simultaneously, 3 elements v_1, v_2, v_3 appear as false positives because a query to the filter would incorrectly yield a positive result for these elements.

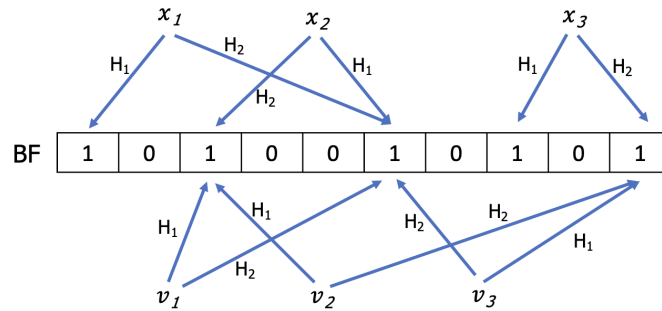


Fig. 3.11 Elements v_1, v_2, v_3 belonging to the hiding set of a 10-bit BF, storing x_1, x_2, x_3 . Arrows indicate the bits that are set to one according to the two hash functions H_1 and H_2 .

Definition 2

An element $x \in S$ inserted in $BF(S)$ is defined *deniable* if $\forall i \in \{1..k\}$ exist at least one element $v \in V$, such that $\exists j \in \{1..k\}$ s.t. $H_i(x) = H_j(v)$. A $BF(S)$ is γ -deniable whenever a randomly chosen element $x \in S$ is deniable with probability γ .

In other words, an element is considered deniable if it can be replaced with items that were not originally part of the Bloom filter's stored set, all while maintaining the integrity of the bit map without any changes.

Using (3.9), introduced in [21], it is possible to calculate the size of the hiding set.

$$\gamma(BF(S)) \simeq \left(1 - \exp\left(\frac{hk}{m(1 - e^{-kn/m})}\right) \right)^k \quad (3.9)$$

with

$$h = (|U| - n)\psi(m, k, n) = (|U| - n)(1 - e^{-kn/m})^k \quad (3.10)$$

depicting the mean number of elements in the hiding set.

Definition 3

Considering a Bloom Filter $BF(S)$ and $x \in S$ inserted in $BF(S)$, x is K -Anonymous if exists at least $K - 1$ hiding set elements $\langle v_1 \dots v_{K-1} \rangle \in V$, with $K \geq 2$, such that $\forall i \in \{1 \dots k\} \exists \langle j_1 \dots j_{K-1} \rangle \in 1..k$ s.t. $H_i(x) = H_{j_1}(v_1) = \dots = H_{j_{K-1}}(v_{K-1})$. Hence, it is accurate to state that a Bloom filter $BF(S)$ is γ - K -anonymous if, for each randomly chosen element, there is a probability γ that is K -anonymous.

Equation (3.11), extracted from [21], provides a method to compute the γ - K -anonymity for a particular Bloom filter configuration.

$$\gamma(K, BF(S)) \simeq \left(1 - \exp\left(-\frac{hk}{m(1 - e^{-kn/m})}\right) \sum_{i=0}^{K-2} \frac{\left(\frac{hk}{m(1 - e^{-kn/m})}\right)^i}{i!} \right)^k \quad (3.11)$$

3.5.3 Anonymization noise

In light of the definitions introduced in Section 3.5.2, we propose the utilization of the γ -deniability property to ensure the ability to deny the membership of all inserted elements in a Bloom filter. However, as delineated in Section 3.5.2, γ signifies the likelihood of being able to disclaim the presence of a randomly chosen element using another element that is not genuinely part of the stored set. To establish this capability for all elements within the Bloom filter, it is imperative for γ to attain its maximum value, which is 1. This ensures that any inserted element can be unequivocally denied.

Nonetheless, at the outset, the value of γ is equal to 0. In fact, when the Bloom filter is entirely empty and the configuration parameters m and k , which are constants, are disregarded, the value of n (equal to 0) renders (3.10) ineffective, consequently setting the value of (3.9) to 0. To elevate the value of γ it becomes imperative to insert elements into the Bloom filter, and after a certain number of insertions, γ will eventually reach 1.

However, to safeguard privacy, it is impractical to employ MAC addresses captured before γ attains its maximum value. Therefore, we introduce the concept of *anonymization noise*. This noise comprises n_{min} randomly generated elements, which are exempt from privacy constraints, and are inserted as soon as the Bloom filter is created to expedite the achievement of a γ value of 1.

Let us revisit the same example previously discussed in Section 3.5.5, where we have $m = 10,000$ and $k = 7$. Starting with an empty filter, we began inserting one random MAC address at a time and subsequently calculate the value of γ , using (3.9). The outcome, as depicted in Figure 3.12, reveals that for the adopted Bloom filter configuration, approximately 30 insertions are adequate to reach a γ value of 1.

In this scenario, to adhere to privacy standards, when a new Bloom filter is created, it can be promptly filled with a minimum of 30 random MAC addresses. This approach guarantees the privacy of insertions without the necessity of relying on cryptographic functions, which could potentially strain the system and introduce additional complexities associated with the safeguarding and administration of encryption keys.

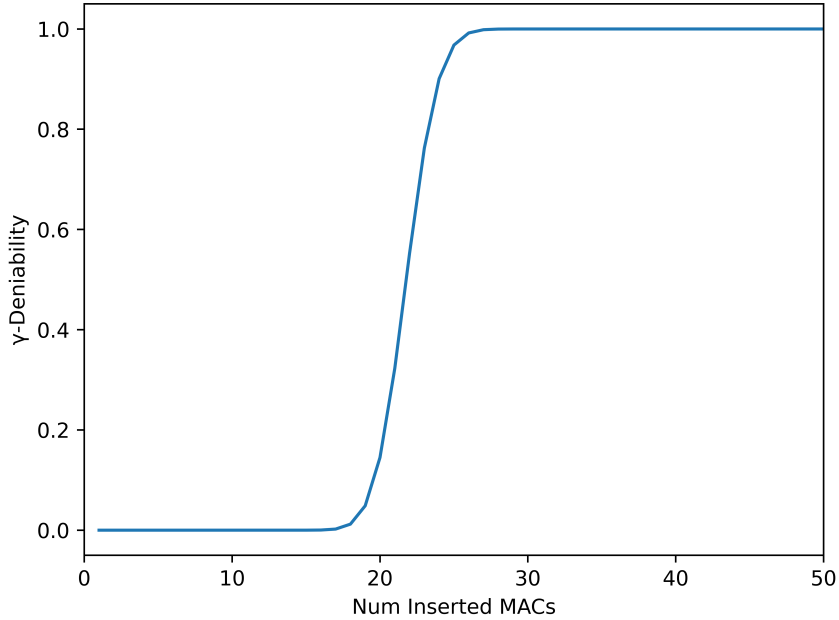


Fig. 3.12 γ -deniability value vs. number of inserted MAC addresses into a Bloom filter.

3.5.4 Multiple anonymity

The concept of *anonymization noise*, as introduced in Section 3.5.3, ensures that there is always at least one element that is not part of the Bloom filter, covering an actually inserted element.

To further enhance the provided protection, we can leverage the definition outlined in Section 3.5.2. By selecting a specific value for K and optimizing (3.11) to reach its maximum value (i.e., 1), it is possible to achieve coverage of at least $K - 1$ elements that were not actually inserted, rather than just a single element. In this context, like in the earlier scenario described, it remains crucial to employ *anonymization noise* to ensure that from the very first insertion of detected MAC addresses, each of them is shielded by at least $K - 1$ additional elements.

In a manner akin to the example presented in Section 3.5.3, once a specific value for K is determined, it becomes imperative to define the value of n_{min}^K in a way that, for each element randomly drawn from the filter, there are $K - 1$ non-inserted elements available to provide cover. Figure 3.13 illustrates the outcome of applying (3.11) to a Bloom filter with $m = 10,000$ and $k = 7$ for various values of K .

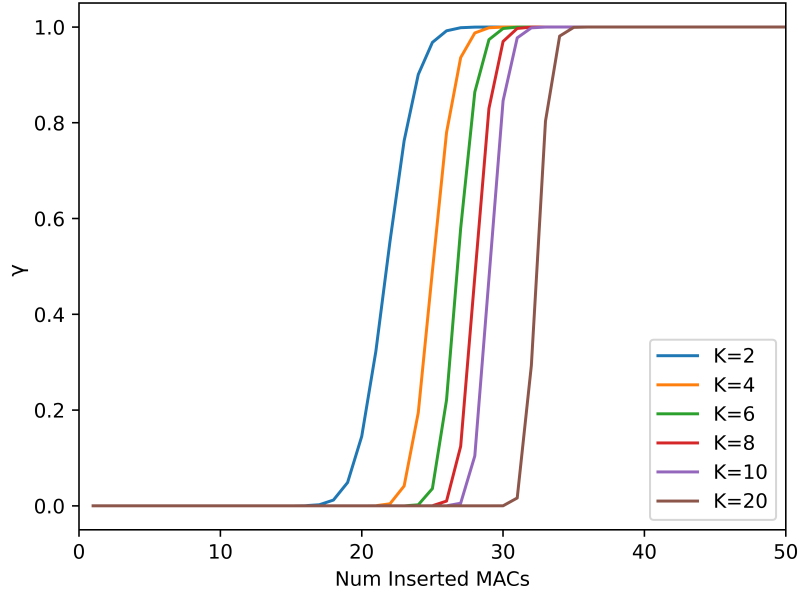


Fig. 3.13 γ -deniability value vs. number of inserted MAC addresses into a Bloom filter, for different values of K .

3.5.5 Counting elements stored in a Bloom filter

In this Section, we focus on the data extraction from the Bloom filter and in detail on how to count the number of inserted elements in a Bloom filter.

The task of counting elements within a Bloom filter can be accomplished using the well-known formula (3.12), derived from [22].

$$c_1 = -\frac{m}{k} \ln \left(1 - \frac{t}{m} \right) \quad (3.12)$$

where m is the size of the filter, k the number of hash functions and t the number of bits set to one.

Let us revisit the same example involving a Bloom filter with $m = 10,000$ and $k = 7$. Figure 3.14 illustrates the accuracy of counts derived from (3.12) when a total of 10,000 elements are inserted into the Bloom filter. It is worth noting that this number significantly surpasses the originally intended value of $n = 1,000$. Following each insertion, (3.12) yields a count of the contained elements, which is then compared to the actual number of elements present. As depicted in Figure 3.14,

the count is not notably overestimated or underestimated, despite the fact that the number of insertions exceeds the initial design value for the filter.

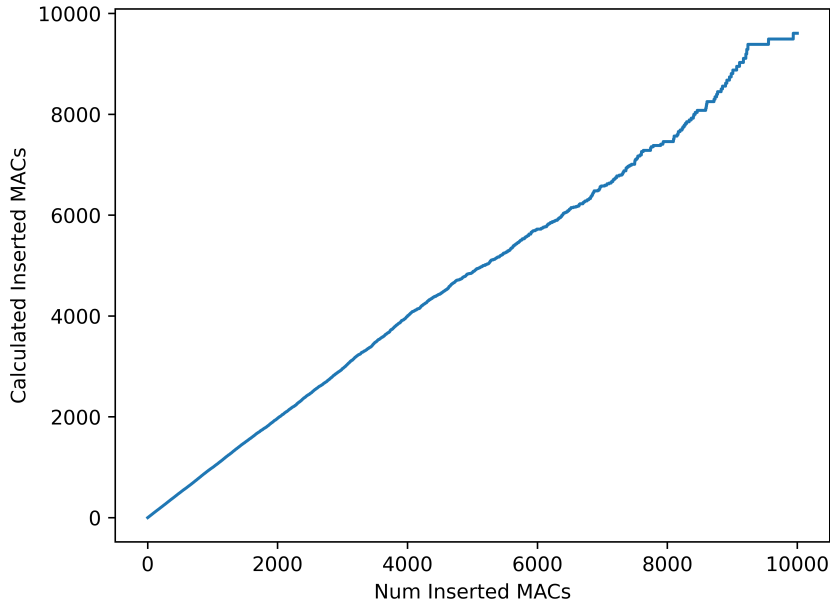


Fig. 3.14 Accuracy evaluation of (3.12) when comparing number of elements inserted in the Bloom filter and counted ones.

We will now delve into the counting procedure within a Bloom filter while maintaining 1-deniability through the introduction of the anonymization noise. The pseudocode in Algorithm 1 illustrates the essential steps involved, encompassing Bloom filter initialization, insertion, and count estimation.

To elaborate further, at the onset of each iteration, the Bloom filter undergoes initialization, where all bits are set to 0. Subsequently, random noise, comprised of n_{min} “fake” MAC addresses, is introduced into the Bloom filter. Importantly, leveraging the properties of independent hash functions employed in Bloom filters, an efficient implementation of this step necessitates generating $n_{min} \times k$ random positions in the Bloom filter. This equivalence arises from adding n_{min} elements with k hash functions. Following this initialization phase, with each newly detected probe request, the MAC address is incorporated into the filter by setting the corresponding bits to 1. The indices of these bits are determined by the output of the k hash functions applied to the MAC address. Upon the conclusion of the capturing window, the initial operation involves counting the number of ones in the Bloom filter. Subsequently,

Algorithm 1 Counting algorithm with anonymization noise

Input: Bloom filter (BF) of size m bit, Number of elements c_{\min} for the anonymization noise, k independent hash functions

Output: Estimated count of elements inserted

```

procedure RESET( $BF, c_{\min}$ )
  for  $i = 1, \dots, m$  do
     $BF[i] = 0$  ▷ Reset each bit
  end for ▷ Add the anomization noise
  for  $c = 1, \dots, c_{\min} \times k$  do
     $i = \text{random-int}(1, m)$  ▷ Choose a random bit to set
     $BF[i] = 1$  ▷ Update the bit
  end for
end procedure

procedure INSERT( $BF, \text{mac}$ )
  for  $i = 1, \dots, k$  do
     $BF[H_i(\text{mac})] = 1$  ▷ Set bit to 1 in the  $H_i$  index
  end for
end procedure

procedure COUNT( $n_{\min}$ )
   $t = 0$  ▷ Init  $t$ 
  for  $i = 1, \dots, m$  do
    if  $BF[i] = 1$  then
       $t = t + 1$  ▷ Count number of 1 in  $BF$ 
    end if
  end for
   $c = -\frac{m}{k} \log\left(1 - \frac{t}{m}\right)$  ▷ Apply (3.12)
  return  $c - n_{\min}$  ▷ Compensate for the anonymization noise
end procedure

```

equation (3.12) is applied to estimate the number of elements present in the filter. Ultimately, the estimated value undergoes subtraction of a n_{\min} value, accounting for the anonymization noise introduced at the beginning.

3.5.6 Intersection of different Bloom filters

Monitoring crowd flows across various scanners becomes possible by recognizing shared MAC addresses within different Bloom filters. This involves the intersection of these Bloom filters and the extraction of valuable count information.

Let us consider two subsets, $S1$ and $S2$, which are represented by their respective Bloom filters, denoted as $BF(S1)$ and $BF(S2)$. These Bloom filters share the same configuration parameters. To find the intersection between them, a bitwise logical AND operation is necessary, resulting in a combined Bloom filter $BF(S3)$. From this intersection Bloom filter, we can calculate the number of elements it contains using two distinct equations. The first formula, expressed in (3.12), can be applied replacing t with t_3 . The second formula, proposed in the work [67] and presented in (3.13), offers an alternative approach for counting.

$$c_2 = \frac{\ln\left(m - \frac{t_3 \times m - t_1 \times t_2}{m - t_1 - t_2 + t_3}\right) - \ln(m)}{k \times \ln\left(1 - \frac{1}{m}\right)} \quad (3.13)$$

where t_1 denotes the count of bits set to 1 in BF_1 , t_2 signifies the number of bits set to 1 in BF_2 , and t_3 represents the count of bits set to 1 in BF_3 .

Our objective is to facilitate anonymous tracking and counting of devices passing through and detected by various scanners. To achieve this, we conducted a series of comparisons to determine the accuracy of the proposed formulas, namely (3.12) and (3.13).

Similar to the examples proposed in the previous Sections we used two Bloom filters, namely BF_1 and BF_2 , with the configuration of $m = 10,000$ and $k = 7$. Our experiment began with the two initially empty Bloom filters, and we proceeded through the following steps:

1. Insert 500 random MAC addresses into BF_1 .
2. Insert 500 random MAC addresses into BF_2 .
3. Insert 1 identical MAC address into both BF_1 and BF_2 .
4. Compute a new Bloom filter BF_3 , by finding the intersection of BF_1 and BF_2 .
5. Count the elements within this intersection using both Equations 3.12 and 3.13.
6. Iterate again from step 3 until 500 MAC addresses are commonly inserted.

In Figure 3.15, the results obtained from our experiment are displayed, allowing us to compare the two equations. Notably, the two curves exhibit a nearly parallel

relationship. However, it is worth mentioning that (3.12) consistently tends to overestimate the count of elements within the intersection.

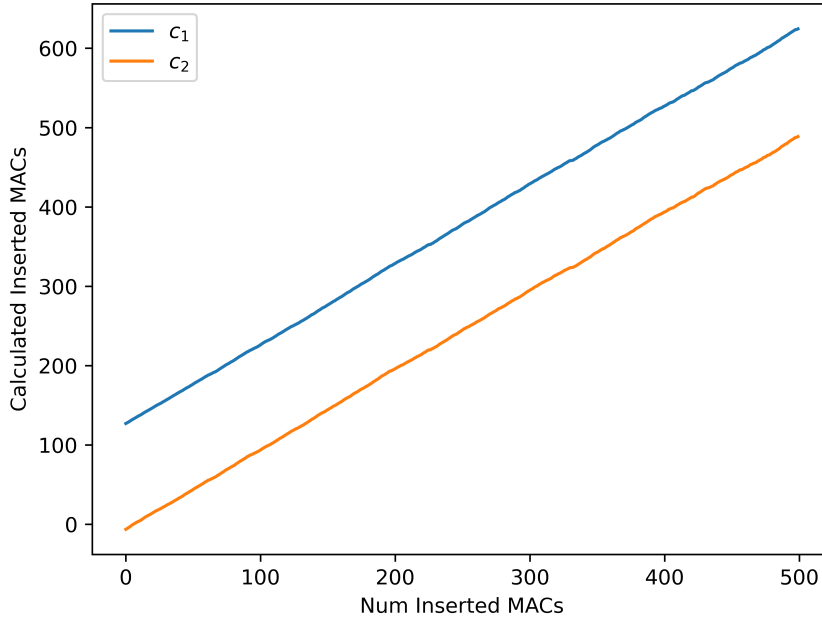


Fig. 3.15 Comparison of flow monitoring accuracy between (3.12) denoted as c_1 and (3.13) denoted as c_2 .

To further validate our earlier observations, Figure 3.16 provides a comparison of the relative errors associated with the two equations. Notably, the relative error for c_1 is significantly higher when dealing with a smaller number of stored MAC addresses. It only becomes acceptable (falling below 100%) when more than 100 MAC addresses are stored. In contrast, the relative error of c_2 remains consistently low, even with a small number of stored MAC addresses. Indeed, this observation is in line with the findings in [67]. It underscores a distinct trade-off between accuracy, complexity, and privacy when calculating the count of MAC addresses within the intersection of BF_1 and BF_2 .

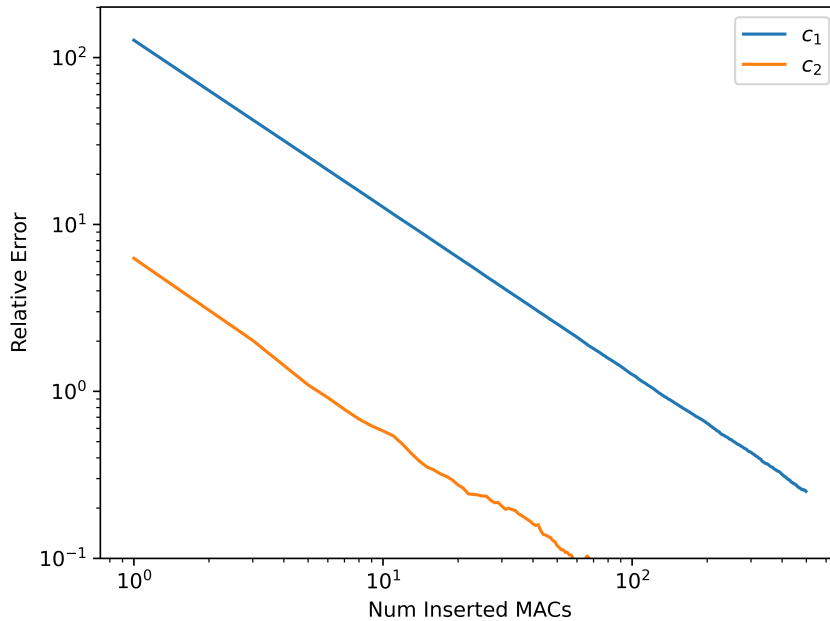


Fig. 3.16 Comparison of flow monitoring relative errors between c_1 and c_2 .

3.6 Conclusions and future works

Given the stringent European General Data Protection Regulation and the proactive measures adopted by leading smart device manufacturers, like the implementation of MAC randomization, counting the number of people in an area or event track people movements in various settings has grown increasingly challenging. This challenge not only pertains to the complexity of developing effective algorithms but also to the critical aspect of ensuring privacy.

Within the scope of this study, we have developed a sophisticated WiFi traffic generator that accurately simulates the behavior of actual devices, particularly in generating probe request messages. This generator possesses the capability to generate realistic traffic patterns that span several minutes in a matter of seconds for a single device. Furthermore, it allows for the concurrent emulation of multiple devices, even in the hundreds. This achievement holds significant practical importance as it facilitates the creation of a ground truth scenario, where the number of devices is known in advance. This environment becomes instrumental in evaluating the accuracy of different counting algorithms based on probe requests.

Probe Request Generator and Privacy-Aware People Flow Monitoring through Bloom Filters

To address privacy concerns, we have leveraged Bloom filters in our approach, which can guarantee the 1-deniability property, thanks to the introduction of the anonymization noise. We have established a comprehensive methodology for configuring this noise effectively. Additionally, we have assessed the impact of various formulas, drawn from existing literature, for estimating the number of individuals transitioning from one WiFi scanner to another. This estimation is performed while ensuring that MAC addresses remain protected within privacy-preserving Bloom filters.

The combination of these two contributions sets the stage for the development of innovative counting algorithms that can effectively deal with MAC randomization and the associated privacy considerations. Detailed analysis of these aspects can be found in Chapter 4, providing insights and solutions for addressing these contemporary challenges.

Chapter 4

People counting and crowd monitoring in real use cases

The COVID-19 pandemic and its aftermath have underscored the critical need for monitoring and ensuring community safety by detecting gatherings of people and accurately counting individuals in transit areas. In this context, the role of Internet of Things (IoT) devices has become indispensable. Safety applications and services tailored for the convenience of mobile users now rely heavily on the ability to detect and analyze patterns in people's movements at various times of the day and week, as well as assessing the density of individuals in specific areas. This Chapter introduces two comprehensive solutions deployed in distinct use cases and environments. The knowledge acquired from these applications, coupled with the evolving privacy measures related to MAC address randomization, has paved the way for the in-depth investigation documented in Chapter 2 and Chapter 3. This research culminates in an enhanced version of the crowd monitoring solution. Leveraging advanced machine learning techniques, this solution surpasses the performance of state of the art counting algorithms while remaining fully compliant with GDPR regulations.

4.1 Research motivation

In November 2018, the Italian National Institute of Statistics (ISTAT) conducted a comprehensive study on the daily commuting habits of Italians [15]. Leveraging data from the preceding year's national census, this research shed light on the preferences

and choices of Italian commuters when it comes to transportation. The findings from this investigation revealed a multifaceted landscape: more than 40% of the Italian population opted not to use private transportation as their primary means of daily travel. Instead, 19% of the population preferred the eco-friendly alternatives of walking or cycling, while 23% relied on public transportation options, including buses, trams, trains, school buses, and subways.

Additionally, in May 2022, ISTAT released another noteworthy study [16], shedding light on the shifting trends in transportation preferences following the COVID-19 pandemic. The findings revealed a remarkable transformation in commuter behavior. Specifically, more than 80% of the Italian population indicated a preference for private cars as their primary mode of transportation. However, this preference started to wane in the subsequent months, paralleling the decline in the risk of COVID-19 infection and the concurrent escalation in fuel prices.

These statistics underscore the adaptive nature of the Italian populace in response to the pandemic. Before the outbreak of COVID-19, many Italians relied on non-private forms of transportation for their daily commutes. However, the pendulum has swung back towards private car usage as people sought to minimize potential health risks. As the situation has improved and concerns over infection have diminished, the pendulum is once again shifting, this time toward a resurgence in the utilization of public transportation. The driving force behind this shift lies in the rising costs associated with private car ownership and operation, emphasizing the significant impact of economic factors on transportation choices. This dynamic transformation in transportation preferences calls for adaptable and data-driven urban planning to cater to the evolving needs and concerns of the Italian population. To maintain a high level of service and uphold the ideals of a smart city, continuous and precise monitoring of public transportation users, as well as other modes of transportation, is an imperative.

As we have discussed in earlier Chapters and corroborated by existing literature, numerous solutions have been proposed to address the challenge of people counting. Despite the wealth of ideas and technologies explored, the majority of these solutions remain in the realm of Proof of Concept (PoC) or theoretical frameworks. Only a handful of them have transitioned into practical implementations, often tailored to specific use cases. This underscores the need for further research and the development

of tangible real-world applications, to effectively monitor and enhance transportation services in an era of growing urbanization and mobility.

4.2 Main contributions

The present study, as detailed in [43, 82], introduces two initial solutions and an innovative framework that utilizes machine learning techniques and probe request messages to monitor crowds while prioritizing privacy preservation. Firstly, one of our implemented solutions focuses on quantifying the number of individuals within the coverage area of six commercial access points, along with tracking the flows between them. This monitoring is achieved through the deployment of a 5G infrastructure in the “innovation mile” of Turin, providing valuable insights into crowd dynamics over a span of more than two years and the detection of over 100 million probe requests. In response to the unique challenges posed by the COVID-19 pandemic, our second solution was designed and deployed on a bus to ascertain the number of passengers onboard. Additionally, it offers real-time information to the bus driver, aiding compliance with strict Italian government regulations regarding maximum indoor/closed space occupancy during the pandemic. Building on the knowledge garnered from these initial projects and our comprehensive study of probe request behavior, we present a wholly redesigned solution. Our framework harnesses probe request messages to address the challenges of MAC address randomization and GDPR-induced privacy concerns. It integrates machine learning techniques and employs a Bloom filter data structure while upholding the essential 1-deniability property.

