Towards Privacy-sensitive Mobile Crowdsourcing

Lakhdar Meftah

Towards Privacy-sensitive Mobile Crowdsourcing

Lakhdar Meftah

Supervisors: Pr. Romain Rouvoy
Pr. Isabelle Chrisment

Thesis Committee: Dr. Sonia Ben Mokhtar
Pr. Yérom-David Bromberg
Dr. Chadi Barakat
Pr. Gilles Roussel

Inria Lille Nord Europe
University of Lille

This dissertation is submitted for the degree of
Doctor of Philosophy in Computer Science

University of Lille December 5th 2019

Vers une externalisation des tâches respectueuse de la vie privée des usagers des applications Mobiles

Lakhdar Meftah

directeurs de thèse: Pr. Romain Rouvoy
Pr. Isabelle Chrisment

Jury: Dr. Sonia Ben Mokhtar
Pr. Yérom-David Bromberg
Dr. Chadi Barakat
Pr. Gilles Roussel

Inria Lille Nord Europe
Université de Lille

thèse présentée et soutenue le 5 décembre 2019 à l'université de Lille pour obtenir le grade de
docteur en informatique

Université de Lille 5 Décembre 2019

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Lakhdar Meftah
5 Décembre 2019

Acknowledgements

I would like to express my gratitude to all the people who have contributed in some way in this thesis.

In particular, my supervisor Romain Rouvoy and Isabelle Chrisment. I would like to offer my special thanks to Romain Rouvoy for his time, his valuable and constructive suggestions during all the three years of this work. For encouraging me to do all the scientific communications, winter school, research visits, working with interns, attending the project meetings. Also, for encouraging me to work with the latest available technologies. My grateful thanks are also extended to Isabelle Chrisment who followed the progress of this work and provided insightful feedback.

I would like to thank Inria Lille research center for all the suitable work environment. I would like to thank the doctoral school, the university of Lille and CRIsTAL research lab.

I would like to thank Lionel Seinturier the Spirals team leader, who was an example of the real researcher, I have learned a lot from him, he would anticipate and deal with my requests right on the spot, thank you for all the support.

I would like to acknowledge the Inria IPL Betternet project for funding my work and my scientific communications.

I would like to thank the ANR Bottlenet project for the feedback I got for my work.

I would like to acknowledge the Grid'5000 for allowing me to launch my heavy experimentation, special thanks go to Guillaume Fieni for maintaining the G5K Docker driver.

I would like to thank all the members of the Spirals team for welcoming me during these three years.

I would like to thank the Inria AGOS association for all the events they organized.

I would like to thank all the interns that have contributed in some way in my work.

I would like to thank Nearchos Paspallis for welcoming me for 4 days in a research visit in his lab in Larnaca.

I would like to thank all the people who have participated in the DayKeeper mobile app testing.

I would like to thank Karine our research assistant for her reactivity.

Sarra Habchi for the insightful discussions we had together reviewing, criticising and defending our works.

Maria Gomez Lacruz with whom I have written my first research paper.

Alexandre, Aurelien, Chakib and Maxim, my office mates.

I would like to thank Tedjani for welcoming me in Lille on my first days. I would like to thank Oussama as he was my counselor. I would like to thank Khomaysi for his traditional dishes and his infinite invitations and advises in Lille.

I would like to thank my soul friend Aissa for all the moments we shared during these three years.

I would like to thank all of my ESI friends in all France for the unforgettable parties.

I would like to thank the sports association for all the organized sport sessions and for finding new cool friends. I would like to thank my fitness club for keeping me in shape and all the help I got during the long run.

Finally, I would like to thank my parents for their infinite support, thank you mom, thank you dad. I would like to thank my brothers and sisters, especially Taha who was by my side in the short weekends. I would like to thank Amina for her infinite support.

Abstract

With the widespread adoption of mobile phones, devices are used to track the user's activity and to collect insightful reports from the environment (*e.g.*, air quality, network quality). Most of these collected data are systematically tagged with the user location which may inevitably lead to user privacy leaks by discarding sensitive information *a posteriori* based on their potential points of interest.

This thesis introduces an anonymous data collection library for mobile apps, a software library that improves the user's privacy without compromising the overall quality of the crowdsourced dataset. In particular, we propose a decentralized approach, named FOUGERE, to convey data samples from user devices using *peer-to-peer* (P2P) communications to third-party servers, thus introducing an *a priori* data anonymization process that is resilient to location-based attacks.

To validate our approach, we propose a testing framework to test this P2P communication library, named PEERFLEET. Beyond the identification of P2P-related errors, PEERFLEET also helps to tune the discovery protocol settings to optimize the deployment of P2P apps.

We validate FOUGERE using 500 emulated devices that replay a mobility dataset and use FOUGERE to collect location data. We evaluate the overhead, the privacy and the utility of FOUGERE. We show that FOUGERE defeats the state-of-the-art location-based privacy attacks with little impact on the quality of the collected data.

Abstract

L'adoption globale des téléphones mobiles a rendu possible leur utilisation pour tracer l'activité des utilisateurs et pour collecter des données pertinentes de l'environnement (par exemple, qualité de l'air, qualité du réseau). La plupart de ces données collectées sont systématiquement étiquetées avec la localisation de l'utilisateur, ce qui peut inévitablement conduire à des fuites de la vie privée de l'utilisateur en inférant des informations sensibles a posteriori en fonction de ses points d'intérêts potentiels.

Dans cette thèse, nous proposons une bibliothèque de logiciels de collecte de données anonyme pour les applications mobiles, une bibliothèque de logiciels qui améliore la confidentialité des utilisateurs sans compromettre la qualité globale du jeu de données externalisé. En particulier, nous proposons une approche décentralisée, nommée FOUGERE, pour acheminer des échantillons de données provenant de périphériques utilisateurs via des communications Peer-to-Peer (P2P) vers des serveurs tiers, introduisant ainsi un processus a priori d'anonymisation des données résistant aux attaques visant les données de localisation des utilisateurs.

Pour valider notre approche, nous proposons un framework de test pour tester cette bibliothèque de communication P2P, nommée PEERFLEET. Au-delà de l'identification des erreurs liées aux communications P2P, PEERFLEET aide également les développeurs à ajuster les paramètres du protocole de découverte afin d'optimiser le déploiement d'applications P2P.

Nous validons FOUGERE à l'aide de 500 émulateurs qui rejouent un jeu de données de mobilité et utilisent FOUGERE pour collecter des données de localisation. nous évaluons la surcharge, la vie privée et l'utilité de FOUGERE. Nous montrons que FOUGERE contrecarre les attaques de confidentialité reposant sur la localisation de l'état de l'art, avec un impact minimal sur la qualité des données collectées

Table of contents

List of figures

List of tables

Chapter 1

Introduction

As mobile devices are getting smarter every day, they are used to collect insightful data about the user [1]. The sensors incorporated to these mobile devices give the mobile apps valuable information about the user's context and activity as well as her surrounding environment [2]. The provided user context can help these mobile apps provide the best user experience, like detecting when the phone is in the pocket [3] or detecting if the user is indoor/outdoor [4]. These context data also allow getting the user's current activity: like walking or driving [5]; it can help find parking places [6] and what types of places the user is visiting [7]; it can also provide a way to predict the user personality traits [8]. These context data may contain the user location history which is used to locate the user [9] or to provide the user with targeted advertisements [10].

Some scientists want to use these valuable sources of data to run realistic experiments and get real world data from users all around the world, these mobile devices offer a cost-effective way to collect data about the participant users. The scientists will recruit some users to participate in their crowdsourcing campaign, where their aim is to collect as much valuable data as they can. They can later scale this solution to get thousands of participants and these crowdsourcing campaigns can last for years [5].

The scientists are trying to maximize the data utility, this utility can cost the privacy of the participants. While the scientists are only interested in the crowd data and not in one participant specific data, some adversaries can get access to these data and try to identify each participant; then, these adversaries can infer personal information about the participants, their environment [11] and their routines [12].

In this thesis, we aim at protecting the location privacy of users participating to crowdsourcing campaigns. We propose a user-centric approach where the user can decide on how her location data are anonymized and with whom these data are shared. In our approach, we leverage the opportunistic dissemination using peer-to-peer communications between mobile devices to upload anonymized data to the server from another mobile device (other than the data producer) in order to cut the link between the server and the data producer. We explore the proximity between users to confuse the adversary on the server; this proximity can be explored thanks to *peer-to-peer* nearby connections (*e.g.*, WiFi, Bluetooth). So, the users only share their location data if there are at least K other users in their proximity. They thus upload *k-anonymized* location data to confuse the adversary on the server. This can also help prevent the crowdsourced data from tagging isolated users who cannot find at least K nearby users.

To complement our location privacy preserving solution, as we lack location providers in indoor environments, we propose a user contextualization approach based on WiFi scans which offers crowdsourcing platforms a simple way to collect the user's WiFi places in a privacy friendly way. We show how our WiFi places are created, anonymized and shared; then, we show how our WiFi places can help mobile apps provide more contextualized services based on the user's WiFi places in an anonymized way.

While the core of our location privacy preserving solution is based on peer-to-peer collaboration between mobile apps' users to hide from the crowdsourcing server. We propose the first testing framework for P2P mobile apps. We show how our framework can help developers test and scale their P2P mobile apps, our framework can also help validate the deployment of P2P privacy preserving solutions in an emulation environment using realistic datasets. We also show how our testing framework helped us tune the privacy and performance settings of our location privacy preserving solution.

The rest of this chapter is organized as follows. Section 1.1 motivates the research contributions. Section 1.2 discusses the problems we are trying to tackle in this thesis. Section 1.3 presents the main thesis goals. Section 1.4 briefs our contributions. Section 1.5 summarizes our publications derived from this research. Section 1.6 outlines the structure of this document.

1.1 Motivation

Mobile crowdsourcing platforms aim at encouraging users to share more data, but at the same time, the users' main concern is their privacy which would discourage them from sharing more data. Most of the crowdsourcing campaigns would collect raw data [13], store these data on the remote server and then they will try to figure a way to anonymize the collected data; these data holders will not inform the participating users when their data are anonymized or when their personal sensitive data are disposed.

We believe that the user is the owner and the responsible for her data, an informed decision must be taken by the user regarding her personal data; to this end, users have to be in the center of their data collection and anonymization process. These users have to decide upon the amount of data they are feeling comfortable to release; at the end, only these users know the impact of these data on their privacy. These crowdsourcing campaigns need to be user-centric.

Mobile crowdsourcing platforms can offer exclusive services, like the driving map service Waze [14]. In this example, Waze is collecting real-time location data from users, these data are tagged with spatio-temporal tags, and if not anonymized correctly, these data can disclose the *Points of Interest* (POIs) for any given user. When sharing crowdsourced data among users, these platforms are not necessarily interested in a specific user; rather, they are interested in the raw data, so this data does not have to contain users' identifiers and need to be anonymized as soon as possible in a priori way.

A lot of location privacy preserving solutions (including ours) use collaborative peer-to-peer communications to help mobile apps' users hide from the crowdsourcing server. Both developers and scientists are challenged with the lack of testing support for P2P mobile apps. Mobile crowdsourcing platforms need to scale to thousands of users [8] which makes testing these P2P mobile apps a challenge for developers. Not to mention, optimizing and tuning some settings when scaling the app is a tedious task that requires multiple deployments on real users' mobile devices, which can rapidly become unpractical, especially if these parameters concern the privacy of the participating users which can harm their privacy. Thus, before any deployment, we need a testing environment where we can scale the P2P mobile apps, tune the parameters of these apps and ensure the preservation of our users' privacy.

1.2 Problem Statement

To preserve the privacy of participating users in the crowdsourcing campaigns; and then, to test these P2P approaches before deployment, some challenges are faced by the campaign stakeholders:

- Crowdsourcing platforms must be user-centric, as the user is the owner of her data and she has to provide an informed decision on who can have access to her data and at what level her own data have to be anonymized;
- Managing user privacy in crowdsourcing platforms must be transparent to developers, as they do not necessarily master the needed skills, thus, they can harm the users'

privacy without them even knowing about the consequences of their privacy preserving choices;

- The need for a privacy friendly approach to collect the user context, as most of the proposed approaches come with either a privacy or data utility cost;
- Developers are challenged with the lack of support for testing P2P mobile apps;
- Finding the best parameters to deploy P2P mobile apps has to be automated and scaled.

In particular, this thesis aims to answer the following research questions:

- Can we increase the user's location privacy while maintaining a quality crowdsourced data?
- Can we put the user in center of her privacy control?
- Can we share the WiFi places between users while preserving both their privacy and the data utility?
- Using WiFi scans, can we detect if the user is moving between places or not?
- Can we have a distance between the WiFi places?
- Can the P2P testing framework support the developers in validating the P2P mobile apps deployment?
- Can the P2P testing framework support the developers in tuning crowdsourcing campaign parameters before deployment?

1.3 Thesis Goals

The first goal of this thesis is to protect the users' privacy when participating in a crowdsourcing campaign. We want to put the user in the center of control of her privacy when she is participating into a crowdsourcing campaign. With this control, users will find it easy to participate to crowdsourcing campaigns as their first concern is always about their privacy and how their data are going to be anonymized and stored.

As we do not trust the servers for protecting the crowdsourced data (due to potential vulnerabilities), we do not want to keep raw data on the server, as the users do not have to trust the remote server. Furthermore, when the users' data are in the server they are subject to legal obligations like the *EU General Data Protection Regulation (GDPR)* and the *Privacy Act of 1974* of the USA.

Another thesis goal is to provide a way to collect the user context using a cost-effective method while preserving the users' privacy. We want to study the creation the WiFi places and analyze them in order to provide a good user contextualization. We want to provide a way to share these WiFi places with other users in a way that preserves their privacy while still providing an insightful information on the crowd level.

Finally, we want to provide a testing framework to test the privacy preserving solutions that uses P2P communications, before risking real users' privacy or involving any deployment cost. Because our proposed privacy preserving solution involves P2P communications, we also want to scale the testing of P2P mobile apps to provide as realistic deployment as possible using real-life datasets.

1.4 Contributions

This thesis proposes to put the user in the center of the privacy equation and to exploit the nearby properties between users to confuse the adversaries willing to identify isolated users. Our intention is to help preserve and increase the user's privacy while maintaining an acceptable overall data utility. Because data scientists want to maximize the data utility without caring about endangering the privacy of the participating users, we believe that the privacy settings should be coming from the users who produce the data. To this end, we want to put the user in control of her privacy. The user will thus decide how comfortable she feels sharing her location data with others and what privacy guarantees her data must provide.

Second, to complement the first contribution, for indoor environments, we propose to use WiFi places to grab the user's context in a privacy friendly way. In particular, we define a WiFi places recognition algorithm based on WiFi scans data and we show how this WiFi places data can be shared between users while preserving their privacy. We propose multiple ways of sharing the WiFi places, including sharing them with other apps, with other devices and with other users. Thus, the WiFi places data can be adapted to each use case and to the user's wanted privacy level. Our algorithm is able to detect if the user is moving between places or she is staying in the same place.

Finally, while the core of our location privacy preserving contribution is based on peer-to-peer collaboration between the workers of the crowdsourcing campaign, we propose a testing framework to test peer-to-peer privacy preserving collaborative solutions. In our P2P testing framework, we support the developers testing their P2P mobile apps, we help the campaign designers choose the best privacy preserving parameters for by running peer-to-peer privacy preserving solutions in an emulated environment using realistic mobility datasets. This helps the campaign designers find the best privacy, utility, performance and cost-effective parameters to tune, before risking any real users' privacy with the untested crowdsourcing campaign.

1.5 Publications

Several works presented in this thesis manuscript have been published in international peer-review conferences. We present below all the published work related to this thesis; we also present the work under ongoing submission.

1.5.1 Publication details

- **Lakhdar Meftah**, Maria Gomez, Romain Rouvoy, and Isabelle Chrisment. *"ANDROFLEET: testing WiFi peer-to-peer mobile apps in the large."* In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, pp. 961-966. IEEE Press, 2017. [15]

- **Lakhdar Meftah**, Romain Rouvoy, and Isabelle Chrisment. *"Testing nearby peer-to-peer mobile apps at large."* In Proceedings of the 6th International Conference on Mobile Software Engineering and Systems, pp. 1-11. IEEE Press, 2019. [16]

- **Lakhdar Meftah**, Romain Rouvoy, and Isabelle Chrisment. *"FOUGERE: User-Centric Location Privacy in Mobile Crowdsourcing Apps."* In IFIP International Conference on Distributed Applications and Interoperable Systems, pp. 116-132. Springer, Cham, 2019. [17] **(Best Paper Award)**.

Two other local research communications (in France) are:

- **Lakhdar Meftah**, Maria Gomez, Romain Rouvoy, Isabelle Chrisment, "Androfleet : Testing WiFi P2P Mobile Apps in the Large". GdR GPL 2017, Montpellier France.

- **Lakhdar Meftah**, Romain Rouvoy, Isabelle Chrisment, "Testing Nearby Peer-to-Peer Mobile Apps at Large", CIEL Conference 2019, Toulouse France.

In addition, two works are under evaluation:

- **Lakhdar Meftah**, Romain Rouvoy, and Isabelle Chrisment. *"Improving User-centric Privacy in Data-intensive Mobile Apps."* Journal of Parallel and Distributed Computing, Special Issue on "Distributed Applications and Interoperable Systems: Recent Advances and Future Trends", Submitted on Sept 1st. [18]

- **Lakhdar Meftah**, Romain Rouvoy, and Isabelle Chrisment. *"Leveraging WiFi Places for Anonymous Location Data Collection."* To be submitted, pp. 1-10. 2020. [19]

1.5.2 Proof of Concept

During this thesis, open source tools have been developed to support the proposed work:

- **FOUGERE** [20]: An anonymous location data acquisition middleware for mobile apps.
  - Provides a more concise control over the user's Location data,
  - Gives more privacy guarantees to the user.
- **ANDROFLEET** [21]: A framework to test WiFi Peer-To-Peer mobile apps in the large.
  - Provides a testing environment for apps that uses Android,
  - Scales the testing with hundreds of peers of mobile devices.
- **PEERFLEET** [22]: An extension of ANDROFLEET to test and scale P2P mobile apps before deployments.
  - Provides a testbed for peer-to-peer privacy preserving solutions,
  - Helps tune the P2P mobile apps' settings.
- **WiFi Places** [23]: An automatic user contextualization framework based on WiFi fingerprints.
  - Provides an automatic WiFi contextualization,
  - Provides a privacy friendly sharing of the user WiFi places.
- **DayKeeper** [24]: An Android crowdsourcing mobile app using the WiFi places framework.
  - Generates an automatic user timeline from WiFi scans.
  - Detects when the user is out of routine.

1.5.3 Awards

This thesis has received one award:

- **Best Paper Award**. **Lakhdar Meftah**, Romain Rouvoy, and Isabelle Chrisment. *"FOUGERE: User-Centric Location Privacy in Mobile Crowdsourcing Apps."* [17]. In the international conference : DAIS 2019 - 19th International Conference on Distributed Applications and Interoperable Systems. Lyngby, Denmark.

1.6 Thesis Outline

The remainder of this thesis is organized as follows.

In Chapter 2, we first present the research background, we then review the state of the art on the main research areas related to our work. We also present some of the industrial tools and methods related to our research.

Because we use ANDROFLEET to validate our first contribution, we start by describing our third contribution in Chapter 3, a nearby P2P mobile apps testing frameworks (ANDROFLEET and PEERFLEET) to test P2P privacy preserving solutions.

Then, in Chapter 4, we present our first contribution, a privacy preserving solution called FOUGERE where we put the user in the center to control her privacy. To this solution, in Chapter 5, we add our second contribution, a new approach to collect the user's context in indoor environments that is privacy friendly and cost-effective using WiFi scans.

Finally, in Chapter 6, we conclude and share our vision for upcoming research directions.

Chapter 2

State of the Art

Thanks to the wide adoption of mobile devices, mobile crowdsourcing has emerged as a convenient approach to gather meaningful and scalable environmental datasets by involving citizens in the process of performing measurements in the wild [25–28]. While the development of mobile crowdsourcing apps is clearly leveraged by the *Software Development Kits* (SDK) made available by Android and iOS, mobile crowdsourcing platforms are bringing another level of abstraction to ease the design and the deployment of mobile crowdsourcing campaigns [29–32].

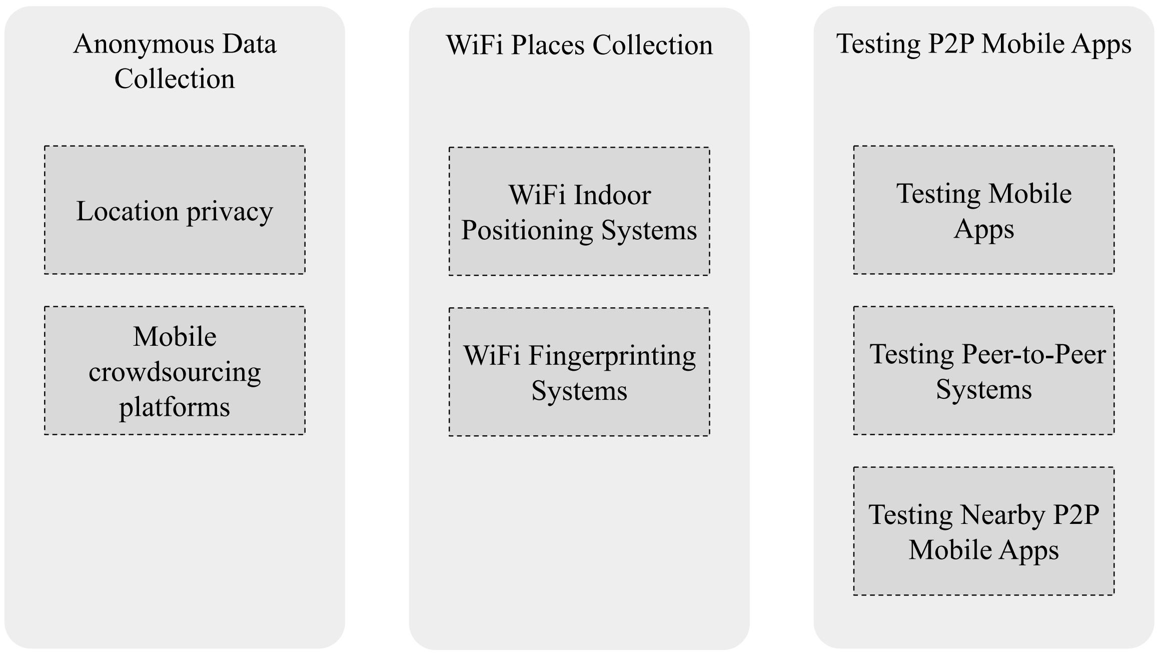

Anonymous Data Collection
Location privacy
Mobile crowdsourcing platforms
WiFi Places Collection
WiFi Indoor Positioning Systems
WiFi Fingerprinting Systems
Testing P2P Mobile Apps
Testing Mobile Apps
Testing Peer-to-Peer Systems
Testing Nearby P2P Mobile Apps

Fig. 2.1 Research Areas and subareas related to this thesis

This chapter reviews the state of the art in the research areas and subareas (cf. Figure 2.1) related to this thesis. In particular, we consider that this thesis contributes to the following three research areas: *Anonymous Data Collection*, *WiFi Places Collection* and *Testing P2P Mobile Apps*.

The remainder of this chapter is therefore organized as follows. Section 2.1 discusses the state-of-the-art of privacy-preserving methods in crowdsourcing mobile apps. While Section 2.2 discusses the related work in collecting the user WiFi places. Finally, Section 2.3 reviews the research area of testing P2P mobile apps.

2.1 Anonymous Data Collection

With the ubiquity of the mobile devices, both scientific and industrial communities leverage the sensors of the mobile devices to collect insightful information about the user. As depicted in Figure 2.2, mobile crowdsourcing campaigns typically consist of several stages: *i)* the description of the data to be crowdsourced, *ii)* the deployment and the gathering of the dataset in the wild, *iii)* the aggregation and storage of datasets in the Cloud, *iv)* the processing and *v)* publication of the campaign results. However, along all these stages, *Sensitive Personal Information* (SPI) can be conveyed by the mobile app and potentially be subject to attacks from adversaries, therefore motivating the development of a better privacy support.

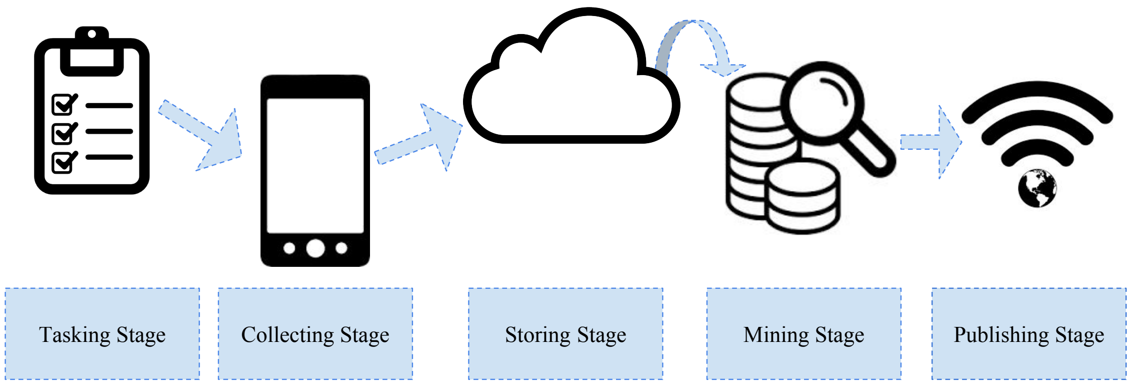


Fig. 2.2 Anatomy of a mobile crowdsourcing campaign

We identify 4 categories of SPI that can be collected by crowdsourcing campaigns and might be exploited by adversaries:

Identifiers group all persistent or transient identifiers that can take the form of a device ID (IMEI) or Google account ID, for example, to explicitly identify a worker from the perspective of a mobile crowdsourcing system. However, such identifiers may directly name the worker or be used to perform context linking attacks by combining several measurements;

Point of Interest (POI) groups all the forms of geolocated data that can deliver some spatial information on the location of a worker. This includes GPS locations, but also places' check-in, cell tower ID or location, which can be used to produce maps from crowdsourced measurements. However, these POIs may also reveal the home, office, shopping and/or leisure locations of workers that can uniquely identify them [33–35];

Routines concern any information that can be used to capture a recurrent activity of a worker. This category of SPI covers in particular any form of timestamp, no matter the format and the precision. While this precious information often appears as harmless, it may also be used by context linking attacks to group crowdsourced data and observe correlation along time (*e.g.*, nights, week-ends);

Markers finally focus on information whose entropy in terms of values can be exploited to detect outlier workers and thus be indirectly used as an identifier by an adversary. There can be a wide diversity of such markers depending on the purpose of the mobile crowdsourcing system. For example, in the case of MobiPerf (a mobile app for measuring network performances) [27], the properties of device manufacturer, model, OS version and network carrier can be considered as unique if a worker uses some original/old mobile device. Beyond individual values, the combination of markers can also lead to the disclosure of a unique fingerprint, which can be used by an adversary to extract insightful information from a group of workers [36].