Our work significantly contributes to the existing body of literature in the following key areas:

- **Real-time presence sensing on a 5G infrastructure:** We offer a robust people counting and crowd monitoring solution using commercial scanners in Turin’s “innovation mile”. This solution provides valuable insights by analyzing more than two years of data collection, comprising over 100 million detected probe requests.
- **Passive crowd monitoring inside a bus:** We conduct experiments on people counting inside a bus, leveraging not only probe request messages but also

additional data from the bus's infonet to enhance people detection within the boundaries of a bus. This has proven especially valuable during the COVID-19 pandemic.

- **Machine learning driven privacy-preserving framework for crowd management:** Our enhanced framework addresses the challenges of MAC address randomization and privacy concerns by integrating advanced machine learning techniques and the Bloom filter data structure, fortified with anonymization noise. This approach represents a significant stride towards safeguarding user privacy in crowd monitoring applications.

In the upcoming Sections, our attention turns to the various WiFi sniffers at our disposal for detecting probe request messages, where we dissect the principal advantages and disadvantages of each. Subsequently, we embark on a comprehensive exploration into the array of de-randomization techniques existing in literature, shedding light on their efficacy and applicability. Moving forward, we transition to a discussion of the two implemented solutions, offering an in-depth analysis of their outcomes. These solutions have been instrumental in our ongoing involvement in a European-funded project, where a novel framework is currently under development and in the testing phase. This framework represents our commitment to advancing the state of the art in the field. Lastly, we draw our exploration to a close, summarizing the key findings and insights gained throughout the study.

4.3 WiFi probe request sniffers

In this Section, we introduce two distinct WiFi probe request sniffing devices, delving into the software options available and highlighting their primary strengths and weaknesses. In Section 4.3.1, we explore the commercial access point known as Meshlium, developed by Libelium [59]. Additionally, in Section 4.3.2, we delve into the more cost-effective Raspberry Pi [74] solution.

4.3.1 Meshlium scanner by Libelium

The Meshlium scanner, is a commercial WiFi and Bluetooth probe request detector developed by Libelium [59], boasts several noteworthy features. Encased in a rugged IP67 waterproof housing, it is well-suited for outdoor applications. The device offers versatile connectivity with both an Ethernet port, facilitating Power over Ethernet (PoE) for system power, and a nano SIM slot for 3G/4G cellular connectivity.

In terms of antennas, it is equipped with two dipole antennas rated at 5 dBi, used for Bluetooth, BLE, WiFi, and Xbee-PRO 802.15.4 applications, as depicted by the white antennas in Figure 4.1. Additionally, it includes a 4.5 dBi Dipole antenna, suitable for XBee 868LP and XBee-PRO 900HP, along with two antennas designed for 4G/GPS functionality.

About hardware specifications, Meshlium scanner features a 1 GHz Quad-Core (x86) processor, 2 GB of DDR3 unified memory, and 16 GB of disk storage. It operates on a Debian-based Linux operating system. Furthermore, it offers a wide operating temperature range, from -20 degrees Celsius to 50 degrees Celsius, and when housed in its enclosure, it has a weight of approximately 2.2Kg.



Fig. 4.1 Meshlium scanner by Libelium [59].

4.3.2 Raspberry Pi

The Raspberry Pi, often referred to as Raspi [74], is a single-board computer developed by Raspberry Pi Ltd in collaboration with Broadcom. While there are various Raspberry Pi models with varying specifications, we will focus on the most powerful iteration to date, which is the Raspberry Pi 4B.

The Raspberry Pi 4B boasts impressive hardware components compared to the low cost price, featuring a 1.8 GHz Quad-core Cortex-A72 (ARM v8) 64-bit processor produced by Broadcom. Depending on the model, it can be configured with up to 8 GB of LPDDR4 unified memory. The available disk space depends on the micro SD card used, with support for micro SD cards offering more than 1TB of storage capacity.

In terms of connectivity, the Raspberry Pi 4B is well-equipped, providing two USB 2.0 ports and two USB 3.0 ports for versatile peripheral connections. It also includes two micro-HDMI ports for video output, offering great flexibility. Wireless connectivity options encompass support for both 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless standards, in addition to Bluetooth 5.0 and BLE (Bluetooth Low Energy). Furthermore, it is equipped with a Gigabit Ethernet port that can also support Power over Ethernet (PoE) for streamlined power delivery, along with a USB-C connector. For a visual reference, Figure 4.2 displays an image of the Raspberry Pi 4B.

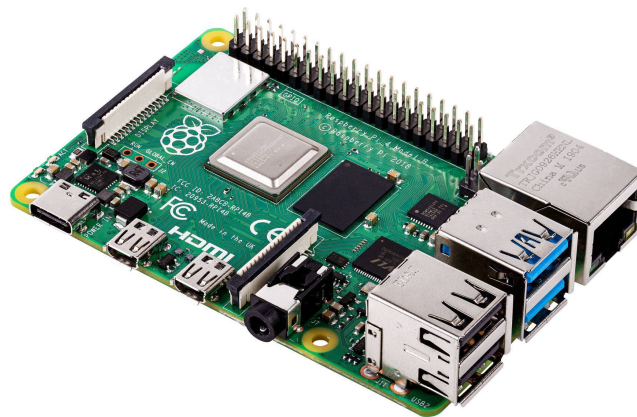


Fig. 4.2 Raspberry Pi 4B [74].

4.3.3 Meshlium vs Raspberry Pi

Table 4.1 provides an insightful comparison of the advantages and disadvantages of Meshlium by Libelium and the Raspberry Pi 4B, aiding in making an informed choice between these two compelling solutions for various applications.

	Meshlium by Libelium	Raspberry Pi 4B
Price	Cons: - Very expensive	Pros: - Low budget solution - Excellent price-to-performance ratio
Performance	Pros: - Optimized for packet capture - Hardware acceleration for specific tasks	Pros: - Powerful quad-core processor (1.8 GHz) - Up to 8 GB of RAM for multitasking
Connectivity	Pros: - Built-in 3G/4G support - Waterproof and rugged casing Cons: - No HDMI or USB ports for direct GUI interaction	Pros: - Multiple USB ports for peripherals - Micro-HDMI ports for displays Cons: - Not waterproof by default
Wireless Capabilities	Pros: - WiFi - Bluetooth 2.1 and BLE - Xbee-PRO 802.15.4	Pros: - Dual-band WiFi support (2.4 GHz & 5 GHz) - Bluetooth 5.0 and BLE
Operating System	Cons: - Limited to Debian-based Linux	Pros: - Versatile OS options, including Raspbian OS
Customization	Cons: - Limited to predefined use cases	Pros: - Highly customizable and expandable
Weight	Cons: - Relatively heavy (2.2 Kg with enclosure)	Pros: - Lightweight and easy to carry

Table 4.1 Comparison between Meshlium by Libelium and Raspberry Pi 4B.

Sniffing software

In the realm of network packet capture, there exists a range of software tools designed to effectively capture and manage packet data. These tools are invaluable for monitoring network activity, allowing users to capture packet data from a live network or extract data from previously saved capture files. The captured data can then be presented in a decoded format, either displayed on the standard output or saved to a file for further analysis.

Among the numerous options available, three notable command-line tools are commonly used for packet capture and analysis:

- **tcpdump [96]:** A versatile and widely recognized packet capture utility, tcpdump offers powerful capabilities for monitoring network traffic.
- **tshark [99]:** Tshark is a command-line packet analyzer based on Wireshark. It provides robust packet capture and analysis features, making it a favorite among those who prefer a command-line interface.
- **Airodump-ng [17]:** Specifically tailored for wireless network analysis, Airodump-ng is a component of the Aircrack-ng suite. It excels in capturing and analyzing packets in wireless environments, making it indispensable for wireless security assessments and troubleshooting.

4.4 WiFi probe requests people counting algorithms

In this Section, we present an overview of crowd monitoring systems that rely on WiFi probe requests. We will begin by discussing older methods used before the implementation of MAC address randomization and progress to the latest techniques. These systems all share the common goal of estimating the number of devices within a specific area, effectively gauging the presence of individuals. These methodologies typically operate under the assumption that each person possesses a single device, simplifying the process of estimating the number of people present based on the count of detected devices.

It is worth noting that there are certain exceptions to this assumption. For instance, some elderly individuals and babies may not have smart devices, while

some people may carry multiple devices. However, this simplifying assumption proves highly valuable in contexts such as crisis management and urban traffic analysis. In such scenarios, the goal is to estimate a figure that closely approximates the actual number of individuals present, rather than achieving perfect accuracy. This is why the one-device-one-person assumption remains a pragmatic and effective approach.

4.4.1 Naive algorithms

Before the advent of MAC address randomization techniques, tracking individuals through the sniffing of probe request messages was a relatively straightforward process. Simply tallying the count of unique MAC addresses detected sufficed for the task of headcount.

An example implementation of this algorithm is detailed with the provided pseudo-code in Algorithm 2.

Algorithm 2 Algorithm to compute the device counting with a Naive approach.

Require: *.pcap* file with the capture
MACaddresses_list \leftarrow *empty_list*
for *packet* in *capture* **do**
 MAC_address \leftarrow *getMACaddress(packet)*
 if *MAC_address* is not already in *MACaddresses_list* **then**
 MACaddresses_list \leftarrow *insert(MAC_address)*
 end if
end for
number_of_devices \leftarrow *length(MACaddresses_list)*

By running this algorithm iteratively across all captures, the count of detected devices was easily obtainable by simply tallying the elements within the list of MAC addresses. However, with the introduction of MAC address randomization, these straightforward methods could no longer be applied. This represented a crucial step forward in safeguarding user privacy. Nevertheless, it substantially complicated the feasibility of monitoring crowds through probe requests.

4.4.2 De-randomization algorithms

In the realm of WiFi probe requests, there have been concerted efforts to tackle the issue of MAC address randomization. The primary objective is not to uncover the original MAC addresses, as this would demand extensive computational resources. Instead, the focus is on developing methods that allow to discern which MAC addresses are more likely to be linked to the same device. This is what we refer to when discussing de-randomization algorithms.

Non-ML-driven algorithms

Among the early algorithms developed for MAC addresses de-randomization, one notable solution is known as iABACUS [63]. Unlike traditional approaches that focus on MAC addresses, this algorithm leverages basic computer science constructs like if-then-else statements, loops, cycles, and recursion. What sets it apart is its consideration of fields other than the MAC address, particularly the capabilities found in the header sections of probe requests. The iABACUS system employs a recursive algorithm to investigate the header's fields and assigns a probability to a probe request with a randomized MAC address belonging to a set of previously observed probe requests. In essence, it estimates the likelihood that the probe request corresponds to a device that has already been counted. By the end of this process, it can provide an accurate count of the number of devices present in the captured environment.

While methods based on conventional algorithms and capabilities fields have demonstrated their effectiveness in device counting, they do come with certain limitations. Firstly, they can demand significant computational power, especially when recursion is involved. In cases where the message trace contains a substantial number of probe requests, there is a risk of overwhelming the memory of the devices running these algorithms.

Unsupervised ML-driven algorithms

In recent years, the emergence of machine learning techniques has given rise to new de-randomization algorithms that capitalize on clustering methods. Clustering, an unsupervised machine learning approach, is designed to categorize similar objects or

data points based on their intrinsic characteristics or attributes. The primary objective is to uncover patterns, structures, or natural groupings within a dataset. Importantly, this process is carried out without any predefined labels, making it a crucial part of unsupervised methods. Clustering aids in uncovering insights, identifying similarities or dissimilarities between data points, and structuring extensive datasets into more manageable groups. To illustrate this, Figure 4.3 provides a representation of a clustering schema where data is depicted as points on a plane. These data points are clearly grouped into distinct clusters, each assigned a label.

The clustering process is usually made of some key steps:

1. **Data representation:** The data is formatted in a convenient way, allowing the algorithm to discern relationships between data points.
2. **Selection of distance measurement:** A distance metric is employed to determine the proximity of two data points. The most common distance metric is the Euclidean distance.
3. **Determination of the number of clusters:** When the number of clusters to be created is established in advance we call it *hard* clustering, instead, when it is determined automatically by the algorithm, we call it *soft* clustering.

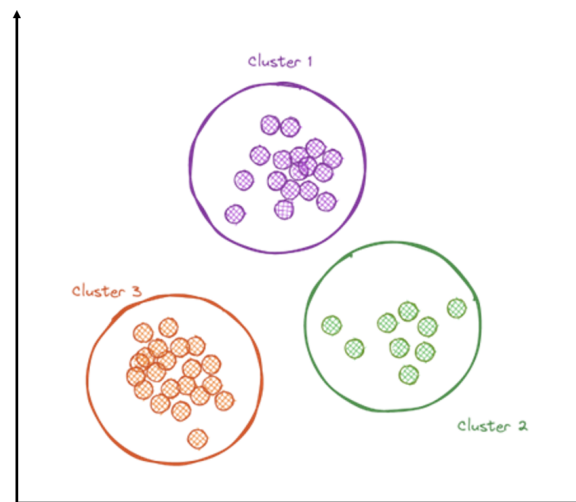


Fig. 4.3 Example of clustering.

4. **Selection of a clustering algorithm:** Various clustering algorithms are available, each with its unique strengths and weaknesses. Some popular ones include k-means, hierarchical clustering, density-based clustering, Gaussian Mixture Models (GMM), and others.
5. **Clustering process:** The clustering algorithm is applied to each data point, and it iteratively assigns data points to clusters based on their similarity evaluated through the distance measure selected.
6. **Evaluation process:** Once the clustering process is complete, the resulting clusters are evaluated to gauge their quality and coherence. Different evaluation metrics may be used, depending on the nature of the data and the specific task objectives.

In crowd monitoring systems, clustering can be employed with a straightforward mechanism: grouping the probe requests, with each request representing a device, and subsequently counting the number of clusters formed to extrapolate the number of devices. Instead, the selection of attributes to use for forming clusters is indeed a non-trivial task and often a critical aspect of the clustering process. The choice of attributes significantly influences the quality and meaningfulness of the clusters produced.

In the domain of clustering algorithms, there exists a notable category known as density-based clustering, a part of the soft clustering group. Two prominent density-based clustering algorithms are DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure). While DBSCAN can encounter challenges with datasets containing clusters of varying densities and irregular shapes, OPTICS addresses these issues by establishing a hierarchical data representation, thereby enhancing the flexibility and robustness of cluster identification. Research endeavors, exemplified in studies [90] and [100], have harnessed these algorithms to achieve precise device counting and crowd analysis across diverse scenarios.

In the case of [90], the authors initiate the process by dividing detected probe requests into two categories: those with global MAC addresses and those with locally administered MAC addresses. They then focus exclusively on packets featuring locally administered MAC addresses and extract essential fields and frame capabilities. Subsequently, they apply two clustering methods, DBSCAN and OPTICS,

to allocate the probe requests into distinct clusters. In the final phase, the clusters undergo a matching algorithm that attempts to link a cluster with one of the globally administered MAC addresses detected. In cases where no matches are found, the group is considered an individual device.

Conversely, in [100], the authors opt not to consider the values of the fields but rather their lengths. Building upon the framework previously described, they initially categorize probe requests into two groups: one based on globally administered MAC addresses and the other based on locally administered MAC addresses. Subsequently, they further subdivide the probe requests based on burst value. From these burst-based groups, they extract information such as burst rate and inter-packet time. Finally, they apply the clustering algorithm, factoring in the time-based features extracted. The overall device count is then determined by adding the number of clusters to the count of distinct globally administered MAC addresses identified at the outset of the analysis.

In the following three Sections, we offer real-world use case examples showcasing the implementation of techniques, including both non-machine learning-driven and machine learning-driven approaches.

4.5 Real-time presence sensing on a 5G infrastructure

In this Section we represent the first use case within our commitment to the European project known as 5G EVE [60]. Our objective was to address an outdoor scenario, with a specific focus on the detection and counting of people traversing various modes of transportation within a designated area in Turin, Italy. Within this testbed, a multitude of off-the-shelf scanners, detailed in 4.3.1, were strategically deployed along a stretch referred to as the *innovation mile*. This mile spanned the distance between the Politecnico di Torino university campus and Porta Susa train station, which stands as the primary transit hub in Turin.

Within this area, as illustrated in Figure 4.4, we not only successfully identified pedestrians commuting daily to our campus but also tracked a substantial influx and efflux of individuals utilizing diverse transportation modes, such as bicycles, electric scooters, cars, and motorcycles.

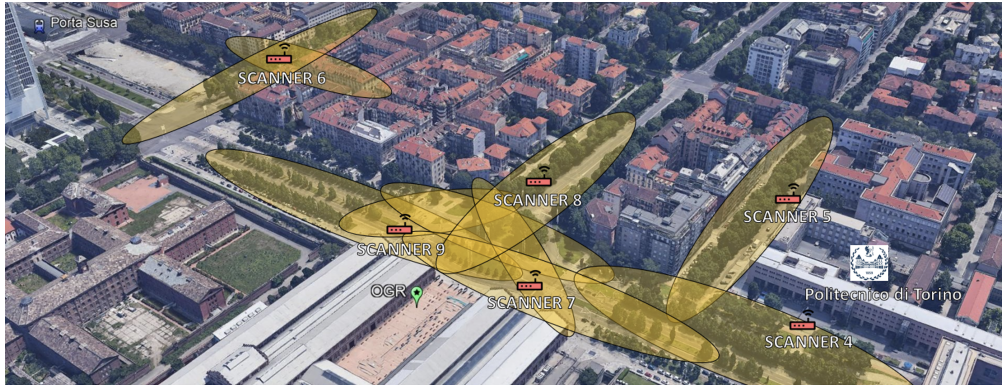


Fig. 4.4 WiFi scanner coverage map in the *innovation mile* in Turin.

4.5.1 Implemented architecture

Figure 4.5 illustrates the architecture of the testbed we established to address our use case. It encompasses an edge cloud, housing two applications realized by integrating multiple Virtual Network Functions (VNFs). The WiFi scanners are interconnected via the Radio Access Network (RAN) with the OneM2M server [9], an open architecture catering to IoT service provisions. An MQTT broker facilitates the connection between the OneM2M server and the edge cloud, hosted within a data center at Politecnico di Torino.

The architecture’s scalability is greatly empowered by the utilization of the edge paradigm. It can expand horizontally as the number of deployed scanners increases. Moreover, the use of Virtual Network Functions (VNFs) to orchestrate the entire mobility application confers a high degree of flexibility concerning the requisite hardware resources.

Through MQTT connectivity, data residing in the OneM2M server is retrieved by the MOBility tracking (MOB) VNF and subsequently stored in a local MySQL database, enhancing performance for subsequent data analyses. Finally, users or experimenters have real-time access to all data collected by the scanners through the VISualization (VIS) VNF, featuring a web-based visualization tool.

As portrayed in Figure 4.4, we installed six Meshlium by Libelium WiFi scanners [59]. Two of these were placed on our campus near entry gates, while the remaining scanners were positioned atop traffic light poles. Each scanner is linked to the platform developed within the 5G EVE EU project [60] via a cellular connection.

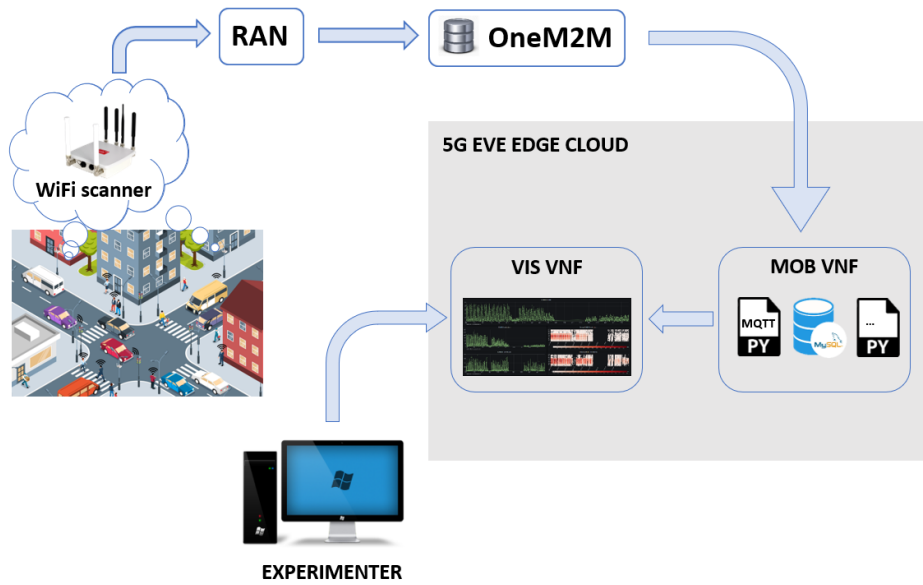


Fig. 4.5 Edge cloud architecture in the 5G EVE Eu project.

Notably, all MAC addresses identified are anonymized using an SHA-224 function immediately upon collection.

4.5.2 Mobility framework

The core mobility framework comprises two integral components known as MOB and VIS. MOB is the Mobility VNF responsible for the retrieval, cleaning, and storage of data collected by the sensors. VIS, on the other hand, is tasked with visualizing the data stored in MOB's database.

MOBility VNF

The MOB VNF is primarily responsible for retrieving data from the OneM2M platform using an MQTT (Message Queuing Telemetry Transport) client. MQTT is a lightweight, open transport protocol founded on the publish-subscribe model. This protocol operates over TCP/IP, ensuring reliability and preventing data delivery in the wrong order.

When the scanners transmit a new message to the remote platform, it is initially saved in the server's local database. Subsequently, it is relayed to the MQTT client,

where the message undergoes parsing, analysis, and eventual storage within the MySQL database of MOB.

Before the data is preserved in the MOB database, two sequential operations take place:

- **Address Digesting:** This step reduces the data's footprint in the database, optimizing storage efficiency.
- **Stationary Device Removal:** Here, devices classified as stationary objects (e.g., access points, fixed computers, etc.) are identified as outliers and eliminated from the dataset.

VISualization VNF

The VIS VNF offers a comprehensive overview of all the meticulously collected and analyzed data. These data are accessible through a standard web browser via two distinct means: an interactive, real-time visualization dashboard integrated through Grafana [50] tool, and a dynamic web page with a Python backend.

Grafana [50] is a versatile, open-source analytics and interactive visualization web application that operates across multiple platforms. It empowers the creation of sophisticated monitoring dashboards by employing interactive query builders linked to databases. Grafana enables the generation of interactive charts, graphs, and alerts, making it possible to establish threshold values for alerts—both above and below—which trigger notifications.

4.5.3 Results

Now, let us delve into some of the results stemming from the copious data collected via our testbed. Over the span from October 2019 to May 2022, we recorded a staggering total of 97,728,830 detection events, signifying 51,255,486 distinct MAC addresses. Notably, due to MAC address randomization, this latter figure represents an upper limit concerning the number of detected devices. It is worth to mention that, our inability to access the Meshlium software has hindered our capacity to implement any de-randomization procedures on the collected data.

In a more detailed examination, Figures 4.6 and 4.7 provide a comparative analysis of data captured during the month of March for three consecutive years: 2020, 2021, and 2022.

Figure 4.6 employs a logarithmic scale heatmap to illustrate the detection occurrences for each scanner across various hours of the day. Scanner 7 data is excluded due to outages. The impact of COVID-19 restrictions is evident. In March 2020, a strict lockdown led to significantly reduced outdoor mobility, reflected in the diminished numbers of smart devices detected. In March 2021, as restrictions began to ease, more devices appeared. Finally, in March 2022, with schools, universities, and companies returning to normal operations, the graph depicts a substantial increase in the presence of individuals in the testbed area.

Figure 4.7 continues to reflect the aforementioned trend but provides a more consolidated view of the overall detections for each scanner in the month of March across the three years. As anticipated, the three curves do not align but exhibit gaps between them. Notably, scanner 6, located near the Porta Susa train station, detected

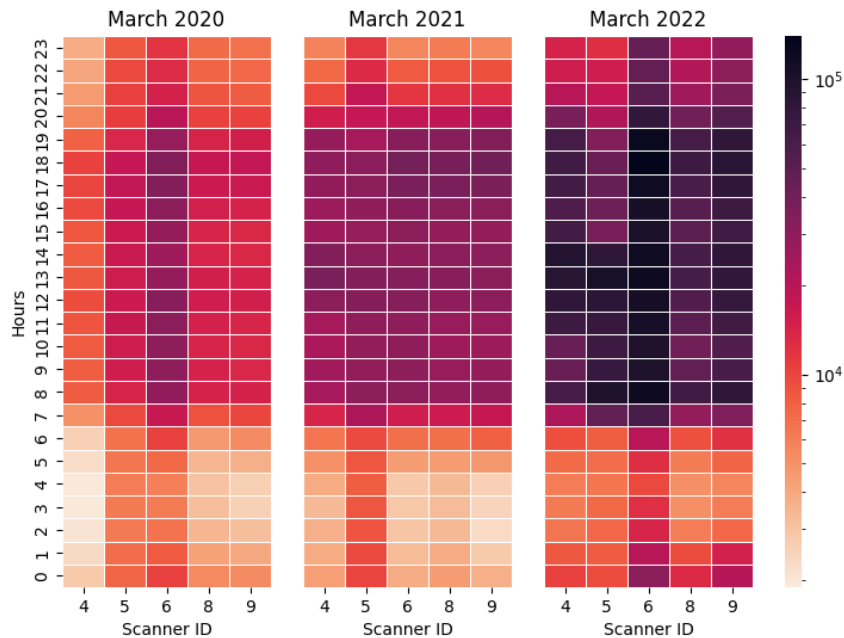


Fig. 4.6 Heatmap illustrating hourly log-scale detection frequencies for March 2020, 2021, and 2022.

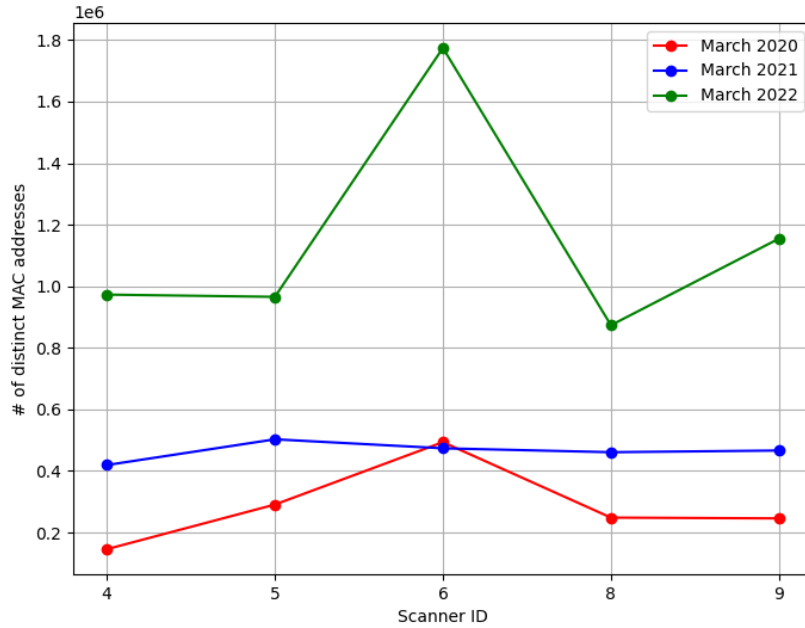


Fig. 4.7 Count of unique MAC addresses identified by each scanner throughout March for 2020, 2021, and 2022.

a consistent number of devices in both 2020 and 2021 but recorded a considerably higher count compared to other scanners in 2022.

4.6 Passive crowd monitoring inside a bus

In this Section, our focus shifts to a more complex real-world scenario: onboard a bus. In this research, we undertook the challenging task of counting passengers aboard buses in the city of Turin. To achieve this, we employed a cost-effective Raspberry Pi 3B, a cutting-edge de-randomization algorithm, and leveraged the WiFi probe request messages emitted by passengers' devices.

4.6.1 Capturing framework

To uphold privacy and data security, the probe requests intercepted by our sniffer (depicted in Figure 4.8) are processed in real-time. This processing involves directing

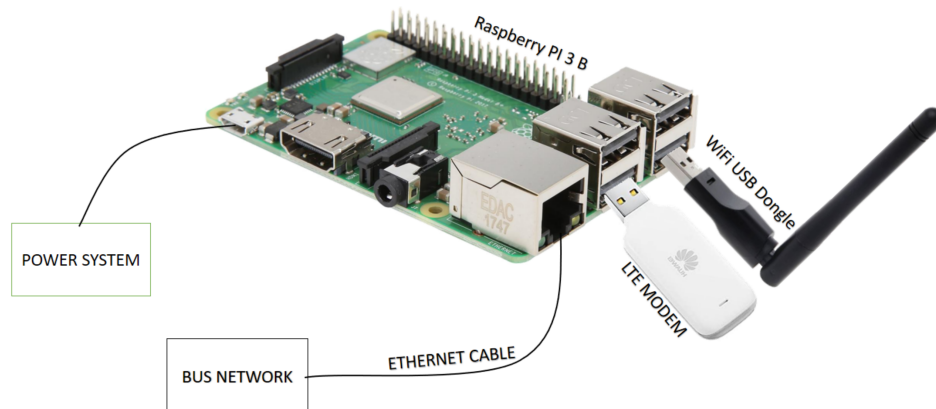


Fig. 4.8 Sniffer solution featuring a Raspberry Pi 3B, a USB WiFi dongle, and a USB LTE modem.

the output of the network sniffer from two interfaces - one capturing WiFi probe request packets and the other gathering internal bus information via Ethernet - into scripts designed for message analysis. Our analysis consists of four distinct steps, each executed sequentially within a processing pipeline.

1. Probe request sniffing

To determine the optimal capturing timeframe, we developed a script that monitors the bus door status obtained via UDP from the bus's Local Area Network (LAN). We initiated the capture process as soon as the bus doors closed following a door opening status, and terminated it as soon as the doors opened again. This approach ensures that we capture a continuous sample encompassing the entire duration between two consecutive door openings, corresponding to two consecutive bus stops. This methodology streamlines our passenger counting task, eliminating the need to account for noise stemming from people waiting at bus stops.

2. Probe request filtering

In our pursuit of detecting only onboard devices, it is imperative to employ filtering mechanisms that restrict our data capture to the boundaries of the bus. It is quite common for our sensor to capture probe request frames originating from devices outside the bus. This can occur frequently, especially when the bus halts at traffic lights or is surrounded by other vehicles, but it is a scenario that can unfold at any time

during the journey. To address this challenge, we establish two crucial parameters: the power level threshold, denoted as P_l , and the occurrence level threshold, referred to as O_l . These thresholds are defined as follows:

- **P_l (Power Level Threshold):** P_l signifies the minimum acceptable average power level of a burst of probe requests. A lower average power level is an indicator that a device might be in proximity to the bus for a brief moment, implying that it is not onboard.
- **O_l (Occurrence Level Threshold):** O_l establishes the minimum count of times a MAC address must appear within a specified sample window to be recognized as a valid onboard device. A lower number of frames within a burst may suggest that a device is merely passing by, necessitating its exclusion from our analysis.

If the average power of a burst of probe requests falls below P_l , or if the burst contains fewer than O_l frames, the respective input record is discarded. It is crucial to emphasize that the values of P_l and O_l should be fine-tuned for each unique environment, such as each individual bus. These thresholds require recalibration when transitioning to different bus settings to ensure accurate filtering and detection of onboard devices.

3. De-randomization algorithm

At the core of our probe request counting procedure lies the MAC address de-randomization method. This technique endeavors to determine if two sets of probe requests, each containing randomized MAC addresses, likely originate from the same device. Our input data comprises a series of bursts of probe request frames, all sharing an identical MAC address. We rely on the iABACUS method [63], as elaborated in Section 4.4.2, to underpin our de-randomization process. The output of the iABACUS algorithm, which we have implemented following the flowchart available in [63], comprises a collection of lists, each containing various randomized MAC addresses assumed to be linked to a single device. In the final step, we ascertain the count of distinct devices on board by tallying the number of separate lists generated through the de-randomization process. This approach enables us to

accurately count the number of onboard devices by discerning and grouping probe requests originating from the same source.

4. Counting, storage, and visualization

The preceding step yields the crucial result: the count of individuals on board the bus. The final script within our pipeline is responsible for transmitting this count, along with the detection timestamp, to our server through the LTE dongle integrated into the Raspberry Pi. On the server side, a script continually awaits incoming UDP datagrams. Upon receiving new data, it diligently stores the detected count and its corresponding timestamp in a local SQL database.

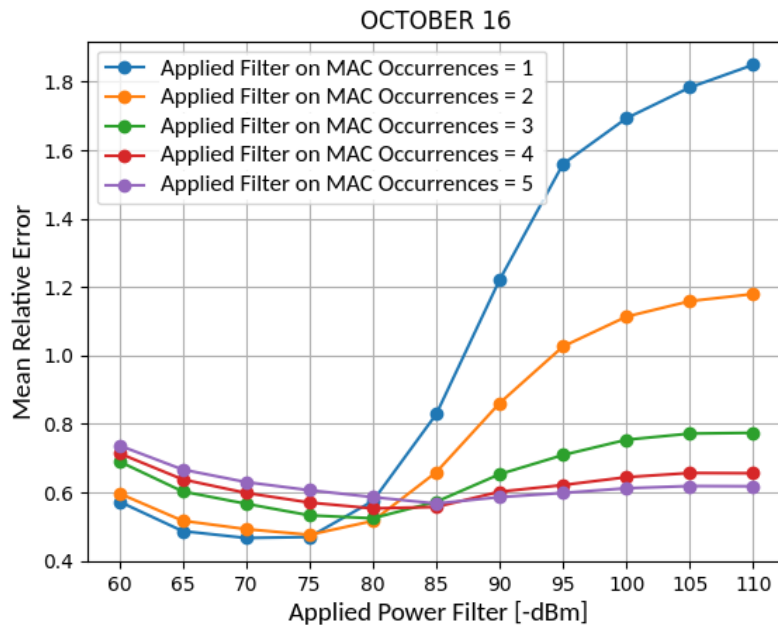
To provide real-time insights and visual representation of the data, we employ a Grafana dashboard [50]. This dashboard allows us to visualize the data originating from the sniffer as it arrives, offering a real-time perspective on the passenger count and enabling us to monitor passenger activity and trends as they occur.

4.6.2 Validation and parameter tuning

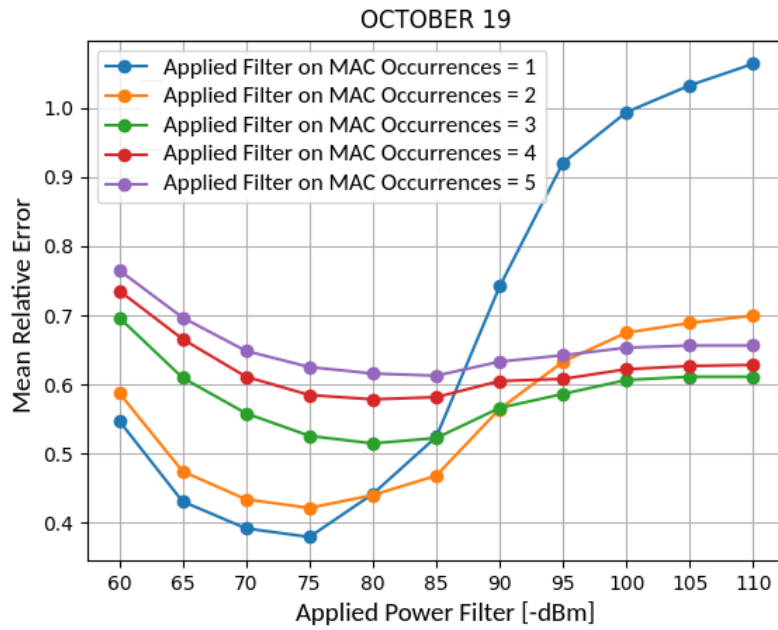
To ensure the accuracy and effectiveness of our system, it was essential to compare the results of our counting process with ground truth data. We scheduled numerous manual counting sessions to establish the precise count of individuals on board a public bus while our system was actively in operation. This critical step allowed us to align our system parameters with the capturing environment accurately. In Figure 4.9, we can observe the mean relative error for various combinations of power and occurrence thresholds. By using specific values such as $O_l = 1$ and $P_l = -75$ dBm as thresholds, we effectively filter out spurious detection occurrences. This minimizes the mean relative error across the board, as evident in Figure 4.9.

4.6.3 Performance evaluation

With our parameter tuning efforts, we can provide rather precise estimates of the number of passengers on a bus. Our findings demonstrate that we can predict the actual passenger count with a high degree of accuracy, particularly for scenarios where we had access to ground truth data.



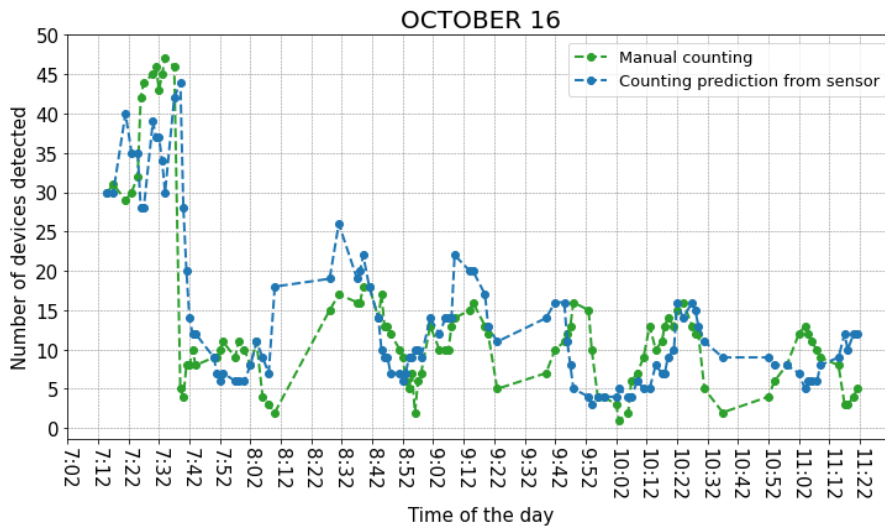
(a) October 16, 2020



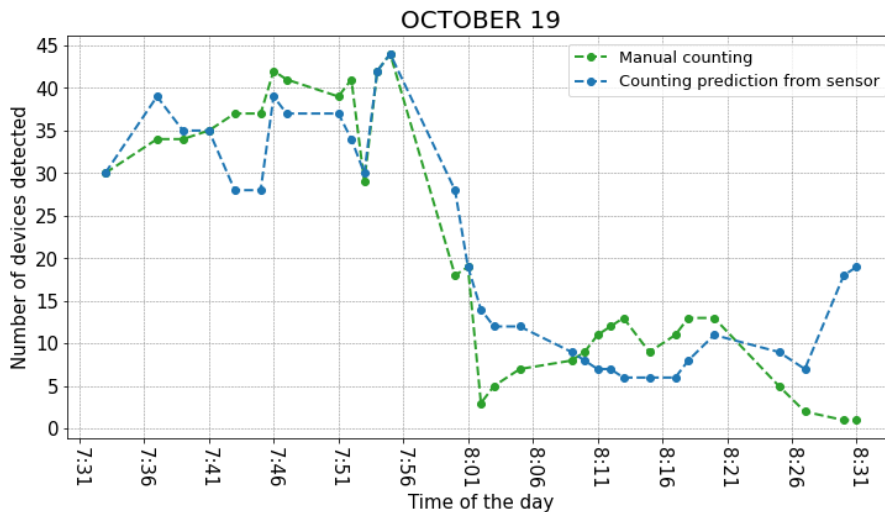
(b) October 19, 2020

Fig. 4.9 Mean relative error comparison for two metrics: power filter and MAC occurrence filter.

Figure 4.10 illustrates the comparison between our passenger count predictions and the manual counts conducted on two different days, October 16 and October 19, 2020. In summary, we achieved an accuracy level of approximately 85% for the first day and 90% for the second. These results reflect the effectiveness of our system in accurately estimating the number of passengers on board, demonstrating its potential for reliable real-time passenger counting on public buses.



(a) October 16, 2020



(b) October 19, 2020

Fig. 4.10 Performance evaluation with manual counting for two days.

4.7 Machine learning-driven privacy-preserving framework for crowd management

In this Section, our focus shifts to the refinement of the de-randomization algorithm, a fundamental component of our people counting technique that leverages probe request messages. Building upon the insights gained from our analysis in Chapter 2 and the probe request generator introduced in Chapter 3, we now introduce our innovative machine learning-driven privacy-preserving framework for crowd monitoring.

4.7.1 DBSCAN clustering method

Our crowd monitoring framework harnesses the power of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. DBSCAN is employed to classify probe requests into distinct categories based on specific criteria related to device or model differentiation. Unlike traditional clustering methods, DBSCAN excels at effectively handling noisy data points, i.e., those that fail to meet the density criteria required for inclusion within a cluster. As outlined at the outset of this Chapter, clustering algorithms follow unsupervised learning techniques. They involve training models on unlabeled data, lacking access to explicit target values or predefined categories for the input. Instead, they aim to unveil underlying patterns and relationships within the data sets.

Key components and parameters of the DBSCAN algorithm include:

- **Epsilon (ϵ):** This represents the cluster radius.
- **Min samples:** It signifies the minimum number of data points required to form a cluster.
- **Evaluation metric:** This metric is used to calculate the distance between data points.
- **Core points:** Data points considered core points must have a specified number of other data points (as defined by the min samples parameter) within a certain radius (expressed by the epsilon parameter). These core points serve as central components of clusters.

- **Border points:** Data points not classified as core points but falling within the radius of a core point are designated as border points.
- **Noise points:** Data points that neither qualify as core points nor border points are identified as noise points or outliers.

In our case, we employ the Euclidean distance as the evaluation metric, applied in an N-dimensional space. Each probe request is regarded as a point, with its unique features determining its coordinates along each dimension. Among the various features contained in probe request frames, we specifically focus on VHT, HT, and Extended capabilities. The epsilon and min samples parameters play pivotal roles in the effectiveness of counting algorithms, and their calibration is contingent upon the specific scenario and expected traffic. Typically, epsilon is minimized to maximize differentiation between clusters, while min samples is tailored based on the expected average number of probe requests within the designated time window.

To facilitate the clustering process, it is essential to generate distinct model identifiers for each probe request. These identifiers serve as coordinates within the space in which the clustering method will be subsequently applied. The pseudo-code for the method used to generate these model identifiers is provided in Algorithm 3. The Algorithm is designed to create individual identifiers for each probe request, allowing for the unique recognition of its model. The deliberate separation of the three capabilities serves a specific purpose: it mitigates potential problems that could arise if messages from different models share the same identifier after the aggregation, despite possessing differing capability values. This separation of fields ensures a more robust and effective approach.

4.7.2 DBSCAN clustering algorithm

To ensure the precision of probe request analysis, it is crucial to confine these requests within a predefined capture area, typically in close proximity to the sniffer device. Occasionally, probe requests from devices located outside this designated region may be captured. To address this, a signal power threshold of -70 dBm is applied to compare the strengths of probe requests signals. The specific value of this threshold can be adjusted to tailor the capture area according to the particular use case at hand.

Algorithm 3 Algorithm to compute the probe requests model identifiers.

Require: *.pcap* file with the capture
ids_list \leftarrow *empty_list*
for *packet* in *capture* **do**
 HT_counter \leftarrow 0
 VHT_counter \leftarrow 0
 Extended_counter \leftarrow 0
 for *field* in *packet* **do**
 if *field* contains HT info **then**
 HT_counter \leftarrow *HT_counter* + *value*
 else if *field* contains VHT info **then**
 VHT_counter \leftarrow *VHT_counter* + *value*
 else if *field* contains Extended info **then**
 Extended_counter \leftarrow *Extended_counter* + *value*
 end if
 end for
 ids_list \leftarrow *insert*(*HT_counter*,*VHT_counter*,*Extended_counter*)
end for

Before employing clustering techniques to group probe requests based on device models, it is essential to handle devices that do not employ MAC address randomization. These are the devices that transmit their original MAC addresses, and distinguishing them from those using randomized MAC addresses is achieved by inspecting the Globally/Locally unique bit. Counting the devices utilizing globally unique MAC addresses is straightforward; a Python set can be employed to store these addresses, ensuring uniqueness. The size of this set, as revealed during the parsing of the *.pcap* file, provides the count of devices not implementing MAC address randomization.

For probe requests coming from devices employing randomization techniques, a model identifier is computed as detailed in Algorithm 3. These model identifiers represent the data points used in the DBSCAN clustering algorithm, with the primary goal of grouping probe requests based on the device models from which they originated.

By default, DBSCAN assigns the label *-1* to items classified as noise. Since this group may contain probe requests from various models, it is not representative for our crowd monitoring analysis, and thus, it is discarded. To determine the number of devices for each model within the clusters formed by the clustering algorithm,

probe request rates are taken into consideration. Each model sends probe requests at a specific frequency, depending on the phase it is in. Instead of attempting to separate clusters by phase, this framework focuses on the average sending rate for each model. In cases where crowd monitoring is conducted in environments where a particular phase predominates over others, it may be useful to consider message rates for each phase separately.

The formula employed for this purpose is the (4.1).

$$N = K \cdot L \cdot T \quad (4.1)$$

Where: N represents the total number of probe requests sent by devices of a specific model, K denotes the count of devices belonging to the model, L signifies the average probe request rate for the model, and finally T represents the time window in which the packets were sent.

It is important to emphasize that while parameters N and T are readily available, while the rate L must be determined from a reliable data source. Finding a specific L value for each formed cluster is essential, as each model exhibits significantly different rates during the experiments.

4.7.3 Counting pipeline

The core concept behind the counting pipeline is that when two models share similar identifiers, there is a higher probability that their probe request rates are also alike. Our probe request generator's database contains multiple models, each with its unique identifiers and rates. This approach facilitates the comparison of each generated cluster with all models in the database based on their identifiers. This, in turn, enables the identification of the most similar model, from which the rate can be extracted as the parameter L . To obtain these values, the probe request generator (referenced in 3.4) allows for simulations with individual devices from the database. This simulation makes it possible to calculate the message rate associated with a specific model.

The counting pipeline is illustrated in Figure 4.11. To provide a more detailed overview, the system initialization process begins by configuring all the necessary environment variables and executing a wireless interface configuration script through

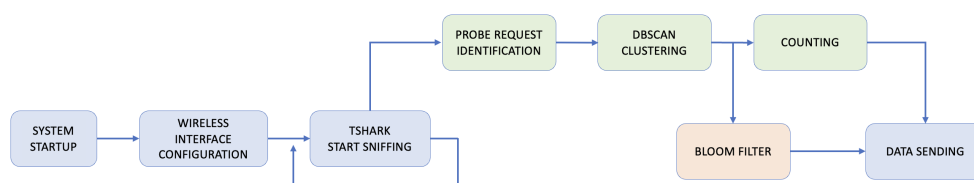


Fig. 4.11 Counting pipeline.

the *rc.local* file. This script serves the purpose of switch off the onboard WiFi interface of the Raspberry Pi and switch on the one connected with the USB WiFi dongle. This is a crucial step because the embedded WiFi interface cannot be placed in promiscuous mode. After all configurations are set, the primary sniffing script is initiated. Using tshark [99], we begin the process of capturing probe request messages within a two-minute time window. Following this, the resulting *.pcap* file is handed off to the subsequent stage of the processing pipeline, while the tshark script restarts for another capture cycle.

Each time a new *.pcap* file is generated, a script commences the analysis of all the detected probe requests. All messages are categorized into two groups based on the global/local bit in the MAC address. For each probe request with a locally administered MAC address, we proceed with generating model identifiers. Specifically, we utilize the throughput capabilities to construct 3-dimensional data points that are then fed into the DBSCAN clustering algorithm, as discussed in Section 4.7.1. After this clustering process, the pipeline then splits into two separate paths, as highlighted in Figure 4.11 using distinct colors: green for the single scanner counting pipeline and orange for the multi-scanner flows analysis.

In the first scenario, we obtain the initial timestamp of the time window and a numerical value representing the count within that specific time window. In the multi-scanner analysis case, the resulting MAC addresses are stored in a Bloom filter. This filter is specially initialized with n random MAC addresses to meet the 1-deniability requirement, necessary for ensuring GDPR compliance. Consequently, we obtain the initial timestamp of the time window and the populated Bloom filter.

Both sets of results are subsequently transmitted to a server and saved in a local database.

4.7.4 Counting results

Our primary goal in these studies is to conduct an extensive analysis of crowd monitoring system performance under various environmental conditions. By employing the probe request generator, we can generate a range of scenarios, including those highlighting a limited number of devices with a diverse set of distinct models, as well as scenarios featuring the opposite condition. Additionally, we explore scenarios characterized by mixed situations. Moreover, the algorithms have undergone extensive testing using genuine *.pcap* traces, with associated ground truth data. The list of some of the datasets used in our analysis is the following:

- **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 [69].
- *Dataset G*: comprises a collection of 10 two-minute captures performed within room 14 at Politecnico di Torino, on the first day of the 2023/2024 academic year.

The experiment results have been organized into Table 4.2, where each row corresponds to a specific set of tests conducted using a *.pcap* file simulating a particular environment. The columns in the table present various details, including the dataset identifier, the ground truth (i.e., number of devices), and the count of detected and accuracy, for two different frameworks (our and an implementation of iABACUS [63]). To calculate the accuracy, we employ a method that takes

Dataset	Ground truth	Our framework	iABACUS [63] implementation
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%)	not able to compute
E	120	110 (91.66%)	not able to compute
F	22	23 (95.45%)	not able to compute
G	up to 121	overall 91.48%	overall 27.63%

Table 4.2 Crowd monitoring results.

into account the difference between the obtained result and the ground truth. More precisely, the counting error is determined by computing the absolute difference between the ground truth and the obtained result, which is then normalized relative to the ground truth. Accuracy can be readily obtained by subtracting this error from one.

In Table 4.2, we present the results derived from our framework and compare them to the outcomes obtained using an implementation of the framework proposed in [63]. It is noteworthy that the authors of [63] introduced a de-randomization framework renowned for its high accuracy in their experiments. This framework employs multiple if-then-else constructs and incorporates several levels of recursion to assess the likelihood that multiple MAC addresses belong to the same device.

During our experiments, we ran our implementation of the iABACUS framework, following the flow chart outlined in [63], on an Apple MacBook Pro laptop equipped with the M1 Pro CPU. However, despite the substantial computational power at our disposal, we encountered challenges when processing datasets D, E, and F. The script failed to complete its execution due to recursion errors, surpassing the maximum recursion depth.

Real-life scenario counting results

To evaluate our framework within a real-life scenario, we introduced *Dataset G*. This dataset 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 sets this dataset apart is its dynamic nature. Instead of a static use case, we aimed to capture a scene a few minutes before the start of a lesson and continued recording for a few minutes afterward. As a result, at the outset of the scenario, a few individuals were already present in the room. Over the next twelve minutes, more people entered the room, seeking seats. In the final three capturing windows, the lesson commenced, with only a handful of additional individuals arriving. This dynamic recording scenario allowed us to simulate the gradual entry of people into the room, mirroring a real-world situation.

Table 4.2 provides an overall accuracy for both frameworks under consideration. Additionally, Figure 4.12 offers a detailed representation of the results obtained.

The progressive presence of individuals in the room is clearly evident when examining the ground truth line in Figure 4.12. Our framework’s results are presented in red, while the results obtained with the implemented version of the iABACUS

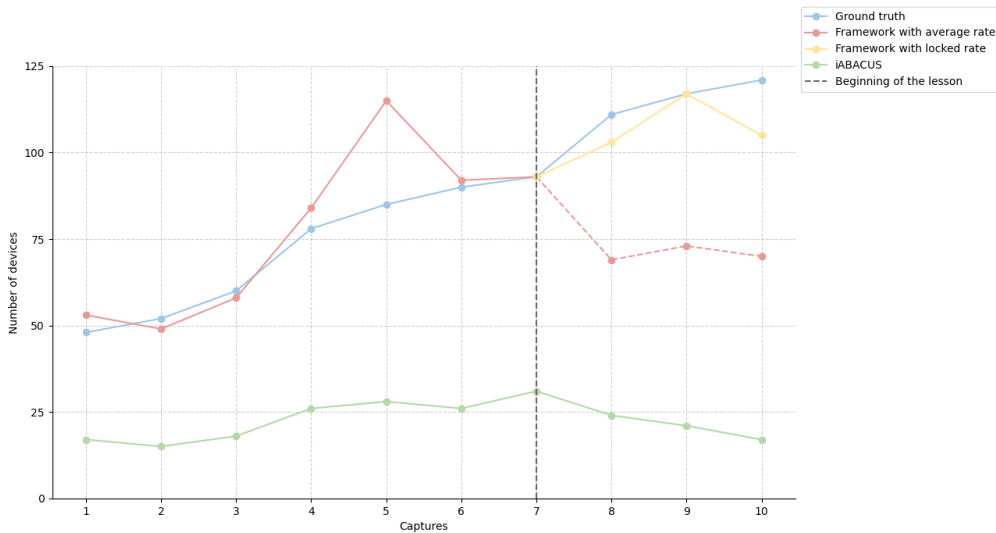


Fig. 4.12 Comparison between ground truth and results obtained from our framework and an implementation of iABACUS one, for Dataset G.

framework are shown in green. The x-axis represents the various captured windows, while the y-axis indicates the number of detected people/devices. Notably, our framework outperforms the iABACUS framework, which tends to significantly underestimate the number of detected devices.