We believe that this classification is sufficient to identify sensitive personal information and support the privacy of workers. While an ontology could be defined to further organize the diversity of SPI along these 4 categories, the definition of such an ontology remains out of the scope of this thesis.

2.1.1 Location Privacy

Location Privacy Threats

When studying the location privacy threats in crowdsourced datasets, we consider that the adversary can exploit two dimensions of knowledge [37]: *spatio-temporal information* and *context information*.

In the context of mobile crowdsourcing systems, *spatio-temporal information* refers to the capability of the adversary to access a history of crowdsourced data—*i.e.*, several measurements reported by a single worker. In the case of a compromised storage server (or connection to the storage server), such an assumption holds as the adversary can get access to sufficiently large volume of crowdsourced data to build some spatio-temporal knowledge.

Beyond spatio-temporal information, *context information* refers to any additional information that an adversary can exploit. This covers embedded knowledge that is included in the crowdsourced dataset (*e.g.*, markers) or side knowledge that an adversary can obtain from other information sources (*e.g.*, the number of involved workers).

Several location privacy attacks exploit crowdsourced data to reveal the identity of workers. For example, *identity matching* [38] can be used to attack several pseudonyms of a worker. The adversary can link several pseudonyms based on equal or correlating attributes to the same identity, this way, the provided privacy of the obfuscated pseudonyms is no longer useful.

A *location tracking attack* [39] makes use of several location updates known to the adversary. For example, this attack can be used against randomly changing pseudonyms without using mix zones. Here, the adversary can correlate succeeding pseudonyms by linking spatial and temporal information of succeeding location reports even if an obfuscation mechanism is used. For instance, the adversary can try to reconstruct the trajectory of a worker based on the provided locations of several pseudonyms.

In a *maximum movement boundary attack* [40], the adversary computes the maximum movement boundary area, where the worker could have moved between two succeeding position reports. For example, the position of the first measurement uploaded at time t_1 helps the adversary to increase the precision of the measurement uploaded at t_2. In this example, only a small part of the area reported at t_2 is reachable within the maximum movement boundary. Therefore, the remaining area of the position update can be excluded by the adversary.

Then, *location distribution attacks* [41] observe that workers are not often distributed homogeneously in space and use outlier locations in sparsely populated areas to link and extract the crowdsourced data of the same worker. In particular, spatial-temporal clustering algorithms, like ST-DBSCAN [42], can be used in such a case to group the worker's traces together. In the case of dense areas, an adversary can use the speed of the workers to group the locations related to a given trajectory [43]. However, the performance issue of such techniques must be taken into consideration [44].

Finally, given a dataset of workers' location traces with timestamps, *Mobility Markov Chain* (MMC) attack [33] reports on the workers favorite POIs with labels, such as home, work, or shopping. Then, by correlating these POIs with a map, it can reveal SPI about her work and her home, which combined with a public phone directory can deduce her full name and leave opportunity to query social networks, etc.

Most of these attacks are implemented within privacy evaluation tools [45–48]. In particular, the *Location-Privacy and Mobility Meter* (LPM^2) tool [45] provides a reusable toolkit to evaluate location privacy.[1] LPM^2 uses statistical methods (such as Bayesian inference, hidden Markov model, and Markov-Chain Monte-Carlo methods) to formalize and implement the location inference attacks that quantifies the location privacy of mobile users, given various location-based applications and *location-privacy preserving mechanisms* (LPPM).

Location Privacy Protection Mechanisms (LPPMs)

LPPMs are an interesting mechanisms that can be used to limit user privacy leaks [49]. A large body of the related work has been devoted towards the latest stages of mobile crowdsourcing campaigns by improving the privacy properties of datasets once uploaded to remote servers [50–55, 35]. These techniques contribute to preserving the privacy of workers while limiting the sanitization impact on the quality of the resulting dataset.

In particular, Gambs *et al.* [56] propose a toolkit called GEPETO to enhance the privacy of spatio-temporal datasets by applying a sanitization process. They provide a tool to sanitize spatio-temporal datasets and to measure the privacy and the utility trade-off of the resulting dataset.

Ma *et al.* [57] discuss some vulnerabilities found in anonymized mobility traces. In a formally detailed approach they explain how an adversary can identify workers using complementary metadata collected while observing workers.

Baik *et al.* [58] propose a new algorithm to achieve guaranteed anonymity in a spatio-temporal dataset; they focus on the home identification and user tracking privacy threats.

While all of these works focus on protecting the collected dataset, raw datasets stored on a remote server may be leaked through security breaches. Moreover, early stages of mobile crowdsourcing campaigns remain exposed to privacy attacks (*e.g.*, man-in-the-middle). This issue has been identified as an open challenge by Christin *et al.* [59]. Later, Christin *et al.* [60] have proposed a detailed approach that requires the involvement of participants in the anonymization process.

To ensure that workers will not upload any corrupted data, reputation systems [61, 62] have been proposed to build a trust between the collecting server and the workers, thus limiting the adversaries impact on the corrupted data that they want to upload to the server. In the crowdsourcing phase, Mousa *et al.* [63] introduce a new attack in crowdsourcing environments that are using reputation systems exploiting their attack, they can link all the data of one user without her identifier in the collected data. They propose a privacy-preserving

[1]http://icapeople.epfl.ch/rshokri/lpm

reputation system, called PRIVASENSE, in which they integrate the reputation system while ensuring the worker's anonymity against the reputation system itself. These works motivate the need to protect our users from the server (reputation system) as they show how they can link all the users' data. This highlights the need to protect the users from the collection server itself.

A lot of LPPMs have been proposed to anonymize the mobility traces. A major limitation of the existing LPPMs is that they are used on the server side and they require all the users' collected data as input, which forms a single point of failure from the privacy point of view, as our users do not have to trust the server. In our work, we use the most effective LPPMs and adapt them to work in mobile devices. We thus anonymize the data before they reach the server. In order to upload anonymized data to the server, we propose to send the data through other users (other than the data producer). Next, we will review some of the works that propose user collaboration to exchange data between users in a privacy friendly way.

Peer-to-Peer Collaborative Privacy-preserving

Approach	Technique	LPPM	Communication
Show *et al.* [64] (2011)	Multi-Hop queries	Spatial cloaking	P2P protocols
Shokri *et al.* [65] (2011)	Shared buffer	MobiCrowd	WiFi *Access Point* connection
Shokri *et al.* [66] (2014)	Shared context	MobiCrowd	WiFi Direct
Peng *et al.* [67] (2017)	Fake queries	Spatial Cloaking	WiFi Direct

Table 2.1 Summary of related work using Peer-to-peer collaborative privacy-preserving solutions

Some LPPMs provide collaborative privacy preserving methods which are particularly interesting in protecting the users' privacy in *Location-Based Services* (LBS) [68].

Some privacy protection mechanisms used in LBS can be adapted to mobile crowdsourcing platforms. As both domains share the same purpose from the location privacy view angle, which is hiding users' locations from the remote servers collecting users' locations. In LBS, users query the remote server to get some context information about their locations. For example, the user must provide their location to a remote server to get the satellite image corresponding to their location. Most of the the proposed solutions would benefit from the nearby users to mix the user's location with theirs to confuse the adversary on the server side.

In Table 2.1 we consider the solutions where users collaborate to hide their locations data from the server [64, 45, 66, 67].

In particular, Show *et al.* [64] use spatial cloaking to hide users' locations from the LBS, the user's query passes through several hops before arriving to the server, providing some spatial cloaking on each hop, thus, they prevent the server from linking the query creator with the query itself, they use communication over *peer-to-peer* (P2P) protocols to disseminate the query over multiple hops to the server.

Shokri *et al.* [65] present a novel LPPM to hide the user location against LBS. They use a shared buffer to query the server only when the buffer lacks that information. A WiFi *Access Point* connection allows a collaboration between users. They need to change the LBS server architecture in order for their solution to work.

Shokri *et al.* [66] introduce a user-collaborative privacy-preserving approach for LBS. The users keep context information in a buffer and pass it to other peers. They only seek the LBS if the shared buffer does not contain the sought information. WiFi Direct communications are used to exchange the shared context between users. Their solution does not require changing the LBS server architecture nor involve third party servers.

Peng *et al.* [67] present a more complex scheme where the users obfuscate their actual trajectory by issuing fake queries to confuse the LBS. They propose to issue other users' queries along with the user real queries to confuse the LBS. WiFi Direct is used to exchange queries between users. They will need to issue a lot of fake queries to confuse the server, especially when they rely on just one location obfuscation LPPM.

Yet, such approaches are not widely adopted by LBS solutions as they fail to demonstrate their effectiveness and justify their cost in a realistic deployment. In particular, caching all the users' trajectories can take an important storage space in mobile devices, not to mention the delays added by such solutions. To these points, one can add the new threats evolving from including other adversaries in the scheme, like the *man-in-the-middle* (MITM), which has to be considered in each adversary model.

For all of the above reasons, we consider the following research work that proposes a cost effective and privacy preserving solution for peer-to-peer communication. In particular. Luxey *et al.* [69] propose a novel decentralized dissemination protocol that exploits opportunistic interactions between peer-to-peer users. Their solution called Sprinkler, a gossip-based protocol that enables an interaction between users while preserving their privacy. Bromberg *et al.* [70] present a peer-to-peer decentralized communication protocol called Cascade, to share the user's sessions between her devices in a seamlessly way while preserving her privacy. Aditya *et al.* [71] introduce EnCore, a peer-to-peer nearby communication for opportunistic encounters using Bluetooth communication to give the user the control over her privacy. Tsai *et al.* [72] propose enClosure, a peer-to-peer communication based on nearby encounters for mobile devices which ensures a privacy preserving communication between users. The above mentioned works provide a privacy preserving communications between users. The main intention of these work is either to reduce network costs or to explore the proximity of the users providing a secure and privacy preserving communication. In our work, we use the same opportunistic communication privacy preserving scheme, our intention is not to propose another communication scheme, but to extend existing communication schemes to disseminate crowdsourced data to help users hide from the collection server. Furthermore, in our work we anonymize the crowdsourced data on each encounter on their way to the server.

2.1.2 Mobile Crowdsourcing Platforms

Mobile crowdsourcing has gained a lot of attention for the last decade [13, 73], and this is due to the capabilities that offer these mobile phones. Scientists and decision makers can now run their experiments on the user mobile device to collect real life scenarios. The collected data can be of different types: geospatial data [13], behavioral data [74], environment monitoring [75, 76], Internet quality monitoring [77] and health data [78].

To help scientists, crowdsourcing platforms leverage all the technical kit to deploy their crowdsourcing collection campaign, to monitor the campaign and to store the collected data in the cloud servers.

In our work, we test our privacy preserving techniques on both APISENSE crowdsourcing platform and MobiPerf crowdsourcing mobile app:

APISENSE Platform

The crowdsourcing platform APISENSE [79] offers the scientists, with no technical background, the ability to acquire insightful data from the field. APISENSE manages the deployment of the dada acquisition campaigns as dedicated tasks described in JavaScript, which are remotely deployed to all the participants in real-time. APISENSE offers a generic mobile app that can be used by various scientists for different use cases. This mobile app is in charge of directly executing the deployed tasks on the participant's mobile device. The collected data comes from most of the sensors available on a mobile device, but the scientists can also ask the participant to perform a specific action whenever a given event happens, like answering a question whenever the participant comes back home.

The APISENSE platform provides a web interface to monitor and to control the crowdsourcing campaign. Through this web interface, the campaign manager (the scientist) can invite new users, update the data acquisition script and update it in real-time. The campaign manager can visualize the anonymized user data, as soon as they are uploaded to the server. Although, these received data do not contain any user identifiers, it needs to be further anonymized before starting to process the collected data.

MobiPerf App

MobiPerf is open source mobile app [27] for measuring network performances at regular intervals in the background. The MobiPerf mobile app use the open source library Mobilyzer [80] that facilitates the network measurement crowdsourcing campaigns. It contains most of the network measurement tools which can be used just by configuring a file that contains the measurement tasks (*e.g.*, *ping*, *traceroute*, *HTTP GET*) to be executed on the user mobile device. To use the MobiPerf mobile app, the scientists need to customize the app to meet their needs and to change the server address by their own server address.

Privacy-preserving Mobile Crowdsourcing Platforms

Approach	Collaborative	Integration	Trust model
ANONYSENSE [31] (2008)	Yes	Library	Third-Party server
PEIR [81] (2009)	No	Platform	Server
PRISM [82] (2010)	No	Sandbox	Server
HP3 [83] (2010)	Yes	Platform	Third-Party server
SPEAR [84] (2014)	NO	Platform	Third-Party server

Table 2.2 Summary of related work in privacy of mobile crowdsourcing platforms

Mobile crowdsourcing platforms are used to collect data about the contributing users, these data include some *Sensitive Personal Information* (SPI), which introduce privacy issues. This is why researchers are actively working on privacy protection mechanisms for crowdsourcing platforms [85, 86].

Table 2.2 summarizes the work related to the privacy of mobile crowdsourcing platforms.

ANONYSENSE. Cornelius *et al.* [31] have proposed ANONYSENSE: a mobile platform for opportunistic sensing. Because the server hosting the collected dataset can trace the worker's wireless access points, they use an anonymization network to hide the worker locations and they rely on a third-party server for routing the data. ANONYSENSE also supports reporting data with a statistical guarantee of *k*-anonymity. The workers' data are blurred and combined before being reported to the remote server.

PEIR. Mun *et al.* [81] introduce a platform for participatory sensing where they include an access control mechanism to let workers decide who can access to their data. In this approach, they control which sensors the mobile app has access to and who has access to the data on the server.

PRISM. Das *et al.* [82] present a *Platform for Remote Sensing using Smartphones* (PRISM). They use a sandbox environment to prevent mobile apps from accessing mobile sensors. Users participating in crowdsourcing campaigns will install the PRISM runtime and then the untrusted sensing app. This app can only fetch data after ensuring that the user cannot be identified using these data. PRISM acts thus as an intermediate to guarantee the user privacy preservation. As discussed in [82], both ANONYSENSE and PRISM suffer from similar privacy leaks as the mobile app collects data using local sensors made available by their mobile device, allowing data to be linked to the worker identifiers easily.

HP3. Hu *et al.* [83] present a collaborative privacy-preserving platform, which uses social networks to hide workers from the server. The user data will travel through other devices (randomly chosen) before reaching the collection server, and the server thus cannot guess who is the data owner. In their approach, they rely on third-party servers (the social network) that can store all the exchanged locations along with workers identifiers.

SPEAR. Gisdakis *et al.* [84] introduce a privacy-preserving architecture for crowdsourcing platforms. They address all the challenges of participatory sensing like security, privacy and worker incentives, as well as limiting the participation to legitimate workers and authenticate them using third-party servers.

These works use either a third-party server or propose a postoriori anonymization techniques. In our work, we have a different adversary model. As we do not trust the collection server or any third-party server, we apply a priori data anonymization scheme to hide workers from the crowdsourcing server.

2.1.3 Synthesis

To the best of our knowledge, the state of the art fails to appropriately address the anonymization schemes along the earliest stages of a mobile crowdsourcing campaign in order to limit potential privacy threats, especially when the adversary model includes trusting a remote server. Therefore, in this thesis, we intend to address this limitation by proposing an a priori approach that uses existing privacy protection mechanisms in the mobile device by providing the first decentralized dissemination to adjust location privacy in mobile crowdsourcing systems.

2.2 WiFi Places Collection

Mobile devices are equipped with a variety of sensors. Therefore, they can collect insightful data about both the user surrounding environment and the user activities to feed the mobile apps with the user context. These context data include the user GPS location and the surrounding WiFi networks.

One of the methods to contextualize the user with WiFi networks is the *indoor positioning systems* and *WiFi fingerprinting systems*. These systems use the surrounding scanned WiFi *Access Points* (APs) to localize indoor users or to track users' places.

2.2.1 WiFi Indoor Positioning Systems

Due to the absence of the *Global Positioning System* (GPS) signal inside buildings, many new systems have been proposed as an indoor positioning system [87]. Among these systems, WiFi-based systems, which take advantage of the already spread WiFi *Access Points* (APs) and the users phone that are equipped with WiFi scanning abilities. WiFi signals can be used to calculate the position of a mobile device down to 1.5 meters precision [88]. While some research works require a prior training phase (online mode) to survey the location of the antennas in the map [89, 90], other proposals have been proposing an automatic way to build a radio map without any prior knowledge about the antennas [91–93]. In particular, Liu *et al.* [94] propose a peer assisted localization method to improve localization accuracy. Gorski *et al.* [95] present a multi-building indoor localization system using WiFi fingerprint based on the *K nearest neighbors* (KNN) method. Salazar *et al.* [96] use a Type-2 fuzzy inference systems to cluster WiFi fingerprints to localize the indoor users. Li *et al.* [97] introduce a privacy preserving WiFi indoor localization system. They argue that the localization query can inevitably leak the client location and lead to potential privacy violations. Jin *et al.* [98] propose a real-time WiFi positioning algorithm with the assist of inertial measurement unit to overcome the RSS variation problem. Pulkkinen *et al.* [89] present an automatic fingerprint population using theoretical properties of radio signals. They rely on the locations of WiFi APs and collecting training measurements. Salamah *et al.* [99] use the *Principle Component Analysis* (PCA) to reduce the computation cost of the WiFi indoor localization systems based on machine learning approach. Yiu *et al.* [100] apply training measurements and a combined likelihood function from multiple APs to measure the indoor and the outdoor user position. Capurso *et al.* [101] present an indoor and outdoor detection mechanism which can be used to optimize GPS energy usage. Yiu *et al.* [90] review the various methods to create the radiomap. Then, they examined the different aspects of localization performance like the density of WiFi APs and the impact of an outdated radiomap. Ahmed *et al.* [102] provide a new optimized algorithm for fast indoor localization using WiFi channel state information. Caso *et al.* [103] introduce an indoor positioning system that relies on the *Received Signal Strength* (RSS) to generate a discrete RSS radiomap. Li *et al.* [93] present SoiCP, a seamless outdoor-indoor crowdsourcing positioning system without requiring site surveying. Crowdsourced WiFi signals are used to build a radiomap without any prior knowledge.

2.2.2 Synthesis

Indoor localization systems using WiFi scans are proven to be very useful and precise. These indoor localization systems are still limited to a restricted number of buildings, which have to be either equipped specifically to localize users with WiFi antennas, or contain an important number of users to reduce the errors. Some approaches propose to learn the position of these WiFi antennas for every building. In our work, we do not require any prior knowledge about the building and the location of the WiFi antennas. Moreover, we do not intend to localize the user but to contextualize the user by knowing if she is in a different place or not, which can be answered without the need for a priori extensive WiFi surveying. Next, we will review some of the works that do WiFi fingerprinting to contextualize the user without the need to prior knowledge about the buildings, and can work with new buildings on the go.

2.2.3 WiFi Fingerprinting Systems

Due to the cost-effective of WiFi signals [104] against GPS regarding battery usage and accuracy, a lot of recent work in user contextualization were interested in using WiFi signals to contextualize the users both indoor and outdoor.

With the location permission granted, surrounding WiFi APs can be provided to the app, these WiFi APs will act as the place fingerprint, different WiFi APs signals received mean different places. Tracking the user for several hours using just WiFi scans can provide a valuable information about the user activities [105], personality traits [106] and the user routines [107]. These data can also be used to find a unique fingerprint per users [108, 109], find social networks [110] or track users in an offline mode [111, 112]. To localize the user, her WiFi fingerprint has to be compared to other collected WiFi fingerprints. Once a similar fingerprint is found, the corresponding location is attributed to the user. Some research works are focusing on optimizing the similarity distance between two WiFi fingerprints [113, 114, 100, 115].

In particular, Zhang *et al.* [116] introduce POLARIS, a localization system using cluster based solution for WiFi fingerprints. They explain how to collect and compress a city scale fingerprints, then, how to find the location of a user using similar WiFi fingerprints. Sakib *et al.* [117] present a method to contextualize the mobile user using the WiFi APs fingerprints, with a simple clustering algorithm that creates a place for each group of WiFi APs. They validate their results with cellular cell ids datasets. Sapiezynski *et al.* [118] use WiFi to enhance GPS traces localization, with one GPS location per day; they can localize 80% of mobility across the population. Wind *et al.* [119] use WiFi scans to infer stop locations for users, their algorithm can tell if a user is moving or in a stationary state using just WiFi fingerprints. Choi *et al.* [120] propose a method for an energy efficient WiFi scanning for user contextualization; they try to minimize the number of scans depending on the scanned WiFi APs.

2.2.4 Synthesis

While a lot of work have been done to localize the user using WiFi indoor localization techniques, most of them require either a training phase or user input to locate the user using the WiFi antennas. Several research works have been focusing on the WiFi fingerprinting based on the clustering algorithms and similarity distances, but they do not report on the utility of the collected data, its cost or its privacy preserving methods. Furthermore, they do not provide any libraries or frameworks in order for their methods to be used and further improved. The WiFi places collection methods and algorithms have to be well documented and the resulted data have to be analyzed so that new contributions can be built on top of it.

2.3 Testing P2P Mobile Apps

Most of the reviewed privacy preserving methods propose to collaborate between mobile apps' users using peer-to-peer communications to hide the workers from the crowdsourcing server. In this section, we overview the state of the art approaches used to test these peer-to-peer mobile apps. To overview these testing approaches, we first start overviewing the works that have been done in testing the mobile apps in general.

2.3.1 Testing Mobile Apps

Testing mobile apps has taken attention in recent years both in industry and research due to the proliferation of mobile devices and the need to deliver quality apps to demanding mobile users.

Table 2.3 summarizes the state of the art in the domain of testing mobile apps.

To test P2P mobile apps, some research contributions add a set of *test oracles* [121–123]. Testing P2P mobile apps involves considering the hardware pervasive communication which therefore lead us to consider the *dynamic context* of the app being tested [124–128], and to consider controlling the executed *event sequences* in the tested app [129–132]. Also, when testing context based mobile apps, we need to do *black-box* testing as it is becoming a requirement for mobile apps testing [133–137]. Some literature work is based on *test automation* [138–140]. A bunch of approaches [141, 124, 129, 125, 126, 131] discuss *GUI testing* of mobile apps. The goal of these approaches is to provide exploration strategies to

Method	Related work
New test oracles	[121–123]
Dynamic context	[124–128]
Control event sequence	[129–132]
Black-Box testing	[133–137]
Test automation	[138–140]
GUI testing	[141, 124, 129, 125, 126, 131]
Cloud based testing	[142–144]
Multi-platform	[145, 136, 137]
Crowdsourced testing	[146]
Parallel testing	[147]
Simulation testing	[148, 149]
Reverse engineering	[150]

Table 2.3 Summary of related work in the domain of testing Mobile Apps

maximize test coverage for the UI elements and routes. Another family of approaches have presented *cloud-based* solutions to test mobile apps [142–144].

In *literature reviews*, Muccini *et. al.* [151] investigates new research directions in mobile testing automation. Kirubakaran *et. al.* [152] provide an overview of what mobile testing is, and answer some research questions about mobile testing requirements. Mendez-porras *et. al.* [153] present a systematic literature review on existing testing approaches for mobile apps. Amalfitano *et al.* [154], compare the existing online automatic testing technique for Android mobile application. Linares-Vasquez *et al.* [155] overview the state of the art of the test automation solutions for mobile apps in terms of frameworks, tools and services available to help developers testing their apps. They open new challenges concerning the test case generation, the model-based testing and the scalability of mobile testing .

In *crowdsourced testing*, Zhang *et al.* [146] present a crowd-sourced testing approach to better test real-life scenarios with freelancer testers. While this approach can be used to test Nearby P2P mobile apps, the tests are not reproducible and cannot be automated. Also, it will take the test an important amount of time (several days) to cover all the possible scenarios. This approach can be used as a complementary testing approach, but it cannot replace the main testing method.

In *simulated environments*, Serfass *et al.* [148] use a simulation solution for Android apps that are using Near Field Communication (NFC) P2P data exchange. Richerzhagen *et al.* [149] introduce a framework to test distributed mobile apps. They provide an API that can communicate with their built-in runtime on the network simulators. In this testing method, no real mobile app is running on the test-bed, hence, this testing method is used to asses the protocol and the data exchange functionalities. Further testing with other methods is still required like testing the real app in a mobile device.

In *parallel testing*, Wen *et al.* [147] present a framework for a parallel UI testing for Android apps. Their intention is to speed up the tests by running them in parallel, and not to run the app on different devices that interact with each other in a synchronized way.

Morgado *et al.* [150] introduce an approach for testing mobile applications using *reverse engineering* and behavioral patterns.

In *multi-platform* mobile testing, Song *et al.* [145] present a mobile multi-platform testing framework. They extend a common testing framework where the developers can write tests in a high code level with a common interface for the common functionalities in each platform. Other approaches have extended the multi-platform Calabash framework to support the testing of additional features, such as touch gestures [136] and sensors (*i.e.*, accelerometer, GPS, etc.) [137].