It is important to note 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 such cases, we calculated an average rate that considered both the locked, awake, and active phases. However, starting from the eighth capture, corresponding to the beginning of a lesson, using the average rate is no longer an accurate choice, as it tends to underestimate the count. Instead, focusing solely on the locked rate yields greater accuracy. This underscores the significance of considering the context in which these captures are conducted, as it is crucial for achieving even better results.

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

4.8 Conclusions and future works

Detecting human presence in both outdoor and indoor settings holds significant utility in a variety of contexts. It aids in the design of urban spaces, facilitates public transportation planning, and supports the implementation of mobility restrictions, such as those enacted during the COVID-19 pandemic.

In the course of our study, we have developed several solutions, each tailored to tackle diverse and challenging environments. Moreover, we have been adapting to the continuous evolution of privacy techniques, always with a commitment to keep safe user privacy. Our cumulative experience from these projects has culminated in the creation of a machine learning-driven privacy-preserving framework. This framework serves a dual purpose: accurate people counting and in-depth flow analysis.

Looking ahead, we are currently in the final stages of refining the hardware solution that will run the crowd monitoring framework described in Section 4.7. In the near future, we are preparing to deploy this solution in a real testbed scenario, which is the core component of our participation in the TrialsNet EU project [13]. This project presents an opportunity to validate both our people counting algorithm and the comprehensive flows analysis, facilitated by the deployment of two scanners strategically positioned at opposing ends of a spacious events area.

Chapter 5

Federated Learning Empowered Vehicular Networks

The relentless evolution of automobiles has consistently revolved around two fundamental objectives: enhancing safety for both passengers and pedestrians, and concurrently, improving the overall quality of life within urban centers, with a particular emphasis on the driving experience for drivers. This progressive journey has brought forth an era where vehicles come equipped with an extensive array of sensors, capable of meticulously monitoring both their internal operations and external surroundings. As we approach the future, the trend appears to be unambiguously directed towards an exponential increase in the volume of data amassed by these vehicles. Such data holds the key to delivering services that are not only highly utilitarian but also exceedingly efficient. Inevitably, this exponential data growth calls for advanced technologies to process and extract valuable insights. In this context, machine learning models are poised to play an instrumental role. However, it is essential to acknowledge that the effective training of these models demands substantial computing and energy resources.

Within this Chapter, we embark on a journey to explore innovative solutions centered around cooperative learning, specifically focusing on the application of Federated Learning (FL). Our primary goal is to harness the power of FL in an urban environment, where numerous vehicles work together. In this collaborative paradigm, vehicles, each functioning as a node, securely keep their data onboard while collectively training a Neural Network (NN) model. This approach not only

ensures data privacy but also leverages the collective intelligence of an interconnected fleet of vehicles, with a server positioned at the network's edge orchestrating the cooperative learning.

5.1 Research motivation and State of the Art

According to the most recent report from the World Health Organization, road accidents continue to claim the lives of approximately 1.3 million individuals annually, with a substantial portion of these tragic events attributed to human errors [66]. Moreover, a study conducted by the European TRACE project underscores the significance of intersections, where around 43% of all road injuries occur [89]. These distressing statistics emphasize the pressing need for the development of more robust trajectory prediction systems to enhance road safety.

In the contemporary era, the proliferation of automated vehicles has resulted in the generation of a prodigious amount of data – around 25 GB per hour. This data emanates from a plethora of onboard sensors integrated into modern vehicles. Furthermore, the imminent future promises an even more staggering data volume, with autonomous vehicles poised to produce up to 3 TB of data per hour. This wealth of data holds immense potential for the realization of data-driven solutions, thereby raising the bar for safety and comfort standards while facilitating the effective implementation of convenience applications. However, it is essential to acknowledge that the utilization of machine learning models, particularly Neural Networks (NNs), which are instrumental in these applications, often demands substantial computational and energy resources for their training. Consequently, a pivotal challenge arises around how a network of interconnected vehicles can contribute to the efficient training of neural networks.

In the landscape of potential solutions explored in existing literature, one approach shines as particularly well-suited for this scenario: Federated Learning (FL). Federated learning is an innovative machine learning paradigm in which multiple clients collaboratively engage in the iterative training of a model, all under the coordination of a central server. A defining advantage of FL, setting it apart from other distributed learning approaches, is its capacity for localized model training at each client, with only the resultant training outcomes, i.e., the updated model parameters, being transmitted to the server. This ensures that clients' private data

remains shielded, a crucial feature, especially in an open and distributed environment like a network of vehicles.

Several studies have begun to explore the implementation of FL in a vehicular context, delving into the intricacies of this emerging distributed machine learning paradigm. Works such as [32] and [103] focus on the client selection process in FL. In particular, [32] introduces a data distribution-aware FL orchestration specifically tailored for vehicular networks, establishing protocols based on the Vehicular Knowledge Query format, initially introduced in [33]. These protocols empower the server to glean insights into potential client candidates and make more informed selections based on factors like the requisite training environment or the nature of the training data held by these candidates. Similarly, [103] endeavors to identify the most suitable vehicles to participate in learning tasks, optimizing resource allocation while taking into account the positions and velocities of the vehicles.

In contrast, research explored in [106] and [107] delves deeply into the aspect of parameter aggregation. The former, [106], proposes a selective model aggregation approach, leveraging two-dimensional contract theory as a distributed framework to facilitate interactions between the central server and vehicular clients. This approach yields promising results, especially with datasets like MNIST [31] and BelgiumTSC [98], outperforming the traditional FedAvg [58] aggregation method. In [107], a novel multi-layer heterogeneous model selection and aggregation scheme is introduced, based on a two-layer FL model designed to harness the capabilities of a 6G network architecture.

Several other studies are oriented toward designing edge-based frameworks tailored for vehicular applications. For instance, [54] presents an enhanced service for collision avoidance deployed at the network's edge, effectively minimizing latency in Cellular Vehicle-to-Infrastructure (C-V2I) communication. Meanwhile, the authors of [108] propose a framework that employs deep learning to predict vehicle density to be served by a Multi-access Edge Computing (MEC) host, thereby determining the optimal computing resources required for collision avoidance applications.

5.2 Main contributions

The present study, as detailed in [49], introduces an innovative application of federated learning within a vehicular context. In our scenario, vehicles collect trajectory data at specific intersections using their onboard sensors, and this data is used for local training of a trajectory prediction model within a federated learning process coordinated by the server located at the edge of the network. For a concrete implementation, we adopted the Long Short-Term Memory (LSTM) model for trajectory prediction, as recently presented in [88]. To model the behavior of vehicles, we utilized real-world mobility traces [73] specific to the city of Turin, Italy. Finally, our federated learning process is implemented through the well-established FLOWER platform [20].

Our work makes significant contributions to the existing body of literature in the following key areas:

- **Federated learning in a vehicular context:** We introduce a novel application of federated learning in a vehicular context where vehicles collaboratively train an LSTM model for trajectory prediction.
- **Real-world mobility traces:** We apply the federated learning algorithm in a real-world scenario, harnessing actual mobility traces pertaining to the *innovation mile* in the city of Turin, Italy.

In the upcoming sections, our focus shifts towards the federated learning approach, where we elaborate on the various implementations available, including synchronous vs. asynchronous and centralized vs. decentralized methodologies. Following this, we introduce vehicular networks, emphasizing the trajectory prediction algorithm and the urban environment simulator at our disposal, which enables the high-fidelity simulation of urban environments in a safe manner.

Subsequently, we provide an exhaustive and detailed description of the vehicular framework we have implemented, leveraging federated learning. Finally, we present a performance evaluation obtained from our framework and we conclude this Chapter with our final remarks.

5.3 Federated Learning

Federated Learning, as introduced in [105], represents a decentralized approach to machine learning. This method harnesses data residing on diverse clients' devices to collaboratively train a shared model, in stark contrast to the conventional distributed ML paradigm. In distributed ML, a central server partitions the dataset among multiple servers, thereby distributing the computational load. Furthermore, distributed ML allows for manual allocation of dataset portions to various servers. Conversely, in FL, a key distinction is that the central server does not have access to the raw data of clients.

At the heart of federated learning lies the principle that participant clients maintain privacy and do not reveal their individual datasets. They independently train a local model using their local data and subsequently transmit model parameter updates to a central server (in the case of centralized federated learning). When the server receives these parameters from the clients, it employs a predetermined aggregation strategy to construct an updated global model, which is then transmitted back to the clients. This iterative process continues until predefined performance metrics, such as achieving a specific accuracy level, are met. A visual representation of the federated learning framework can be observed in Figure 5.1.

Federated learning encompasses two primary network topologies: centralized and decentralized. The former involves a central entity responsible for coordinating the training process, aggregating local model parameters from each federated client. The

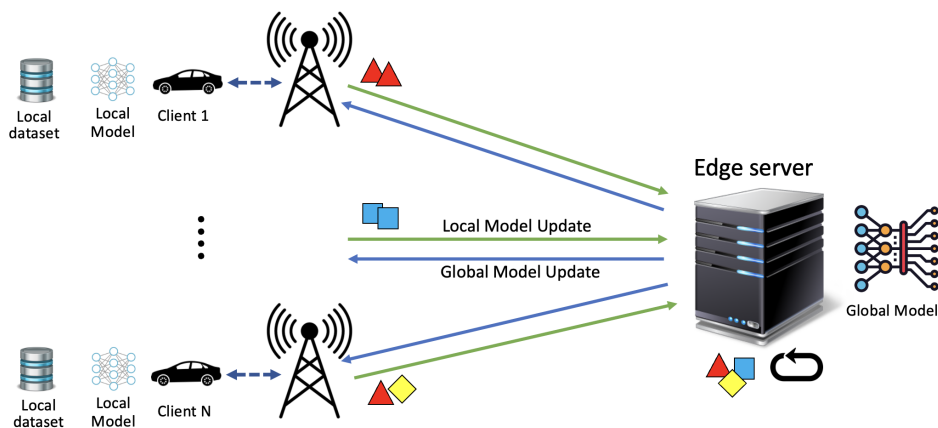


Fig. 5.1 Federated learning architecture overview.

latter relies solely on direct communication among participating clients to coordinate the training process. A more in-depth examination of these network topologies is provided in Section 5.3.1.

Furthermore, federated learning can be classified as either synchronous or asynchronous, depending on whether the server waits for results from all clients before starting a new iteration. The synchronous mode corresponds to the former, while the asynchronous mode pertains to the latter. A comparative analysis of these two execution modalities is discussed in Section 5.3.2.

5.3.1 Centralized vs. Decentralized

As previously outlined, the federated learning literature explores two primary network topologies:

- **Centralized topology:** In this centralized architecture (depicted in Figure 5.2a), a central server takes charge of coordinating a group of available clients. The server assumes the roles of both a controller and coordinator, selecting which devices will participate in each training round and overseeing the communication and aggregation processes. This centralized approach provides several advantages. Notably, the server offers a more dependable connection, as it is typically located at a fixed point, often at the network's edge. Moreover, the server's substantial computational power allows for rapid aggregation of model parameters, minimizing delays. However, a notable drawback is the single point of failure inherent to this topology. If the central server experiences any issues or goes offline, the entire federated learning network comes to a halt.
- **Decentralized topology:** In the decentralized network topology (illustrated in Figure 5.2b), there is no central coordination unit. Instead, devices are interconnected through peer-to-peer links. In each training round, every client conducts its local training without enforced client selection, and the global model update is generated by collecting model parameters from neighboring devices. The decentralized approach eliminates the single point of failure concern. Each client can communicate directly with all other clients, enhancing network robustness. However, it is important to note that this network topology can introduce complexities that may impact the performance of the learning process.

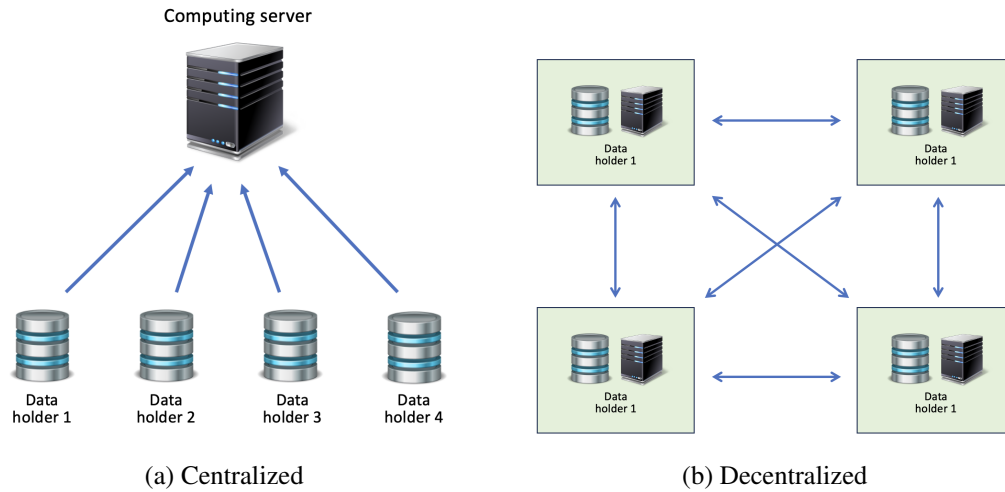


Fig. 5.2 Comparison between centralized and decentralized federated learning.

5.3.2 Synchronous vs. Asynchronous

Now, we will explore the two operational modes in federated learning: synchronous and asynchronous.

- Synchronous Federated Learning:** Synchronous federated learning follows a coordinated, time-locked approach. In this setting, the central server orchestrates training rounds that are synchronized across all participating clients. During each round, clients perform their local training and send their model updates to the central server. The central server waits until it receives updates from all the involved clients before proceeding with aggregation and model update distribution. This mode offers a clear advantage in terms of synchronization, as it ensures that all clients' contributions are processed simultaneously, potentially leading to a more consistent and predictable convergence of the global model. However, it does have some drawbacks. Synchronous federated learning may result in higher communication and computation overhead, especially when some clients experience delays, leading to potential bottlenecks.
- Asynchronous Federated Learning:** Asynchronous federated learning takes a more flexible and decentralized approach. In this configuration, there is no strict synchronization of training rounds among clients. Each client independently conducts its local training and transmits its model updates to the central

server whenever it is ready. The central server aggregates these updates as they arrive, without waiting for the other clients' results, and continuously updates the global model. The key advantage of this mode is its flexibility, as it avoids the need for strict synchronization, which can be especially beneficial in scenarios with varying client capabilities and network conditions. However, the trade-off is that asynchrony can lead to more challenging coordination and convergence issues. Inconsistent update timings can make it harder to achieve a consistently accurate global model.

5.3.3 Federated learning frameworks and implementations

The landscape of federated learning is rich with various frameworks and implementations, catering to both research and real-world deployment needs. Let us delve into the most noteworthy options, each offering unique features and capabilities:

- **FLOWER [20]:** FLOWER stands out as an open-source, platform-independent framework renowned for its exceptional customizability in all facets of federated learning. With FLOWER, it is possible to modify client/server behavior, message exchanges, learning strategies, and even the overall system architecture. FLOWER's versatility makes it a valuable asset for innovation and experimentation in the FL domain.
- **IBMFL [53]:** IBMFL, a black-box Python framework developed by IBM, streamlines the customization process by focusing on two key elements: the dataset and the neural network model. This framework is designed to offer rapid startup times for real-world application deployment and provides a straightforward experimental environment for testing machine learning models within a federated setting.
- **FLSlim [51]:** FLSlim, supported by Facebook research, is a standalone library that simplifies federated learning through high-level API calls. It is known for its scalability and open-source nature, allowing for straightforward customization. What sets FLSlim apart is its foundation on the PyTorch library, which takes an object-oriented approach.
- **Astraea [35]:** Astraea introduces a unique system architecture for federated learning, departing from the conventional client-server model. It introduces

an intermediary role, known as *mediators*, positioned between the clients and the server. Mediators facilitate and moderate client-server communication. Astraea’s distinctive approach includes a pre-training scheduling phase where clients are dynamically assigned to mediators, aiming for diverse and representative client datasets. The objective is to reduce bias and enhance the quality of the federated learning process. In this framework, the server supports a global model with synchronous updates from the mediators, while the mediators handle asynchronous updates from their connected clients.

5.4 Methodology

In this Section, we delve into the methodology used to conduct experiments within a simulated urban environment, where various vehicles harness federated learning for trajectory prediction. Section 5.4.1 provides an extensive overview of the implemented architecture, encompassing all the pertinent software components and stakeholders, while also elucidating the primary objective of our study. Section 5.4.2 introduces the simulated environment, offering a comprehensive description of the specific area under consideration and the mobility traces employed. Lastly, in Section 5.4.3, we explore the federated learning process, elucidating how the clients (i.e., vehicles) collaboratively train the machine learning model in a privacy-compliant manner, without disclosing any data to the server, only sharing the model’s parameters.

5.4.1 Implemented architecture

The implemented architecture, depicted in Figure 5.3, comprises three key components: urban simulation, the federated learning process, and the interface connecting these two elements.

Urban simulation

The primary goal of the urban simulator is to create custom mobility traces within the area of interest. To achieve this, we harnessed ns-3 [8], a discrete-event network simulator, in conjunction with SUMO (Simulation of Urban MObility) [18], a

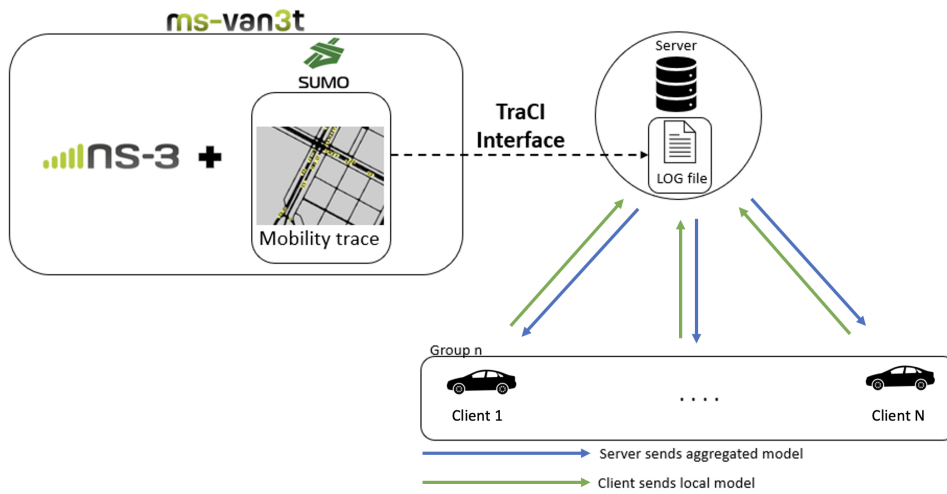


Fig. 5.3 Implemented architecture.

microscopic and continuous traffic simulator. Specifically, we utilized the ms-van3t framework [55], which constructs and simulates ETSI-compliant VANET (V2X) applications using SUMO and ns-3.

Federated learning process

In implementing the federated learning process, we made use of the FLOWER framework [20]. The FLOWER framework is an open-source Python library that streamlines the development of federated learning. It offers a high-level API for federated learning workflows, making it easy to construct and manage machine learning models across a network of decentralized clients while upholding data privacy and security.

TraCI

To bridge the FLOWER platform with the mobility traces, we employed TraCI, SUMO's built-in Traffic Control Interface. TraCI functions as an interface that allows external applications and tools to interact with and control the traffic simulation within SUMO. This interface provides access to real-time data and enables the manipulation of various aspects of the simulation, including vehicle movement, traffic signals, and environmental conditions.

The primary objective of this study is to assess the performance of federated learning in two distinct scenarios: a real-world setting and a synthetic one. In the real-world scenario, we aim to employ actual vehicles operating in an urban environment as clients within the federated network. The synthetic scenario, on the other hand, represents a more controlled environment, where clients change with a fixed periodicity and less frequently compared to their real-world counterparts.

The purpose of this comparative analysis is to gain insights into how the dynamics of an urban mobility scenario can influence the performance of federated learning. To evaluate the real-world scenario, we intend to establish a connection between the behavior of real vehicles entering and leaving a specific area and the actions of simulated federated clients. This approach will allow us to explore the impact of real-world urban dynamics on the efficacy of federated learning processes.

5.4.2 Urban Environment simulation

To replicate a real urban environment using SUMO, we utilized a dataset known as TuST (Turin SUMO Traffic), as detailed in [72]. This dataset was created to model the traffic patterns in the metropolitan area of Turin and its adjacent regions, drawing from 24 hours of data collected by 5T, a public information mobility company in Turin.

Specifically, we focused on a specific segment within the comprehensive TuST coverage area, encompassing an area of 3.5 square kilometers surrounding the Politecnico di Torino university, as illustrated in Figure 5.4. Our analysis was centered on the vehicles operating within this city section during the two-hour morning traffic peak, from 7 a.m. to 9 a.m. The geographical coordinates defining this area are as follows:

- Latitude: [45.054223, 45.073081]
- Longitude: [7.651316, 7.672293]

This area also boasts the radio signal coverage of seven cellular base stations (eNBs), which were configured to mirror the actual real-world cellular deployment, as depicted in Figure 5.5. To construct the network topology, we employed the ns-3 framework, creating a star network configuration with the central node comprising

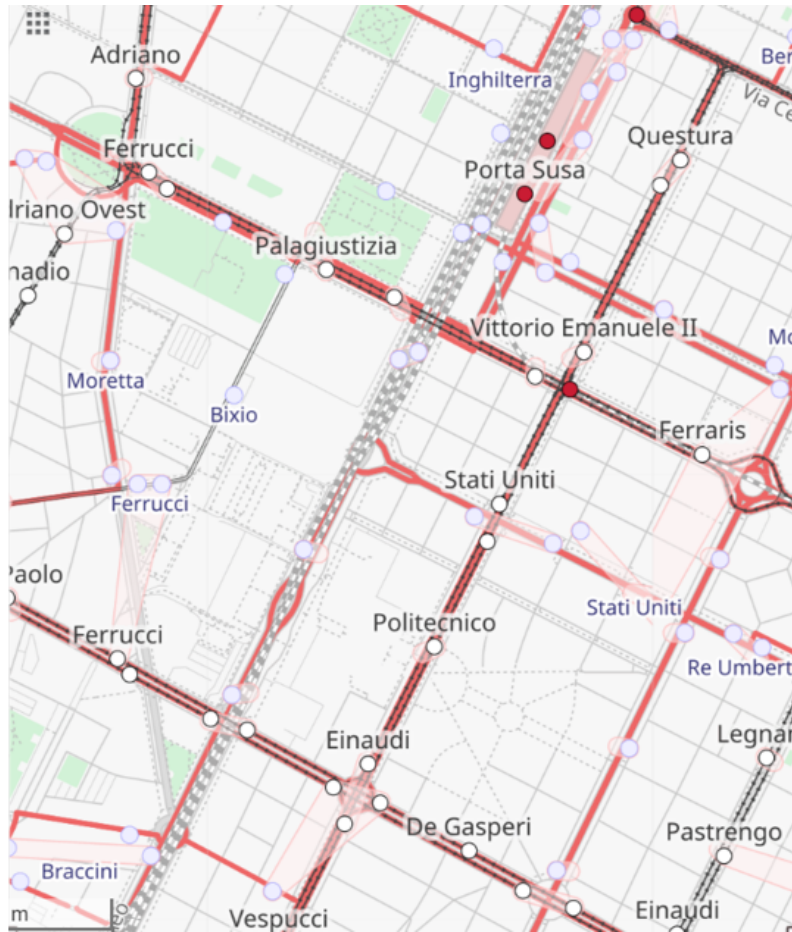


Fig. 5.4 Considered area in Turin.

the edge server, and the connecting nodes representing the seven eNBs. Then each vehicle was emulated as straightforward client nodes equipped with an On-Board Unit (OBU) capable of packet exchange with the seven eNBs and, consequently, the single edge server. The choice of the TCP network protocol was deliberate, ensuring reliable and controlled packet transmission for the exchange of parameters in both local and global machine learning models, a crucial consideration for trajectory prediction applications.

Throughout the entire mobility simulation, we captured each vehicle's dwelling time, which signifies the duration a vehicle spends within a specific area, in our case, the time it remains within the coverage of at least one of the seven antennas. This dwelling time information is invaluable for simulating real traffic conditions, which, in turn, can be leveraged to train a model within our federated learning framework.

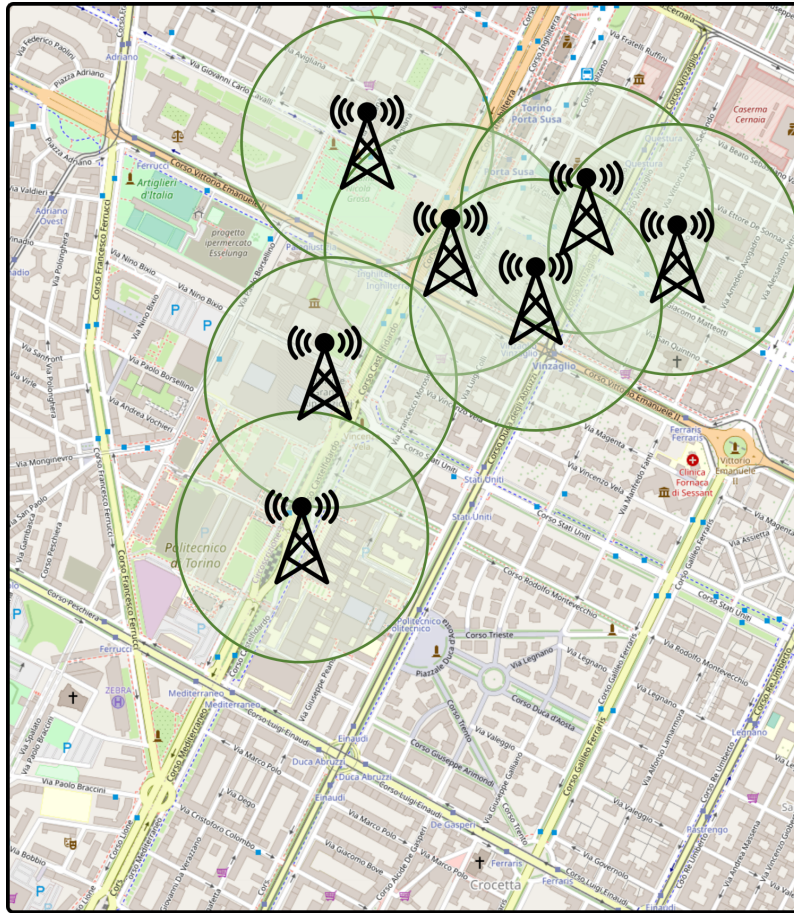


Fig. 5.5 Considered area in Turin with seven eNBs.

It is worth noting that obtaining accurate dwelling time data require conducting several experiments. Ultimately, we determined that using antennas with a 500-meter signal radius, vehicles could reliably be considered within coverage. This setup provided a stable bandwidth, low delay, and virtually negligible packet loss.

The result of the entire simulation process is a file that records, at 100-millisecond intervals over the two-hour duration, each vehicle’s unique ID, and its remaining dwelling time within the coverage area. An example of this output is illustrated in Figure 5.6. It is important to mention that if a vehicle has less than 60 seconds of dwelling time remaining, it is excluded from the dataset, as this duration is deemed insufficient to complete the training phase and transmit the results back to the server during the federated learning process.

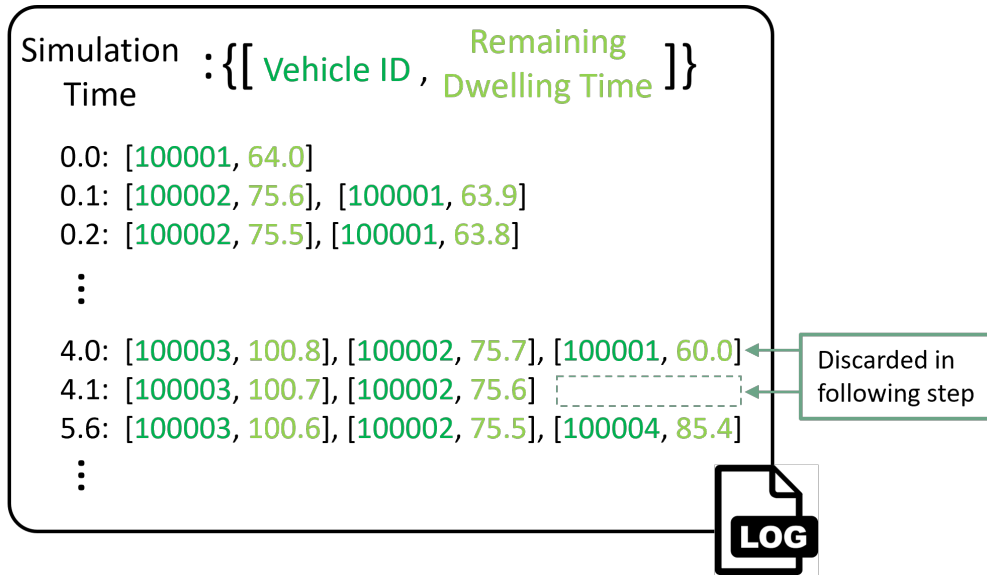


Fig. 5.6 Urban mobility simulation output file.

5.4.3 Federated learning framework

As previously introduced in Section 5.4.2, we rely on the FLOWER framework [20] to implement our customized federated learning framework.

Federated learning operates through a series of rounds during which clients and the server exchange the parameters of trained machine learning models. While the conventional federated learning setup assumes a static and ideal scenario with stationary, always-connected, and reliable clients, our scenario introduces unique challenges. In our case, the clients are mobile, constantly on the move during the federated learning simulation. Consequently, the server must dynamically select different groups of clients in each iteration based on different parameters, such as higher dwelling time, better connectivity quality, higher computation resources available, clients recently used, and others.

In the real-world scenario, client behavior is emulated through the utilization of the ms-van3t framework. In contrast, in the synthetic scenario, clients exhibit a more fixed periodicity and lower frequency of movement when compared to their real-world counterparts.

Figure 5.7 illustrates a schema of the federated learning process. Typically, the simulation comprises K federated learning cycles, each intended to achieve a desired level of accuracy or another termination criterion (e.g., max available time, max

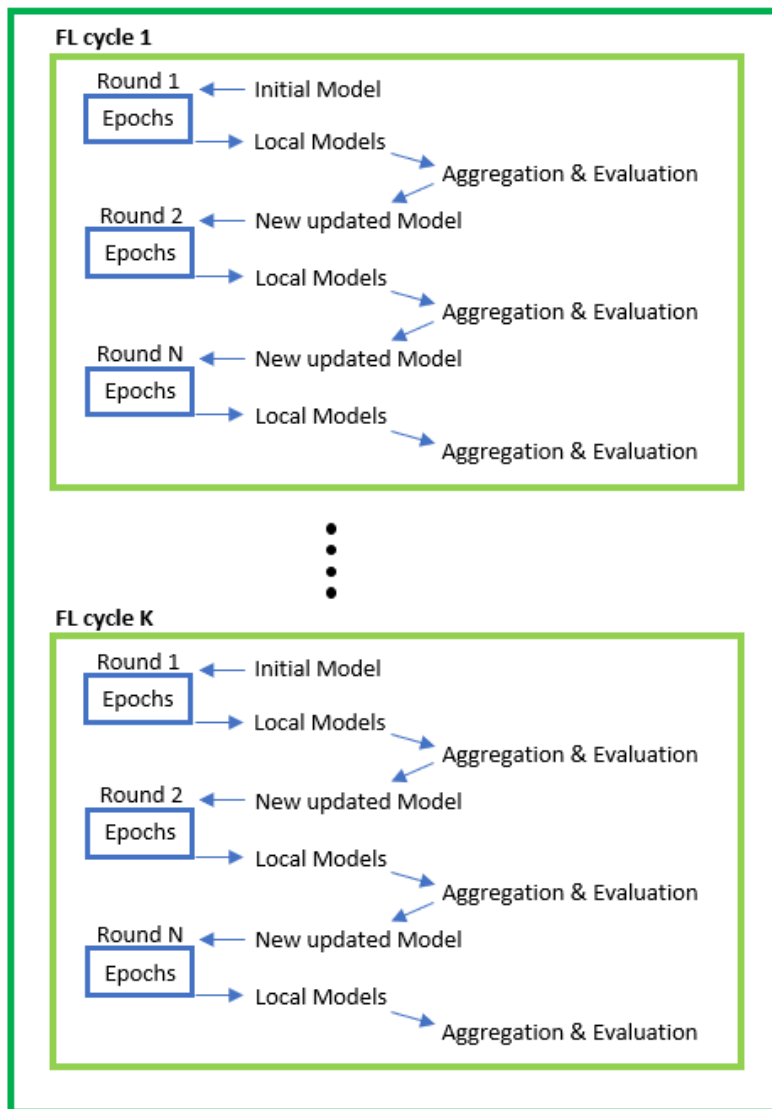


Fig. 5.7 Federated learning schema.

number of FL cycles, etc.). Each cycle consists of N rounds, where the trained model parameters are exchanged between the server and clients to facilitate collaborative learning. This dynamic and adaptive approach ensures the federated learning process effectively accommodates the mobility and behavior of clients in a real-world setting.

Federated learning enhancements

We started from the base implementation available of FLOWER framework and subsequently we introduced several key enhancements. These enhancements include:

- **Early stopping:** To prevent overfitting and optimize model training, two early stopping criteria have been implemented. The first criterion assesses the variation of the `val_loss` variable between two consecutive epochs during an FL round. If the variation falls below a defined threshold, denoted as min_{delta} , it signals a lack of improvement, prompting the round to terminate. The second criterion operates similarly but is applied at the level of the FL cycle, comparing the `val_loss` of the current cycle with the value from the preceding cycle.
- **Consecutive FL cycles:** The default behavior of the FLOWER framework allows for the execution of only a single cycle of federated learning before concluding model training. However, in many use cases, a single cycle may not be sufficient to attain the desired model accuracy. To address this limitation, we implemented a while loop to facilitate multiple consecutive FL cycles. Each cycle, except the initial one, commences training from the final global model of the preceding cycle. This approach ensures that model refinement can continue iteratively until the desired performance level is achieved.

5.5 Results

In this Section, we present the results of our experiments, which encompass a performance evaluation in two distinct client selection scenarios within the federated learning process. Section 5.5.1 introduces the trajectory dataset utilized in the machine learning training process and the LSTM trajectory prediction algorithm. Section 5.5.2 details the experimental setup, including the number of participating clients in the federated learning process, the policies employed, and the network configuration for both scenarios. Finally, in Section 5.5.3, we delve into a discussion of the obtained results and their implications.

5.5.1 Trajectory dataset and LSTM algorithm

The dataset used for training and evaluating the neural network is the TLS Dataset [5]. As depicted in Figure 5.8, this dataset was generated from an aerial drone shot captured over a busy intersection in Cyprus during peak hours. Post-processing of this shot yielded trajectory data for 5,105 vehicles. Of these, 2,699 vehicles, roughly 53% of the total, were recorded as traveling straight through the intersection, while the remaining 2,406 vehicles were categorized as turning vehicles. This near-even distribution of vehicle types ensures that the model is trained consistently for both behaviors at the intersection.

To facilitate federated learning, the dataset is further divided into multiple subsets, each allocated to federated clients. For the LSTM NN evaluation phase, a dataset comprising 200 car trajectories was created, sampled from the larger pool of 5,105 vehicles. During the training phase, the remaining dataset was divided into 170 subsets, each containing 30 vehicle trajectories. When a new client joins the FL process, it is assigned one of these subsets, selected randomly from the 170 available subsets.

The criteria governing the number and size of these subsets are based on the necessity to ensure that each client's round duration is approximately one minute, given the use of 2 vCPUs per client. The pivotal factor guiding this decision is the simulation timing, intrinsically tied to the subset's size. Given that vehicles are mobile and remain within the coverage of eNBs for a limited timeframe, the timing of the federated learning process must be meticulously calibrated. If the round time is too extended, some vehicles may exit the base station's coverage area before completing their training, rendering their local models ineffective in contributing to the aggregation of the global model. On the other hand, overly brief round times result in shorter FL cycles, leading to poorly trained models with reduced accuracy.

The LSTM trajectory prediction algorithm employed in federated learning is detailed in [88]. Notably, this LSTM algorithm, when applied to trajectory prediction training, demonstrates results that surpass the outcomes of other tests performed in a less demanding scenario.

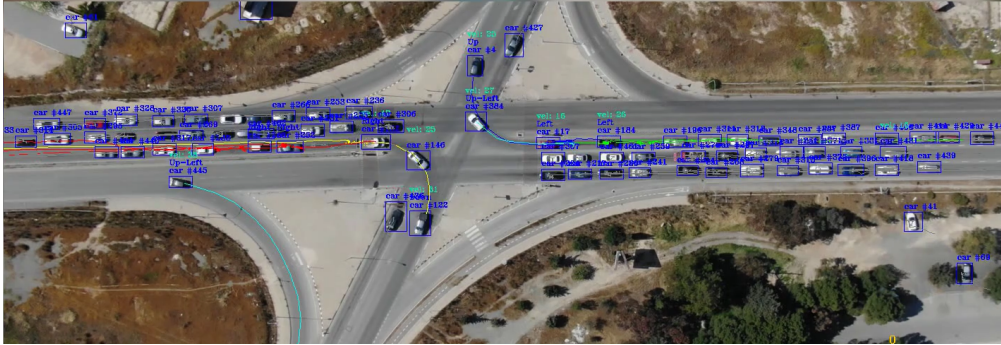


Fig. 5.8 Frame of the video footage took by a drone in Cyprus.

5.5.2 Federated learning framework setup

We have established a federated learning framework, as illustrated in Figure 5.1. In this framework, each individual vehicle, referred to as a *client* in our context, possesses its own local dataset and the computational resources required for training a neural network (NN) model. While the specific process of creating the vehicle dataset lies beyond the scope of this work, we can conceptualize a procedure in which a vehicle, while passing through an intersection, gathers trajectory data from other vehicles at that intersection through vehicle-to-vehicle (V2V) data exchange. The central server, situated at the network's edge, assumes several critical responsibilities within the FL process. These include coordination of the entire FL framework, aggregation of the local models trained by the clients, and dissemination of the updated global model to initiate a new iteration or for inference purposes.

To generate our experimental results, we explored two distinct scenarios, denoted as the *Synthetic* and *Real world* settings. In both scenarios, a fixed number of vehicles, hereafter referred to as *clients*, are engaged in the federated learning process.

In the first scenario, named the *Synthetic* setting, we examine an ideal condition in which the set of clients designated for participation in the FL process remains constant for an entire FL cycle. In a simpler way, a group of clients is chosen at the outset of the FL cycle and remains unchanged throughout the cycle, conducting all rounds. Between successive cycles, a new group of N clients is selected by randomly sampling N sub-datasets from a larger main dataset, comprising a total of 170 sub-datasets. To prevent the repetition of the same subset in consecutive or closely spaced rounds, we maintain a record of the selected sub-datasets in a list. Once all 170 sub-datasets have been used, the list is reset and refilled, cycle by cycle.

Conversely, in the second scenario, we opt for a fixed number of clients for each round. The selection of clients follows a mobility trace derived from a traffic simulation within the Simulation of Urban MObility (SUMO) environment, simulating real-world vehicular traffic in Turin, Italy. This mobility trace is seamlessly integrated with the FLOWER platform via SUMO's built-in Traffic Control Interface (TraCI). More precisely, we identify vehicles predicted to have the longest dwelling time, indicating their suitability for carrying out FL operations. If the initially specified number of clients for a given round is not readily available, the file is continually read until the desired number of clients is reached. The composition of clients may remain consistent from one round to the next, partially change, or undergo a complete overhaul, depending on the mobility trace, particularly with regard to the dwelling time of each client.

Regarding the experimental settings, we have two set of parameters configurations: one related to federated learning and the other concerning the network and system settings.

Regarding the network settings, we assume the following:

- the bandwidth between each eNB and the server is set to 1 Gbit/s bidirectional;
- the eNB-server link has zero delay and no packet loss;
- the emulated network is managed by Docker.

To conduct our runs on the FLOWER platform, we utilized a Linux virtual machine (VM) equipped with 16 virtual CPUs (vCPUs), with both the server and clients running within individual Docker containers.

In terms of FL configuration, the following assumptions are made:

- each client is allocated a maximum of 2 vCPUs. Given the machine's available computational resources, we concurrently run between 2 to 6 clients, alongside the server. It is worth noting that the server has practically no limitations regarding vCPU usage, as it consumes computational resources primarily when the clients are idle, awaiting the next round;
- the maximum number of rounds for every FL cycle is set to 10;
- the maximum number of epochs for each round is set to 10;

- the strategy to aggregate clients' model parameters is FedAvg [58];
- the accuracy evaluation is conducted at the end of every round, and at the end of each FL cycle.

In order to evaluate the accuracy of the trained model, we assess the actual level of accuracy against a testing dataset that encompasses the total number of features observable across all clients' training datasets. The accuracy evaluation is carried out at the end of each FL cycle, and the FL process concludes when a predefined threshold value, representing the target accuracy level, is attained. The evaluation of accuracy is primarily based on the Mean Euclidean Distance (MED) metric. MED is determined during the evaluation phase of the neural network after the training phase. It involves comparing the car trajectories predicted by the NN (in meters) to the real car trajectories. Given a testing dataset comprising 200 car trajectories, this comparison is performed for all 200 trajectories, and the results are averaged to yield the final MED. In accordance with [88], we set the MED threshold at 1.299 meters, as this value is preferred for more challenging scenarios.

Moreover, as introduced in 5.4.3, each round features an early stopping criterion. Specifically, an FL cycle is considered complete when the average validation loss (`val_loss`) among all clients, at the end of a round, reaches 0.001. The threshold value of 0.001 is adopted, in line with the choice made for the original LSTM model. Similar to the early stopping criterion for rounds, there is also an early stopping criterion for epochs. In this case, a round concludes if the `val_loss` between the last two epochs reaches 0.001.

Figure 5.9 shows the final experimental setup schema comprising SUMO for the mobility trace creation, FLOWER for the federating learning process and TraCi for the connection between the two.

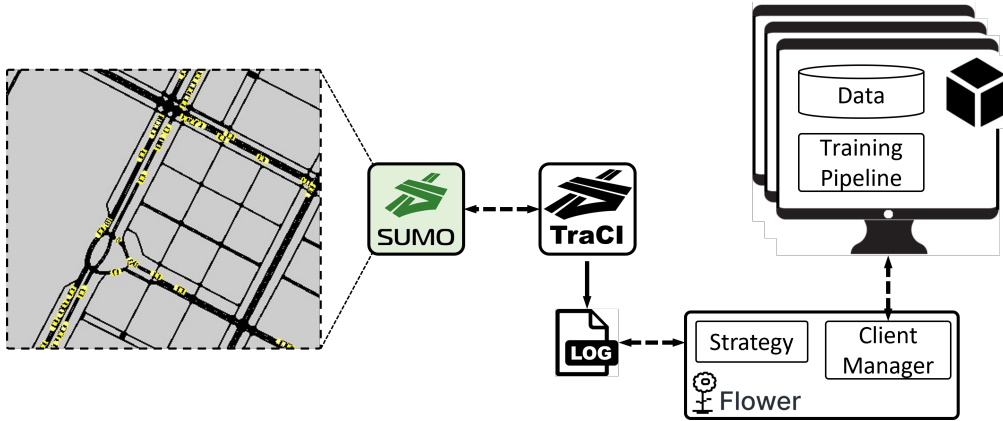


Fig. 5.9 Experimental setup scheme.

5.5.3 Numerical results

As explained in Section 5.5.1, our focus lies on scenarios where a single simulation round does not exceed 1 minute. Examining Table 5.1, specifically the *Time per round* column, we find that the first case, involving a subset of 30 vehicles' trajectories per client, appears to be the most suitable for meeting the simulation time requirements. The second case exhibits round and simulation parameter times that are generally too prolonged to be adaptable in a vehicular dynamic scenarios. This extended timing implies the presence of identical vehicles for an extended duration within the coverage area, a condition that is more challenging to achieve. Consequently, this leads to a diminished availability of vehicles for federated learning training.

Num. vehicles' trajectories	Time per round	Avg. num. of rounds per FL cycle	FL cycle time
30	50 seconds	2.3	3 minutes
120	190 seconds	4.7	15 minutes

Table 5.1 Simulation outcomes varying with different sizes of vehicle trajectories.

The experimentation involved testing both synthetic and real-world scenarios under three different conditions, wherein the maximum number of clients selected for a federated learning round varied. Due to constraints in the available computational resources of the virtual machine, the scenarios were characterized by 2, 4, and 6 maximum clients per FL round. In the next Figures, the Synthetic scenario is represented in blue, while the real-world scenario, based on a trace file derived from a realistic traffic simulation in SUMO, is depicted in yellow.

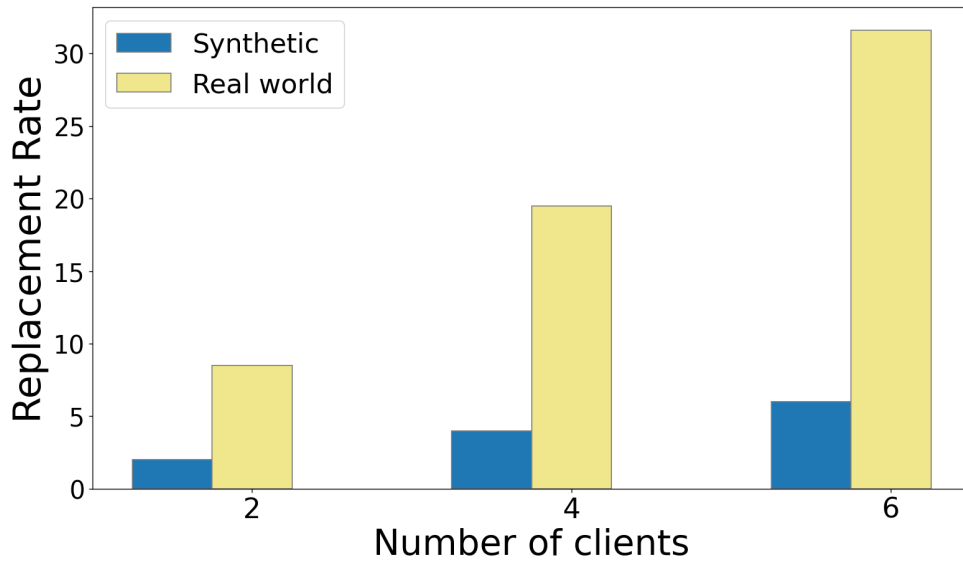


Fig. 5.10 Comparison of replacement rates in synthetic and real-world scenarios across three distinct client selections.

Specifically, Figure 5.10 illustrates the average number of clients replaced after each FL cycle. In the synthetic scenario, where clients change every FL cycle, the values are fixed at 2, 4, and 6. In contrast, for the real-world scenario, the values depicted are determined at the simulation's conclusion by dividing the number of clients used by the number of FL cycles required to achieve the target accuracy.

The high replacement rate in the real-world scenario is attributed to clients changing over time. In each round, the server selects vehicles traveling over the considered area with the highest remaining dwelling time as FL clients. Consequently, the results in Figure 5.10 indicate that, in the real-world scenario, the selection of a new group of clients at each round is more frequent than the usage of the one used in the previous round. It is essential to note that each vehicle or client corresponds to a different dataset, and changing the client results in a change of the dataset. The increased variety of different datasets used in an FL cycle in the real-world scenario allows for training the model with a broader range of information, ultimately leading to a lower MED at the end of the FL cycle when evaluating the model on the server side.

Analyzing the statistics related to the number of federated learning cycles, FL rounds, and training time provides a clearer understanding of the results.

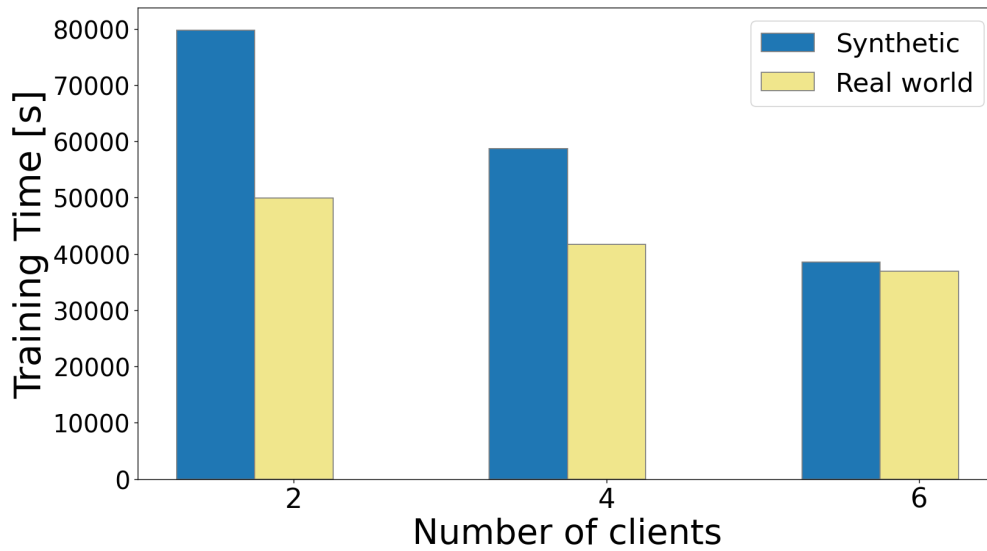


Fig. 5.11 Comparison of training times in synthetic and real-world scenarios across three distinct client selections.

Figures 5.11, 5.12, and 5.13 reveal a consistent pattern. Real-world scenarios consistently demonstrate quicker achievement of model accuracy, requiring fewer FL rounds and cycles compared to synthetic settings.

The shared characteristic among these graphs is the inverse relationship with the replacement rate. As the total number of clients used in an FL cycle increases, training incorporates a larger quantity of data, resulting in the utilization of a more diverse dataset at each FL cycle. This, in turn, enhances accuracy within a shorter timeframe, indicating fewer rounds per cycle.

Figure 5.11 highlights an interesting observation: both synthetic and real-world scenarios exhibit similar training times with 6 clients, while training times with 2 and 4 clients differ significantly. The discrepancy arises from the insufficient data available in the training session with only two clients. Repeating rounds with the same two clients becomes futile, as model performance plateaus early. Consequently, the synthetic case requires more FL cycles compared to the real-world scenario, where clients change every round, allowing for more substantial improvements in each FL cycle. In contrast, with 6 clients, the neural network has enough data in each cycle, eliminating the need for an early stop, and accuracy consistently improves round after round.

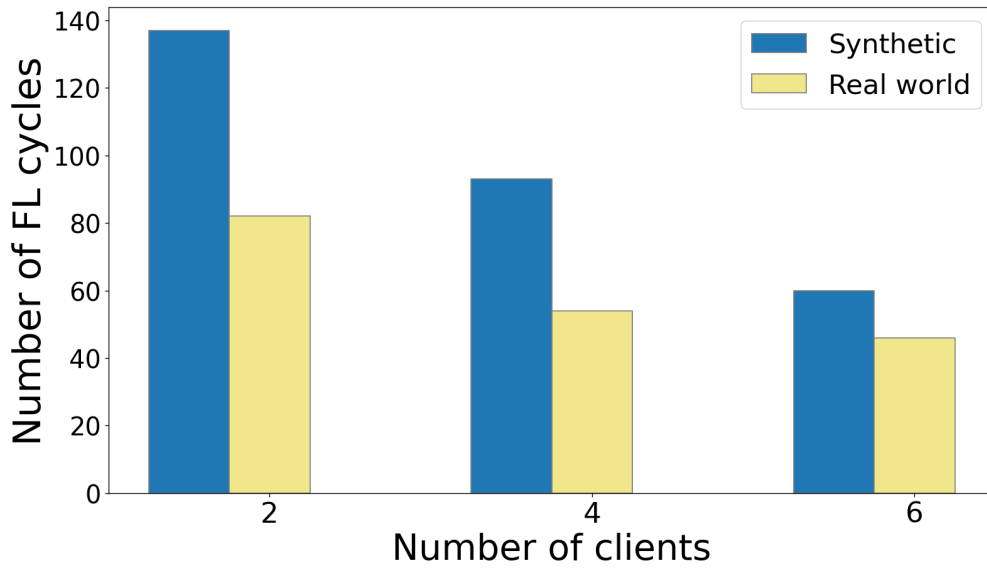


Fig. 5.12 Comparison of FL cycle counts in synthetic and real-world scenarios needed to reach a set threshold across three distinct client selections.

Figure 5.13 corroborates this trend, demonstrating that real-world settings necessitate fewer FL rounds per cycle than synthetic settings, contributing to the overall lower execution time in the real-world scenario.