Despite the prolific research in this area, current testing frameworks for mobile apps lack support for peer-to-peer interactions for mobile apps. In the next section, we will explore the literature related to testing peer-to-peer systems in general, as these approaches can be adapted to mobile devices.

2.3.2 Testing Peer-to-Peer Systems

Almeida *et al.* [156] present a framework and a methodology for testing P2P applications, The framework is based on controlling nodes individually, allowing test cases to precisely control the volatility of nodes during their execution. Zhou *et al.* [157] introduce a framework to test peer-to-peer multi-player games. Sunye *et al.* [158] explain a model-based testing for large-scale distributed systems and peer-to-peer communications. Charaf *et al.* [159] propose an agent-based architecture for distributed system testing. Marroquin *et al.* [160] use a testing approach based on test cases dependencies for communications protocols.

Butnaru *et al.* [161] present a tool for measuring peer-to-peer platform performance using log messages to get feedback from different peers. Boldman *et al.* [162] explain in a patent a system and methods for testing peer-to-peer network application by orchestrating unit testing between peers. Gorodetsky *et al.* [163] use a peer-to-peer emulation environment for testing mobile P2P agent applications.

Although testing general P2P systems has been extensively studied by the research community. Testing P2P mobile apps still uncovered, as it requires some specifications, like detecting a nearby device and including the mobile device resource consumption (battery, cpu) in the testing environment. The research community fails to cover this specific testing support for P2P mobile apps which become a requirement as the developers and scientists are left without any support for testing their existing mobile apps. This is why we want to fill this gap, by proposing the first testing framework for mobile P2P framework for nearby encounters, by including all of the specificities of mobile devices like resource consumption and opportunistic encounters.

2.3.3 Testing Nearby P2P Mobile Apps

Nearby P2P mobile apps are standard mobile apps leveraging a nearby P2P API provided by the OS's SDK. These apps are using P2P communication through nearby communication technologies, like the WiFi, Bluetooth, BLE, NFC or ultrasonic. These P2P Nearby Apps present some testing challenges, as they require managing the peer discovery and the connection inside the mobile app. As such, one can consider that the state-of-the-art testing frameworks cannot address the challenges of assessing the quality of these apps.

While the literature lacks testing P2P mobile apps, some developers have moved ahead and proposed their own way to test this kind of apps. We report on the possibilities that can be considered with the current solutions.

Mocking the Google Nearby API. One of the solutions that a developer can consider consists in mocking the Nearby framework API. As a matter of example, we share a template file to illustrate how these apps can be unit tested using a mocked version of the Google *Nearby Connections API*.[2] When mocked, with a framework like Mockito,[3] the Google Nearby Connections API hardens and checks the behavior of the app regarding the expected sequence of calls to the API. For example, if the app calls `acceptConnection` before calling `startDiscovery`, the test should fail. However, this approach remains limited to test the nearby P2P mobile apps as:

1. the peer interactions requires to be hardcoded, and can hardly cover all the possible scenarios,
2. the forged mocks may not really reflect the actual behavior of the underlying hardware components,
3. while the developers can test the app logic, they cannot perform black-box testing, which is required when testing dynamic context-based apps.

Testing with Physical Devices. Because it is hard to mock the Nearby hardware components, developers tend to consider physical devices to perform black-box testing using frameworks

[2]https://github.com/m3ftah/NearbyTest
[3]https://github.com/mockito/mockito

like calaba.sh[4] or Cucumber.[5] However, this solution fails to scale testing with more than two devices, which quickly become a tedious task to automate and consider for any developer:

1. only the simplest scenarios can be covered. It is not possible to automate black-box tests for outlier scenarios (*e.g.*, getting out of the connectivity range requires taking devices away from each other),
2. this approach fails to scale when considering deployment to hundreds of devices,
3. the test must be kept agnostic from the underlying hardware and the OS diversity,
4. when combining with cloud providers, the tests need to control the proximity between devices, and this proximity needs to be handled by the test.

Crowdsourced Testing with Beta-testers. One of the alternative solutions to test nearby P2P mobile apps consists in using crowdsourced testing platforms to try the app on different mobile devices and manually follow the test scenarios [146]. Nonetheless, crowdsourced testing has some major drawbacks:

1. the tests are not reproducible,
2. the tests are not automated,
3. covering all possible scenarios may take a long time.

Testing using simulators. Simulating the peers' behavior is possible to assess the connection behavior and peers' interactions [149, 148]. This approach can help developers to test the interaction between peers, but it fails to test the exact behavior of the app, as mobile apps have their own dynamic context that cannot be simulated. Moreover, black box testing is not available in simulators, as there is no real app that is executed by the peer.

2.3.4 Synthesis

While a lot of work have been done in the domain of mobile apps testing and the testing of P2P systems, the state-of-the-art fails to provide a solution to test Nearby P2P mobile apps. These apps that uses Nearby technologies to communicate with other peers—using Bluetooth, for example—exist, but with no support from the research community to help test and maintain these kind of apps.

[4]https://calaba.sh/
[5]https://cucumber.io/

2.4 Conclusion

This chapter reviews the closest related work in our research areas. In this thesis, we try to fill the gap between peer-to-peer anonymous data collection for mobile apps and properly testing these data collection techniques.

Despite the prolific work that have been done to study and propose new data collection privacy preserving schemes, some of these approaches have become obsolete and require new privacy preserving schemes, especially with the new privacy attacks, the advance of the computing capabilities of the adversaries and the recent *(The EU General Data Protection Regulation)* legislation on handling user personal information.

Our privacy-preserving schemes use the collaborative P2P systems, where the state-of-the-art fails to propose appropriate testbeds and testing techniques specific to the mobile apps using this communication methods.

This thesis therefore intends to contribute an anonymous data collection solution based on peer-to-peer collaboration to help scientists collect users' data while preserving their privacy. To test this peer-to-peer collaborative solution, we propose a testing framework for P2P mobile apps that will help validate our privacy preserving solution using an emulation of crowd of devices to emulate a real life crowdsourcing campaign.

To prepare the ground for our first contribution called FOUGERE, we first start by presenting our third contribution, which is a framework for testing peer-to-peer mobile apps that we need to use to validate FOUGERE. In the next chapter, we present our testing framework for Nearby P2P mobile apps.

Chapter 3

Testing Nearby Peer-to-Peer Mobile Apps at Large

This chapter present the thesis contribution on testing P2P mobile apps, it starts by motivating the need for a new testing framework for P2P mobile apps, the motivation contains an empirical study of some Google play users' reviews implying that developers are shipping buggy apps. It also overviews existing P2P APIs and shows how developers are failing to stay up to date with new P2P API versions. A summary of all challenges the developers are facing to test these apps, is presented along with the main requirements that any testing framework has to cover. Later, this chapter introduces ANDROFLEET a WiFi P2P testing framework that can scale the testing using a GPS mobility dataset, its architecture and a use case are also presented. Then, PEERFLEET (an extension of ANDROFLEET) which adds support for all nearby P2P mobile apps is presented, PEERFLEET also supports the P2P discovery parameters tuning. The architecture of PEERFLEET is presented, along with its evaluation following two research questions, the first one is about detecting bugs in P2P mobile apps, and the second one is about tuning the discovery parameters for P2P mobile apps. The results show how PEERFLEET can detect different types of bugs, categorized into (permission, scalability, protocol and pervasive bugs). For the second research question, using different proximity datasets, PEERFLEET can report on the impact of discovery parameters (5*mn*, 10*mn*, 20*mn*) and the used communication (WiFi, Bluetooth) on the performance of the P2P mobile apps. A discussion is given to suggest that each set of parameters is suitable for a specific use case. Finally, a brief summary of battery consumption is presented to give an idea about the importance of discovery parameters.

3.1 Introduction

Mobile devices are intensively used to support different forms of interactions among users in the physical world. However, most of these interactions are currently conveyed through remote communications over the Internet, even when the involved peers are co-located. This indirect communication scheme may suffer from communication bottlenecks depending on the *Quality of Service* (QoS) of the underlying network, as well as disclosure of sensitive data by relying on third parties (*e.g.*, public local area network).

The GOOGLE NEARBY framework has been released in 2014 to fulfill such weaknesses and to deliver a native support to enable *peer-to-peer* (P2P) communications among neighboring devices. In particular, GOOGLE NEARBY supports device-to-device exchanges by implementing network advertisement, discovery and communication features atop of standard wireless technologies (including WiFi hotspots, Bluetooth, BLE). Unlike previous *ad-hoc* communication APIs, users are not prompted to turn on Bluetooth or WiFi—GOOGLE NEARBY enables these features when required, and restores the device to its prior state once the app is done using the API—thus ensuring a smooth user experience. GOOGLE NEARBY, therefore, supports the design of collaborative whiteboards, local multiplayer gaming, multi-screen gaming, offline file transfers and many other applications.

However, 4 years after releasing this framework, only a limited number of Android apps have integrated GOOGLE NEARBY in production (only 272 apps according to `42matters.com` when we wrote this chapter). We guess that this lack of adoption is probably due to the difficulty to properly configure and test *peer-to-peer* mobile apps. Moreover, we also observed that users of *peer-to-peer* mobile apps tend to complain from app crashes and battery issues, which inevitably impacts the adoption of these apps at large.

This chapter, therefore, addresses this limitation and promotes new testing abstraction to enhance the development of *peer-to-peer* mobile apps based on the GOOGLE NEARBY framework. Because faithfully reproducing testing conditions with physical devices is extremely hard for developers. We propose two testing frameworks, we first present ANDROFLEET, a WiFi P2P testing framework for WiFi P2P mobile apps, then, we extend ANDROFLEET and propose our second testing framework PEERFLEET that adds support for other nearby P2P communication (Bluetooth, BLE and ultrasonic) and supports tuning the associated discovery parameters of thes nearby P2P communications.

More specifically, ANDROFLEET and PEERFLEET can control a crowd of device emulators through specific *actions* and check the expected behavior of mobile apps through the definition of specific *assertions*.

The remainder of this chapter is therefore organized as follows. We motivate this work by introducing the Google Nearby technology and reporting on some empirical evaluation

Table 3.1 50^{st} nearby apps ratings in the Google Play store (April 2018)

App package name	Rating	Bad ratings	WiFi Direct issues	*Not working*	*Crashing*	*Battery*
`com.remaller.android.wifitalkie_lite`	4.2	352	1	6	1	-
`com.elmoha.Wifitalkie`	3.8	174	5	16	1	-
`org.servalproject`	4.2	175	7	21	6	1
`com.opengarden.firechat`	3.6	9,592	2	21	15	2
`com.offlinechatapp.android`	3.7	118	25	-	15	-
`com.estmob.android.sendanywhere`	4.7	4,019	-	3	-	-
`com.everwasproductions.demo.wifidirect`	3.5	275	-	13	1	-
`ru.kolif.wffs`	4.5	28	-	1	-	-
`com.ftp.WifiDirectFileTransfer`	3.5	106	28	-	1	-
`ca.nickadams.wifi.direct.file.transfer`	3.1	118	2	10	6	-
`aaqib.shareonwifi`	4.1	11	-	1	-	-

of its adoption in the wild in Section 3.2. Section 3.3 introduces and discusses how *peer-to-peer* mobile apps based on GOOGLE NEARBY can be effectively tested by state-of-the-art frameworks. Section 3.4 presents ANDROFLEET our WiFi P2P testing framework for WiFi P2P mobile app. Then we extend it and introduce our PEERFLEET framework in Section 3.5, which is then evaluated in Section 3.6 before concluding in Section 3.7.

3.2 Background & Motivations

3.2.1 Overview of the Google Nearby Framework

Google Nearby is a broad framework developed by Google to support the development of pervasive, peer-to-peer, mobile apps by leveraging the diversity of communication interfaces made available by mobile devices. This framework supports not only the discovery of peers, but also communication capabilities and interoperability with Apple iOS.

Android Wi-Fi Peer-to-Peer. The legacy *Android Wi-Fi peer-to-peer* framework includes a support for both *peer* discovery and *service* discovery.[1] This core framework therefore provides the primitives to discover and pair two devices, by exposing IP addresses for both endpoints. However, beyond this base support, developers are required to develop their own application protocol (*e.g.*, by exposing a TCP server) to exchange data across multiple devices, which often leads to errors and makes the testing of these apps particularly hard.

Google Nearby Connections. To leverage this issue, Google developers recently released a Nearby framework,[2] which is integrated into any Android devices via Google Play Services. The *Google Nearby Connections* delivers an offline peer-to-peer communication layer without the need for a cloud provider or a WiFi router. The connection is achieved as follows:

[1]https://developer.android.com/guide/topics/connectivity/wifip2p
[2]https://developers.google.com/nearby

whenever a device discovers services advertised by other devices (printer, beacon, etc.), it automatically connects to one of them and—once connected—the two devices can exchange data through `Payload` objects, which can convey bytes, files or streams. While the *Android Wi-Fi peer-to-peer* framework handles creating a connection between nearby devices, *Google Nearby Connections* adds another layer for handling data transfer between nearby devices, which raises the communication abstractions and eases the test of the mobile apps using this framework.

Google Nearby Messages. The *Google Nearby Messages* extends the Nearby framework to support both Android and iOS devices. In this mode, while device discovery is supported by hardware interfaces (WiFi, Bluetooth, etc.), the communication is achieved through an Internet server for small payloads size. However, this API can only be used for small payloads size (from 3 KB to 100 KB) because it passes by a cloud server, which imposes a daily quota. Usually, it is used to pair different devices by exchanging ids, then the file exchange can be done without Internet.

iOS Multipeer Connectivity. The iOS *Multipeer Connectivity* framework provides a support to discover nearby services and connect with them.[3] The connection is completed in two phases, first by discovering nearby peers, and then by initiating a session between the peers, the session also handles sending payloads between peers.

3.2.2 Presence of Google Nearby in the Play Store

Table 3.1 reports on the ratings and issues raised by the users of peer-to-peer Android apps. These apps were selected from the Google Play store by querying the *WiFi Direct* keyword, which is one of the technology supported by Google Nearby. The *bad ratings* column refers to the number of times the app obtained the lowest rating—*i.e.*, 1 out of 5 stars. By processing user reviews, we have observed that the users are complaining about the app crashing, the app is not working and on WiFi Direct issues. Examples of complaints mentioned in reviews are:

- *Keeps asking for WiFi On/Off Permission every single time I unlock my device.*
- *Stops after startup if I have any internet connection.*
- *Turns wifi off and on even when not using it.*
- *Unstable & drop wifi speed when running app background!!!!*

Given the feedbacks reported by the app users, one can guess that the developed mobile apps could benefit from a more advanced testing support to anticipate and isolate the situations

[3]https://developer.apple.com/documentation/multipeerconnectivity

reported online in order to improve the quality of experience of their users and avoid bad ratings that severely impact their promotion.

Similarly, the Thali project[4] is a Cordova Plugin that brings GOOGLE NEARBY P2P communications to Cordova mobile apps. In particular, they clearly report that *"Getting peer to peer working on Android has been and continues to be a heck of a challenge."* They discuss the issues they had got using nearby P2P, especially with Wi-Fi Direct Service Discovery API.[5]

Finally, Table 3.2 reports on the 10 most downloaded apps that are using the GOOGLE NEARBY Connections API.[6] We have observed that these apps fail to update to the latest version of the API, although this version proposes some major improvements. The reason behind this may be the fact that migrating to the new version will take too much time, and they will have to walk through testing the app again to get a stable version without any support from the testing frameworks.

Table 3.2 Apps in the Google Play app store using the Google Nearby Connections API

App (package name)	downloads
WhatsApp Messenger (`com.whatsapp`)	1,000,000,000+
WeChat (`com.tencent.mm`)	100,000,000+
BBM - Free Calls & Messages (`com.bbm`)	100,000,000+
Hola Free VPN Proxy (`org.hola`)	50,000,000+
Email App for Mail.Ru (`ru.mail.mailapp`)	50,000,000+
AirDroid (`com.sand.airdroid`)	10,000,000+
XShare (`com.infinix.xshare`)	10,000,000+
iPair-Meet (`com.ipart.android`)	5,000,000+
DataBot IA (`com.testa.databot`)	1,000,000+
FITAPP (`com.fitapp`)	1,000,000+

Conclusion. Overall, one can observe that Google Nearby is adopted by the apps published in the Play store, including some major ones like WhatsApp Messenger, WeChat or Hola VPN. Nonetheless, most of these apps seem to suffer from bad reviews, some pointing the weaknesses of the integration of Google Nearby. Furthermore, we have also observed that the most famous apps integrating Google Nearby do migrate to the latest version of the API, thus potentially exposing their users to security leaks and quality of experience degradations.[7]

[4]http://thaliproject.org

[5]http://thaliproject.org/androidWirelessIssues

[6]Obtained from 42matters.com analytical website, then using *apkanalyser* to filter the APKs depending on the `ConnectionsClient` class from Google Nearby Connections, which unavailable in the latest versions of the framework.

[7]https://developers.google.com/nearby/connections/v11-update

We therefore think that the support for Google Nearby can be strengthened by extending existing development tools and methodologies to help Android developers better integrate and test peer-to-peer mobile apps, which remains a tedious task so far.

3.3 Testing Nearby P2P mobile apps

To test nearby P2P mobile apps, we need to take into consideration all the challenges that comes with the nearby technologies, this includes the management of the discovery and the connection phases. When we are trying to optimize our execution with the opportunistic connections our app can have, we first need to set our objectives to target them in an optimal way.

3.3.1 Objectives when Testing Nearby Mobile Apps

Unit testing is known to be a partial solution to assess the expected behavior of the app. In particular, when the execution context is known to be unpredictable, black-box testing becomes critical, especially for mobile devices [128]. Moreover, testing exhaustively a mobile app with every possible scenarios is desirable to reduce the risk of receiving bad user reviews and ratings on the app store. Beyond issues faced by standard mobile apps, the inclusion of the Google Nearby framework within a mobile app requires to address the following additional testing concerns:

Peer Scalability. Nearby mobile apps are meant to support large number of users. Thus, developers are expected to test that their apps can effectively discover and potentially interact with a large number of nearby users;

Connection Resilience. Given the pervasive nature of the underlying network layer, the developer has to consider and test that unexpected disconnections of peers does not crash or negatively impact the nearby app;

Protocol Robustness. As nearby apps are interacting and exchanging messages by implementing a dedicated application protocol, the developer has to assess that the contextual interactions of peers do not lead to deadlocks;

App Interoperability. Given the device and OS fragmentation of Android,[8] different configurations need to be supported by the peers. More specifically, the nearby app is expected to work correctly, no matter the underlying configuration of nearby peers. Moreover, as Google Nearby also targets iOS, this implies that the test scenarios should

[8]https://developer.android.com/about/dashboards/

be described in platform-neutral languages to maximize the interoperability of nearby apps;

App Reactivity. Nearby apps are expected to be opportunistic by triggering some actions upon the discovery of a peer. This includes message broadcasting, forwarding a message to a specific user or detecting the maximum number of users. This reactive behavior heavily depends on the configuration of the discovery protocol (*e.g.*, advertisement periodicity) to adjust the effectiveness of the triggered actions. Furthermore, these parameters might also have some critical implications for the end users (*e.g.*, battery consumption), thus forcing the developer to carefully tune them.

3.3.2 Challenges for Testing Nearby Apps

Given the elicitation of specific testing concerns and the limitation of state of practice when it comes to testing nearby mobile apps, we identified the following key challenges:

Emulators Support. Mobile apps that are using the Nearby framework can only be tested on physical devices and cannot be tested on emulators, or in the cloud (as Nearby requires some physical proximity between the tested devices). Even if the developer needs to test other functionalities (other than nearby features) using automated black-box testing on emulators, these tests will fail to execute without the hardware components associated to the Nearby framework;

Automation. The existing automation testing tools do not support testing nearby P2P mobile apps, so developers are expected to test their mobile apps manually, which is generally acknowledged as a poor approach for context-based mobile apps;

Scalability. Exploring the scalability of nearby mobile apps remains difficult. When interacting with a large number of nearby devices, the mobile app, including its user interface, has to adjust itself in order to visualize a large number of neighbors and the app needs to support a large number of communications without crashing or being forced to sleep by the mobile battery/resource manager;

Context Predictability. Nearby communications adopt an opportunistic communication scheme, which fits with the dynamics of the surrounding environment and the unpredictable nature of mobile user context. However, developers cannot anticipate and manually enumerate all the possible scenarios and test them to maximize the test coverage criteria. Furthermore, the continuous change in context information tends to explode the permutations of the available test space [128];

Cross-platform Tests. Both industrial and research communities do not provide any standardized way of testing nearby P2P mobile applications. Because the nearby framework has to provide cross-platform communications, the test framework needs to be cross-platform too;

Black-box Testing. The interactions between nearby devices cannot be only covered by unit tests. The dynamic context events cannot be expected in controlled experiences and the nearby communication hardware cannot be statically mocked. Beyond the need to support integration testing scenarios, nearby apps also need to be tested in black-box due to the dynamic context, the variability of devices, APIs and screen sizes.

These challenges therefore pave the way of an extended support for nearby app testing. In the next sections, we introduce ANDROFLEET and PEERFLEET two testing frameworks to support testing P2P mobile apps.

3.4 The ANDROFLEET Testing Framework

WiFi *peer-to-peer* (WiFi P2P, also known as WiFi Direct) enables mobile devices to discover, connect, and transfer data to nearby devices via WiFi; without an intermediate access point as long as they comply with the WiFi Alliance's WiFi Direct certification [164].

With the challenges identified in the last section 3.3.2, the existing automated testing methods are falling short on answering all the challenges. In this section, we introduce ANDROFLEET, an acceptance testing framework for WiFi P2P Android apps.

ANDROFLEET contributes the following features:

1. ANDROFLEET builds on a standard acceptance testing framework (*i.e.,* Cucumber[9]) to provide new primitives for describing contextual and distributed scenarios;

2. ANDROFLEET implements an emulated WiFi P2P communication stack to control the P2P interactions of mobile apps through emulated devices;

3. Finally, ANDROFLEET provides a large-scale deployment platform that can be used to assess the robustness of a WiFi P2P app prior to its deployment in the field, along an alpha testing phase. This feature collects insightful metrics on the execution and behavior of WiFi P2P apps.

ANDROFLEET is open-source and publicly available on this link: https://github.com/m3ftah/androfleet. The demo video of ANDROFLEET is made available from https://youtu.be/gJ5_Ed7XL04.

[9]https://cucumber.io

The goal of ANDROFLEET is to automate WiFi P2P *User Acceptance Testing*. ANDROFLEET enables developers to create and execute common WiFi P2P test scenarios. In particular, the framework provides two main features:

1. A DSL (*Domain-Specific Language*) to describe WiFi P2P testing scenarios, such as peer discovery and peer interactions;

2. A large-scale emulation platform that supports parallel testing on multiple devices as well as the WiFi P2P communications among them.

3.4.1 Describing Test Scenarios in ANDROFLEET

To describe the testing scenarios, ANDROFLEET provides a DSL to developers. Previous researches have extended the Calabash framework[10] to add some context-based testing [136, 137]. ANDROFLEET adopts a similar approach by extending Calabash to introduce a WiFi P2P specific vocabulary. We chose Calabash because it is multi-platform and delivers a rich and extensible syntax, based on Cucumber.

Figure 3.1 describes an example of a test scenario, created with ANDROFLEET, to simulate P2P unexpected contextual events. This scenario represents a WiFi Direct connection to another peer. The app starts the discovery operation. Then, it receives the list of available peers, from which, the user will choose one peer and the connection will be initiated. While we described the best case scenario, a much complex scenario can face the developers. Each of the possible scenarios has to be well tested, which requires the use of automated tests

In P2P systems, different unexpected contextual events can happen, such as a peer discovery, a peer out of range or a peer connection lost. However, some of these events can lead to failures of apps. ANDROFLEET lets developers to simulate these contextual events that can emerge along an execution. Listing 3.1 depicts the different ANDROFLEET testing directives that can be used to describe typical communication scenarios among peers in WiFi P2P apps. In particular, these directives provide a way to activate WiFi P2P, grant or revoke the WiFi P2P permission, discover other peers, configure the WiFi P2P settings and cause connection lost event.