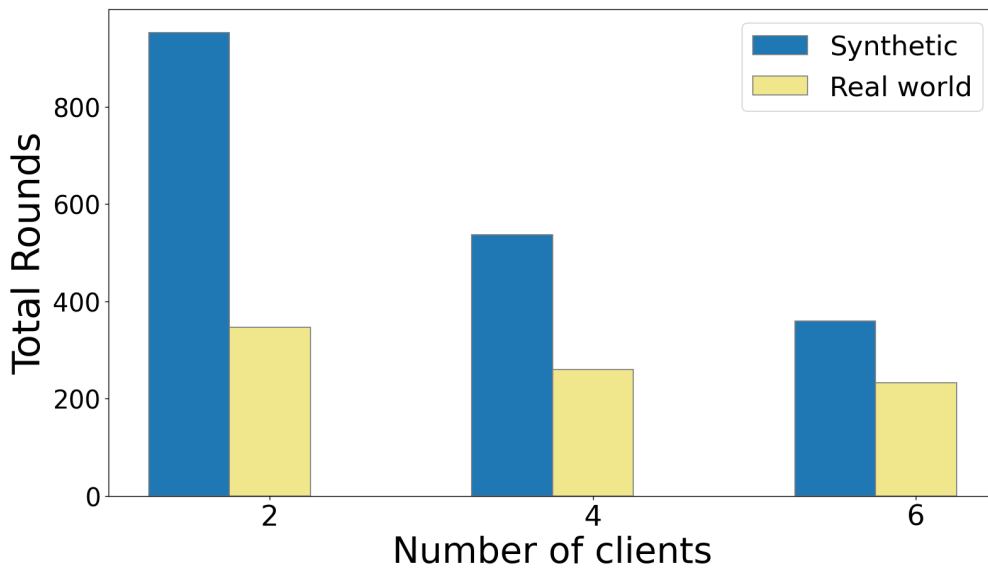


Fig. 5.13 Comparison of FL round counts in synthetic and real-world scenarios to attain a specified threshold across three distinct client selections.

In summary, real-world scenarios exhibit a proportional increase in the number of clients participating in an FL cycle relative to the maximum number of federated participants in an FL round. This trend underscores the efficiency of real-world scenarios in achieving model accuracy within shorter timeframes, highlighting their practical viability for federated learning applications.

Additionally, it is noteworthy that through federated learning, we can attain equivalent accuracy, as evidenced by achieving $MED = 1.299$ m, akin to the centralized ML approach detailed in the study [88]. Notably, this accomplishment is achieved within a reasonable timeframe while preserving data privacy.

5.6 Conclusions and future works

Advancements in automotive technology highlight a dual commitment: a persistent dedication to enhancing safety for both vehicle occupants and pedestrians, paired with a focused effort to improve the urban mobility experience. Modern vehicles seamlessly integrate state of the art sensors and technologies, leading us towards an era where cars will produce extensive data. This growing data landscape holds potential insights and transformative possibilities, significantly influencing service delivery and user experiences. As we actively explore advanced methodologies like federated learning to unlock this data potential, it is crucial to acknowledge the associated challenges. Notably, addressing the computational power and energy resource requirements demands careful consideration. Therefore, in this era of innovation and evolution, finding a balanced approach between technological advancement and sustainable resource utilization becomes imperative.

In the course of our study, we delved into innovative solutions, with a specific focus on applying FL in an urban environment where vehicles collaboratively function as nodes. This cooperative paradigm ensures data privacy, allowing vehicles to securely keep their data localized while collectively training a NN model. The orchestration of this collaborative learning occurs at the network's edge, with a server overseeing the entire process. Utilizing a realistic vehicle trace, we implemented the proposed system using the FLOWER [20] platform, employing federated learning to study its performance with varying numbers of vehicles acting as FL clients.

Looking ahead, our focus will be on overcoming existing hardware limitations to create a more comprehensive emulation scenario. This involves investigating diverse client selection and replacement criteria to further optimize the FL process. Our ongoing research is committed to extending the application of FL in vehicular networks and other domains where there is a demand to distribute the load among different clients while ensuring the privacy preservation of the data used to train machine learning models.

Chapter 6

Radio Frequency mobility scenario for wireless channel emulators

In the contemporary landscape of wireless communication, the surge in large-scale wireless emulation has become a pivotal force, offering substantial potential for advancing the development and deployment of sophisticated use cases within the realm of next-generation wireless networks. This burgeoning field has given rise to various innovative applications, notably encompassing massive Multiple Input Multiple Output (MIMO), millimeter wave beamforming, and Artificial Intelligence (AI)-driven Vehicle-to-Everything (V2X) optimized communication. The endeavor to develop and test wireless applications, particularly when dealing with mobile nodes on a significant scale, encounters multifaceted challenges that extend beyond the capabilities of simulation frameworks alone. As a result, the emergence of massive-scale channel emulators has become imperative, providing the means to emulate realistic scenarios by harnessing authentic hardware and radio signals. However, the complexity of this task is underscored by the scarcity of realistic scenarios grounded in actual datasets.

In this Chapter, we present a novel framework tailored for the design and generation of channel emulation scenarios derived from authentic mobility traces—whether generated through dedicated tools or collected in real-world settings. Our primary goal is to offer a pragmatic approach to the generation of mobility scenarios featuring diverse entities such as vehicles, pedestrians, drones, and other mobile elements. Furthermore, we demonstrate the efficacy of our proposal by designing and con-

structuring a vehicular 5G scenario, and we validated it using the Colosseum channel emulator [24].

6.1 Research motivation and State of the Art

The advent of 5G and the ongoing exploration of the forthcoming 6G network generation have propelled wireless channel emulators into a central role as catalysts for the evolution of mobile networks. Researchers are increasingly directing their focus towards harnessing Artificial Intelligence (AI) and advanced Deep Machine Learning techniques to bolster wireless solutions, particularly in the context of applications such as Massive MIMO, wireless signal beamforming, and Vehicle-to-Everything (V2X) communications.

In response to the stringent demands set by flagship communities like Hexa-X [7] for B5G/6G development, the conventional approach of creating synthetic testbeds with simulated datasets has become impractical. The complexity of developing, deploying, and on-field testing applications necessitates a paradigm shift toward reliable simulation and emulation of networks using real data, especially in the pre-deployment phase and in scenarios involving large-scale networks.

While simulation frameworks offer a viable means of evaluating wireless networks, particularly those involving mobile nodes, they are constrained by mathematical models that focus on essential aspects of real environments, often relying on probabilistic considerations. As such, they fall short in capturing the full spectrum of variables in complex environments. Consequently, there is a need for a progressive move towards the emulation of large-scale Radio Frequency (RF) scenarios, involving real hardware and physical radio signals processed by Software Defined Radios (SDRs) in dedicated testbed environments.

Flexibility and repeatability emerge as pivotal features sought in testbeds by researchers, a need underscored by the authors of [104], where an FPGA-based network channel emulator was developed to support real-time emulation of the interaction of different RF nodes. However, as the demand for large-scale experiments increased, challenges arose, including the proper emulation of numerous signals without depleting available computational resources and the prohibitive cost associated with simulating large-scale scenarios using dedicated hardware.

To address these challenges, DARPA (Defense Advanced Research Projects Agency is USA) developed **Colosseum** [24], the world’s most powerful wireless network emulator, which combines 128 Standard Radio Nodes (SRNs) with a Massive digital Channel Emulator (MCHEM) supported by an extensive FPGA routing fabric. Colosseum, made available to industry, universities, and research groups by Northeastern University, stands as the largest RF emulator globally, enabling the emulation of up to 256 independent radio nodes and 65,535 wireless channels in expansive operational environments. At the heart of Colosseum are RF and traffic scenarios, which emulate not just the channels between transmitters and receivers but also replicate the typical effects of the wireless propagation environment.

Colosseum provides publicly-available experimental scenarios, such as (i) the *alleys of Austin*, which engages 50 nodes (comprising 45 pedestrian users and 5 Unmanned Aerial Vehicles) in the exchange of voice traffic or file transmission, and (ii) *SCE Qualification*, involving 10 nodes in the exchange of UDP packets at a constant bitrate. Additionally, cellular scenarios are accessible, featuring 8 to 10 base stations serving four mobile users each. These scenarios simulate both moving and stationary pedestrians under the coverage of the base stations.

Recently the work in [102], presented a synthetic V2X scenario for the Colosseum emulator, depicting an area around Tampa, FL. Their scenario, based on a tap and channel approximation framework previously presented in [97], mirrors the foundation of our work. This V2X scenario involves one Road Side Unit (RSU) and three On Board Units (OBUs or vehicles) moving on a straight path at a constant speed, with routes and trajectories simulated using the commercial ray-tracing software, Wireless InSite[®] [11].

6.2 Main contributions

The current study, as detailed in [84], pioneers a framework for the practical generation of RF scenarios, originating *from real collected data*. Additionally, we delineate the comprehensive steps essential for crafting V2X RF scenarios within Colosseum, basing our scenario on a recently-released open dataset [75], featuring traces from 19 vehicles navigating an urban environment.

Our research contributes significantly to the existing literature in the following key domains:

- **Data-driven framework for RF mobility scenario creation:** We introduce a groundbreaking framework that initiates RF scenario creation from authentic mobility traces, embracing a data-driven paradigm.
- **Comprehensive Mobility Scenario for Vehicular Network Emulation in Colosseum [24]:** Utilizing our framework, we have crafted the inaugural V2X scenario tailored for Colosseum, integrating real mobility traces and employing a cutting-edge ray-tracing algorithm for precise channel path loss computations.

In the subsequent Sections, we pivot our focus towards elucidating the concept of a wireless channel emulator and highlighting the most renowned and potent options available today. Following this, we present our framework, which employs a data-driven approach and real-world data, for designing an RF mobility scenario.

Ultimately, we furnish a thorough and detailed account of the implementation of a V2X scenario, utilizing the developed framework within the Colosseum emulator, encompassing every aspect from design to rigorous validation.

6.3 Wireless channel emulators

The complexities inherent in wireless communication systems necessitate a refined methodology for testing and validating their performance. Central to this undertaking are wireless channel emulators—sophisticated instruments meticulously crafted to replicate the unpredictable and dynamic characteristics of real-world communication channels. This emulation process enables to test wireless devices and protocols to meticulously controlled yet authentic conditions, facilitating a thorough exploration of their capabilities and constraints.

Moreover, the significance of channel emulators lies in their ability to deliver reproducible results. Every test is conducted within a consistent channel environment, eliminating the impact of external variables such as weather conditions and interferences. This consistency ensures that experiments are not swayed by extraneous factors, allowing for precise and reliable assessments of wireless system

performance. In essence, wireless channel emulators serve as indispensable tools, not only unraveling the intricacies of communication systems but also providing a stable and controlled platform for the systematic evaluation of devices and protocols.

The main characteristics of a wireless channel emulator include:

- **Fading models:** A defining characteristic of wireless channels is the phenomenon of signal fading. As signals traverse the air, they are susceptible to variations in amplitude and phase due to factors such as multipath propagation, atmospheric conditions, and obstacles in the transmission path. To mimic these real-world dynamics, wireless channel emulators incorporate fading models. These models, simulate the fluctuations in signal strength that occur in practical scenarios.
- **Path loss:** Wireless signals experience attenuation as they propagate through space, a phenomenon known as path loss. Wireless channel emulators replicate this distance-dependent signal attenuation to emulate the realistic effects of signal propagation. This capability enables the possibility to study how different devices and protocols perform under varying signal strengths and distances, crucial for optimizing communication systems.
- **Interference generation:** In real-world scenarios, wireless devices must contend with interference from other signals and external noise sources. Emulators simulate interference to evaluate a system's resilience in the presence of competing signals. This aspect is particularly crucial in crowded frequency bands where multiple devices coexist. By introducing controlled interference, it is possible to assess the performance and reliability of wireless systems under challenging conditions.
- **Multipath effects:** Multipath propagation, resulting from signal reflections, can introduce delays and variations in signal amplitude. Emulators recreate these multipath effects, giving the possibility to analyze the impact on signal integrity.
- **Reproducibility:** Channel emulators offer reproducibility by maintaining consistent channel conditions across multiple tests. This characteristic ensures that experiments yield comparable results, allowing for meaningful comparisons and analyses. External factors such as weather conditions, atmospheric

changes, and other environmental variables are minimized or eliminated, contributing to the stability and reliability of the testing environment.

- **Flexibility and configurability:** Many modern emulators adopt a software-defined radio (SDR) approach, providing flexibility for researchers to configure and customize the emulation environment based on specific testing requirements. Furthermore, emulators are designed to scale, accommodating a broad range of testing scenarios and evolving with advancements in wireless technologies.

In recent years, various platforms have emerged in response to the growing demand for high-fidelity testing of wireless channels within the research community.

POWDER [26]: POWDER represents a highly adaptable, remotely accessible, end-to-end software-defined platform designed to support a diverse spectrum of wireless and mobile research. This platform empowers researchers to construct their own 5G networks using open-source software stacks like OpenAirInterface [3] and srsRAN [12]. Offering end-to-end programmability, POWDER provides complete control over both the Radio Access Network (RAN) and core components, along with the services running within. Whether in a confined indoor setting or an expansive outdoor environment with multiple gNodeBs and authentic mobile devices, experimenters can build and evaluate networks.

Drexel Grid [30]: Drexel Grid integrates physical Software-Defined Radios (SDRs) to establish a cohesive experimental framework for the swift prototyping and assessment of a diverse array of wireless systems. This is achieved through the utilization of field measurements, which allow for the real-time evaluation of transceiver and channel-specific effects. Additionally, the platform employs network emulation to scrutinize systems on a large scale, presenting controllable and repeatable propagation channels.

Arena [19]: Arena stands out as an open-access wireless testing platform structured around a grid of antennas positioned on the ceiling within a spacious office environment. Each antenna is intricately linked to programmable SDRs, providing the capability for research in the sub-6 GHz 5G and beyond spectrum. This setup facilitates comprehensive investigations into various wireless scenarios.

Colosseum [24]: Colosseum, distinguished by its assembly of 256 SDRs and 128 remotely accessible SRNs and GPUs, offers a robust platform for testing full-protocol

stack solutions at scale. It supports experimentation with real hardware devices and provides emulated and realistic environments with intricate channel interactions, including path loss, fading, and multipath effects. Beyond its experimentation capacities, Colosseum serves as an innovative space for AI exploration and functions as a wireless data factory, enabling the creation of large-scale datasets, mobility RF scenarios and the training/testing of solutions within a secure and controlled setting.

6.4 Methodology

In this Section, we delve into the methodology we proposed in order to create mobile radio frequency scenarios for wireless channel emulators.

6.4.1 Framework

We introduced a framework designed for the creation of RF scenarios intended for large-scale channel emulators. The generation process leverages real mobility data, sourced either from specialized tools or directly gathered on the field.

The workflow of our solution encompasses distinct steps for input, processing, and output, as illustrated in Figure 6.1. In this Figure, various elements have been

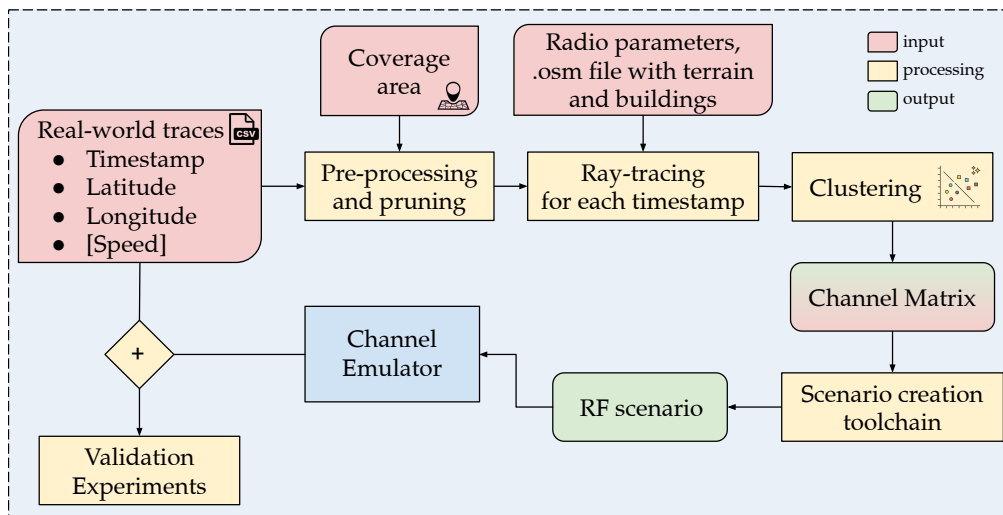


Fig. 6.1 Overview of the proposed framework for generating and validating RF mobility scenarios.

color-coded to provide a visual guide. Red rectangles denote the input components, offering a clear distinction for real-world traces, coverage areas, and radio parameters. Processing stages, represented by yellow boxes, showcase the tasks where the framework manipulates the input data to build dynamic and realistic RF scenarios. Finally, the green rectangle signifies the output stage. Additionally, the MATLAB implementation of the entire framework is available in a public GitHub repository [80].

Our framework mandates three primary inputs:

1. **Real-world traces:** The core dataset comprises real-world traces presented in CSV format. These traces must include a timestamp accurate to the millisecond, along with the precise location of each node denoted by latitude and longitude.
2. **Coverage area:** A well-defined coverage area is a critical input for the framework. This spatial parameter delineates the region within which the RF scenario will be generated, ensuring that the emulation accurately reflects the intended operational environment.
3. **Radio parameters:** The generation of RF scenarios requires specific radio parameters. This includes crucial details such as transmission power for User Equipment (UE) and base stations (eNB or gNB), or the equivalent parameters for WiFi-based scenarios involving mobile nodes and Road Side Units (RSU). Additionally, the set encompasses operating frequency details (e.g., 1 GHz for generic cellular-based scenarios or 5.9 GHz for V2X DSRC scenarios), the spatial positioning of any base station, and the antenna height. An OpenStreetMap (.osm) file can also be incorporated to provide terrain and building information within the designated coverage area, enriching the scenario with realistic channel interactions.

The framework's output manifests as a channel matrix, structured as *Number of nodes* \times *Number of nodes* \times *Number of timestamps*. This matrix encapsulates, for each link and timestamp, the Finite Impulse Response (FIR) filter delay and In-phase and Quadrature (IQ) coefficients, elucidating the path gain and phase shift of every signal.

The interplay within the framework involves various tasks leading to this output, with key steps highlighted in yellow boxes in Figure 6.1.

Task 1 - Pre-processing and pruning

The initial phase entails pre-processing a set of traces, hereafter referred to as the *dataset*, and pruning mobile nodes that do not enter the coverage area during specific time instances. It is noteworthy that, despite the capabilities of the channel emulators, usually the coverage area is limited to a few squared kilometers, thus the selection of a meaningful coverage area is crucial. Consequently, the pre-processing module empowers users to discern the temporal distribution of nodes, aiding in the judicious choice of a geographic area. This is exemplified in Figure 6.2, where all vehicles’ GPS coordinates are represented for a specific timestamp. The green and red dots indicate whether a vehicle is inside or outside the coverage area delineated by the square with blue borders.

Task 2 - Ray-tracing

Commencing from the input radio parameters the framework engages in ray-tracing to calculate path loss gain and phase shift — essential inputs for subsequent

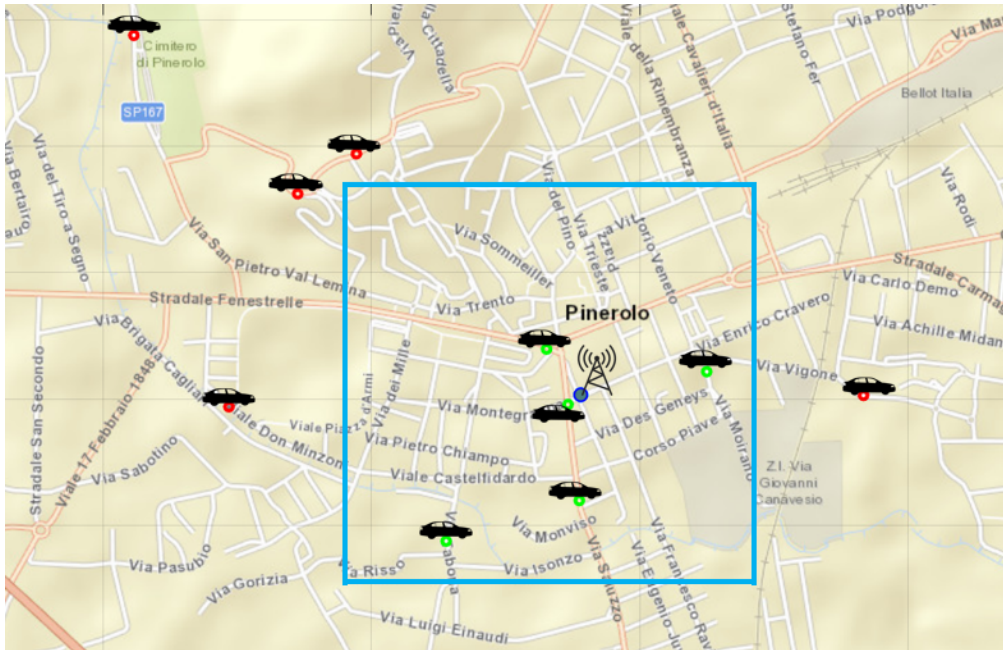


Fig. 6.2 Visualization of vehicles on a map for a specific timestamp, marking with green dots if the vehicle is inside the coverage area (blue borders) and red dots if outside.

tasks. Optionally, users can specify building and terrain materials, along with the maximum number of reflections. If unspecified, the latter defaults to the third order, a configuration found, through our tests, to yield realistic scenarios without introducing excessive low-power reflections. A noise level can also be configured to prune less significant rays.

Ray-tracing stands out as a pivotal task, particularly in wireless communication systems where signals traverse multiple propagation paths, encountering reflections, diffractions, and unpredictable penetration patterns from ground and buildings. Consequently, radio waves taking distinct paths reach the receiver at varying times, each path inducing diverse phase conditions owing to different lengths. Thus, at the receiver, disparate power levels are observed for each frequency component. Figure 6.3 visually illustrates the output of the ray-tracer, designating the antenna (receiver) with a blue point and the vehicle (transmitter) with a red point. Each ray represents reflections and diffractions of the signal and the different color indicate the different power level.

Accurately modeling the behavior of radio waves in mobile scenarios poses a formidable challenge. Relying solely on simplistic path loss models, such as those advocated by 3GPP [37], may fall short of achieving the desired emulation accuracy. In contrast, a ray-tracing algorithm proves invaluable for determining the path loss and phase shift of each ray, involving electromagnetic analysis and considering



Fig. 6.3 Visualization of the ray-tracing output, depicting the receiver with a blue point and the transmitter with a red point. Each ray represents reflections and diffractions of the signal, with varying colors indicating different power levels.

horizontal and vertical polarization. This process entails tracing the propagation path of signals transmitted from mobile nodes (transmitters) to the designated receivers, whether fixed eNodeB (for 4G), gNodeB (for 5G), or other mobile devices.

Task 3 - Clustering

The ray-tracer output often yields numerous multi-path components, but due to computational complexities and storage constraints, channel emulators accommodate up to K non-zero-valued channel taps (i.e, usually $K = 4$). To curtail the number of components while adhering to this limit, our framework incorporates a two-step clustering procedure outlined in [97].

In the first sub-task of clustering, a machine learning clustering algorithm is applied to identify centroids — each corresponding to a channel tap — that effectively capture the characteristics of components within the cluster. We employ a K-means algorithm utilizing the Multi-path Component Distance (MCD) as distance function. As demonstrated by the authors of [29], MCD outperforms classical Euclidean distances (e.g., Squared Euclidean Distance — SED — and Joint SED [97]) in channel data clustering, considering both time of arrival and angles of arrival/departure of multiple paths.

Subsequently, an approximate taps re-sampling step aligns each centroid with the specific FIR tap of the channel emulator, as detailed in [97]. This ensures synchronization of the centroid's time of arrival with the channel emulator's FIR filter indexes.

Algorithm 4 and 5 present the core aspects of the clustering algorithm. Notably, we enhance the code compared to [97] by introducing a random permutation of centroid positions during initialization (line 20). This guarantees that each centroid, representing the approximated position of a channel tap, is assigned at least one path. Subsequently, for reconstructing the approximated taps, the gain of each centroid is computed as the sum of all multi-path components within its cluster. This process iterates for each pair of transmit-receive nodes and for every available timestamp in the set of traces.

Task 4 - Scenario creation and installation

The clustering algorithm produces a channel matrix structured as *Number of nodes* \times *Number of nodes* \times *Number of timestamps*. For each link and timestamp

Algorithm 4 K-means MCD clustering algorithm.

```

1: Input:  $K$  = number of channel taps,  $rays$  = output rays from ray-tracing,  $P_{th}$  =
   power level threshold,  $TXPOWER$  = transmitter power,  $Nrep$  = number of
   iterations,  $tx$  = transmitter,  $rx$  = receiver
2: Output:  $taps$  = approximated taps

3: Create table  $X\_table$  with columns:
4:     PropagationDelay, PathLoss, AngleOfArrival_az, AngleOfArrival_el,
5:     AngleOfDeparture_az, AngleOfDeparture_el, hi
6: and values extracted by  $rays$ 

7: if height( $X\_table$ ) >  $K$  then
8:      $X\_l$  = mcdkmeans( $X\_table$ ,  $K$ ,  $P_{th}$ ,  $TXPOWER$ ,  $Nrep$ ,  $tx$ ,  $rx$ ,)
9:     for  $k = 1, \dots, K$  do
10:          $Hck(k)$  = sum of all multi-path components  $X\_l$  within the cluster
11:     end for
12:     Construct table  $taps$  from  $Hck$  with columns: delay, h
13: else
14:      $K$  = height( $X\_table$ )
15:      $X\_l$  =  $X\_table$ 
16:     Construct table  $taps$  from  $X\_l$  with columns: delay, h
17: end if

18: procedure MCDKMEANS( $X$ ,  $K$ ,  $P_{th}$ ,  $Ptx\_dB$ ,  $Nrep$ ,  $tx$ ,  $rx$ )
19:     Create  $X\_l$  by filtering  $X$  based on  $P_{th}$ 
20:     Initialize cluster centroids  $c_{i/k}$  with a random permutation
21:     for  $nr = 1, \dots, Nrep$  do
22:         for  $xli = 1, \dots, \text{height}(X\_l)$  do
23:             Update  $X\_l$  based on MCD distances
24:         end for
25:         for  $k \leftarrow 1$  to  $K$  do
26:             Update cluster centroids  $c_{i/k}$ 
27:         end for
28:     end for
29:     return  $X\_l$ 
30: end procedure

```

Algorithm 5 CIR re-sampling algorithm.

```

1: Input:  $taps$  = approximated taps
2: Output:  $tap\_delays$  and  $tap\_gains$ 

3:  $ds = 10^{-9}$ 
4:  $fs = 1/ds$  ▷  $fs$  is the FIR filters sampling frequency
5:  $N = 512$  ▷  $N$  is the number of FIR filters
6: Initialize arrays  $tap\_delays$  and  $tap\_gains$  with zeros
7: for  $n = 1, \dots, N$  do ▷ Initialization of  $tap\_delays$  and  $tap\_gains$ 
8:    $tap\_delays(n) = n \times ds$ 
9:    $tap\_gains(n) = 0 + 0i$ 
10: end for
11: for  $k = 1, \dots, \text{height}(taps)$  do
12:    $i = \text{round}(taps(k).delay/ds)$ 
13:    $tap\_gains(i) = tap\_gains(i) + taps(k).h$ 
14: end for

```

within this matrix, it encapsulates the FIR filter delay and IQ coefficients, elucidating the path gain and phase shift of individual signals.