Listing 3.1 ANDROFLEET testing directives to manage WiFi P2P scenarios

```
Switch the device WiFi P2P state
Switch the device WiFi P2P permission
Switch the device WiFi P2P state
Send the available peer list has changed action
```

[10]Calabash is a user acceptance testing framework: http://calaba.sh

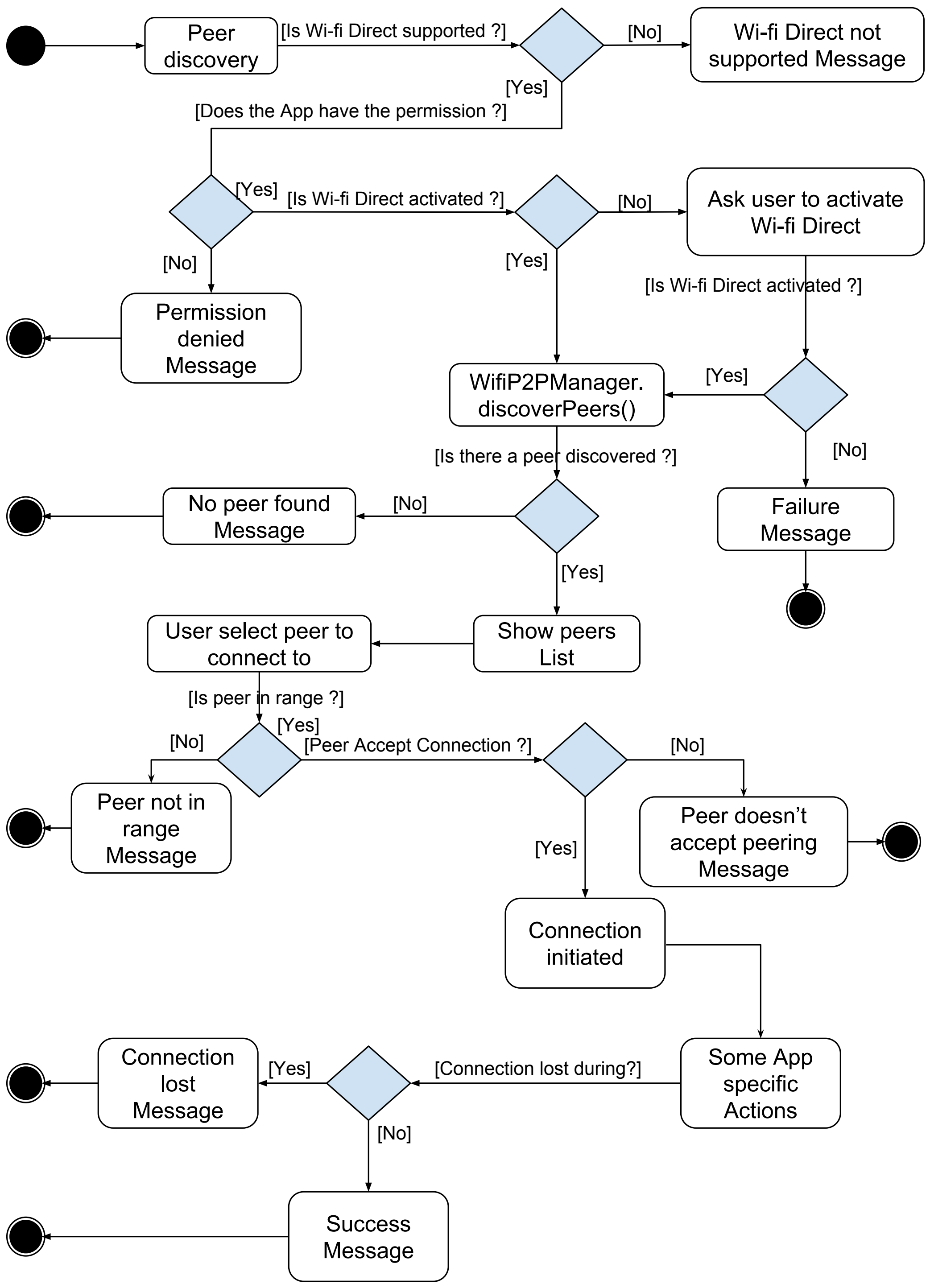

Peer discovery
[Is Wi-fi Direct supported ?]
[No]
Wi-fi Direct not supported Message
[Yes]
[Does the App have the permission ?]
[Yes]
[Is Wi-fi Direct activated ?]
[No]
Ask user to activate Wi-fi Direct
[No]
[Yes]
[Is Wi-fi Direct activated ?]
Permission denied Message
WifiP2PManager. discoverPeers()
[Yes]
[No]
[Is there a peer discovered ?]
No peer found Message
[No]
Failure Message
[Yes]
User select peer to connect to
Show peers List
[Is peer in range ?]
[Yes]
[No]
[Peer Accept Connection ?]
[No]
Peer not in range Message
Peer doesn't accept peering Message
[Yes]
Connection initiated
Connection lost Message
[Yes]
[Connection lost during?]
Some App specific Actions
[No]
Success Message

Fig. 3.1 Example of a communication scenario with WiFi-Direct

```
Change the device WiFi P2P name
Change the device WiFi P2P IP address
Change the device WiFi P2P role as group owner
Send the device WiFi P2P connection lost action
Get the device WiFi P2P state
```

3.4.2 Peer Discovery in AndroFleet

Figure 3.2 (cf. top-right rectangle) shows an example of a testing scenario created with the AndroFleet DSL. This scenario tests the sharing of a file between two nodes (from Node 0 to Node 1) in a WiFi P2P File Sharing App. First, the developer has to specify the running conditions of the scenario using the `peer_discovery.feature` file, these conditions include:

- Number of participant nodes (*i.e.*, emulators)
  - Example: *Given the list of devices: {A, B, C}*
- Discovery configuration
  - Example: *Given devices {A, B} are in range for 5s*
- Apps installed on each node
  - Example: *MyApp.apk is installed on all devices*

Later, for each single node, the developer specifies the behavior to test. Figure 3.2 (cf. bottom rectangles) shows two examples of test scenarios in two nodes for the File Sharing app. The apps asking for the list of nearby peers from the WiFi P2P hardware component will get this list from the *PeerDiscovery*. The list of available peers is configured using the `peer_discovery.feature` file.

3.4.3 Running Test Scenarios in AndroFleet

To run the specified test scenarios, the developer has to provide: (*i*) *test script scenarios* for each single node; (*ii*) *device profiles* (*e.g.*, OS version, hardware specification) to test the app; and (*iii*) `Peer_Discovery.features` as explained in Section 3.4.2.

The scenarios are executed in parallel in multiple devices using Calabash-Android. After the execution of the scenarios, the developer can gather the test results, and improve the app accordingly and test all the scenarios again using AndroFleet.

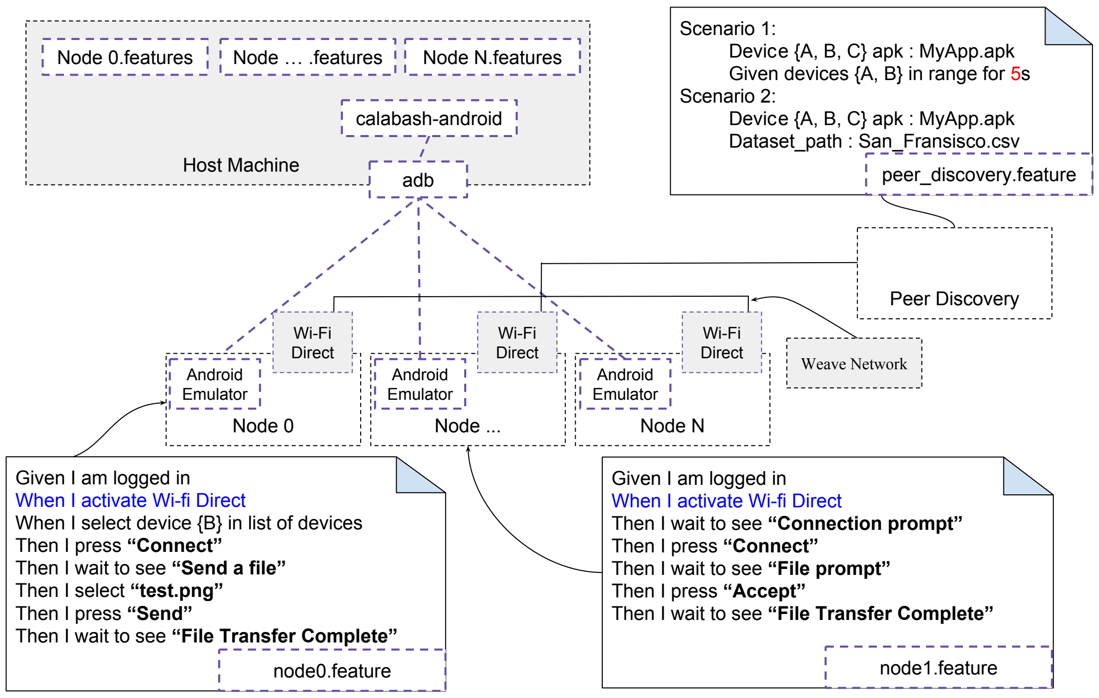

Node 0.features
Node … .features
Node N.features
calabash-android
Host Machine
adb
Scenario 1:
Device {A, B, C} apk : MyApp.apk
Given devices {A, B} in range for 5s
Scenario 2:
Device {A, B, C} apk : MyApp.apk
Dataset_path : San_Fransisco.csv
peer_discovery.feature
Peer Discovery
Weave Network
Wi-Fi Direct
Android Emulator
Node 0
Node ...
Node N
Given I am logged in
When I activate Wi-fi Direct
When I select device {B} in list of devices
Then I press "Connect"
Then I wait to see "Send a file"
Then I select "test.png"
Then I press "Send"
Then I wait to see "File Transfer Complete"
node0.feature
Given I am logged in
When I activate Wi-fi Direct
Then I wait to see "Connection prompt"
Then I press "Connect"
Then I wait to see "File prompt"
Then I press "Accept"
Then I wait to see "File Transfer Complete"
node1.feature

Fig. 3.2 Overview of the ANDROFLEET framework

3.4.4 ANDROFLEET Implementation Details

The ANDROFLEET testing framework implements an extension of the Calabash testing framework to create test scenarios and execute them.

ANDROFLEET is released as a Gradle plugin, which developers can add to run the tests inside Android Studio IDE. The ANDROFLEET gradle plugin executes the Calabash tests in parallel (in multiple devices) and collects the test results from all devices. ANDROFLEET contains three parts: Device Nodes, a Peer Discovery Node, and a Host Machine. We describe the implementation of these three parts, respectively.

Device Nodes. ANDROFLEET uses Docker containers to deploy Android emulator nodes. We use the Docker technology to provide a lightweight and scalable testing environment. The different nodes communicate among them via Weave network.[11]

Peer Discovery Node. The Peer Discovery node is responsible for emulating WiFi Direct behavior in the Android emulator nodes and notify them about nearby devices. The communications between Device Emulator Nodes and the Peer Discovery node is achieved using the Akka framework, which supports large-scale deployment. In addition, the Peer Discovery

[11]https://www.weave.works/oss/net

node can use a GPS mobility dataset to compute the distances between devices and inform the devices when they are in range.

Host Machine. Finally, developers use a Host Machine to define and run testing scenarios. The Android emulators connect to the developer's machine via ADB (*Android Debug Bridge*).[12]

Peer-to-peer mobile apps are complex in nature, since they are based on a distributed environment. Thus, many aspects of the system need to be verified to ensure the correct behavior of the system under multiple circumstances (peer fails, peer refuses connection, peer disappears, etc.).

ANDROFLEET implements an emulated version of the Android WiFi P2P API that can be deployed within an emulator (*e.g.*, GenyMotion) and controlled by the testing environment (*e.g.*, Calabash) to simulate the discovery of a nearby device and assess the user acceptance testing scenarios.

3.4.5 Case Study

#	**Scenario**	**Expected behavior**	**Pass?**
1	A simple image transferring scenario	Image is transferred	Yes
2	Transferring the image from the group Owner	Image is transferred	No
3	Deactivating WiFi P2P before transfer	Ask user to switch on WiFi P2P	Yes
4	Selecting a file instead of image	Open as dialog appears	Yes
5	Sending a second file in the same connection	The file is sent	No
6	Disconnect the second peer before starting the file transfer	Ask user to switch on WiFi P2P	No
7	Connecting to peer without launching discover request on it	The file is sent	Yes

Table 3.3 Testing scenarios for the WiFiDirect app

In this section, we present a case study to illustrate how ANDROFLEET assists developers to automate the testing of WiFi P2P Android apps. As *Application Under Test* (AUT), we use *WiFiDirect*[13], an open-source app to transfer files between Android devices using WiFi Direct connection. The AUT has more than $10,000$ installs on the Google Play Store and its source code is freely available online.[14]

First of all, the developer needs to set up ANDROFLEET in the development IDE (*i.e.,* Android Studio). To do this, she uses the provided *Gradle Plugin* which automatically mocks the Android WiFi P2P API used in the app.

Table 3.3 presents 7 relevant testing scenarios for the AUT. The developer describes the scenarios using the ANDROFLEET framework. Figure 3.3 shows the description of scenario

[12]https://developer.android.com/studio/command-line/adb.html
[13]https://play.google.com/store/apps/details?id=anuj.wifidirect
[14]https://github.com/anuj7sharma/WiFiDIrectDemo

```
Feature: File sharing feature
  Scenario: Simple image sharing - sending
  Given I activate WiFi Direct
  Given I wait up to 60 seconds for "
   Available" to appear
  Given I do not see the text "Disconnect"

  When I take an image
  Then I press the menu key
  Then I press the text view "Discover"
  Then I should see "finding peers"
  Then I wait up to 60 seconds for "
   N192168492" to appear
  Then I wait up to 60 seconds for "
   CHOOSE FILE" to appear
  Then I press the "CHOOSE FILE" button
  Then I take a screenshot
  Then I tap in 150 200
  Then I tap in 150 200
```

(a) Scenario send image file (Node 1)

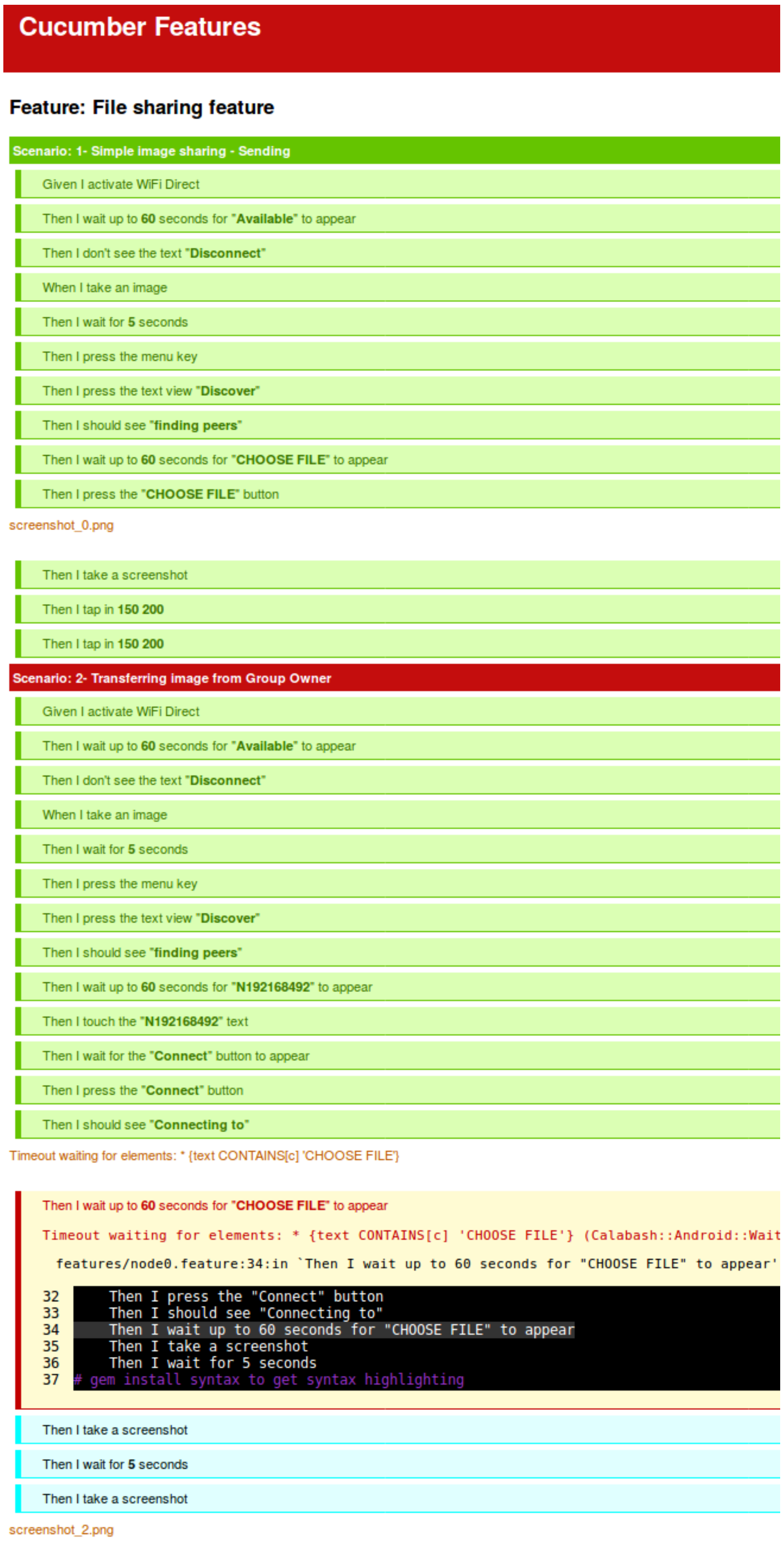
```
Feature: File sharing feature
  Scenario: Simple image sharing -
   receiving
  Given I activate WiFi Direct
  Given I wait up to 60 seconds for "
   Available" to appear
  Given I do not see the text "Disconnect"

  When I take a screenshot
  Then I press the menu key
  Then I press the text view "Discover"

  When I should see "finding peers"
  Then I wait up to 60 seconds for "
   N192168491" to appear
  Then I touch the "N192168491" text
  Then I wait for the "Connect" button to
   appear
  Then I press the "Connect" button
  Then I should see "Connecting to"
  Then I wait for 15 seconds
  Then I take a screenshot
```

(b) Scenario receive image file (Node 2)

Fig. 3.3 Description of sharing file scenario in the AUT

#1, a simple image transferring from one device to another. In particular, Listing 3.3a presents the sending scenario in one device, while 3.3b displays the receiving scenario in a different device.

Next, the developer runs the scenarios in the ANDROFLEET emulators. After execution, ANDROFLEET presents the testing results—*i.e.,* passing and failing tests. Figure 3.4 shows a screenshot of the testing results of the described case study. The screen displays the set of tests executed. Green and read colors mean passing and failing scenarios respectively. In the example, *Scenario 1* passed, while *Scenario 2* failed. Each scenario shows the sequential actions that were executed. If an action fails, ANDROFLEET shows error information and the reason of the failure. For example, in Scenario 2, the action 13 (*"Then I should see 'Connecting to' "*) fails. The remainder actions of the test are printed in blue, meaning that they were not executed as the test execution failed before. After analyzing the testing results, the developer can fix the app and re-run the tests until all scenarios successfully pass.



Cucumber Features

Feature: File sharing feature

Scenario: 1- Simple image sharing - Sending

Given I activate WiFi Direct

Then I wait up to **60** seconds for "**Available**" to appear

Then I don't see the text "**Disconnect**"

When I take an image

Then I wait for **5** seconds

Then I press the menu key

Then I press the text view "**Discover**"

Then I should see "**finding peers**"

Then I wait up to **60** seconds for "**CHOOSE FILE**" to appear

Then I press the "**CHOOSE FILE**" button

screenshot_0.png

Then I take a screenshot

Then I tap in **150 200**

Then I tap in **150 200**

Scenario: 2- Transferring image from Group Owner

Given I activate WiFi Direct

Then I wait up to **60** seconds for "**Available**" to appear

Then I don't see the text "**Disconnect**"

When I take an image

Then I wait for **5** seconds

Then I press the menu key

Then I press the text view "**Discover**"

Then I should see "**finding peers**"

Then I wait up to **60** seconds for "**N192168492**" to appear

Then I touch the "**N192168492**" text

Then I wait for the "**Connect**" button to appear

Then I press the "**Connect**" button

Then I should see "**Connecting to**"

Timeout waiting for elements: * {text CONTAINS[c] 'CHOOSE FILE'}

Then I wait up to **60** seconds for "**CHOOSE FILE**" to appear

```
Timeout waiting for elements: * {text CONTAINS[c] 'CHOOSE FILE'} (Calabash::Android::Wait
  features/node0.feature:34:in `Then I wait up to 60 seconds for "CHOOSE FILE" to appear'
```

```
32      Then I press the "Connect" button
33      Then I should see "Connecting to"
34      Then I wait up to 60 seconds for "CHOOSE FILE" to appear
35      Then I take a screenshot
36      Then I wait for 5 seconds
37  # gem install syntax to get syntax highlighting
```

Then I take a screenshot

Then I wait for **5** seconds

Then I take a screenshot

screenshot_2.png

Fig. 3.4 Testing results screenshot

3.4.6 Discussion

Although the general idea of emulating hardware capabilities (*e.g.*, GPS, sensors) for testing is not new, the novelty of ANDROFLEET resides in providing an emulated WiFi Direct environment for testing. Given the increasing popularity of WiFi P2P mobile apps, this feature is crucial because none of current testing frameworks support WiFi P2P.

ANDROFLEET reports the following advantages to the mobile development and testing communities:

1. Extend the capability of an Android testing framework to enable the control of peer-to-peer interactions among mobile apps via WiFi Direct,
2. Provide enriched vocabulary for testing mobile apps using WiFi Direct,
3. Automate parallel testing on multiple emulated devices, as well as WiFi Direct communications among them,
4. Reproduce behaviors on failing test scenarios to help developers to fix the apps.

ANDROFLEET is practical for developers, who lack access to a farm of real devices to test. But even when a crowd of real devices is available, current testing frameworks lack primitives to assess distributed behavior.

3.5 The PEERFLEET Testing Framework

In this section we propose PEERFLEET testing framework which comes as an extension to the ANDROFLEET testing framework. Table 3.4 presents the differences between ANDROFLEET and PEERFLEET frameworks. In particular, PEERFLEET testing framework can test all nearby peer-to-peer nearby mobile apps. While WiFi peer-to-peer communications present some challenges, as the provided OS's API will only setup a connection between peers and leave the data transfer protocol for the developers to implement. In PEERFLEET, we opt for more generic APIs that also handle the data transfer phase which makes testing the data transfer protocol more simple, as all the developers use the same data transfer protocol provided by the API. These generic APIs can use any nearby communication method (BLE, Bluetooth or WiFi), as the communications between peers share the same Publish/Subscribe protocol.

The PEERFLEET testing frameworks do not only leverage the challenges we identified in Section 3.3.2, but also users' feedback from the Google Play store (cf. Table 3.1) and the GitHub issues of the Google Nearby framework [165] to propose appropriate abstractions and primitives for testing nearby apps at large.

Table 3.4 Comparing ANDROFLEET and PEERFLEET

Features	**ANDROFLEET**	**PEERFLEET**
WiFi P2P communications	Yes	Yes
Nearby P2P communications	No	Yes
Test the data transfer phase	No	Yes
Used dataset	GPS mobility traces	Proximity encounters traces
Control the P2P connection's state	Yes	Yes
Control discovery parameters	No	Yes

3.5.1 Framework Overview

The PEERFLEET testing framework intends to support the orchestration of a fleet of mobile devices, which can be physical devices connected to the development environment or virtual ones (*e.g.*, emulators), as depicted in Figure 3.5. PEERFLEET instruments these environments with a nearby adapter that connects a fleet of remote devices through a socket.io event bus. PEERFLEET exposes an API that is used to send specific commands to remote devices (cf. Listing 3.2). This API is used by the PEERFLEET orchestrator according to actions that are triggered by the test scenarios or evolutions in the devices vicinity. These evolutions are driven by the configured proximity dataset and the discovery protocol settings. The PEERFLEET API accommodates a wide diversity of black-box testing frameworks, such as monkeyrunner, Calaba.sh[15] or JBehave.[16] In particular, we provide a support for the Gherkin specification language to leverage behavior-driven specifications.[17]

Listing 3.2 PEERFLEET API

```
public interface NearbyActions {
  void peerJoin(String peerId);
  void peerLeft(String peerId);
  void peerRequestConnection(String peerId);
  void peerAcceptConnection(String peerId);
  void peerRejectConnection(String peerId);
  void peerConnect(String peerId);
  void peerSendBytes(String peerId, byte[] bytes);
  void peerSendFile(String peerId, byte[] file);
  void peerSendStream(String peerId, byte[] stream);
  void peerDisconnect(String peerId);
  void nearbyPeers(String[] peers);
```

[15]https://calaba.sh
[16]https://jbehave.org
[17]https://docs.cucumber.io/gherkin

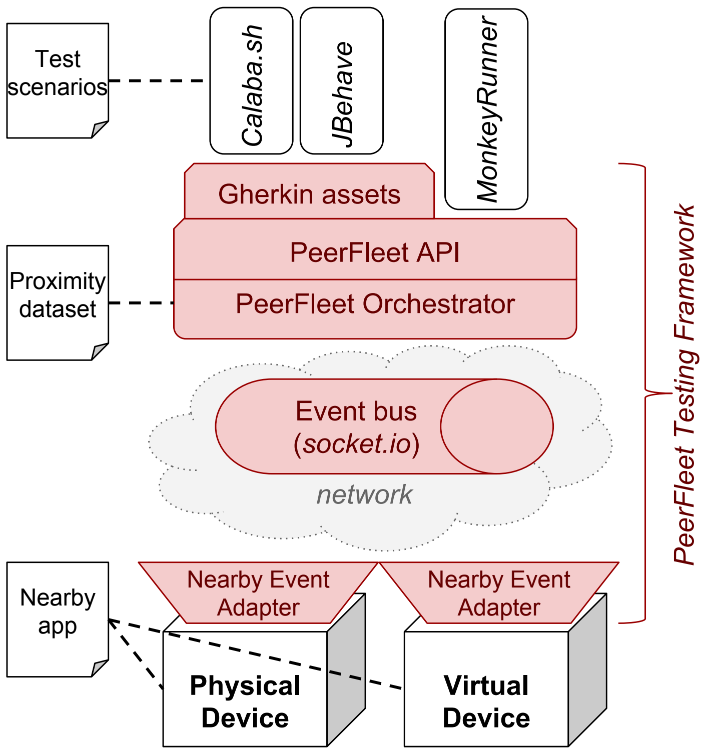


Fig. 3.5 Overview of PEERFLEET.

```
}
```

3.5.2 The PEERFLEET Orchestrator

The PEERFLEET orchestrator is composed of 2 parts: the PEERFLEET Test Runner and the PEERFLEET Bridge Component.

The PEERFLEET Test Runner is in charge of interpreting a testing scenario and interacts with the set of remote devices to check that the mobile apps behave accordingly to a given specification. For example, Listing 3.3 provides the snippet of the specification, based on Gherkin, of a nearby app that support device-to-device file transfer. This acceptance test describes 3 scenarios that can be triggered by the test runner according to the context. The PEERFLEET Bridge Component will follow the proximity dataset which, for each trace in the dataset, it yields the *IDs* of the peers that are in proximity. The test runner then receives these *IDs* and picks one of the scenarios to execute. The test runner completes when the bridge component notifies the completion of the proximity dataset.

The PEERFLEET Bridge Component is also in charge of managing the interactions among the remote devices by controlling their Nearby framework through socket.io messages.

Table 3.5 summarizes the mapping of the PEERFLEET framework API to socket.io events, which are produced by the bridge component and consumed by the nearby adapters of remote devices, depending on the call that are triggered from the PEERFLEET API.