Subsequently, this channel matrix feeds into a specialized scenario generation toolchain tailored for the designated channel emulator. The outcome is an RF scenario ready to be installed into the emulator. Once the RF scenario is installed, it facilitates realistic experimentation with interconnected mobile nodes, such as WiFi clients or UEs, through the channel emulator.

A relevant example of toolchain, designed explicitly for the Colosseum emulator, is the Channel Emulation Generator and Sounder Toolchain, abbreviated as *CaST*, as introduced in [102]. Taking the previously defined channel matrix as its input, *CaST* outputs and installs the provided Colosseum scenario.

6.5 V2X scenario in Colosseum

In this Section, we elucidate the development of a realistic V2X scenario within the Colosseum emulator, leveraging our innovative methodology and a recently published, high-precision open dataset [75]. Section 6.5.1 provides an in-depth introduction to the vehicular dataset essential for extracting authentic real-world traces. Section 6.5.2 meticulously outlines the comprehensive processing pipeline

employed to transform these mobility traces into a channel matrix. This matrix is pivotal for configuring the RF scenario within the Colosseum emulator. Finally, in Section 6.5.3, we delve into the validation of the V2X scenario showing the effectiveness of our approach when crafting, implementing, and harnessing a realistic communication environment featuring mobile nodes.

6.5.1 Vehicular dataset

At the heart of our scenario lies a comprehensive vehicular dataset, capturing the traces of 19 vehicles navigating across both urban and suburban terrains, around the city of Pinerolo, Italy. This dataset, aptly named the Synthetic Accurate Multi-Agent RealistiC Assisted-gNss Dataset, abbreviated as *SAMARCANDA*, has been graciously provided to the research community by the authors referenced in [75].

Each vehicle’s dynamic information is stored in CSV files, encompassing metrics such as latitude, longitude, heading, speed, acceleration, and precise timestamps corresponding to each data point. A visual representation of this data structure is available for reference in Figure 6.4.

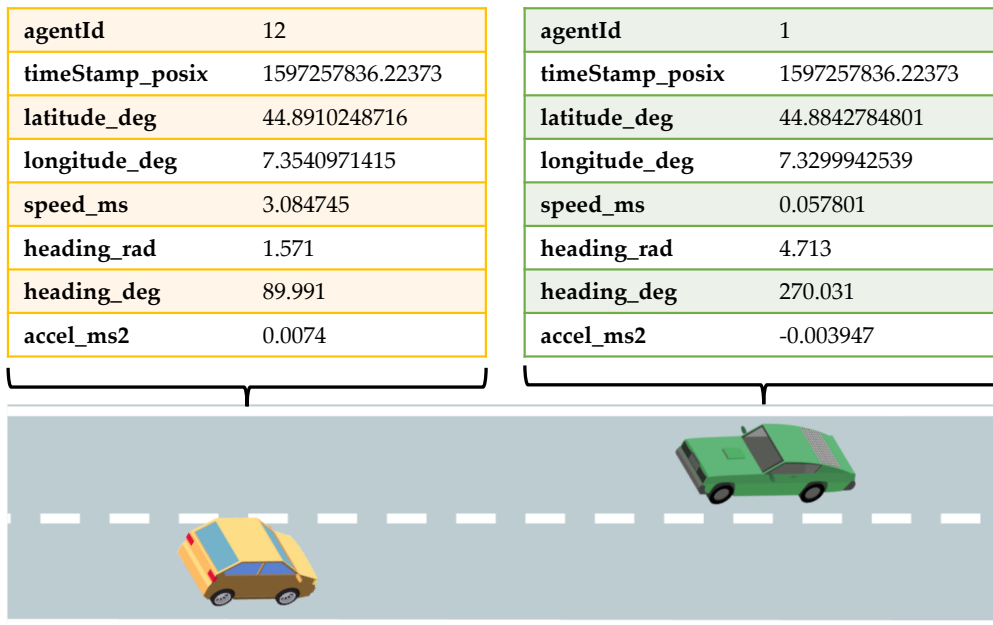


Fig. 6.4 A sample of the data available within the open dataset [75] leveraged for the creation of our V2X scenario.

The dataset captures updates for each of the 19 vehicles at a frequency of approximately 100 ms, setting the minimum time granularity achievable within our scenario. This granularity proves sufficient for emulating the majority of V2X use cases that necessitate frequent dynamic data updates. Additionally, the 10 Hz periodic message frequency aligns with the maximum frequency stipulated by the European Telecommunications Standards Institute (ETSI) [38].

Employing real traces ensures that the dataset authentically mirrors the behavior of actual vehicles, empowering researchers to evaluate V2X applications with a level of precision that surpasses conventional mathematical mobility models. The data collection process leveraged a sophisticated Global Navigation Satellite System (GNSS) device operating at a 10 Hz update rate, complemented by Real-Time Kinematic (RTK) corrections and an Inertial Measurement Unit (IMU). This configuration enhances the dataset's granularity, capturing nuanced dynamic information like acceleration.

The emulated scenario encapsulates an urban environment centered around Pinerolo's city center, exemplifying the characteristics of a mid-sized city with a bustling downtown area. Given Colosseum's capability to define an area spanning up to 1 km², we judiciously selected Pinerolo's city center as the coverage zone. This locale predominantly houses a significant concentration of vehicles, thereby facilitating a more nuanced emulation of urban dynamics, particularly the impact of architectural structures. This central area choice further refines the emulation's fidelity, especially when juxtaposed with the more rural regions encompassed by the SAMARCANDA dataset.

As previously highlighted, our framework necessitates a choice between emulating a cellular-based or DSRC-based scenario, thereby setting the requisite input radio parameters. Given the prevalent requirements of innovative V2X services, characterized by low latency and high throughput, our emphasis gravitates towards leveraging 5G connectivity facilitated through a centralized base station, specifically a central gNB.

The adoption of centralized methodologies for automated and connected vehicles, including techniques like centralized Federated Learning (FL) and advanced automated maneuver management, has garnered significant attention. Numerous European Projects have substantiated the efficacy of these centralized approaches, particularly when synergized with a dependable 5G infrastructure [1]. Moreover, we

posit that such centralized strategies hold substantial appeal for the research community. Centralized frameworks often yield a more refined and precise understanding of road dynamics compared to their purely decentralized counterparts [76]. Considering the reasonable range for a 5G base station, when focusing on centralized V2X scenarios, a 1 km^2 square appears to be technically sound as maximum emulation area, out of which vehicles can be considered out of coverage.

In light of the operational parameters inherent to a 5G base station, a coverage area encompassing a 1 km^2 square emerges as a judicious choice for maximum emulation. This delineation ensures that vehicles positioned beyond this boundary are pragmatically considered to be outside the coverage ambit, aligning with the technical constraints and objectives of centralized V2X scenarios.

For visual clarity, Figure 6.5 illustrates the primary area encompassed by the SAMARCANDA traces in conjunction with the designated 1 km^2 coverage zone.

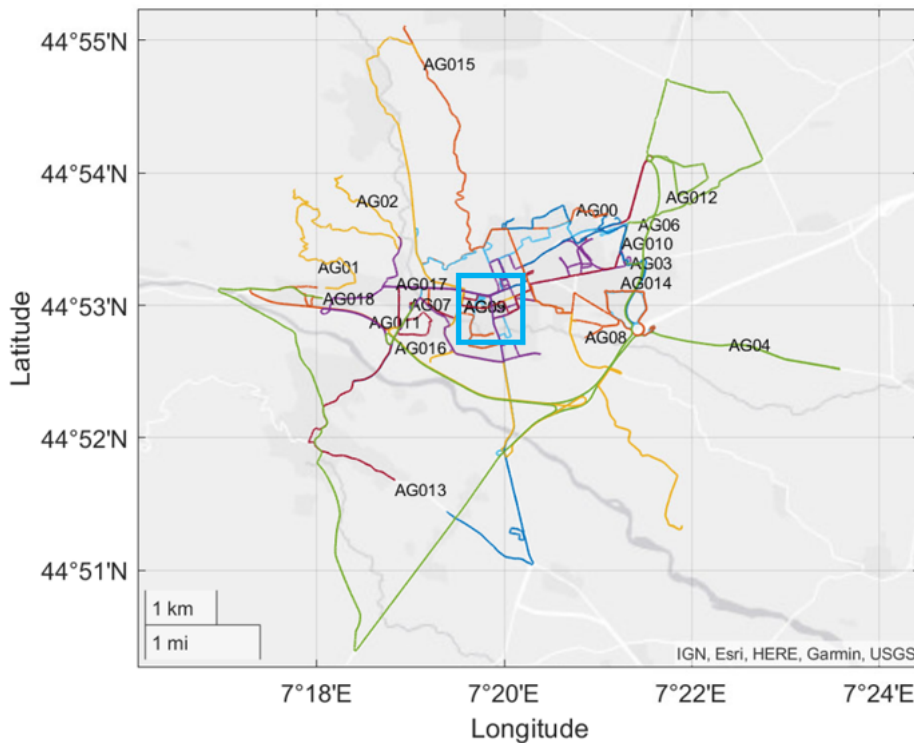


Fig. 6.5 SAMARCANDA traces with 1 km^2 square area in blue. Reproduced from [75].

6.5.2 Scenario creation

We will now explore the step-by-step process of generating the channel matrix using the SAMARCANDA dataset of real-world vehicle traces.

Task 1 - Dataset pre-processing and refinement

When crafting the Colosseum scenario, our foundation rested upon the mobility traces sourced from the SAMARCANDA dataset. Given Colosseum's current limitations, which cap the emulation to geographical regions not exceeding 1 km^2 , we judiciously pruned our dataset to fit this criterion. The selected area, delineated based on our previously discussed rationale, is visually denoted by the blue square in Figure 6.5.

Following an initial pre-processing phase, we embarked on refining the dataset further. A pivotal step involved filtering out vehicles that never ventured beyond the designated 1 km^2 area. This selective removal culminated in a pruned dataset that featured 13 vehicles. Additionally, to optimize the dataset's efficiency, timestamps—along with their associated dynamic data—corresponding to periods devoid of vehicular activity within our specified area were systematically expunged. While these instances constituted a minuscule fraction of the overall timestamps, their elimination streamlined the dataset, priming it for subsequent ray-tracing phases. Consequently, the final pruned dataset encompasses data from 13 vehicles, spanning 5,200 timestamps per vehicle, translating to an approximate emulation duration of 8.7 minutes.

While configuring the scenario, we strategically positioned a 5G antenna within the confines of the 1 km^2 area, precisely at the geographic midpoint as depicted by the blue marker in Figure 6.2. This placement mirrors the coordinates of a real Vodafone LTE antenna, as corroborated by data in [2], pinpointed at $[44.88338; 7.33152]$. Moreover, aligning with Colosseum's operational parameters, which predominantly focus on the 1 GHz frequency band, we designated this as our foundational frequency. Nevertheless, it is worth highlighting that our framework's versatility allows seamless adaptation to alternate frequency bands. For instance, orchestrating a scenario centered on the 5G N77 band at 3.7 GHz necessitates merely tweaking the transmitter frequency parameter, underscoring the flexibility inherent in our approach.

Task 2 - Ray-tracing

We utilized the ray-tracing feature within the MATLAB Communications Toolbox, recognizing its exceptional versatility compared to other available software solutions. We employed the *raytrace* function provided by MATLAB, as referenced in [4], tailoring its settings based on the radio parameters outlined in Table 6.1. Our configuration specified a 1 GHz transmitter frequency and a transmission power of 23 dBm, equivalent to 199.53 mW as indicated by [47]. Furthermore, we established the maximum reflection order to the third level and designated the building and terrain material as concrete.

The *raytrace* function produces a *Ray* object, encapsulating all detected rays originating from the designated transmitter to its corresponding receiver. Each ray is characterized by specific attributes such as *PathLoss*, *PhaseShift*, *PropagationDelay*, *AngleOfArrival*, and *AngleOfDeparture*. Subsequently, this collected data was integrated into the clustering phase to formulate the channel matrix.

Figure 6.6 illustrates a representation of these propagation rays, accompanied by their respective power levels, as delineated in the adjacent legend, linking a vehicle to the antenna.

Parameters	Value
Transmitter Power	23 dBm
Transmitter Frequency	1 GHz
Transmitter Antenna Height	1.5 m
Transmitter Antenna Gain	0 dBi
Receiver Antenna Height	30 m
Receiver Antenna Gain	0 dBi
Max Number of Reflections	3
Buildings and Terrain Material	Concrete
Ray-tracing method	SBR
Noise level	-89.1 dBm [97]

Table 6.1 MATLAB simulation parameters.

Task 3 - Clustering

We employed the refined K-means algorithm, as elaborated in 6.4.1, utilizing the MCD as the distance metric on the results of the ray-tracing phase. Figure 6.7



Fig. 6.6 Ray-tracing output for a specific timestamp between vehicle 9 and the antenna.

showcases the outcomes of this algorithm when applied to the multi-path components obtained from the ray-tracer process. Specifically, the algorithm yielded K clusters, where we set $K = 4$.

Figure 6.7 represents each multi-path component, plotting path loss (in dB) against its respective propagation delay (in seconds). The \times symbols on the plot indicate the centroids of individual clusters, pinpointed by the K-means algorithm. Upon consolidating all multi-path contributions, the resulting channel matrix exhibited up to $K = 4$ delay values, accompanied by IQ parameters for each connection directed towards the gNB across all the timestamps. Given our objective to emulate a centralized environment, we focused on capturing all link interactions between vehicles and the base station. Consequently, we omitted values pertaining to vehicle-to-vehicle links within this matrix.

Task 4 - Scenario creation and installation

Upon finalizing the channel matrix generation, the next pivotal step involved leveraging this data as the primary input for the *CaST* toolchain [102]. Finally, after few days of processing the scenario was installed inside Colosseum emulator with $id = 33013$.

In the subsequent Section, we validate the efficacy of the constructed V2X scenario by evaluating two critical metrics: Round-Trip Time (RTT) and Signal-to-Noise Ratio (SNR).

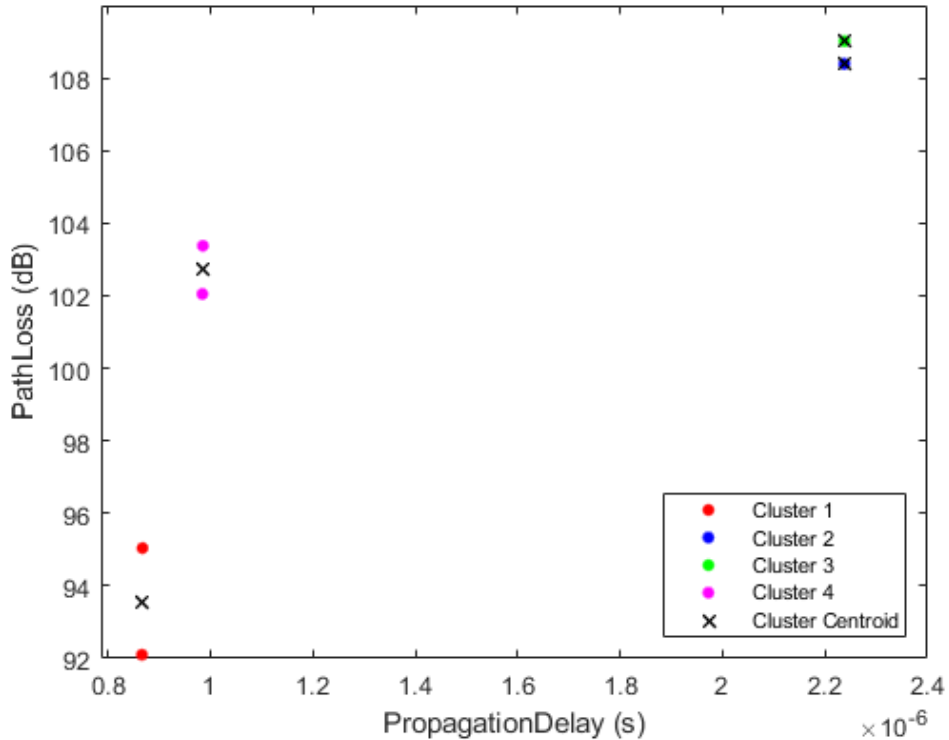


Fig. 6.7 Visualization of the K-means algorithm applied to a collection of rays from the ray-tracing process.

6.5.3 Scenario validation

To validate the scenario developed using our framework, we executed multiple experiments within the Colosseum emulator, utilizing the implemented RF scenario. Specifically, each SRN was set up to operate with SCOPE [23], simulating a comprehensive cellular network system on both the base station and the 13 vehicles.

SCOPE environment automates the instantiation processes for the base station and the Evolved Packet Core (EPC), along with the user equipment (UE) association procedures, which in our context are the vehicles. By default, SCOPE designates the SRN with the lowest identification number as the base station, and the remaining nodes function as UEs, given that the correct number of UEs is defined in the .config file. Given that our scenario designates the 14th SRN as the base station, we adjusted this default setting to ensure the 14th node assumes the role of the base station, designating all other nodes as UEs.

Upon initializing the scenario, we captured various metrics, encompassing the RTT as vehicles navigated within the coverage area and the downlink SNR over time. Subsequently, these metrics were juxtaposed with each vehicle's position and their respective distance from the gNB, utilizing coordinates from the SAMARCANDA dataset.

Figure 6.8 delineates the relationship between the distance from the gNB over time, juxtaposed with the RTT metrics between the vehicle and the base station, as gauged by the *ping* utility. The dotted lines in purple demarcate intervals wherein the vehicle remains within the coverage area (the 1 km² area). Concurrently, vertical black lines segment the time instances encompassed in the pruned dataset, representing the authentic emulated timestamps (from 250 s to 770 s).

As expected, effective communication is observed only when the vehicle remains within the coverage zone, hence inside the 1 km² area. Notably, the average RTT improves as the vehicle's distance decreases, owing to enhanced SNR and received power metrics. Given the emulation of a genuine urban setting, fluctuations in both parameters are evident, influenced by multi-path effects and structures that either obstruct or reflect signals.

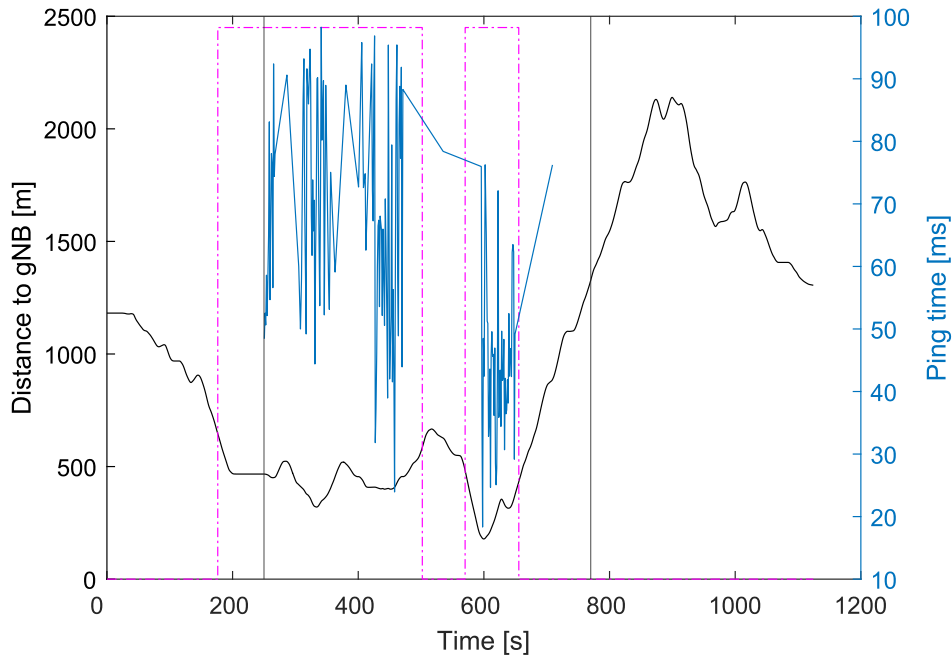


Fig. 6.8 Comparison between RTT and distance from the base station. Data refer to vehicle 9 of SAMARCANDA [75].

Figure 6.9 illustrates the SNR over time juxtaposed with the vehicle’s proximity to the gNB. The presented data represents averaged results from five different experiments, accompanied by 99% confidence intervals. Echoing the prior observations, the graph underscores that a diminished distance from the gNB corresponds to elevated SNR levels, punctuated by terrain and structural variables, realistically captured through our framework and the Colosseum emulator. Additionally, Figure 6.9 underscores the necessity of an SNR surpassing approximately 3.0 to ensure a consistently reliable communication link between the vehicle and the gNB. Importantly, the minimal size of the confidence intervals underscores the repeatability of results achievable with our chosen wireless emulation system. These findings validate both our proposed framework and the ensuing 5G scenario, affirming the efficacy of our methodology.

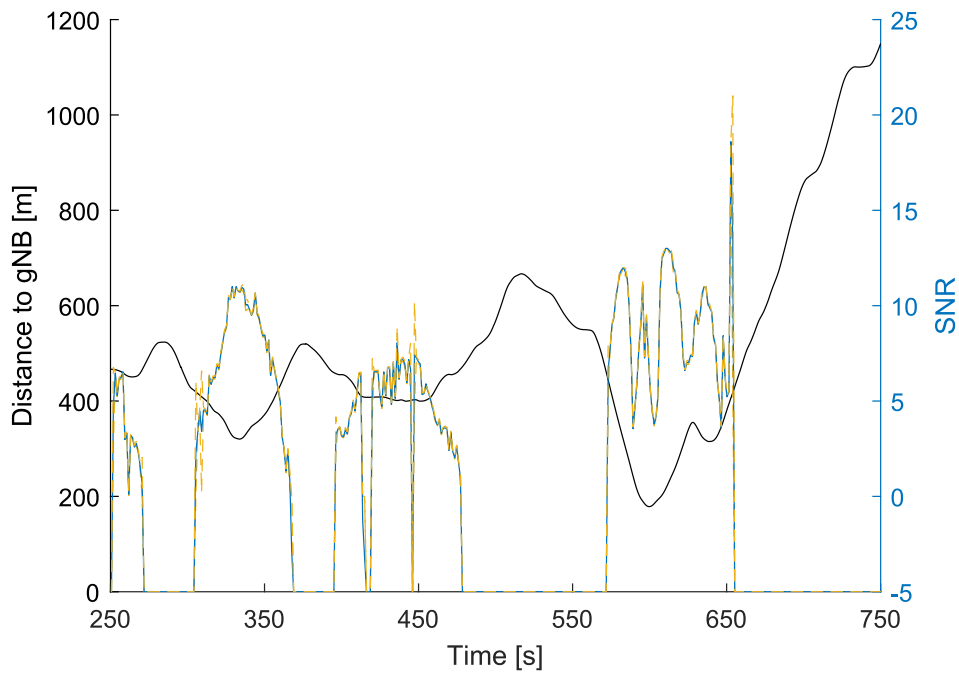


Fig. 6.9 Comparison between SNR and the distance from the base station. Data refer to vehicle 9 of SAMARCANDA [75]. The blue line represents the average SNR over 5 different experiments, with the 99% confidence intervals represented by orange dotted lines.

6.6 Conclusions and future works

In the current dynamic landscape of wireless communication, the increasing use of large-scale wireless emulation has become more than just a trend; it is a significant force influencing the development of next-generation wireless networks. As we have observed, the combination of this growth with technologies like massive MIMO, millimeter-wave beamforming, and AI-driven V2X communication has led to various innovative applications. Despite the potential, addressing challenges in exploring wireless applications, especially those involving numerous mobile nodes, proves complex for traditional simulation frameworks alone. Understanding this gap emphasizes the importance of our work.

Our effort has resulted in the development of an innovative framework specifically designed for creating channel emulation scenarios based on real-world mobility traces. Drawing inspiration and extending the groundwork presented in [102], our methodology emphasizes a data-centric approach, utilizing real-world datasets to emulate scenarios with unparalleled authenticity. With careful attention to detail, we have outlined each procedural nuance, leading to the introduction of an open-source MATLAB framework [80].

Our framework boasts remarkable versatility, serving as a facilitator for the seamless creation and deployment of scenarios encompassing both cellular and WiFi domains on expansive platforms like Colosseum [24]. To substantiate its capabilities, we meticulously constructed a V2X 5G centralized scenario within the Colosseum emulator, grounded in the comprehensive SAMARCANDA dataset [75]. The creation and evaluation of this scenario offered a tangible platform for validating our methodology, showcasing the effectiveness of our solution in constructing and deploying realistic communication environments with dynamic mobile nodes.

Looking forward, our focus will shift to implementing a federated learning process atop the established V2X mobility scenario. In this undertaking, we will utilize vehicles as federated learning clients, leveraging the high-fidelity mobility scenario uniquely achievable through an emulator of Colosseum's caliber.

Chapter 7

Conclusions

In recent years, there have been notable advancements in crowd monitoring, largely driven by the necessity to manage large gatherings during the COVID-19 pandemic. This progress has prompted academic exploration into new ways of monitoring crowds, with a focus on enhancing user privacy and public safety. However, implementing people counting algorithms and predicting traffic flow comes with its share of challenges. One major concern revolves around privacy issues related to collecting and storing sensitive data, especially when using WiFi probe request messages, along with complexities arising from MAC address randomization.

Our journey into crowd monitoring and people counting began in early 2020, coinciding with the escalating importance of crowd monitoring due to the COVID-19 pandemic. We recognized the need for passive people counting systems that operate seamlessly, prompting us to develop systems leveraging WiFi probe request messages. However, evolving privacy regulations and widespread MAC address randomization practices posed new challenges to our research. To address these challenges, we started from scratch and we found a gap in literature about the actual behaviour of this methodology. Consequently, we delved into a comprehensive exploration of probe request message behavior, examining how user-device interactions impact the frequency of message transmission, and how MAC address randomization is actually implemented. With a clearer understanding, we realized that traditional and simplistic de-randomizer algorithms previously utilized were no longer sufficient. Consequently, we opted to transition towards exploring machine learning methodologies to address these challenges more effectively.