Listing 3.3 Gherkin specification for a file transfer app based on Nearby

```
Feature: File exchange

  Rule: File transfer should never crash

    Background:
      Given there are <d> devices
      And discovery advertises every <x> seconds

    Scenario: File transfer should succeed
      Given there are more than one devices alive
      When 2 devices meet, they will connect
      And one device send a file
      Then the notification should be "file transferred" on the screen

    Scenario: File transfer fails
      Given there are more than one devices alive
      When 2 devices meet, they will connect
      And one device send a file
      And one device disconnect
      Then the notification should be "peer disconnected" on the screen

    Scenario: File transfer aborts
      Given there are more than one devices alive
      When 2 devices meet, they will connect
      And one device send a file
      And one device abort
      Then the notification should be "transfer aborted" on the screen
```

Figure 3.6 summarizes the involved parties and interactions among the PEERFLEET components when triggering the scenario `File transfer fails` introduced in Listing 3.3. The execution of this scenario can be controlled, step-by-step, from the testing environment (*e.g.*, developer laptop, continuous integration). The test runner initiates the bridge component by configuring the proximity dataset to be used, as well as the discovery settings for the nearby framework. This separation of concerns allows the developers to test the same scenario in different conditions and scales (number, density and diversity of devices). It also

Table 3.5 Mapping PEERFLEET API to Socket.io events

PEERFLEET API	Socket.io events
`peerJoin(id)`	`emit("peer_join",id)`
`peerLeft(id)`	`emit("peer_left")`
`peerRequestConnection(id)`	`emit("request",id)`
`peerAcceptConnection(id)`	`emit("accept",id)`
`peerRejectConnection(id)`	`emit("reject",id)`
`peerConnect(id)`	`emit("connect", id,p)`
`peerSendBytes(id,p)`	`emit("bytes",id,p)`
`peerSendFile(id,f)`	`emit("file",id,f)`
`peerSendStream(id,s)`	`emit("stream",id,s)`
`peerDisconnect(id)`	`emit("disconnect",id)`
`nearbyPeers(ids)`	`emit("nearby",ids)`

supports the investigation of the effects of the discovery protocol settings on the performance of the mobile app under test.

Once the bridge component initialization process is completed, the test runner waits for the proximity notifications of the devices. The bridge component autonomously process the proximity dataset and, for each input step, it computes the potential proximity among devices, according to the discovery protocol settings. Upon detecting such a proximity, the bridge component fires a nearby event that triggers a scenario from the test runner. The execution of this scenario results in interactions with the instances of the mobile apps deployed in the virtual or physical devices. Each interaction can eventually result in calls to the Google Nearby framework, which are converted into socket.io events consumed by the bridge component and forwarded to the appropriate peers according to the current context state. Each peer can eventually react accordingly to the consumed events by producing an event that is further propagated by the bridge component.

3.6 Evaluation

To evaluate the PEERFLEET testing framework, we focus on the following research questions:

RQ1: Does the PEERFLEET testing framework leverage the identification of bugs related to the usage of the Google Nearby framework?

RQ2: Can the PEERFLEET testing framework support the developer in tuning the discovery protocol settings optimally?

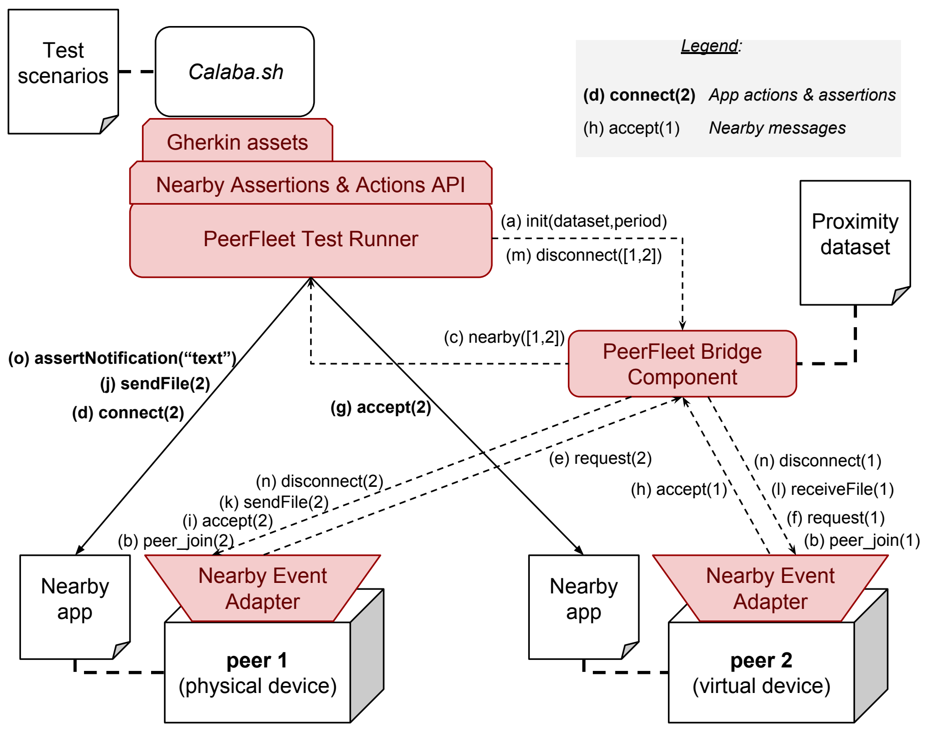


Fig. 3.6 Interactions among the PEERFLEET components.

3.6.1 Proximity Datasets

As detailed in Section 3.5.1, to make our emulation as close as possible to real life scenarios we use real life collected proximity datasets. In our experiment, we decided to use The Cambridge Haggle One datasets (v. 2016-08-28) [166] thanks to their variety of use cases. Each dataset corresponds to a different use case, for example, the cases where the peers are moving or just stationary, indoor or outdoor peers, extensive or casual encounters and short-range or long-range peers. These varieties of use cases give us the possibility to choose the best use case that corresponds to our testing objective. The characteristics of each dataset are summarized in Table 3.6.

Table 3.6 Characteristics of Haggle One proximity datasets

dataset	emulators (#)	duration (days)	interactions (#)	type
Intel	9	4.15	2,728	*mobile and stationary nodes*
Computer-lab	12	5.27	8,456	*mobile nodes*
Infocom2005	41	2.94	44,918	*mobile nodes*
Cambridge-city-complete	52	11.42	21,746	*mobile and stationary nodes*
Infocom2006-short-range	78	3.87	257,958	*short-range nodes*
Infocom2006-complete	98	3.90	341,202	*short-range and long-range nodes*

Each dataset contains Bluetooth proximity traces for different number of days, each trace contains the following fields:

1. The *timestamp* field is the time of the encounter, it starts from 0.
2. The *first-peer* field is the *ID* of the first peer.
3. The *second-peer* field is the *ID* of the second peer.
4. The *event* field is either 1 when two peers connect with each other or 0 when they disconnect.

3.6.2 Bug Identification

Experimental Setup

To address **RQ1**, we have selected 8 Android open source apps (cf. Table 3.7) to be tested with our PEERFLEET testing framework, we try to detect nearby bugs in these apps and, when no bug were detected, we artificially inject them to evaluate the precision of the bug detection. These mobile apps are run in the emulators, while the test runner executes a monkey that randomly chooses an action from the app UI and triggers the widget corresponding to it.

Classification of Nearby Bugs: In this chapter. We consider the following classification of nearby bugs:

1. *Permission bugs* occur when the runtime permissions are revoked using the nearby framework syntax, which requires the app to request for these permissions;
2. *Scalability bugs* occur when the nearby app fails to discover and communicate with a large number of neighboring peers for several days, resulting in app crashes or if the emulator is out of memory;
3. *Protocol bugs* occur when the nearby app fails to establish the connection with remote peers in some contexts, because the nearby protocol is incorrectly implemented by the nearby app. Given that the Google Nearby framework specifies 17 error codes,[18] the mobile app under test may provide a partial implementation resulting in crashes or unexpected behaviors;
4. *Pervasive bugs* may occur when devices succeed in connecting and, given the pervasive nature of their interactions, loose the connection, but the app is still sending data.

[18]https://developers.google.com/android/reference/com/google/android/gms/nearby/connection/ConnectionsStatusCodes

Table 3.7 Open Source Android nearby apps coverage

Apps — Bugs	Permission	Scalability	Protocol	Pervasive
`com.p2psample`	Injected	Found	Found	Injected
`com.example.nearbyplayground`	Injected	Injected	Found	Injected
`de.bigabig.hotmessages`	Injected	Found	Found	Injected
`com.example.hotelca.poolnumbergenerator`	Injected	Injected	Found	Found
`com.jhony.jester.play`	Found	Injected	Found	Found
`com.google.location.nearby.apps.walkietalkie`	Injected	Injected	Found	Found
`com.google.location.nearby.apps.rockpaperscissors`	Injected	Injected	Found	Found
`com.supercilex.autohotspot`	Injected	Injected	Found	Found

Results

The results for the nearby apps under test are reported in Table 3.7. The apps were tested against the classes of nearby bugs identified in Section 3.6.2. Table 3.7 reports if, for each app under test, PEERFLEET succeeds in identifying at least an occurrence of nearby bugs.

Discussion. One can observe that each vanilla app under test exhibited at least one class of nearby bug that could be detected by PEERFLEET testing framework. Interestingly, PEERFLEET succeeds in detecting *protocol bugs* in all the mobile apps we selected, which highlight the partial implementation of the nearby support, thus failing to work in a wide diversity of execution conditions. Typically, after manual investigation, we could observe that most of the mobile apps manage only one of the 17 error codes (`STATUS_OK`), thus leading to app crashes when it faces unexpected situations. Regarding *permission bugs*, most of the mobile apps seem to correctly support the runtime permissions. We believe that the low rate of detected bugs, is related to the associated Android linter rules, which warn the developers about this error. Yet, one can observe that PEERFLEET succeeds in detecting the artificial bugs we injected. Exploring *scalability bugs* requires to stress the mobile app by considering more complex proximity datasets (Table 3.6) that exhibit a large density of devices and potentially long periods of executions. In such situations, PEERFLEET highlights symptoms such as the app UI fails to display multiple peers, the app fails to accept connections from more than one peer, and the app fails to properly disconnect from peers—*i.e.*, resources are not properly released and the app cannot reconnect to the same peer. Finally, regarding the occurrences of *pervasive bugs*, we observe that the developers of the incriminated mobile apps missed to properly handle connection lost when implementing the listener `EndpointDiscoveryCallback.onEndpointLost()`, which is triggered whenever a peer disappears from the device's vicinity.

3.6.3 Parameter Tuning

Experimental Setup

To address **RQ2**, we consider the following objectives:

- *To explore the maximum number of users.* This applies, for example, when considering a Bluetooth beacon in a store, which interacts with users passing nearby. As part of our evaluation, we used a modified version of the open source app HotMessages;[19]
- *To reach the maximum number of users.* For example, we consider the case of an emergency Android app that dispatches a message to the maximum number of users through Nearby framework. As part of our evaluation, we used a modified version of the open source app P2PMessaging.[20]

We believe that these two objectives cannot be only assessed by a simple guess, one developer rather needs to experiment with their mobile app on a representative dataset of users to estimate how much the objectives are reached. Then, developers can investigate which discovery protocol settings are the most suitable for their scenarios and objectives. Along with these objectives, the impact on the battery consumption can also be considered as an optimization objective.

Experiment Settings. To optimize the scalability or the performance of the objectives of her app, the developer can explore several discovery protocol settings:

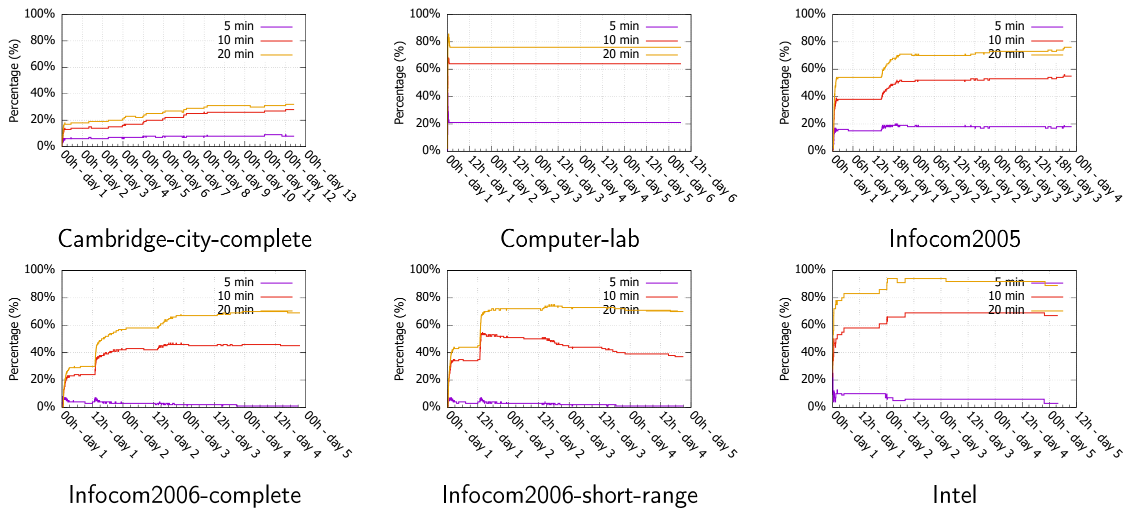
1. *Discovery period*, which is the period between two consecutive discovering operations, each discovering operation takes 2 minutes. In our experiment, we have considered 3 discovery periods: 5 minutes, 10 minutes and 20 minutes;
2. *Connection strategy*, which is the topology used to connect the devices: *one-to-many* (WiFi) or *many-to-many* (Bluetooth).

We adopt the following experimental protocol:

1. We launch a crowd of Android mobile emulators (composed of 9, 12, 41, 52, 78 and 98 devices), depending on the size of the associated proximity dataset [166]—the characteristics of each dataset are reported in Table 3.6;
2. The PEERFLEET test runner executes an acceptance test that randomly chooses an action from the app UI and trigger the widget corresponding to it;
3. These devices will interact with each other following the *Bluetooth* or *WiFi* encounters in both *indoor* or *outdoor* situations depending on the proximity dataset;
4. For each run, the new variant explores a different combination of settings for discovery period and connection strategy;

[19]https://github.com/bigabig/hotmessages
[20]https://github.com/MrPKGupta/P2PMessaging

Table 3.8 Experiment statistics - Evolution of missed peers for the HotMessages app



5. At runtime, we collect statistics for each objective from the bridge component.

Battery Consumption

In order to show the effect of each discovery period on the battery consumption, we deploy mobile apps with PEERFLEET on two physical mobile devices: Moto Z (Android 7.1.1) and Galaxy Tab A (2016, Android 5.1.1). We have run the HotMessages app on the two mobile devices, with different discovery settings per run. For each variant, we keep the device screen unlocked to avoid the app being sent to the background and to disable the battery manager strategies, which can put the app on idle state and interfere with the measurements. The measurements were collected using the `adb shell dumpsys batterystats` command that provides statistical data about battery usage on the device.

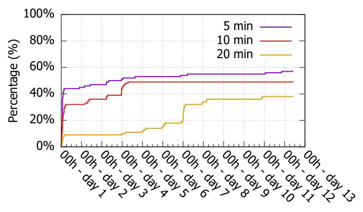
Results

Applications' performances may depend on a large diversity of parameters, ranging from network interface efficiency, to hardware capabilities, to code optimizations. In our study, we are interested in code-level optimizations that positively impact the app performances.

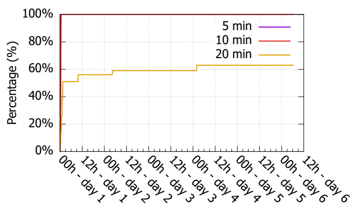
In particular, Tables 3.8 and 3.9 reports on the collected statistics of the experiment we described in Section 3.6.3.

Table 3.8 plots the percentage of missed peers (nearby dataset peers failed to be detected by the mobile app) depending on the discovery period. As expected, the shorter the period is, the less number of peers are missed. However, PEERFLEET succeeds in delivering some concrete insights on the device coverage in the emulated conditions depending on the settings

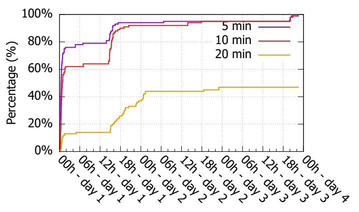
Table 3.9 Experiment statistics - Evolution of reached peers for the P2PMessaging app



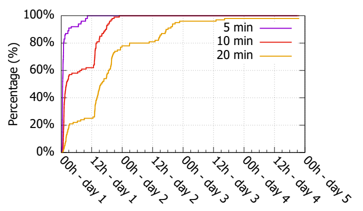
Cambridge-city-complete



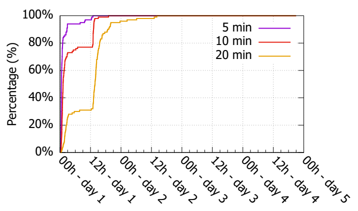
Computer-lab



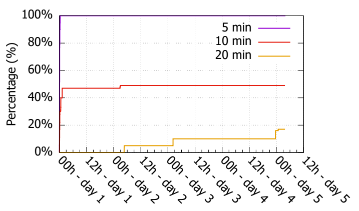
Infocom2005



Infocom2006-complete



Infocom2006-short-range



Intel

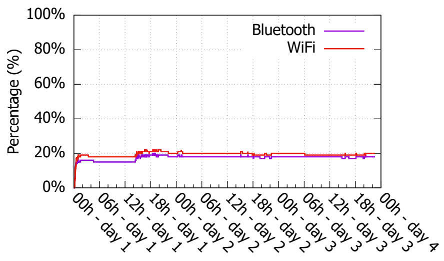
values. Thus, the developer can take informed decisions on the most appropriate values thanks to PEERFLEET.

Similarly, Table 3.9 reports on the percentage of reached peers. Again, the 5*min* discovery period setting maximizes the number of reached nodes. More surprisingly, one can observe that the 10*min* discovery period results performs close to the 5*min* settings (except for the *intel* datset) meaning that the 5*min* discovery period is not really needed especially if the app is intended to run for several days (the two lines converge). We can also note that the number of reached nodes depends also on the number of the dataset nodes, so if the developer expects to have more than 50 collocated peers, then choosing the 20*min* discovery period will get the same results as the 5*min* discovery period. If the developer estimates that the number of users is less than 10 (like in the *Intel* dataset), then choosing 20*min* will lead to poor performances, thus recommending the developer to set 5*min* as discovery period.

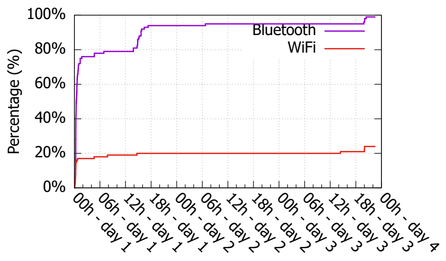
Table 3.10 reports on the percentage comparison between missed and reached peers using Bluetooth and WiFi connections. We can see that there is no large difference in the percentage of missed peers, but there is a big difference in the percentage of reached ones. This is due to the simultaneous connections that the Bluetooth connection offers over the WiFi connection. While apps using WiFi connection (*one-to-many*) cannot initiate another connection during the current session, the apps using Bluetooth connections (*many-to-many*) can pair with several devices without the need to disconnect any one of them. Thus, adopting the Bluetooth connection strategy can better support the efficient dissemination of messages compared to the WiFi strategy.

Finally, Table 3.11 reports on the power measurements obtained from physical devices connected to PEERFLEET. The *witness* app setting disables the discovery protocol. We can

Table 3.10 Experiment statistics - Impact of the connection strategy (using Infocom2005)



Missed peers



Reached peers

Table 3.11 Impact of the discovery protocol settings on battery lifespan

Settings	Moto Z	Galaxy tab
witness	1d 04h 46mn 26s	1d 03h 38mn 40s
20 `min`	17h 04mn 44s	1d 01h 25mn 55s
10 `min`	16h 44mn 42s	1d 01h 02mn 49s
5 `min`	13h 36mn 26s	23h 45mn 38s

see that different discovery periods have a direct impact on the battery consumption that can be noticed by the user. For example, a Moto Z user who installs the app configured with `5 min` discovery period, may notice that the app is responsible for consuming 52% of the battery power, thus encouraging her to uninstall it.

Discussion. Depending on the objectives and the number of the targeted users, the developers can tune their app settings. For example, if the main objective is to disseminate a message to the maximum number of peers, the setting 20 `min` discovery period will give good results (up to 98 %).

When deciding between using Bluetooth or WiFi. If the developer's objective is to exchange messages between direct users, given that WiFi and Bluetooth have almost the same missed peers percentage, the decision can depend on other factors, like the active connection method or the other peer's chosen connection method. If the developer objective is to disseminate the message to maximum number of users than Bluetooth is the best fit.

By running these experiments, developers and product managers can gather some insights on how to choose the best parameters and connection strategy, depending on the user context. Thanks to Google Nearby, the developer can also consider the implementation of an hybrid strategy, thus combining Bluetooth and WiFi, to detect neighbors using Bluetooth then exchange data using WiFi to benefit from the high bandwidth transfer of the WiFi connection.

The PEERFLEET testing framework therefore supports the exploration of app performance along different dimensions, including the proximity dataset characteristic but also the device fragmentation and discovery protocol settings. This exploration of the app performance space can guide the developer in tuning these parameters optimally according to her requirements.

3.6.4 Threats to Validity

While the goal of our study is to increase the quality of mobile apps by detecting the nearby bugs through the application of black-box testing at large, a threat lies in the possibility that we missed some classes of nearby bugs.

Another threat lies in the implementation of our PEERFLEET testing framework. In particular, the nearby event adapter provides an emulated support of the hardware discovery mechanism that is implemented by the operating system. Although we extensively tested our implementation of this nearby event adapter, we cannot be sure that it faithfully complies with all the implementation of the Google Nearby framework, and their potential bugs.

Finally, a possible threat lies in our experimental framework. We did extensive testing of our experimentation, and we manually verified the data from our experiments. However, as for any experimental infrastructure, there may be bugs. We hope that they only change marginal quantitative results and not the quality of our contribution.

3.7 Conclusion

Developers are facing problems developing P2P Nearby apps, there are no guidelines for testing and developing such apps. In this chapter, we have proposed two testing frameworks, ANDROFLEET and PEERFLEET, both frameworks will help developers debug and test their P2P Nearby apps. We show that PEERFLEET framework can also help the developers find the best set of parameters that are the most suitable to their objectives.

Further research routes would be extending this testing approach and generalizing it for other P2P communications, and adding custom support for wireless printers, wireless headphones and car stereo. There is also a need to detect further scaling issues and document them as guidelines for developers.

In the next chapter, we present our privacy preserving solution that uses peep-to-peer nearby communications to disseminate data between users in order to hide them from the server. We show how we can leverage the ANDROFLEET framework to evaluate the privacy preserving properties and tune the parameters of used LPPMs.

Chapter 4

FOUGERE: User-Centric Location Privacy in Mobile Crowdsourcing Apps

This chapter presents the thesis contribution on the location privacy preserving solution, called FOUGERE. It starts by motivating the need for protecting the worker's location privacy, as existing crowdsourcing apps continue doing extensive location data collection without using appropriate privacy preserving solutions or involving the user in the privacy control. Then, it presents the adopted adversary model, in which, the workers do not have to trust the collection server or any third-party server. FOUGERE is introduced as a software library to empower workers with LPPMs while using P2P collaboration between users to hide them from the server. Next, FOUGERE dissemination process is described in 3 phases (SEND, ONDISCOVER and ONRECEIVE) along with the integration of the LPPMs. Then, this chapter presents the privacy settings given to the user to control her privacy, the LPPMs are simplified to give the user a simple interface to configure them (privacy filters, distortions and aggregations). After that, FOUGERE implementation details are presented, including the integration of FOUGERE library in existing mobile apps, which requires setting both the dissemination and the required LPPMs. An evaluation of FOUGERE is conducted. The evaluation protocol involves using ANDROFLEET to emulate a crowdsourcing campaign, where workers collaborate using WiFi P2P communication to hide from the server, the LPM^2 toolkit is used to attack the data of 15 workers that are producing location data for several days. As FOUGERE intention is to put the user in center of her privacy control, in the evaluation, 4 privacy profiles were set to represent different user choices, ranging from weak to strong privacy profiles. Then, an analysis of performance, utility and privacy, had led to conclude that FOUGERE can increase workers privacy with little impact on the quality of the crowdsourced dataset.

4.1 Introduction

Mobile crowdsourcing platforms and applications (or apps) are being widely used to collect datasets in the field for both industrial and research purposes [25, 26, 28]. By relying on a crowd of user devices, mobile crowdsourcing delivers an engaging solution to collect insightful reports from the wild. However, the design of such platforms presents some critical challenges related to the management of users, also known as *workers*. In particular, the privacy of the workers is often underestimated by the crowdsourcing platforms and it often fails to be addressed effectively in practice [167]. Official approvals from *Institutional Review Boards* (IRB) and regulatory bodies addressing worker privacy do not only require consent from workers to collect data from their mobile devices (*e.g.*, app permissions, privacy policies, end-user license agreements), but suggest the use of data anonymization mechanisms to minimize the risk of privacy leaks [168].

While data anonymization is commonly achieved *a posteriori* on the server side [53, 56, 32, 27], this approach is subject to adversarial attacks, even when protocols for the communication and the data storage are claimed to be secured [60, 59]. Furthermore, the workers may be reluctant to share *Sensitive Personal Information* (SPI) with third parties (*e.g.*, students contributing to a crowdsourcing campaign initiated by a professor). Gaining the confidence of workers is extremely difficult and we argue in this chapter that the adoption of *a priori* data anonymization mechanisms contributes to delivering a trustable component to better mitigate privacy leaks in the data shared by workers.

For example, the worker's location is not only the most requested but also the most sensitive data collected by mobile crowdsourcing platforms [85]. Our scheme therefore explores the physical proximity of workers to agree on a dissemination strategy for reporting the crowdsourced data. By altering the link between workers and data *consumers* on the server, our approach intends to mix data contributed by several workers within a collaborative data flow that exhibit similar crowd-scale properties and without discarding any SPI. In particular, we propose a system-level service that acts as a proxy within the mobile device for sharing crowdsourced data and from which workers can control their privacy settings. FOUGERE is our implementation of this anonymization scheme and is available as an open source library[1] that can be used by legacy mobile crowdsourcing apps. We illustrate the benefits of FOUGERE by integrating it within the state-of-the-art MOBIPERF mobile crowdsourcing app

[1]https://github.com/m3ftah/fougere

as well as the APISENSE mobile crowdsourcing platform. We evaluate the effectiveness and the impact of our anonymization scheme on these two mobile crowdsourcing systems by deploying and orchestrating a crowd of 15 emulated mobile devices. More precisely, we replay the SFCABS cab mobility traces [169] and we show that FOUGERE defeats state-of-the-art privacy attacks [170, 171, 57] with little impact on the quality of the resulting datasets.

The remainder of this chapter is organized as follows. Section 4.2 provides an overview of the privacy threats in crowdsourcing apps and platforms. Section 4.3 introduces our anonymization scheme and the integration of LPPMs to increase the workers' privacy. Section 4.4 describes the implementation of the FOUGERE open source library on Android. Section 4.5 introduces our evaluation protocol of FOUGERE on the MOBIPERF mobile crowdsourcing app and discusses the results we obtained on an experimental setup involving 15 emulated workers. Section 4.6 discusses the threats to validity of our contribution. Finally, Section 4.7 concludes on this chapter.

4.2 Privacy Threats in Mobile Crowdsourcing Systems

This section discusses the potential threats in mobile crowdsourcing systems along 2 axes: the *system model* and the *sensitive personal information*.

Mobile crowdsourcing system model. The architecture we consider is a mobile crowdsourcing campaign that involves three components, namely, *mobile devices*, *crowdsourcing apps*, and *storage servers*.

We consider that the mobile *crowdsourcing apps* can be trusted as we believe that the owner of the mobile crowdsourcing app or platform is interested in gathering insightful datasets with the consent of workers, especially if this mobile app is open sourced.

However, we consider that the *storage server* can be compromised and reveal some sensitive personal information on behalf of the owner and the workers. While crowdsourced datasets are expected to be anonymized prior to any online publication or reuse [49, 172], the potential threats we consider encompass a remote or physical access to the storage server to steal the raw crowdsourced dataset prior to the application of any anonymization technique. Another similar threat may involve a *man-in-the-middle* (MITM) attack to intercept the crowdsourced data while it is uploaded to the remote storage server. For example, no matter if they are deployed in the cloud or on-premise, the remote storage servers may suffer from security leaks that can be exploited by an adversary. Moreover, stakeholders who get the data before its anonymization can hold it without anonymizing it, knowing that anonymization will decrease data utility, or because they do not know exactly how to process the dataset. Thus, it may be stored on the server until they figure out how to anonymize it. Furthermore, storing the crowdsourced data on the server must comply with *The EU General Data Protection Regulation (GDPR)* and the *Privacy Act of 1974* of the USA. With crowdsourced data, it is difficult to comply with the regulations, for example: giving the users the right to delete their own data whenever they want. FOUGERE does resolve issues related to these regulations as it does not store personal identifiers on the server side and the users' data are anonymized before being uploaded to the server.

Sensitive personal information in mobile crowdsourcing. In Section 2.1 we considered 4 categories of Sensitive personal information (SPI): identifiers, Point of Interest (POI), routines and markers. These SPI can be collected by the mobile apps and can be subject to attacks from adversaries.

Existing mobile crowdsourcing systems may have collected some sensitive personal information. For example, the MOBIPERF mobile crowdsourcing app[2] publishes a privacy statement on the nature of data that is collected by the app:

> MOBIPERF regularly collects information about the network performance perceived from your device. This information includes the following:
>
> - Device properties:
>   - Device manufacturer, model, OS, and OS version,
>   - Google account ID, if the data is posted non-anonymously. You can choose to post anonymously, or not. Posting non-anonymously provides valuable data that can allow the MOBIPERF researchers to better understand networks and allows you to access your historical data.
>   - Salted hash of device ID (*e.g.*, IMEI)
>   - Coarse-grained Cell ID location information
> - Network properties:
>   - Current network connection type (*e.g.*, HSPA or LTE)
>   - Current carrier (*e.g.*, Verizon)
>   - Current cell tower ID and signal strength
>
> We **DO NOT** collect any personally identifying information in addition to that listed above: no names, no private data from other apps, and so on.

While MOBIPERF claims that *data released to the public domain will not contain any identifying information*, some of these information chunks may contribute to indirectly

[2]http://mobiperf.com

identify the workers, like the cell tower ID or the carrier, or even some timestamp (not mentioned online but included in the source code[3]).

4.3 FOURGERE: Empowering Workers with LPPMs

To overcome the above privacy threats and strengthen the location privacy of workers, this chapter introduces FOUGERE, which allows the anonymization and dissemination of the workers' crowdsourced data across the network. This section introduces the key design principles we adopted, a description of how crowdsourced data flows across multiple devices, as well as the core *Location Privacy Protection Mechanisms* (LPPMs) that are provided by FOUGERE.

Collaborating with apps & workers. In order to be trusted and gather a large crowd of workers, we assume that mobile crowdsourcing apps and platforms are doing their best to enforce privacy and security support. However, developers are not necessarily aware of privacy threats and implementing a comprehensive support for such a support might be time-consuming and error-prone. FOUGERE therefore offers mobile crowdsourcing apps the possibility to offload the management of the worker privacy settings and the data dissemination across the network, thus letting developers focus on the core business of the mobile app. By making FOUGERE available as a system service within the mobile device, a worker can configure her privacy settings for each installed mobile crowdsourcing app and decide upon the level of privacy she requires to be enforced for each of the apps integrating FOUGERE.

From the perspective of the mobile crowdsourcing app, FOUGERE mostly expects the developer to *i)* declare the SPI collected by the mobile app (*e.g.*, location, timestamp, identifier), *ii)* implement a data forwarding task in charge of connecting to the remote storage server and uploading the data, and *iii)* forward any crowdsourced data to the library, instead of uploading it directly to the server.

More specifically, FOUGERE offers the workers control over worker's privacy preferences, thus providing a preference panel to *i)* explore the list of mobile crowdsourcing apps and respective SPI, *ii)* monitor and control the volume of crowdsourced data reported by each app, and *iii)* configure the list of LPPMs to be enforced by a given mobile crowdsourcing app.

By following these principles, FOUGERE can collaborate with the mobile app and the worker to ensure the anonymization and the dissemination of crowdsourced data. Figure 4.1

[3]https://github.com/Mobiperf/MobiPerf

overviews these principles and illustrates how a mobile app can disseminate crowdsourced data without and with FOUGERE. In particular, mobile crowdsourcing apps that do not fulfill the design principles—or do not integrate FOUGERE—will upload crowdsourced data directly to the remote server, thus exposing the workers to the privacy threats introduced in Section 4.2. By integrating FOUGERE, any mobile crowdsourcing app simply delegates the data dissemination to the library. FOUGERE enforces the worker's privacy settings and applies the appropriate LPPMs to the forwarded data. Such mechanisms include *privacy filters* (to discard the data), *privacy distortions* (to alter the data) and *privacy aggregation* (to group the data).

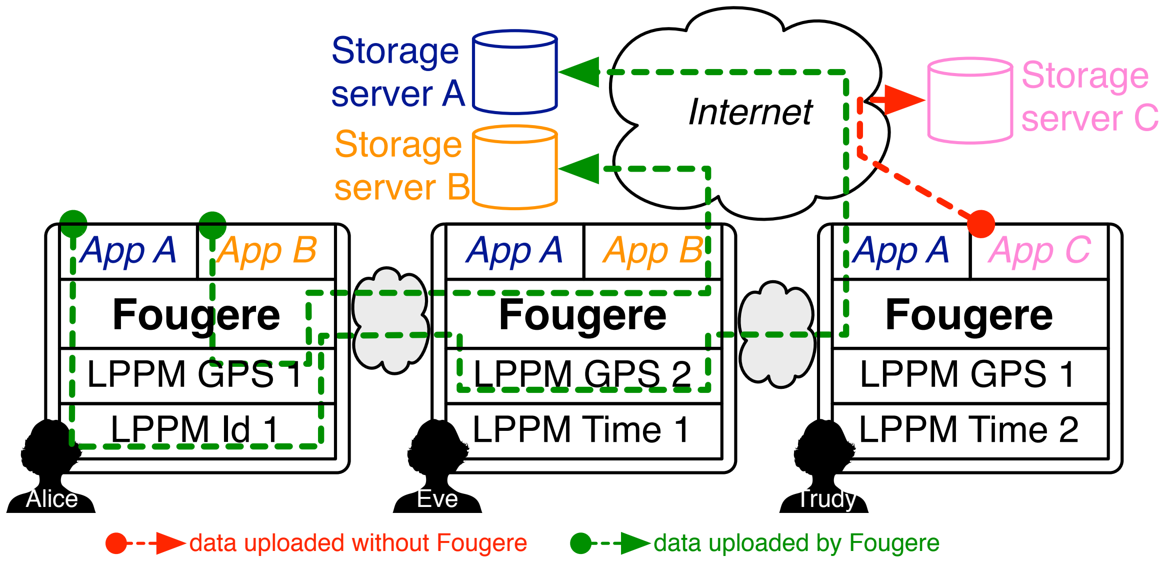


Fig. 4.1 Overview of FOUGERE

Enabling crowdsourced dissemination.

Algorithms 1, 2, 3 summarize our dissemination process, these three algorithms (SEND, ONDISCOVER, ONRECEIVE) are explained as follows:

1. **SEND** (Algorithm 1). If the crowdsourced data has not been discarded by one of the configured LPPMs, FOUGERE stores a message for dissemination that is composed of *i)* a *payload*, *ii)* a *configuration* of remote LPPMs, *iii)* a *bloom filter* of forwarder devices, and *iv)* a *time-to-live* (TTL) for the dissemination process. While the *payload* refers to the crowdsourced data, which has eventually been altered by the local LPPMs, the message also includes some *configuration* parameters for LPPMs that can be executed by remote instances of FOUGERE (*e.g.*, replacing the location of the source by the location of the forwarder). In order to avoid a given message to be forwarded by the same set of mobile devices, FOUGERE also includes a bloom filter that encodes

the list of forwarder nodes, without discarding their identifiers. The *bloom filter* is a data structure that can tell us if the worker's *id* is present in the bloom filter or not, without discarding the ids of other workers. The *bloom filter* is configured with a false positive probability of 0.1 and a number of expected elements equals to the TTL. Finally, the message encloses a *TTL* to define the numbers of workers' hops requested by the worker to disseminate the message.

2. **ONDISCOVER** (Algorithm 2). This procedure is triggered by FOUGERE discovery service whenever other workers are detected in vicinity. FOUGERE filters out the known workers by querying the bloom filter, and randomly picks one candidate. FOUGERE updates the bloom filter with the worker's *id* so that worker will not receive the same data again. Once a data is forwarded, FOUGERE discards it from the forwarding queue.

3. **ONRECEIVE** (Algorithm 3). Upon receiving a forwarded data, a remote FOUGERE node eventually applies the LPPMs listed in the configuration of the data. If the TTL equals 0, then FOUGERE stores the payload in the uploading queue to be uploaded by the mobile crowdsourcing app to the remote storage server. Otherwise, FOUGERE decreases the TTL and stores the resulting data in the forwarding queue for further dissemination.

The adoption of this dissemination scheme prevents server-side adversaries (including peers) to attack the forwarded data in order to reveal some worker's SPI. In particular, by mixing the origins of crowdsourced data uploaded by a single device, FOUGERE confuses adversaries who would run attacks on a specific worker.

Mobile crowdsourcing apps share similarities with *Delay Tolerant Networks* (DTN) by considering that the crowdsourced data does not have to be immediately uploaded to the remote server and can tolerate delays ranging from minutes to hours. We exploit this property to adopt a multi-hop forwarding scheme in FOUGERE, which ensures that at least k neighboring devices with the same mobile app are also potentially collecting data in the same area, thus preventing the worker to be spotted as an outlier. This opportunistic dissemination schemes does not assume any specific network protocol and can be implemented atop of legacy discovery protocols, such as WiFi-Direct or Google Nearby connections.

Furthermore, FOUGERE complements existing privacy-preserving mechanisms, like the TOR anonymity network [173], that prevents the server from tracing back the worker. Which can also be used by FOUGERE to upload the crowdsourced data to the remote server. Using TOR, therefore, hides workers from the remote server, but it loses the physical proximity information that is useful for local LPPMs. For example, when an isolated worker is contributing from within the countryside, she can still report data using TOR but she will

Algorithm 1 FOUGERE Send algorithm.

```
Ensure: $Q_{forward}$: Messages to be forwarded to devices.
  procedure SEND(app, data)
      for all $lppm_{app} \in \text{LPPM}(app)$ do                ▷ applying local LPPM
          $data \leftarrow \text{APPLY}(lppm_{app}, data)$
          if $data = \emptyset$ then
              return                                          ▷ data discarded by LPPM
          end if
      end for
      $ttl \leftarrow \text{TTL}(app)$
      if $ttl > 0$ then                                       ▷ worker enabled the dissemination
          $cfg \leftarrow \text{REMOTELPPM}(app)$
          $bloom \leftarrow \text{BLOOMFILTER}(this)$
          $Q_{forward} \leftarrow Q_{forward} \cup \langle data, cfg, bloom, ttl \rangle$
      else                                                    ▷ worker disabled the dissemination
          $Q_{upload} \leftarrow Q_{upload} \cup data$
      end if
  end procedure
```

Algorithm 2 FOUGERE OnDiscover algorithm.

```
Require: $Q_{forward}$: Messages to be forwarded to devices.
  procedure ONDISCOVER(peers)
      for all $\langle data, cfg, bloom, ttl \rangle \in Q_{forward}$ do
          $candidates \leftarrow peers$
          repeat
              $candidate \leftarrow \text{RANDOM}(candidates)$
              if $candidate \notin bloom$ then
                  $Q_{forward} \leftarrow Q_{forward} \setminus \langle data, cfg, bloom, ttl \rangle$
                  $bloom \leftarrow bloom \cup candidate$
                  $\text{FORWARD}(candidate, \langle data, cfg, bloom, ttl \rangle)$
                  break
              end if
              $candidates \leftarrow candidates \setminus candidate$
          until $candidates = \emptyset$
      end for
  end procedure
```

Algorithm 3 FOUGERE OnReceive algorithm.

```
Ensure: $Q_{forward}$: Messages to be forwarded to devices.
Ensure: $Q_{upload}$: Data to be uploaded by the app.
  procedure OnReceive($data$, $cfg$, $bloom$, $ttl$)
      for all $lppm_{cfg} \in$ LPPM($cfg$) do                    ▷ applying LPPM
          $data \leftarrow$ Apply($lppm_{cfg}, data$)
          if $data = \emptyset$ then
              return                                          ▷ $data$ discarded by LPPM
          end if
      end for
      $ttl \leftarrow$ TTL($m$) $- 1$
      if $ttl > 0$ then                                    ▷ $data$ to be sent to another device
          $Q_{forward} \leftarrow Q_{forward} \cup \langle data, cfg, bloom, ttl \rangle$
      else                                                 ▷ $data$ to be uploaded by the app
          $Q_{upload} \leftarrow Q_{upload} \cup data$
      end if
  end procedure
```

remain exposed to location privacy attacks. To further improve the privacy of the workers, TOR can be used after the data dissemination phase when the data is ready to be uploaded to the server, thus, the server will not be able to identify even if a worker is participating to the crowdsourcing campaign or not.

Controlling LPPMs from devices. In order to give the worker more control over her own data, FOUGERE includes several LPPMs that can be configured by the worker to decide upon the quality and the volume of crowdsourced data to be obfuscated. In particular, we consider 3 classes of LPPMs: *filters*, *distortions*, and *aggregations*, which can be implemented within a mobile device and used to obfuscate one of the SPI of the user.

Privacy Filters are a group of LPPMs that can decide autonomously if a crowdsourced data can be shared with the crowdsourcing platform or not. For example, a LocationFilter applies to *points of interests* and can be configured by the worker to define *white areas* or *black areas* that delimit zones where the mobile crowdsourcing app can or cannot collect data, respectively. Similarly, a TimeFilter rather applies on *routines* and is used with configured periods along which a mobile crowdsourcing app can or cannot collect data. Finally, a QuotaFilter is a more generic filter that can accept a worker-defined quota of crowdsourced data to be uploaded before discarding once this quota is reached.

Privacy Distortions are another class of LPPMs that can modify the value of an enclosed SPI in the crowdsourced data to be shared. For example, a IdentifierDistortion will change the value of an identifier at a given frequency (every request, hour, day), while a location distortion adds a controlled random noise to the worker's location (depending on radius r with a level of privacy that depends on r) into the reported coordinates [174]. These distortions can be generalized to a wider set of SPI, such as *routines*, to reduce the accuracy of the data (*e.g.*, rounding the timestamp to the nearest hour) [64]. Based on this principle of noise injection, *samples* can also be generated by cloning a crowdsourced data and applying a distortion to one of the declared SPI. This results in the dissemination of several, almost similar, crowdsourced data samples that can be used by the worker to fuzz her location [175].

Privacy Aggregations reflect the last class of LPPMs that are supported by FOUGERE and propose to delay the dissemination of crowdsourced data by grouping them along a given criteria. For example, a TimeAggregation will group data per hour and apply an aggregation operator (like the average, the median, the min or the max) to the enclosed timestamp in order to report the same value for all the aggregated samples before reporting them. A MarkerAggregation is an example of remote LPPMs that will be encapsulated with the crowdsourced data and wait for a given marker (*e.g.*, the ISP name) to appear at least k times before being uploaded. This LPPM is an example of a distributed implementation of the k-anonymity algorithm [176, 177] that we can apply on a wide diversity of SPI, including GPS coordinates.

Summary. By combining an opportunistic dissemination scheme with worker-defined LPPMs, FOUGERE aims at leveraging the privacy properties of legacy mobile crowdsourcing apps and platforms. Before assessing the efficiency of FOUGERE, we now report on the implementation of these principles on the Android platform.

4.4 Implementation Details on Android

On Android, FOUGERE is packaged as an open source library that deploys system service within the mobile device of a worker. This system service currently builds on the Wi-Fi Direct network interface to exchange crowdsourced data between nearby devices of workers. It can be shared by multiple crowdsourcing apps of a given device to centralize the control of privacy settings, which are exposed to the worker as a dedicated preference panel. Thanks to its modular architecture, FOUGERE can be further extended with additional LPPMs, which are not covered by this work.

Application programming interface. Any mobile crowdsourcing app can integrate FOUGERE through a simple API that exposes the following operations:

`hasFields(...)` is called by the mobile crowdsourcing app to declare any SPI as a `PrivacyField` , that refers the classes `IDENTIFIER`, `POI`, `ROUTINE`, and `MARKER`;

`forward(...)` enlists a task in charge of uploading a crowdsourced data sample to the remote server when the TTL expires;

`send(...)` delegates the dissemination of a crowdsourced data to FOUGERE.

All these operations are grouped within the interface `Fougere`, which is the facade used by a mobile app (cf. Listing 4.1). As introduced, in Section 4.2, FOUGERE provides a privacy support for 4 categories of SPI: `IDENTIFIER`, `POI`, `ROUTINE`, and `MARKER`.

Listing 4.1 FOUGERE API

```
public enum PrivacyFields {
  IDENTIFIER, POI, ROUTINE, MARKER;
}

public interface PrivacyField {
  PrivacyFields type();
}

public interface Fougere<D extends Serializable> {
  Fougere<D> hasField(PrivacyField f);
  void upload(Consumer<D> task);
  void send(D data);
}
```

Integrating FOUGERE within a mobile crowdsoucring app requires to request 2 specific permissions: `fougere.permission.SHARE_DATA` and `fougere.permission.CONTROL_PRIVACY`, which are intended to inform the workers of the compatibility of the mobile crowdsourcing app with FOUGERE and the possibility to adjust the privacy settings for this app.

Opportunistic dissemination. The current implementation of the FOUGERE dissemination module builds on the WiFi-Direct technology to discover nearby devices. When a mobile crowdsourcing app forwards a message, FOUGERE triggers the configured LPPMs and accumulates the data in the forwarding queue. For each data accumulated in the forwarding queue, FOUGERE picks a random peer that has never received this data and forwards it.

The second device receiving a connection request will receive the message `onReceive()` (cf. Algorithm 3). If the message reaches the configured number of device hops ($ttl = 0$), then the forwarded data is placed in an uploading queue, which will be emptied as soon as the remote mobile crowdsourcing app runs by invoking the upload handler registered by the app.

LPPM integration. FOUGERE combines the implementation of a decentralized dissemination scheme with the integration of LPPMs that can filter out data or alter its content depending on the worker's privacy settings. More generally, FOUGERE intends to leverage the integration of additional LPPMs to better control the data uploaded by any compatible crowdsourcing app. FOUGERE organizes these LPPMs along the 4 categories of SPI it supports. An LPPM complies to an interface `Lppm<T extends PrivacyField>` that declares the category `T` of SPI it considers and implements a method to apply a privacy mechanism on the uploaded data, which eventually returns the anonymized data to be further processed by FOUGERE.

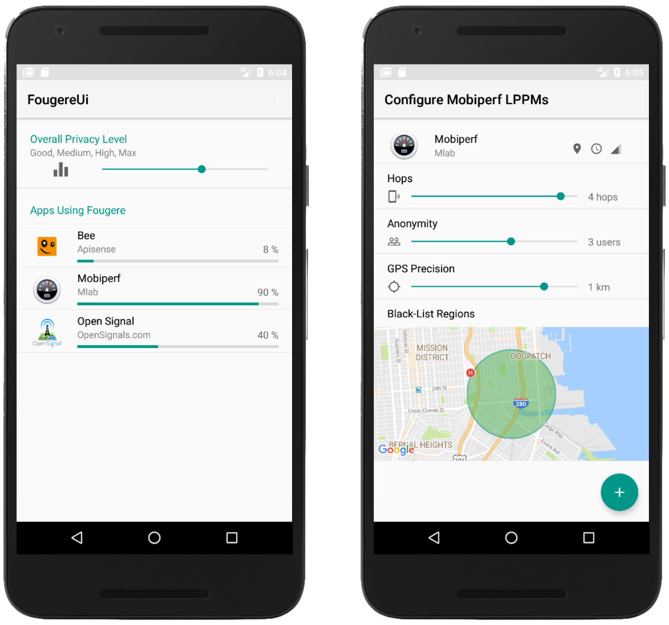
To introspect and eventually modify the crowdsourced data sent by the mobile app, the Android implementation of FOUGERE uses the JXPATH library provided by Apache.[4] This means that, using JXPATH, an LPPM can query the input data, changes its value and inject it back into the original crowdsourced data or discard it. Listing 4.2 provides simplified examples of location privacy filter (`WhiteAreaFilter`) and location privacy distortion (`LocationNoiseDistortion`) that are implemented in FOUGERE by using this LPPM API, according to the heuristics we introduced in Section 4.3.

Listing 4.2 LPPM API

```
public interface Lppm<T extends PrivacyField> {
  <D extends Serializable>
  Optional<D> apply(D data, T field);
}

public class WhiteAreaFilter implements Lppm<PoiField> {
  @Override
  public <D extends Serializable> Optional<D> apply(D data, PoiField field) {
    int lat = field.getLatitude(data), lon = field.getLongitude(data);
    return checkLocation(lat,lon)? Optional.of(data):Optional.empty();
} }

public class LocationNoiseDistortion implements Lppm<PoiField> {
  @Override
  public <D extends Serializable> Optional<D> apply(D data, PoiField field) {
    field.setLatitude(data, randomNoise(radius, field.getLatitude(data)));
    field.setLongitude(data, randomNoise(radius, field.getLongitude(data)));
    return Optional.of(data);
} }
```

In order to effectively apply the worker's privacy settings, FOUGERE operates by first applying the privacy filters, before proceeding with privacy distortions and finally privacy aggregations. In addition to that, privacy distortions and aggregations can also be triggered

[4]http://commons.apache.org/proper/commons-jxpath


FougereUi

Overall Privacy Level
Good, Medium, High, Max

Apps Using Fougere

Bee
Apisense 8 %

Mobiperf
Mlab 90 %

Open Signal
OpenSignals.com 40 %

Configure Mobiperf LPPMs

Mobiperf
Mlab

Hops 4 hops

Anonymity 3 users

GPS Precision 1 km

Black-List Regions

Fig. 4.2 Preference Panel of FOUGERE

remotely to implement decentralized algorithms that build on neighboring samples to increase the privacy of workers [172].

4.4.1 Privacy Settings

FOUGERE aims at informing and supporting workers in the control of their privacy settings. In particular, FOUGERE provides a preference panel to be used by the worker to change her privacy settings (cf. Figure 4.2). Through this preference panel, a worker can list all the mobile crowdsourcing apps installed on her mobile device that requested to use FOUGERE. For each of the listed apps, a worker can add and configure one or several LPPMs she intends to apply on the crowdsourced data, depending on her acquaintance to share with some specific research centers, public institutes, or individuals [178]. To ease the configuration process, instead of requesting raw LPPMs configuration parameters, FOUGERE adopts a slider to configure the privacy level to be considered for a given LPPM and maps to identify POI (cf. Figure 4.2). Finally, the effects of these LPPMs are then depicted to the worker as analytics on the volume of crowdsourced data that has been produced/filtered/forwarded/uploaded by the mobile crowdsourcing app.

4.5 Evaluations of FOUGERE

In our evaluation, we want to answer the following research question:

RQ1: Can we increase the user's location privacy while maintaining a quality crowdsourced data?

RQ2: Can we put the user in center of her privacy control?

To effectively answer this research question, we propose the following evaluation protocol.