The initial step involved developing a realistic probe request generator capable of generating datasets with associated ground truth, allowing us to emulate diverse scenarios and behaviors in a fraction of the time. With the generated data, we proceeded to create and test our first machine learning-driven de-randomizer. This utilized a DBSCAN clustering algorithm, effectively grouping probe requests originating from devices of the same model. Utilizing the refined counting pipeline, which had been rigorously tested in controlled environments, we deployed multiple Raspberry Pi devices in a public park situated in the heart of Turin. This deployment is part of our commitment to the TrialsNet EU project [13].

During our journey we had also the opportunity to explore the potential of federated learning in dynamic resource allocation tasks, particularly in urban mobility solutions and vehicular trajectory prediction. Through a use case scenario, we demonstrated the efficacy of federated learning in training ML models in a privacy-compliant manner while minimizing network footprint and leveraging IoT device computing capabilities. In our scenario we shown that, using real-world mobility traces compared to a synthetic scenario, federated learning converge faster to the predefined termination criterion, thus reaching the same accuracy of the same ML model trained in a classic centralized manner.

Furthermore, we developed an innovative data-driven framework for radio frequency mobility scenario generation, enhancing the realism of V2X scenarios and fostering innovation in urban mobility solutions. By integrating real-world mobility traces with advanced channel modeling techniques, this framework lays the foundation for more accurate and reliable simulation environments.

Future works

Moving forward, our investigation into crowd monitoring, and federated learning opens up numerous promising paths for future research and development.

We are currently investigating the usage of neural networks, particularly LSTM networks to tackle the people counting task. We believe that there may exist temporal patterns among different capturing windows that can be leveraged for improved accuracy. At the same time as privacy concerns continue to escalate in significance, future efforts will concentrate on refining anonymization techniques and exploring the integration of differential privacy mechanisms.

Conclusions

In the realm of federated learning, further research drive towards the application of federated learning to different use cases, particularly in scenarios where federated learning can serve as a viable alternative to centralized machine learning approaches. Additionally, we are delving into the accuracy and efficiency of the asynchronous paradigm of federated learning. This involves exploring the concept of adding an extra layer between clients and servers, akin to the concept proposed in [35], to optimize the convergence time of the entire federated network.

List of acronyms

Acronyms / Abbreviations

AI Artificial Intelligence

AP Access Point

ARP Address Resolution Protocol

BLE Bluetooth Low Energy

C – V2I Cellular Vehicle-to-Infrastructure

CA Collision Avoidance

CID Company Identifier

CV Coefficient of Variation

DARPA Defense Advanced Research Projects Agency

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DES Data Encryption Standard

DPO Data Protection Officer

DSRC Dedicated Short Range Communication

eNB eNodeB

EPC Evolved Packet Core

ETSI European Telecommunications Standards Institute

List of acronyms

<i>FCS</i>	Frame Check Sequence
<i>FIR</i>	Finite Impulse Response
<i>FL</i>	Federated Learning
<i>FPGA</i>	Field Programmable Gate Array
<i>FSM</i>	Finite-State Machine
<i>GA</i>	Globally Administrated
<i>GDPR</i>	General Data Protection Regulation
<i>GMM</i>	Gaussian Mixture Models
<i>gNB</i>	gNodeB
<i>GNSS</i>	Global Navigation Satellite System
<i>GPS</i>	Global Positioning System
<i>HE</i>	Homomorphic Encryption
<i>HT</i>	High Throughput
<i>IE</i>	Information Elements
<i>IEEE</i>	Institute of Electrical and Electronics Engineers
<i>IMU</i>	Inertial Measurement Unit
<i>IP</i>	Internet Protocol
<i>IQ</i>	In-phase and Quadrature
<i>ISTAT</i>	Italian National Institute of Statistics
<i>LAA</i>	Locally Administrated
<i>LAN</i>	Local Area Network
<i>LANMAN</i>	LAN Manager
<i>LiDAR</i>	Light Detection and Ranging

List of acronyms

<i>LSTM</i>	Long Short-Term Memory
<i>M2M</i>	Machine to Machine
<i>MAC</i>	Media Access Control
<i>MCD</i>	Multi-path Component Distance
<i>MCHEM</i>	Massive digital Channel Emulator
<i>MD</i>	Message Digest
<i>MEC</i>	Multi-access Edge Computing
<i>MED</i>	Mean Euclidean Distance
<i>MIMO</i>	Multiple-Input Multiple-Output
<i>ML</i>	Machine Learning
<i>MQTT</i>	Message Queuing Telemetry Transport
<i>MSE</i>	Mean Squared Error
<i>NIC</i>	Network Interface Controller
<i>NN</i>	Neural Network
<i>NTPM</i>	NT LAN Manager
<i>OBU</i>	On-Board Unit
<i>OPTICS</i>	Ordering Points To Identify the Clustering Structure
<i>OS</i>	Operating System
<i>OUI</i>	Organizationally Unique identifier
<i>PHE</i>	Partially Homomorphic Encryption
<i>PIR</i>	Passive Infrared sensors
<i>PoC</i>	Proof of Concept
<i>PoE</i>	Power over Ethernet

List of acronyms

<i>POWDER</i>	Platform for Open Wireless Data-driven Experimental Research
<i>PR</i>	WiFi Probe Request
<i>RAN</i>	Radio Access Network
<i>RF</i>	Radio Frequency
<i>RNN</i>	Recurrent Neural Network
<i>RSSI</i>	Received Signal Strength Indicator
<i>RSU</i>	Road Side Unit
<i>RTK</i>	Real-Time Kinematic
<i>RTT</i>	Round-Trip Time
<i>SAMARCANDA</i>	Synthetic Accurate Multi-Agent RealistiC Assisted-gNss DatAset
<i>SBC</i>	Single-Board Computer
<i>SDR</i>	Software Defined Radio
<i>SED</i>	Squared Euclidean Distance
<i>SHA</i>	Secure Hash Algorithm
<i>SNR</i>	Signal-to-Noise Ratio
<i>SRN</i>	Standard Radio Node
<i>SSID</i>	Service Set Identifier
<i>SUMO</i>	Simulation of Urban MObility
<i>TCP</i>	Transmission Control Protocol
<i>TraCI</i>	Traffic Control Interface
<i>TuST</i>	Turin SUMO Traffic
<i>UE</i>	User Equipment
<i>UUID – E</i>	Universally Unique Identifier-Extended

List of acronyms

<i>V2V</i>	Vehicle-to-Vehicle
<i>V2X</i>	Vehicle-to-Everything
<i>VHT</i>	Very High Throughput
<i>VNF</i>	Virtual Network Function
<i>WPS</i>	WiFi Protected Setup

Bibliography

- [1] 5G-CARMEN. URL <https://5gcarmen.eu/>.
- [2] Cell mapper. URL <https://www.cellmapper.net/>.
- [3] OpenAirInterface. URL <https://openairinterface.org>.
- [4] Reference manual for raytrace. URL <https://www.mathworks.com/help/comm/ref/txsite.raytrace.html>.
- [5] Traffic light status (tls) dataset. URL <https://www2.kios.ucy.ac.cy/harpydata/tlsdataset/>.
- [6] Dropper s.r.l. URL <https://www.dropper.ai>.
- [7] Hexa-X. URL <https://hexa-x.eu/>.
- [8] ns-3. URL <https://www.nsnam.org>.
- [9] OneM2M. <http://www.onem2m.org>.
- [10] Preliminary verification. collection, analysis and processing of data, through the installation of equipment, for marketing and market research purposes. URL <https://www.garanteprivacy.it/home/docweb/-/docweb-display/docweb/9022068>.
- [11] Wireless EM Propagation Software - Wireless InSite - Remcom. URL <https://www.remcom.com/wireless-insite-em-propagation-software>.
- [12] srsRAN. URL <https://www.srslte.com>.
- [13] Trialsnet EU project. URL <https://trialsnet.eu/>.
- [14] Wireshark - go deep. URL <https://www.wireshark.org>.
- [15] Spostamenti Quotidiani e Nuove Forme di Mobilità. Statistical document, ISTAT, Istituto Nazionale di Statistica, 2018.
- [16] Dopo l'emergenza sanitaria il caro-energia: le intenzioni di mobilità nei prossimi tre mesi. Statistical document, ISTAT, Istituto Nazionale di Statistica, 2022.

- [17] Airodump-ng. Airodump-ng. [Online]. URL: <https://www.aircrack-ng.org/doku.php?id=airodump-ng>.
- [18] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2575–2582. IEEE, November 2018. URL <https://elib.dlr.de/127994/>.
- [19] Lorenzo Bertizzolo, Leonardo Bonati, Emrehan Demirors, Amani Alshawabka, Salvatore d’oro, Francesco Restuccia, and Tommaso Melodia. Arena: A 64-antenna sdr-based ceiling grid testing platform for sub-6 ghz 5g-and-beyond radio spectrum research. *Computer Networks*, 181:107436, 07 2020. doi: 10.1016/j.comnet.2020.107436.
- [20] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchu Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. FLOWER: A FRIENDLY FEDERATED LEARNING FRAMEWORK. Open-Source, mobile-friendly Federated Learning framework, March 2022. URL <https://hal.science/hal-03601230>.
- [21] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loret. ”Better than nothing” privacy with bloom filters: To what extent? In Josep Domingo-Ferrer and Ilenia Tinnirello, editors, *Privacy in Statistical Databases*, pages 348–363. Springer Berlin Heidelberg, 2012.
- [22] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
- [23] Leonardo Bonati, Salvatore D’Oro, Stefano Basagni, and Tommaso Melodia. SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems. In *Proc. of ACM Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, Virtual Conference, June 2021.
- [24] Leonardo Bonati, Pedram Johari, Michele Polese, Salvatore D’Oro, Subhramoy Mohanti, Miead Tehrani-Moayyed, Davide Villa, Shweta Shrivastava, Chinenye Tassie, Kurt Yoder, Ajeet Bagga, Paresh Patel, Ventz Petkov, Michael Seltser, Francesco Restuccia, Abhimanyu Gosain, Kaushik R. Chowdhury, Stefano Basagni, and Tommaso Melodia. Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation. In *Proc. of IEEE Intl. Symp. on Dynamic Spectrum Access Networks (DySPAN)*, Virtual Conference, December 2021.
- [25] Tomas Bravenec, Joaquín Torres-Sospedra, Michael Gould, and Tomas Fryza. Exploration of user privacy in 802.11 probe requests with MAC address randomization using temporal pattern analysis, 2022.

- [26] Joe Breen, Andrew Buffmire, Jonathon Duerig, Kevin Dutt, Eric Eide, Mike Hibler, David Johnson, Sneha Kumar Kasera, Earl Lewis, Dustin Maas, Alex Orange, Neal Patwari, Daniel Reading, Robert Ricci, David Schurig, Leigh B. Stoller, Jacobus Van der Merwe, Kirk Webb, and Gary Wong. POWDER: Platform for open wireless data-driven experimental research. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, September 2020. doi: 10.1145/3411276.3412204.
- [27] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004. doi: 10.1080/15427951.2004.10129096. URL <https://doi.org/10.1080/15427951.2004.10129096>.
- [28] Ziqing Chen, Wei Yuan, Ming Yang, Chunxiang Wang, and Bing Wang. SVM based people counting method in the corridor scene using a single-layer laser scanner. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2632–2637, 2016. doi: 10.1109/ITSC.2016.7795979.
- [29] N. Czink, P. Cera, J. Salo, Ernst Bonek, Jukka Nuutinen, and Juha Ylitalo. Improving clustering performance using multipath component distance. *Electronics Letters*, 42, 2006.
- [30] Kapil Dandekar, Simon Begashaw, Marko Jacovic, Alex Lackpour, Ilhaan Rasheed, Xaime Rivas Rey, Cem Sahin, Sharif Shaher, and Geoffrey Mainland. Grid software defined radio network testbed for hybrid measurement and emulation. pages 1–9, 06 2019. doi: 10.1109/SAHCN.2019.8824901.
- [31] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [32] Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Chang-Heng Wang, Jérôme Härrri, and Onur Altintas. On the orchestration of federated learning through vehicular knowledge networking. In *2020 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2020. doi: 10.1109/VNC51378.2020.9318386.
- [33] Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Jérôme Härrri, and Onur Altintas. A definition and framework for vehicular knowledge networking: An application of knowledge-centric networking. *IEEE Vehicular Technology Magazine*, 16(2):57–67, 2021. doi: 10.1109/MVT.2021.3066376.
- [34] Laihui Ding, Shengke Wang, Rui Li, Long Chen, and Junyu Dong. PC-PINet: Partial re-identification network for people counting with overlapping cameras. In *International Conference on Image, Vision and Computing (ICIVC)*, pages 66–71, 2021. doi: 10.1109/ICIVC52351.2021.9526965.
- [35] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In *2019 IEEE*

- 37th international conference on computer design (ICCD)*, pages 246–254. IEEE, 2019.
- [36] Fatih Erden, Ali Alkar, and Ahmet Cetin. A robust system for counting people using an infrared sensor and a camera. *Infrared Physics & Technology*, 72, 08 2015. doi: 10.1016/j.infrared.2015.07.019.
- [37] ETSI. ETSI TR 138 901 V15.0.0 (2018-07) - 5G; Study on channel model for frequencies from 0.5 to 100 GHz (3GPP TR 38.901 version 15.0.0 Release 15). Standard ETSI TR 138 901 V15.0.0, European Telecommunications Standards Institute, 2018.
- [38] ETSI. ETSI EN 302 637-2 V1.4.1 (2019-04) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Standard ETSI EN 302 637-2 V1.4.1, European Telecommunications Standards Institute, 2019.
- [39] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. URL <https://data.europa.eu/eli/reg/2016/679/oj>.
- [40] Julien Freudiger. How talkative is your mobile device? an experimental study of Wi-Fi probe requests. ACM WiSec, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336239.
- [41] M.S. Gast. *802.11 Wireless Networks: The Definitive Guide, 2nd edition*. Sebastopol, CA, USA, 2005.
- [42] M.S. Gast. *802.11 ac: A Survival Guide: Wi-Fi at Gigabit and Beyond*. Sebastopol, CA, USA, 2013.
- [43] Kalkidan Gebru, Marco Rapelli, Riccardo Rusca, Claudio Casetti, Carla Fabiana Chiasserini, and Paolo Giaccone. Edge-based passive crowd monitoring through WiFi beacons. *Computer Communications*, 192:163–170, 2022. ISSN 0140-3664. doi: <https://doi.org/10.1016/j.comcom.2022.06.003>.
- [44] Andrei Günter, Stephan Böker, Matthias König, and Martin Hoffmann. Privacy-preserving people detection enabled by solid state LiDAR. In *International Conference on Intelligent Environments (IE)*, pages 1–4, 2020. doi: 10.1109/IE49459.2020.9154970.
- [45] IEEE. Oui - standards. URL <https://standards-oui.ieee.org>.
- [46] Apple Inc. Wi-fi privacy, 02 2021. URL <https://support.apple.com/en-gb/guide/security/secb9cb3140c/web>.
- [47] Paramananda Joshi, Fatemeh Ghasemifard, Davide Colombi, and Christer Törnevik. Actual output power levels of user equipment in 5G commercial networks and implications on realistic RF EMF exposure assessment. *IEEE Access*, 8, 2020.

- [48] KimiNewt. Pyshark documentation. URL <https://github.com/KimiNewt/pyshark>.
- [49] G La Bruna, C Risma Carletti, R Rusca, C Casetti, CF Chiasserini, M Giordanino, and R Tola. Edge-assisted federated learning in vehicular networks. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, pages 163–170. IEEE, 2022.
- [50] Grafana Labs. Grafana: The open observability platform. <https://grafana.com/>.
- [51] Li Li, Jun Wang, and ChengZhong Xu. FLSim: An extensible and reusable simulation framework for federated learning. In Houbing Song and Dingde Jiang, editors, *Simulation Tools and Techniques*, pages 350–369, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72792-5.
- [52] John D.C. Little. A proof for the queuing formula: $L = \lambda W$. *Operations Research*, 2(4):447–457, 1954.
- [53] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987*, 2020.
- [54] Marco Malinverno, Josep Mangués-Bafalluy, Claudio Ettore Casetti, Carla Fabiana Chiasserini, Manuel Requena-Esteso, and Jorge Baranda. An edge-based framework for enhanced road safety of connected cars. *IEEE Access*, 8:58018–58031, 2020.
- [55] Marco Malinverno, Francesco Raviglione, Claudio Casetti, Carla-Fabiana Chiasserini, Josep Mangués-Bafalluy, and Manuel Requena-Esteso. A multi-stack simulation framework for vehicular applications testing. In *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet ’20, page 17–24, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381215. doi: 10.1145/3416014.3424603. URL <https://doi.org/10.1145/3416014.3424603>.
- [56] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik Rye, and Dane Brown. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 03 2017. doi: 10.1515/popets-2017-0054.
- [57] Célestin Matte. *Wi-Fi tracking: Fingerprinting attacks and counter-measures*. PhD thesis, Université de Lyon, 2017.
- [58] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages

Bibliography

- 1273–1282. PMLR, 20–22 Apr 2017. URL <https://proceedings.mlr.press/v54/mcmahan17a.html>.
- [59] Meshlium by Libelium. Meshlium by libelium. [Online]. URL: <https://www.libelium.com/iot-products/meshlium-scanner/>.
- [60] Fabrizio Moggio, Mauro Boldi, Silvia Canale, Vincenzo Suraci, Claudio Casetti, Giacomo Bernini, Giada Landi, and Paolo Giaccone. 5g eve a european platform for 5g application deployment. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization, WiNTECH '20*, page 124–125, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380829. doi: 10.1145/3411276.3414696. URL <https://doi.org/10.1145/3411276.3414696>.
- [61] MS Windows NT Kernel Description. MS Windows NT kernel description. [Online]. URL: <https://gs.statcounter.com/vendor-market-share/mobile/europe/yearly-2020-2023-bar>.
- [62] Ei Phyu Myint and Myint Myint Sein. People detecting and counting system. In *IEEE Global Conference on Life Sciences and Technologies (LifeTech)*, pages 289–290, 2021. doi: 10.1109/LifeTech52111.2021.9391951.
- [63] Michele Nitti, Francesca Pinna, Lucia Pintor, Virginia Pilloni, and Benedetto Barabino. iabacus: A Wi-Fi-based automatic bus passenger counting system. *Energies*, 13(6):1446, 2020.
- [64] Numpy library. Numpy library. [Online]. URL: <https://numpy.org/>.
- [65] Luiz Oliveira, Daniel Schneider, Jano De Souza, and Weiming Shen. Mobile device detection through WiFi probe request analysis. *IEEE Access*, 7:98579–98588, 2019.
- [66] World Health Organization. Global status report on road safety 2018. geneva: World health organization; 2018. licence: Cc bync-sa 3.0 igo. 2018. URL <https://www.who.int/publications/i/item/9789241565684>.
- [67] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. Cardinality estimation and dynamic length adaptation for Bloom filters. *Distributed and Parallel Databases*, 28:119–156, 2010.
- [68] Cristian Perra, Amit Kumar, Michele Losito, Paolo Pirino, Milad Moradpour, and Gianluca Gatto. Monitoring indoor people presence in buildings using low-cost infrared sensor array in doorways. *Sensors*, 21(12), 2021. ISSN 1424-8220. doi: 10.3390/s21124062.
- [69] Lucia Pintor and Luigi Atzori. A dataset of labelled device wi-fi probe requests for mac address de-randomization. *Computer Networks*, 205:108783, 2022. ISSN 1389-1286. doi: <https://doi.org/10.1016/j.comnet.2022.108783>. URL <https://www.sciencedirect.com/science/article/pii/S1389128622000196>.

Bibliography

- [70] The Android Open Source Project. Implementing mac randomization, 11 2022. URL <https://source.android.com/docs/core/connect/wifi-mac-randomization>.
- [71] Python programming language. Python programming language. [Online]. URL: <https://www.python.org/>.
- [72] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. Vehicular traffic simulation in the city of turin from raw data. *IEEE Transactions on Mobile Computing*, 21(12):4656–4666, 2022. doi: 10.1109/TMC.2021.3075985.
- [73] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. Vehicular traffic simulation in the city of turin from raw data. *IEEE Transactions on Mobile Computing*, 21(12):4656–4666, 2022. doi: 10.1109/TMC.2021.3075985.
- [74] Raspberry Pi. Raspberry pi. [Online]. URL: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>.
- [75] F. Raviglione, S. Zocca, A. Minetto, M. Malinverno, C. Casetti, C.F. Chiasserini, and F. Dovis. From collaborative awareness to collaborative information enhancement in vehicular networks. *Vehicular Communications (Elsevier)*, 36, 2022.
- [76] Francesco Raviglione, Carlos Mateo Risma Carletti, Claudio Casetti, Filippo Stoffella, Girma M. Yilma, and Filippo Visintainer. S-LDM: Server Local Dynamic Map for vehicular enhanced collective perception. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, 2022.
- [77] Alessandro E.C. Redondi and Matteo Cesana. Building up knowledge through passive WiFi probes. *Computer Communications*, 117:1–12, 2018. ISSN 0140-3664. doi: <https://doi.org/10.1016/j.comcom.2017.12.012>.
- [78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
- [79] Riccardo Rusca. Probe request generator, . URL <https://github.com/riccardo-rusca/ProbeRequestGenerator>.
- [80] Riccardo Rusca. Mobile RF Scenario Design, . URL https://github.com/riccardo-rusca/Mobile_RF_Scenario_Design.
- [81] Riccardo Rusca. Probe request sniffing, . URL <https://github.com/riccardo-rusca/ProbeRequestSniffing>.
- [82] Riccardo Rusca, Claudio Casetti, and Paolo Giaccone. IoT for real time presence sensing on the 5G EVE infrastructure. In *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 1–8, 2021.

Bibliography

- [83] Riccardo Rusca, Alex Carluccio, Diego Gasco, and Paolo 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)*, pages 1–6, 2023. doi: 10.1109/ICT-DM58371.2023.10286944.
- [84] Riccardo Rusca, Francesco Raviglione, Claudio Casetti, Paolo Giaccone, and Francesco Restuccia. Mobile rf scenario design for massive-scale wireless channel emulators. In *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 675–680, 2023. doi: 10.1109/EuCNC/6GSummit58263.2023.10188319.
- [85] Riccardo Rusca, Filippo Sansoldo, Claudio Casetti, and Paolo Giaccone. What WiFi probe requests can tell you. In *IEEE Consumer Communications & Networking Conference (CCNC)*, pages 1086–1091, 2023. doi: 10.1109/CCNC51644.2023.10060447.
- [86] Riccardo Rusca, Alex Carluccio, Claudio Casetti, and Paolo Giaccone. Privacy-preserving wifi-based crowd monitoring. *Transactions on Emerging Telecommunications Technologies*, 35(3):e4956, 2024. doi: <https://doi.org/10.1002/ett.4956>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4956>.
- [87] Scapy library. Scapy library. [Online]. URL: <https://scapy.net/>.
- [88] Dinesh Cyril Selvaraj, Christian Vitale, Tania Panayiotou, Panayiotis Kolios, Carla Fabiana Chiasserini, and Georgios Ellinas. Edge learning of vehicular trajectories at regulated intersections. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7. IEEE, 2021.
- [89] M. C. Simon, Thierry Hermitte, and Yves Page. Intersection road accident causation: A european view. *Proceedings: International Technical Conference on the Enhanced Safety of Vehicles*, 2009:–, 2009. URL <http://dx.doi.org/>.
- [90] Aleš Simončič, Miha Mohorčič, Mihael Mohorčič, and Andrej Hrovat. Non-intrusive privacy-preserving approach for presence monitoring based on WiFi probe requests. *Sensors*, 23(5), 2023. ISSN 1424-8220. doi: 10.3390/s23052588. URL <https://www.mdpi.com/1424-8220/23/5/2588>.
- [91] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. Privacy-preserving crowd-monitoring using Bloom filters and homomorphic encryption. In *International Workshop on Edge Systems, Analytics and Networking (EdgeSys)*, page 37–42. ACM, 2021.
- [92] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. Anonymized counting of nonstationary Wi-Fi devices when monitoring crowds. In *International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, page 213–222. ACM, 2022.

- [93] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. Privacy-friendly statistical counting for pedestrian dynamics. *Computer Communications*, 211:178–192, 2023. ISSN 0140-3664. doi: <https://doi.org/10.1016/j.comcom.2023.09.009>. URL <https://www.sciencedirect.com/science/article/pii/S0140366423003195>.
- [94] David Stritzl. Privacy-preserving matching using bloom filters: an analysis and an encrypted variant, April 2019. URL <http://essay.utwente.nl/77733/>.
- [95] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2012. doi: 10.1109/SURV.2011.031611.00024.
- [96] tcpdump. tcpdump. [Online]. URL: <https://www.tcpdump.org>.
- [97] Miead Tehrani-Moayyed, Leonardo Bonati, Pedram Johari, Tommaso Melodia, and Stefano Basagni. Creating RF scenarios for large-scale, real-time wireless channel emulators. In *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2021.
- [98] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine vision and applications*, 25:633–647, 2014.
- [99] tshark. tshark. [Online]. URL: <https://www.wireshark.org/docs/man-pages/tshark.html>.
- [100] Marco Uras, Enrico Ferrara, Raimondo Cossu, Antonio Liotta, and Luigi Atzori. MAC address de-randomization for WiFi device counting: Combining temporal- and content-based fingerprints. *Computer Networks*, 218:109393, 2022. ISSN 1389-1286. doi: <https://doi.org/10.1016/j.comnet.2022.109393>. URL <https://www.sciencedirect.com/science/article/pii/S1389128622004273>.
- [101] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pages 413–424, 2016.
- [102] D. Villa, M. Tehrani-Moayyed, P. Johari, S. Basagni, and T. Melodia. CaST: A Toolchain for Creating and Characterizing Realistic Wireless Network Emulation Scenarios. In *ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization (WiNTECH 2022)*, Sydney, Australia, October 2022.
- [103] Huizi Xiao, Jun Zhao, Qingqi Pei, Jie Feng, Lei Liu, and Weisong Shi. Vehicle selection and resource optimization for federated learning in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 23(8): 11073–11087, 2021.

Bibliography

- [104] Justin Yackoski, Babak Azimi-Sadjadi, Ali Namazi, Jason H. Li, Yalin Sagduyu, and Renato Levy. RfnestTM: Radio frequency network emulator simulator tool. In *MILCOM 2011 Military Communications Conference*, 2011.
- [105] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2019.
- [106] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access*, 8:23920–23935, 2020.
- [107] Xiaokang Zhou, Wei Liang, Jinhua She, Zheng Yan, I Kevin, and Kai Wang. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. *IEEE Transactions on Vehicular Technology*, 70(6):5308–5317, 2021.
- [108] Xiaokang Zhou, Wei Liang, Jinhua She, Zheng Yan, I Kevin, and Kai Wang. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. *IEEE Transactions on Vehicular Technology*, 70(6):5308–5317, 2021.