4.5.1 Evaluation Protocol

Beyond the challenges related to the integration in legacy mobile crowdsourcing systems, FOUGERE intends to deliver an efficient adoption of LPPMs in a decentralized context. The validation of such a capability requires consideration of a realistic deployment of mobile devices in order to assess the benefits of FOUGERE. Given that we are interested in providing a proof of feasibility for FOUGERE, we are not interested in simulating the behaviour of LPPMs, but rather in assessing the reference implementation of FOUGERE. However, testing mobile applications that make use of opportunistic communications is hard to achieve and reproduce with real mobile devices. We propose to deploy a cluster of emulated devices to reproduce the behavior of a crowd of workers who contribute to a mobile crowdsourcing campaign. We use mobility datasets that are publicly available to control the emulated devices and we collect their interactions to trace their actions *a posteriori*. The crowdsourced dataset collected on the remote server are evaluated by the LPM^2 toolkit [45] to evaluate the preservation of workers' privacy. By adopting such an empirical validation, we can evaluate real applications integrating FOUGERE and we can observe the impact of changing the parameters of FOUGERE (number of hops, LPPMs' specific parameters).

In the remainder of this section, we select the legacy MOBIPERF [27] mobile app as the mobile crowdsourcing app that we considered to assess FOUGERE.

Emulating crowds of workers. The assessment of our opportunistic dissemination scheme and the associated LPPMs requires the consideration of a crowd of workers who installed a mobile crowdsourcing app that integrates FOUGERE. While running an emulator on a single machine is rather resource-consuming and cannot scale, we propose to consider the deployment of a cluster of servers to host multiple Android emulators. As Android emulators do not provide any support for ad hoc communications, such as WiFi-Direct, we use ANDROFLEET [179](Chapter 3.4) to control the discovery of nearby peer-to-peer devices within a cluster of emulators. ANDROFLEET gives us the possibility to emulate a real crowdsourcing campaign using GPS mobility dataset instead of proximity encounters dataset when testing with PEERFLEET.

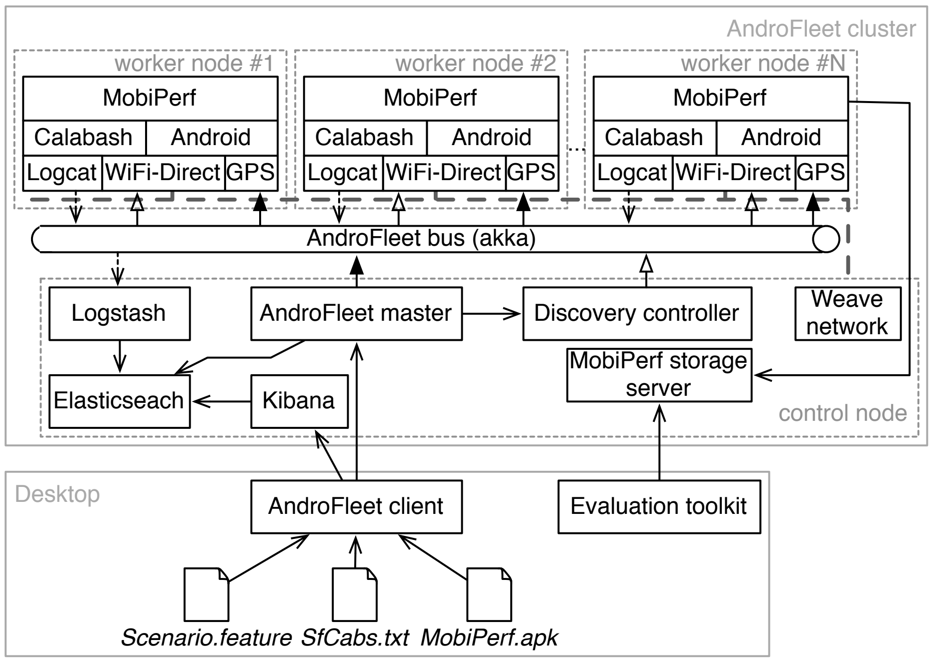


Fig. 4.3 Overview of FOUGERE integration in the ANDROFLEET cluster

In Figure 4.3, we show the integration of FOUGERE in the ANDROFLEET cluster. In particular, the AndroFleet master node takes as input an *app binary* (`MobiPerf.apk`), a *mobility dataset* (`SfCabs.txt`), and a set of *interaction scenarios* (`Scenario.feature`). It deploys the *app binary* and the set of *interaction scenarios* within each emulator and loads the *mobility dataset* into the Elasticsearch service. The ANDROFLEET master collaborates with the Discovery controller to control the virtual location of emulated devices. Each emulated device triggers scenario snippets autonomously and logs their actions via the Android logcat interface, which is automatically forwarded to Elasticseach using Logstash parser. Figure 4.4 reports on the map that can be obtained by querying the logs from Elasticseach using the Kibana UI service. In particular, this map depicts the path followed by crowdsourced messages from the time a crowdsourcing app delegates to FOUGERE to the time the data is effectively uploaded by the crowdsourcing app.

Controlling crowds of workers. To assess the efficiency of FOUGERE using ANDROFLEET, the emulated devices are required to be controlled in order to update their location and eventually internal state, to reproduce the mobility of a crowd of workers. While the choice of such a mobility dataset might be challenging depending on the category of mobile crowdsourcing app, we use the widely used `epfl/mobility` dataset that is publicly available

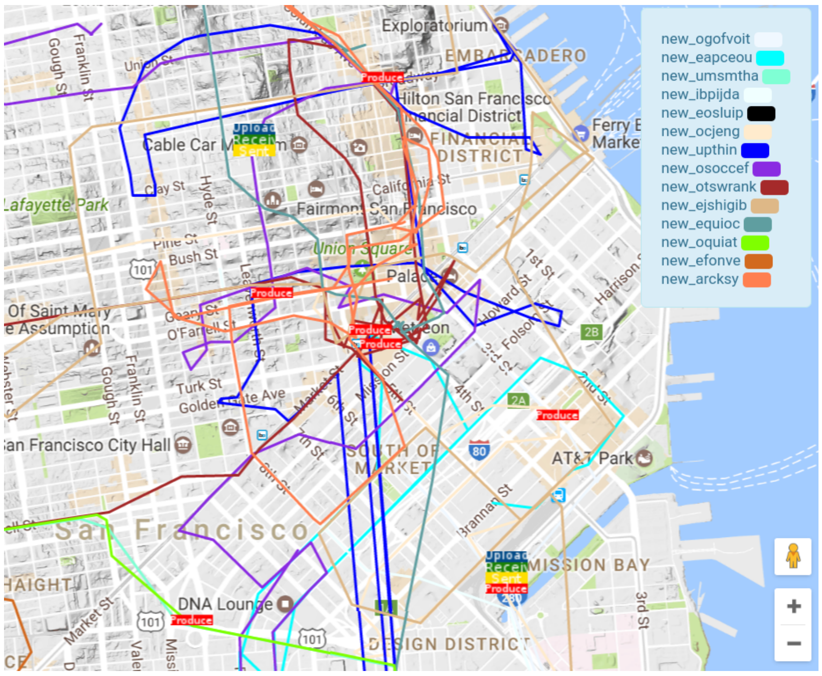


Fig. 4.4 The ANDROFLEET log analytics interface

from CRAWDAD [169] to emulate 15 workers who are performing network measurements with the MOBIPERF mobile app. The crowdsourced dataset contains network measurements reported every 5 minutes by the workers moving in the San Francisco bay area.

Attacking crowdsourced datasets. To evaluate the impact of FOUGERE on the privacy of workers, we use the LPM^2 toolkit [45], which is a state-of-the-art tool for measuring location privacy. In particular, LPM^2 covers the evaluation of the LPPMs that are supported by FOUGERE, like the *privacy filters*, *privacy distortions* and *privacy aggregation*. To validate FOUGERE against privacy attacks, for each configuration, we run an experiment that follows these steps:

1. Run ANDROFLEET with MOBIPERF and FOUGERE (incl. privacy settings),
2. Assign tasks to workers during 3 days, and wait 4 more days for the data dissemination to complete,
3. Gather the logs of data exchanges between workers to evaluate the opportunistic dissemination scheme,
4. Retrieve all the raw crowdsourced data stored on the remote server,
5. Construct the adversary knowledge by tagging the crowdsourced data of one worker (as required by LPM^2),
6. Evaluate the privacy support of FOUGERE with the LPM^2 toolkit,

7. Report on performance, utility, robustness and uncertainty, which are the parameters proposed by Verykios *et al.* [44] to assess LPPMs.

4.5.2 Empirical Evaluation

In this section, we instantiate the above experimental protocol to assess FOUGERE as a practical support to improve the location privacy of workers.

Experimental Setup In particular, thanks to the ANDROFLEET [179] emulation platform, we can reproduce the execution of a deployment of 15 mobile instances emulating a one-week crowdsourcing campaign, thus proposing a realistic input dataset to evaluate FOUGERE. Then, we compare the behaviors of 6 configurations of the MOBIPERF app:

1- VANILLA refers to the reference implementation of the MOBIPERF Android app, as it can be downloaded from *http://www.mobiperf.com.* This configuration is used to demonstrate the vulnerability of legacy mobile crowdsourcing apps with regards to potential privacy threats. It is also used as a witness to evaluate the benefits of the other configurations including FOUGERE;

2- FOUGERE *with no LPPM* refers to the extension of MOBIPERF with the FOUGERE library. This configuration is used to isolate the properties of our opportunistic dissemination schemes independently of the impact of LPPMs. In particular, we consider the following worker configurations for the number of required hops to disseminate the crowdsourced data and the WiFi-Direct discovery scans: (a) $\langle 1\,hop, 5\,min\rangle$, (b) $\langle 4\,hops, 5\,min\rangle$ (default configuration), and (c) $\langle 4\,hops, 10\,min\rangle$;

3- FOUGERE *with LPPMs* refers to the FOUGERE library with the default configuration 2-b selected with 2 privacy distortions—*location noise* and *time noise*—and 1 privacy aggregation—*k-anonymity*, which are representative LPPMs used by the state-of-the-art. To configure these LPPMs, we consider 2 worker profiles, which are mapped to the following values:

(a) *weak privacy profile* where location noise is set to $\langle 1, 0.1, 0.05\rangle$, thus reducing the location precision by 1 digit with a probability of 0.1 and possibly removing the location with a probability of 0.05. Time noise is set to $\langle 30, 0.1, 0.05\rangle$, thus reducing the time precision to half an hour with a probability of 0.1 and possibly removing the timestamp with a probability of 0.5, and finally k-anonymity is set to $\langle 2\rangle$, meaning that at least 2 samples should be produced in the same area to be forwarded;

(b) *strong privacy profile* configured with location noise $= \langle 2, 0.2, 0.1\rangle$, time noise $= \langle 60, 0.2, 0.1\rangle$ and k-anonymity $= \langle 4\rangle$ as privacy settings.

None of these configurations includes a privacy filter, as these LPPMs are expected to be used to hide the living and working places of workers and the input dataset does not include this information. Furthermore, this work does not aim at evaluating the efficiency of individual LPPMs, but rather demonstrating the benefit of combining them in an open framework like FOUGERE.

Performance analysis. FOUGERE implements an opportunistic dissemination scheme to improve the privacy of workers. By doing so, FOUGERE exploits the physical proximity of workers to exchange crowdsourced data and to guarantee that the uploaded data has been forwarded along a number of hops requested by the worker. Figure 4.5 depicts the *time to converge* as a metrics to evaluate *i)* the impact of integrating FOUGERE on a legacy mobile crowdsourcing app like MOBIPERF, and *ii)* the effect of the number of hops and the WiFi-Direct discovery duration parameters. One can observe that, by using FOUGERE, not all the crowdsourced data is reported back to the remote storage server. This can be explained by the fact that some workers are contributing in sparsely populated areas, which prevents FOUGERE from disseminating the collected measurements. This result is actually a strength of FOUGERE as it automatically protects the isolated workers from adversaries who would apply some location distribution attacks to identify them.

Regarding the parameters of FOUGERE, one can note that the delay to upload data and the volume of reported data is more affected by the discovery duration than the number of hops required to upload the crowdsourced data. By increasing the delay of peer discovery, mobile devices miss some other workers in their vicinity in order to improve the time to converge. Therefore, we privilege the configuration 2-b (4 hops and 5 minutes) as the default configuration for FOUGERE. However, the worker remains free to adjust each of these parameters.

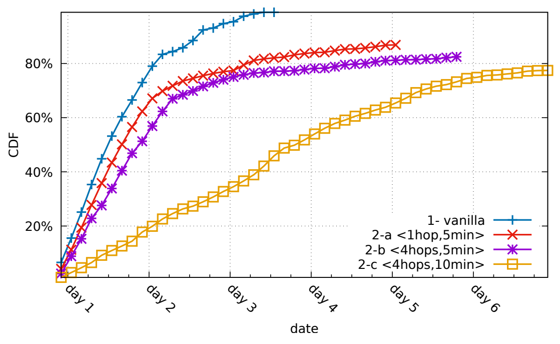


Fig. 4.5 Measurements' time to converge

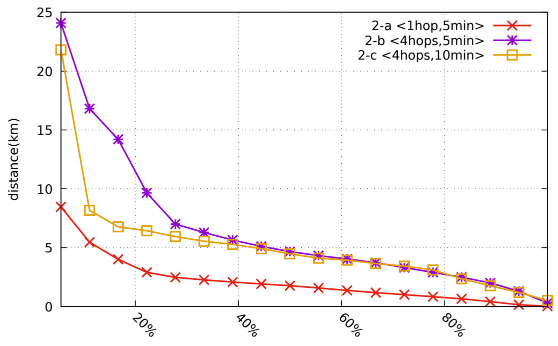


Fig. 4.6 Distance traveled by measurements

The *traveling distance* is another interesting metrics to evaluate the efficiency of the dissemination process and the relevance of peer-to-peer communications. Increasing this data traveling distance with FOUGERE contributes to better shuffle crowdsourced data produced by a crowd of workers. Figure 4.6 reports on this distance traveled by the crowdsourced data before being uploaded back to the remote storage server. In particular, the default configuration of FOUGERE maximizes the traveled distance with 20 % of data that traveled at least 10 *km* (6.2 *miles*), thus ensuring that the data was conveyed by FOUGERE as far as possible from the location where it has been produced.

Utility analysis. While FOUGERE aims at improving the location privacy of workers, the utility of the resulting dataset should not be neglected. The dataset utility is calculated using the LPM^2 toolkit data traces format. Then, the utility corresponds to the ratio of the number of data uploaded to the server to the number of the ground truth data, which is collected directly from the devices without using FOUGERE.

Figure 4.7 reports on the tradeoff between utility and anonymity of the configurations we considered. While the vanilla configuration (1) offers the highest utility with no anonymity, one can observe that the integration of FOUGERE seriously improves the anonymity of workers without seriously impacting the utility of the resulting dataset. As mentioned in Figure 4.5, the loss of 20 % utility is mainly due to crowdsourced data in sparsely populated areas that were retained by FOUGERE. Furthermore, adding some LPPMs (configurations 3-a and 3-b) strongly increase the anonymity of workers.

Interestingly, one can observe that the *weak privacy profile* offers a good privacy/anonymity tradeoff compared to the *strong privacy profile*, which seriously harms the dataset utility without bringing any further improvement over anonymity.

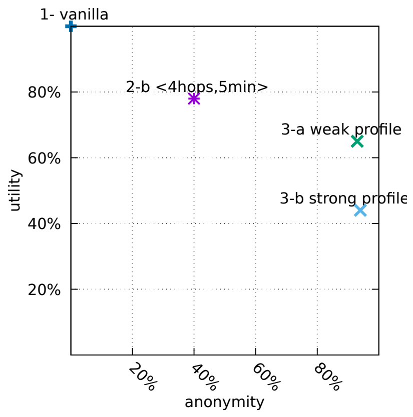


Fig. 4.7 Dataset utility

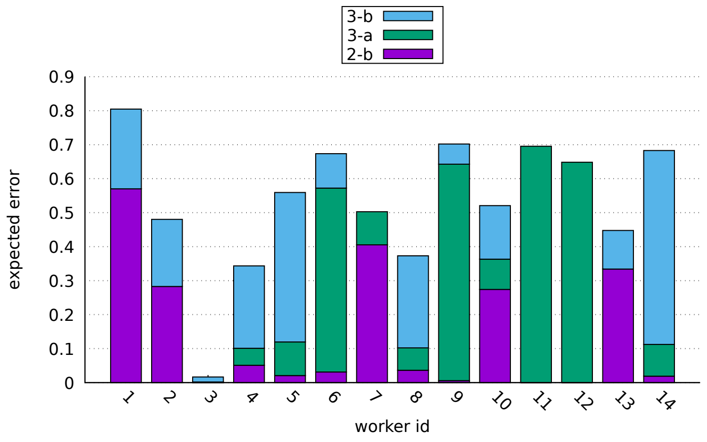


Fig. 4.8 Robustness against location privacy attacks

Robustness analysis. Regarding the effective privacy support offered by FOUGERE, we used the LPM^2 toolkit to evaluate the robustness of crowdsourced datasets that are uploaded through FOUGERE, this metric correspond to the LPM^2 toolkit's distortion metric[5] [180], which measures the location-privacy of the worker at each time instance. We randomly select the crowdsourced data reported by one of the workers as the adversary knowledge required by LPM^2 to apply location privacy attacks and we depict in Figure 4.8 the reported robustness for 14 workers. While LPM^2 successfully defeats worker 3 (used as the adversary knowledge), the other 14 workers clearly benefit from the integration of FOUGERE. In particular, we can observe that the integration of LPPMs complements efficiently our opportunistic dissemination scheme by supporting workers who are not located in a dense area and by offering similar privacy guarantees. Successfully location privacy attacks requires to combine different strategies to cope with the profile of workers.

While FOUGERE offers the worker the possibility to manually adjust her privacy settings, one of the perspectives of this work consists in leveraging this configuration process by delivering privacy risk feedback that would guide her settings accordingly. By recommending the privacy settings of FOUGERE, we aim at maximizing the individual privacy of workers, while preserving the overall utility of the crowdsourced dataset (cf. Figure 4.7).

Uncertainty analysis. Finally, the uncertainty metric evaluates the uncertainty that our LPPMs introduce to the adversary when she tries to reconstruct the hidden information from the crowdsourced dataset on the server. Figure 4.9 reports on the uncertainty metrics computed by LPM^2 [45]. One can observe that FOUGERE succeeds to increase the uncertainty of adversaries when it combines the opportunistic dissemination scheme with LPPMs, which confirms our previous observation. Furthermore, it also assesses that adopting a *weak privacy profile* already brings a reasonable level of privacy that puts adversaries in difficulties.

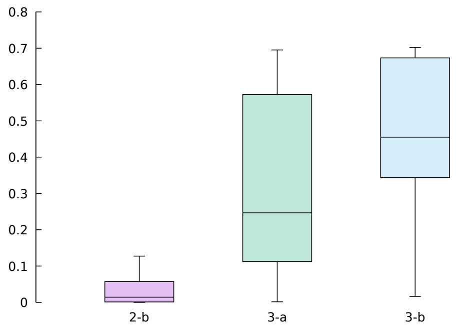


Fig. 4.9 Adversary uncertainty

Indicator	Value
Crowdsourced dataset size	29,712
Exchanged messages	113,785
Contributions per user	59
Messages forwarded per user	227
Detected neighbors	1,730,827
Established connections	127,545
Isolated users	8

Fig. 4.10 Overhead analysis for 500 workers

Overhead analysis. To analyze the overhead induced by our data dissemination process, we report in Table 4.10 the statistics related to an experiment involving 500 emulated workers

[5]https://github.com/rzshokri/quantifying_location_privacy

for 24 hours. Along the experiment, the workers adopt the default configuration of FOUGERE $\langle 4\,hops, 5\,min \rangle$ (2-b). The overhead per user and at the scale of the crowd does not exceed 4 times the initial volume of contributions. FOUGERE also discards 8 users considered as isolated and thus identifiable by tools like LPM^2.

Synthesis. To answer **RQ1**, given the preserved privacy for our workers and the obtained data utility we had for our crowdsourced data using FOUGERE. We can say that using FOUGERE, we can increase the user's privacy while maintaining a quality crowdsourced data. While we have introduced an overhead for the data dissemination, we believe that this overhead is acceptable tradeoff to preserve the users' privacy while maintaining the data utility.

To answer **RQ2**, in our experiment we have used different privacy settings, given that these privacy settings are set from the user's mobile device (Figure 4.2), we can say that the user is the center of control of her privacy.

4.6 Threats to Validity

This section analyzes the factors that may threaten the validity of our results.

Internal validity concerns the relation between theory and observations. In this work, they could be due to measurement errors reported during the experimentation. That is the reason why we did several experiments and we tried to reduce as much as possible external factors as explained in our experimental protocol in Section 4.5.2. We also performed our experiments on a crowd of emulated devices equipped with real mobile apps, instead of a simulation, to reduce the threats that could be due to an integration of the proposed approach in a real mobile crowdsourcing app or platform.

External validity relates to the possibility to generalize our findings. We believe that further validations should be done on different mobile crowdsourcing apps and with different configurations to broaden our understanding of the impact of LPPMs on the privacy of workers. Thus, we are not assuming that our results can be used to generalize the impact of a specific LPPM on privacy. However, we believe that this work contributes to prove that there is a clear positive impact for the privacy threats we considered.

Reliability validity focuses on the possibility of replicating our experiments and results. We attempt to provide all the necessary details to replicate our study and our analysis. Furthermore, the reference implementation of FOUGERE, the input datasets, case studies and testing environment are made available online to leverage its reproduction by the research community.

Construct validity has been covered by considering the convergent validity of privacy and utility properties. We observed that these two properties are related in practice, as the application of LPPMs tends to decrease the utility of the crowdsourced dataset. This observation calls for the identification of a privacy and utility trade-off in the context of mobile crowdsourcing systems, as acknowledged by [49].

Conclusion validity refers to the correctness of the conclusions reached in this work. The empirical evaluation we reported confirms our initial assumption that *a priori* anonymization techniques can be used to leverage the privacy of workers. We were also careful with our conclusion with regards to the impact on the utility of crowdsourced dataset.

4.7 Conclusion

Mobile crowdsourcing apps and platforms are more and more challenged to protect their workers' privacy. To address this challenge, we introduce FOUGERE to increase worker's privacy in mobile crowdsourcing systems. FOUGERE operates a system-level service that collaborates with a mobile crowdsourcing app to declare SPI and delegate the dissemination of crowdsourced data by leveraging the physical proximity of workers. This opportunistic dissemination scheme is complemented by the integration of LPPMs that can be configured by the workers, independently of the installed mobile crowdsourcing apps.

Finally, we consider the deployment of FOUGERE in a realistic Android environment by emulating a crowd of 15 mobile devices hosting different versions of MOBIPERF and FOUGERE to assess our contribution. We show that FOUGERE succeeds in improving the workers' privacy by defeating location privacy attacks implemented by the LPM^2 toolkit.

In the next chapter, we present our WiFi place recognition algorithm that, like FOUGERE, will help the user stay on top of her privacy. While FOUGERE enables the location privacy for GPS data for outdoor environments, our WiFi places recognition algorithm will help preserve the user privacy in the indoor environments, as the WiFi places data is the preferred data to collect for indoor environments. We follow the same anonymization scheme as FOUGERE, only anonymized WiFi places data would be leave the user's device, as we bring the place recognition algorithm in the mobile device.

Chapter 5

Privacy preserving WiFi Places for Mobile User Contextualization

This chapter presents the thesis contribution about the WiFi places privacy preserving solution. This chapter starts by motivating the need for WiFi places and its anonymization, it provides several real life examples, where WiFi places can be of great help. Then, the creation of the WiFi places is explained, along with how they are shared between users. The WiFi places recognition algorithm is described, the algorithm uses the Simhash similarity function to anonymize the WiFi places. The DAYKEEPER mobile app is introduced as a use case for the WiFi places framework, the app generates a timeline for the user, as it is also used to collect anonymized data to asses the utility of the WiFi places recognition algorithm. The data collection campaign was presented, the campaign went for 1 month with 30 users. Later, the evaluation of WiFi places is conducted, it is evaluated on the privacy and the utility aspects. An analysis of WiFi scans, has led to conclude that the WiFi places framework can detect if the user is moving between places or staying at the same place. Several variants of the algorithm has been evaluated with different metrics to assess the shareability and the performances of the WiFi places, the evaluation also shows how a distance similarity can be used between the data of the same place. This chapter concludes on the utility and the privacy preservation properties of the WiFi places.

5.1 Introduction

Mobile devices are getting more capabilities each year. Along with that, the user base is getting expanded, their expectations are becoming higher each year. The users are now more insistent than ever in terms of their user experience. In order to meet the user expectation, the mobile apps are supposed to understand the user environment and act accordingly. Mobile apps tend to collect a lot of information about their user in an attempt to provide the best possible user experience. These collected data are sometimes too much that they are sensitive to the user privacy, but sometimes the collected data cannot give the app enough information about the user and her surrounding environment. Both in the state of the art and in the industry we lack methods of data collection for the user contextualization. A small amount of data, if collected in the right way, can be of a value, and thus provide an informative decision that can be used to improve the user experience and to understand about the user environment. Such data collection methods have to be well studied and documented.

We explore the WiFi scans data, we create a new data type called the *WiFi places* that gives us a good user contextualization. This *WiFi places* data can help the developers and researchers understand more about the user's current place and her immediate surrounding environment.

In this chapter, we introduce a framework that contains our place recognition algorithms, we show how our framework can be tuned to meet certain types of requirements and use cases. Our framework provides the developers with a simple way to collect the user *WiFi places*, our framework's parameters have been studied in this chapter. Our framework produces a simple data type that can be understood by scientists, thus, offering a way to build predictive models on top of it, and a way to aggregate the user data based on her place.

Other developers can use this WiFi places data to provide an environment aware apps. The user experience on these apps can be more personalized taking into consideration the precision of the WiFi places. This WiFi place data can be aggregated on the device level without the need to send these sensitive data to a cloud service. This would give the user all the control over the anonymization scheme on the data, like the privacy level and the quantity of data the user feels comfortable with.

Our approach provides an automatic user contextualization on real-time on mobile device level without the need to any server, or any user input. Our framework can monitor the habits of the user in an offline mode, as it does not require any cloud based service. The *WiFi places* framework does not need any user interactions, a postoriori interaction is only required if the user wants to label the existing places.

The remainder of this chapter is organized as follows. Section 5.2 motivates the use of our framework with real life examples. Section 5.3 provides an overview of the *WiFi places* and its properties. Section 5.4 introduces how our *WiFi places* can be shared between apps, devices and users. Section 5.5 describes our implementation of the *WiFi places* as a software library on Android. Section 5.6 presents our data collection using our *WiFi Places* framework. Section 5.7 shows our analysis of the different place recognition methods using the overall collected data. Finally, Section 5.8 concludes on this chapter.

5.2 Motivation

Scientists and decision makers need to collect some insightful data with the best quality-cost ratio, both in the collected data and the user's privacy.

Although some of the related works have proposed their own way to collect the WiFi places, they lack some of the important information, like the privacy preservation, the collection cost, how this collection can be run and how the *WiFi places* information is perceived per real users. In this section we show some examples that motivates the use of our framework.

Proximity detector. Some information like detecting if two users are in proximity of each other can require collecting a lot of data. Using the WiFi places will simplify getting this information, by simply comparing the *place_ids* of the two users and see if we have a match at the same moment of the day. Another motivating example is to find if our users have passed by the same place or not, this could be useful if this place do not have an Internet access, so next time, other users will be notified that they are going to lose Internet access.

Because of the opportunistic way of finding WiFi APs, we believe that we need to perform an analysis study on the number of WiFi APs that can be found, and if they can cover some of the user important places most of the time or not.

The user routine. One of the most interesting information to capture about the user is if she is on routine or out of routine. When the user is out of routine, she will not be on auto pilot anymore, and she will need more assistance from the used apps. Detecting if user is on her routine or not, can be done with the WiFi places. Our framework will return the current user WiFi place and the time the user has spent in this place. This will give the app the ability to construct a user timetable. When the user is out of routine; the framework will return a new WiFi place or a known place but on a different time of the day. This can be built on top of our framework, by trigging an event each time the user is out of routine.

Get the user Points of Interest (POIs). Most of the GPS collected data will be used to extract the POIs of the user to detect the user routine. For that, most of the mobile apps use a cloud service to offload the computation from the mobile device to the servers. Sometimes, all of the user's sensitive location data are sent to the server just to calculate these POIs. We argue that these user's sensitive location data must remain on the user mobile device. To this end, we can use the WiFi places data, that automatically produces the user's POIs (user places), without the need for expensive computation or exposing the user's sensitive data to third party adversaries. Moreover, the WiFi places are more precise than the GPS [118], this can give us more place granularity, especially in indoor places.

Place shareability. On Android we can find the fences API [1], that provides a way to pin some places, and notify your app whenever the user passes by one of these places. This can be useful for some use cases, but mobile apps would need a more access to the user context, for example, getting a timetable of the user activities to help the apps find the right moment to notify the user about any place dependent event.

Sharing the user current place with the used app can be pretty sensitive, but sharing a temporary identifier for this place can help both the user's privacy and the app. This place tag sharing can be handled at the OS level, so these apps will not have to ask for WiFi scan permission which gives access to a list of surrounding WiFi APs. Instead, the OS can give the app a temporary place identifier that can be managed by the user. This temporary place identifier, can help the app behave differently in different places. Instead of getting some semantic information about the place, which can be harmful to the user's privacy, the app will only receive a place identifier and it is up to the app to add new semantics and adapt to the this place identifier. For example, if the user tend to use the app in the place with "5" as the *place_id*, the app may want to send a notification to the user to remind her to use the app. Each time the user passes by this place, the app will receive an event with the *place_id* "5".

Multiple access level. The user will manage the different data access levels using the WiFi *place_id*, which can provide a unique *place_id* per app or per user. The WiFi *place_id* can also be temporary *place_id* or shareable *place_id* by every app and every user using this app. This multi-level data access provides more control over the user's privacy, for example, if the user feels okay to share the information that she is in the same place with another user, then, her shared *place_id* can be unique between all the users of this app, but this will not be the same *place_id* that is provided to other apps, so the user will not be tracked across different apps. This context id will help the app provide more context based services, which can increase the quality of the user experience, and can help make a lot of decisions on behalf of the user, like detecting when the user is at her home or detecting when the user is at new place.

Environment Awareness. Detecting the current user WiFi places can give the app a lot of information about what the user is doing. This information can be correlated with the user real activities, like sleeping at home or working at the work office, these two activities can be easy detected by the WiFi places framework. Some of the questions we are interested to find answers to in our apps are: where does the user use our app? where has this user installed our

[1]https://developers.google.com/awareness/android-api/fence-api-overview

app and where does the user tap on the notification of our app? Answering these questions can be done using our WiFi places framework.

Place properties. Some collected data need a timestamp and a place identifier, this place identifier can be used to aggregate all of the data collected in one place and on its premises. This can help the app predict the user next WiFi places and all of the next place's properties using the already collected data about this place, like the Internet connection quality, the social presence, the user activities (accelerometer), the user behavior (on the app) and other properties that can be collected and assigned to each WiFi place. These WiFi places can help form the user routine which gives us a practical way of aggregating data based on the WiFi place, which, in most cases follows the user activities (one different activity per place and time).

Predictive Buffering. Our framework can help streaming services provide more predictive buffering, as we can thus assign each place with the corresponding Internet quality; the stream buffering can start before reaching this restrained connectivity place. This WiFi place's Internet quality property can be shared between users, so another user who had never been there can get that place connectivity information. For example, when the user is entering a subway station, the streaming app will start buffering a lot of data, because based on other users, they tend to lose connection when they get underground. This can be done in background and the user will not notice this connectivity issue.

5.3 User WiFi Place

In this section, we show how WiFi places are categorized to grab the user context at best, and how they are built using WiFi scans.

5.3.1 WiFi place identification

Whenever a WiFi Scan is received, the framework will decide if this is a new place, a known place or the user is just moving.

Known place. A known place is found in the local database, this is the case for the already created places.

User is moving. If no known place was detected than the user is considered moving.

New place detection. Whenever a place is recurring for more than $5mn$, then, a new place is created.

5.3.2 User Place Building

The user WiFi place is built using the SimHash similarity algorithms [181] which generates a hash of the list of WiFi Access Points, the time of the day and the user stay period. We can also add some data that depend on the user and not on the place like the accelerometer or the light sensor. Other user independent data can be the pollution or the noise sensors. These user independent place data can be shared between multiple users as they can add additional important information to these users. Some of the user dependant place data can be shared between users, as these data can add some insights or predictions that can be used by other users, like the used apps, the phone use and the social presence.

5.4 Shared WiFi Places

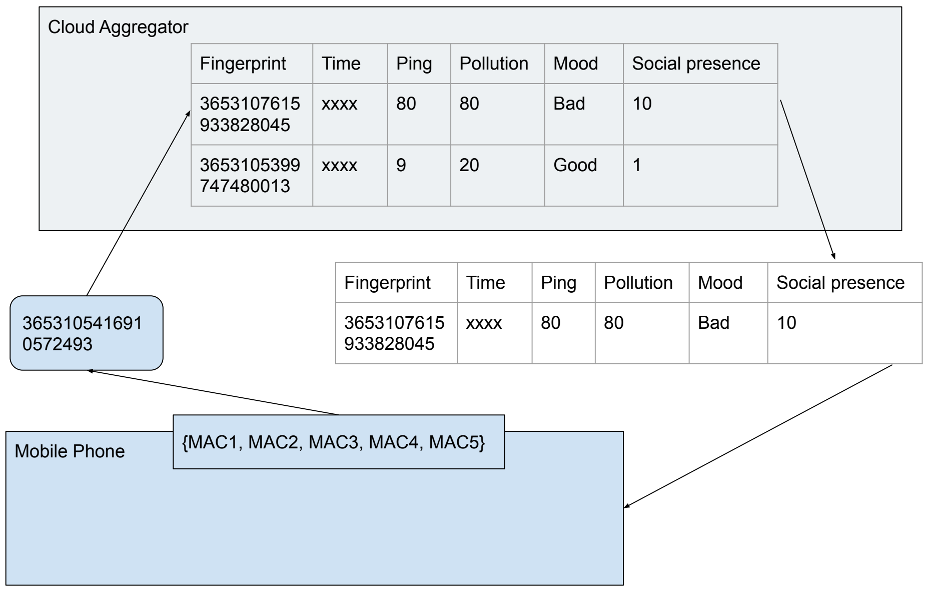
The architecture of the system is presented on Figure 5.1. The users will be sharing their places along with service metrics collected on that place. These *place_ids* can be shared between the apps of the same device. These metrics can for example carry the user preferences, the user timeline, the air quality, the Internet quality, the average mood, social presence and other app's specific metrics. These *place_ids* can be also shared between the user's different devices on real-time, so for example: the Tablet can be brought out of doze mode just because the user's mobile phone is entering the home *place_id*.

The centralized service will work as a middleware service between users of the same app, this service will store all the mappings between the *place_id* (without the user ids) and the service metrics, these users can receive other users service metrics, when entering to that place.

Because the calculated *place_id* is independent from the user (it does only depend on the WiFi APs), this *place_id* and other service based metrics can be shared without disclosing any personal information about the user.

5.4.1 Building atop of the WiFi Places

This *place_id* can be shared with other apps, so these apps can build smart services that take into consideration the user's place. Some other services can be integrated to facilitate the work with places, for example: next context prediction, user timeline and preferences handling. Some other plugins can be added, like social profile detection and users' interaction with other users.



Fingerprint	Time	Ping	Pollution	Mood	Social presence
3653107615 933828045	xxxx	80	80	Bad	10
3653105399 747480013	xxxx	9	20	Good	1

Fingerprint	Time	Ping	Pollution	Mood	Social presence
3653107615 933828045	xxxx	80	80	Bad	10

Fig. 5.1 Overview of the cloud aggregator.

5.4.2 Extending the WiFi Places

For the outdoor contextualization or where there is no WiFi APs, GSM *Cell_id* can replace the MAC Address in the place recognition algorithm. This can provide a temporary replacement place in the offline mode.

GPS location data can also be used to provide a *place_id* in the absence of WiFi APs, these GPS positions can be aggregated using geo-spatial clustering methods [33, 170]. Other methods can be used, like the Open Location Code (OLC)[2] method that can be used to replace the MAC address in our place recognition algorithm.

5.5 WiFi Places Overview

In our work, the user place is defined by the WiFi fingerprint created using the WiFi scan at that place and the time interval the user has spent in that place.

Algorithm 4 WiFi Places recognition algorithm.

```
procedure OnNewScan($scan$, $timestamp$)
    $\langle tag, sim \rangle \leftarrow$ FindTag($scan$)                    ▷ Find the tag in the database
    if $tag <> lastTag$ then
        SaveHit(tag, timestamp)                    ▷ save this hit in the database
        $lastTag \leftarrow tag$
        $start \leftarrow timestamp$
    end if
    if $tag <> NEW\_PLACE$ and $sim < SIM\_TO\_EXPAND$ then
        ExpandPlace($scan$)                    ▷ Create a place expansion
    end if
    if $tag = NEW\_PLACE$ and $timestamp - start > MIN\_WAIT\_TO\_CREATE$ and $sim >$
$SIM\_TO\_CREATE$ then
        CreateNewPlace($scan$)                    ▷ Create a new Place tag
    end if
end procedure
```

5.5.1 WiFi Places Recognition Algorithm

The Algorithm 4 presents how we identify the current place based on the received WiFi scan. Whenever a WiFi scan is received, we search in the device database for a similar WiFi scan, a place tag corresponding to that WiFi scan is returned. If the returned similarity is less than *SIM_TO_EXPAND* (default is 0.30), then, a new place expansion is created. If no similar WiFi scan is found, then, the *NEW_PLACE* place tag is returned. If we were receiving this *NEW_PLACE* tag for more than *MIN_WAIT_TO_CREATE* seconds (default is 10 minutes) and the similarity is greater than *SIM_TO_CREATE*, then, a new place is created.

These parameters (*SIM_TO_EXPAND* : 0.30, *MIN_WAIT_TO_CREATE* : 10 minutes) were tuned using a controlled experiment, where we tried to relaunch the place recognition algorithm with one user's data over one month with different values. The user tuned these parameters to what it seemed to represent her timeline at best.

5.5.2 The similarity function

The Algorithm 5 shows our implementation of the Simhash algorithm. We use a modified version of the Simhash algorithm [181], that has been optimized for our use case. In particular, the words in the Simhash algorithm correspond to each pair of digits in the hexadecimal MAC address format. The procedure *GetHash* is used to get the hash of a scan (the fingerprint), this is the value that will be stored in the local database or shared with other users. To calculate the similarity, like in the Simhash algorithm, we only have to count the number of the shared

[2]https://github.com/google/open-location-code

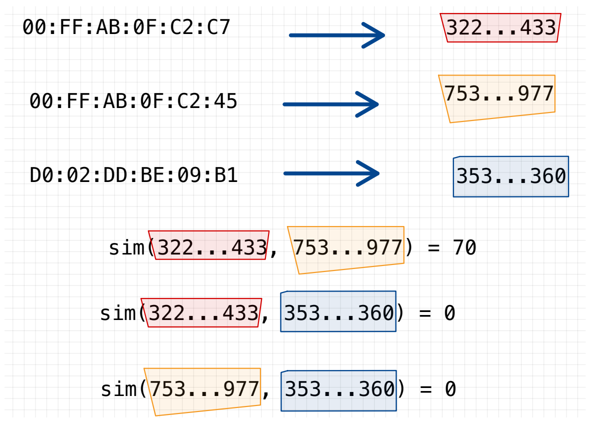

00:FF:AB:0F:C2:C7
322...433
00:FF:AB:0F:C2:45
753...977
D0:02:DD:BE:09:B1
353...360
sim(322...433, 753...977) = 70
sim(322...433, 353...360) = 0
sim(753...977, 353...360) = 0

Fig. 5.2 Overview of the Simhash usage.

bits between the two hashes to get the similarity. Because most of the found similarities are between 0.50 and 1.0, we chose to expand our similarity range, and only consider the values between 0.50 and 1.0 as a the similarity value.

Figure 5.2 shows how we can use our modified Simhash algorithm to calculate the similarity between two scans using only the hashes of these scans. This means we do not need to keep the original scan, but just a hash of that scan, and then, we can calculate the similarity between this place and the places on the local database.

5.5.3 Place notification

Whenever a different place is detected, the target app will receive a *PLACE CHANGED* notification (an intent broadcast in Android). Using the *place_id*, the app can react with the predefined routine.

5.5.4 Place timetable

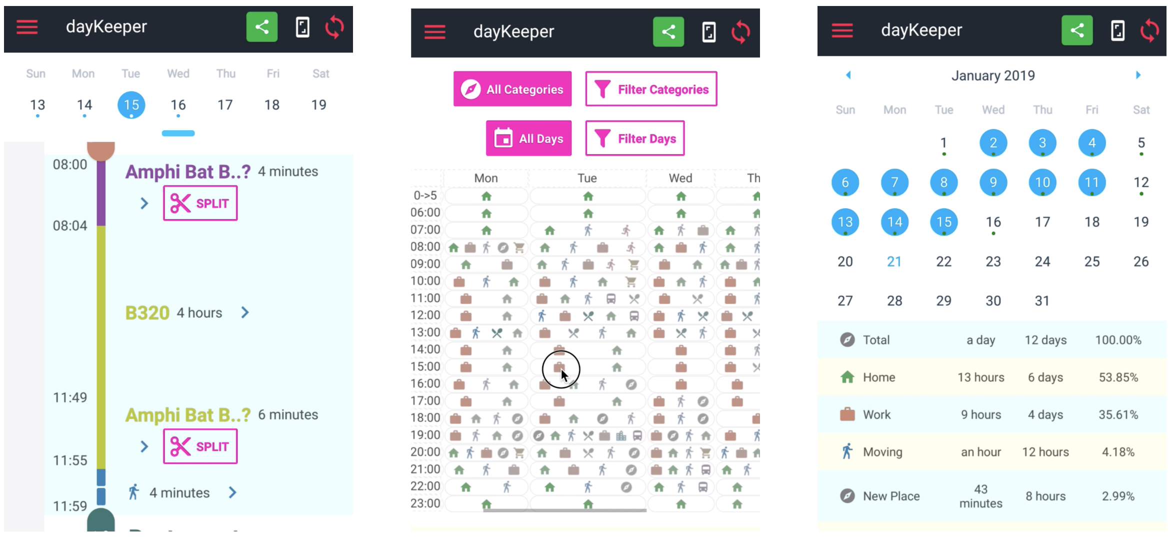
The framework can provide a timetable with user history, this timetable contains events of place changing, the timestamp along with the corresponding *place_id*.

5.6 Data Collection

Figure 5.3 contains an overview of the DAYKEEPER Android app. The DAYKEEPER Android app, was used to asses our algorithms and to collect some useful data to use them as input dataset to our controlled experiment. This helped further prove the utility of our framework. The DAYKEEPER Android app is also our use case to show and motivate our framework usage

Algorithm 5 Simhash modified algorithm.

```
procedure GETHASH($scan$)
    $value \leftarrow MASK$                              ▷ Binary mask used to to sum weights
    for all $\langle bssid \rangle \in scan$ do
        $list \leftarrow$ SPLIT($bssid$, ‘ : ‘)
        for all $element \in list$ do
            $hash \leftarrow$ MD5($element$)
            $value \leftarrow$ BINARYSUM($value, hash$)    ▷ Special Sum used in Simhash
        end for
    end for
    return $value$                                       ▷ $data$ discarded by LPPM
end procedure

procedure SIMILARITY($scan1$, $scan2$)
    $value1 \leftarrow$ GETHASH($scan1$)
    $value2 \leftarrow$ GETHASH($scan2$)
    $differentBits \leftarrow$ XOR($value1, value2$)
    $distance \leftarrow$ BITCOUNT($differentBits$)
    $sim \leftarrow 1 - distance/128$          ▷ Our MD5 algorithm returns hashes with 128 bits
    if $sim < 0.5$ then
        return 0
    end if
    $sim \leftarrow (sim - 0.5) * 2$
    return $sim$
end procedure
```



Total	a day	12 days	100.00%
Home	13 hours	6 days	53.85%
Work	9 hours	4 days	35.61%
Moving	an hour	12 hours	4.18%
New Place	43 minutes	8 hours	2.99%

(a) Daily view (b) Weekly view (c) Monthly view

Fig. 5.3 The DAYKEEPER app overview

with real-time user contextualization, and what can be done with the valuable information we can get from it.

The DAYKEEPER Android mobile app was deployed on the Google Play store, the only incentive for the users to install the app is the automatically generated timeline for the user activities. The users could see the generated timeline and accept or modify the suggestions. Because no incentives were provided except for the utility of the generated timeline, if our app was consuming battery or generating wrong timelines, the user would uninstall the app directly.

Most of the users had accepted the data collection request, especially for the WiFi scans. Some users decided to only share a hashed MAC address instead. We held an IRB authorization to collect data using the Apisense platform [79]. The DAYKEEPER app provides the user UI, while it uses the Apisense platform collection service that runs in the background, to collect the data and upload it to the server. The app receives the WiFi scans from the Apisense service and uses the WiFi places Android library to generate: *i)* an interface (cf. Figure 5.3a) to show the user the current place, *ii)* an interface (cf. Figure 5.3b) to present a weekly timetable of the user places and *iii)* an interface (cf. Figure 5.3c) to summarize the user monthly places.

The anonymously collected data contains: WiFi scans, the ping average to *google.com*, the cellular cells ids, the battery level, the timestamp and the screen status. These data were collected every $5mn$, and every $30mn$ when the phone enters the doze mode (not used). The data collection took 1 month and involved more than 30 users.

In our experiments our users have detected $58,500$ unique WiFi Access Points (APs). These WiFi APs were used as antennas not to locate but to contextualize our users. In one day, one user visits an average of 10 places. Our users have captured an average of 1950 WiFi Access Points in one month.

5.7 Results

To evaluate the WiFi places recognition algorithm, we focus on the following research questions:

RQ1: Can we share the WiFi places between users while preserving both their privacy and the data utility?

RQ2: Using WiFi scans, can we detect if the user is moving between places or not?

RQ3: Can we have a distance between the WiFi places?

5.7.1 Privacy Attacks

Link WiFi MAC Addresses to GPS locations

Most of the WiFi fingerprinting methods use plain text MAC addresses to link the user's WiFi scan with a GPS location. This can be done using a crowdsourced dataset that contains a key-value pair of MAC addresses and the corresponding GPS location [118].

In our case, we only provide a hashed version of a group of MAC addresses. Hashing one MAC address is not enough to preserve the user's privacy, as a rainbow table can be created for all the hashes. In our framework, we hash at least 5 MAC Addresses as input, which can add a lot of calculations regarding the order and the number of the MAC addresses used as input. To enumerate all the possible hashes, a rainbow table has to be created for $240bits$ $(5*48bits)$ with only $128bits$ hashes, which means we can get some collisions. Each entry in the rainbow table can point to more than one WiFi fingerprints which add more difficulty to the adversary.

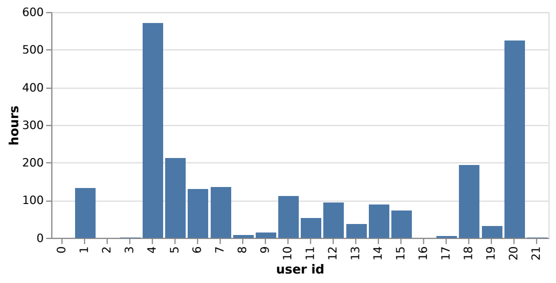
Identify each user's data

Because GPS locations are not resistant to clustering, so each user can be spotted out. This can be done using a simple clustering algorithm like the ST-DBSCAN [42].

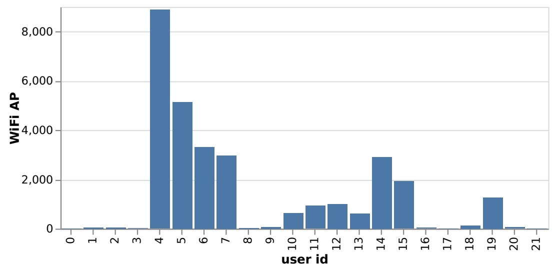
However, these clustering algorithms require a similarity distance between different places. In our case, the similarity works only between the traces of the same WiFi place, so the adversary can link one place's data, but they can't link one user's different places together, as there is no similarity between different places.

Synthesis

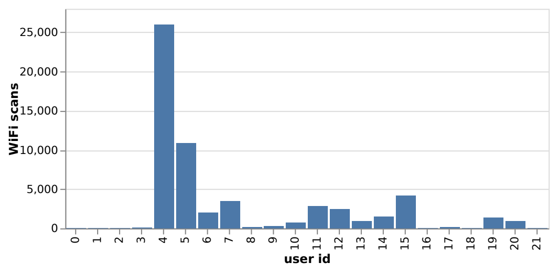
Compared to the GPS mobility dataset, attacking the WiFi places data, is like attacking the users' POIs created from a GPS dataset after being aggregated and the GPS locations being removed from it, which is considered anonymized and can be published online. The WiFi places data itself will not reduce the privacy of the user, but other data that is collected along can present some privacy issues. For example, if the battery level is collected with each WiFi place, this data can be used to group all the places of one user together. To this end, we can use FOUGERE to integrate privacy preserving mechanisms that can deal with different kinds of data. This answers the first part of our **RQ1** regarding the privacy preserving property for WiFi places, we can say that WiFi places preserve the users' privacy.



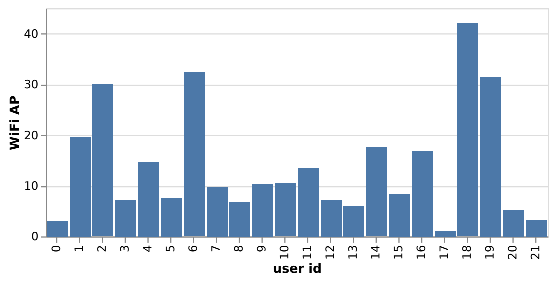
(a) Number of hours contributed per user



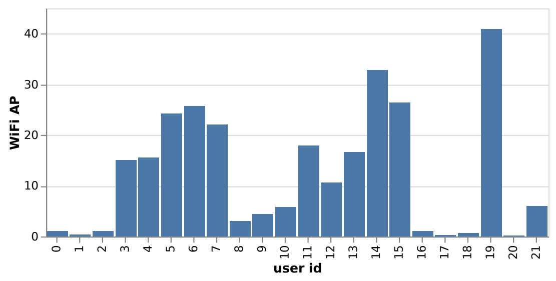
(b) Number of unique WiFi Access Point per user



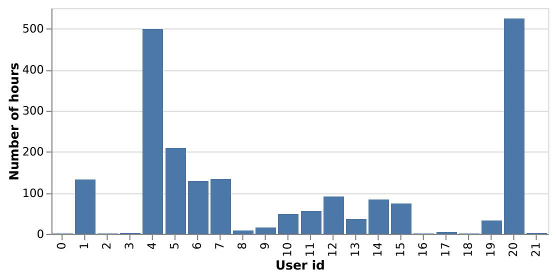
(c) Number of WiFi scans per user



(d) Average number of WiFi Access Points per scan per user



(e) Average number of WiFi Access Points per hour



(f) Sum of Stay per User.

Fig. 5.4 Experiment statistics

5.7.2 WiFi scans analysis

Figure 5.4, present some statistics of our collected data. We filtered the data of 9 users who did not upload an important amount of data. These users may have chosen not to upload their anonymized WiFi data. Figure 5.4a shows the number of hours contributed per user. As users come and go and can activate or deactivate the data collection, the data contributed by each user differ in size. Figure 5.4b shows the number of unique WiFi AP scanned per user, these numbers can reflect on the user being moving or staying fixed, the more the user moves, the more WiFi AP her device can scan. Figure 5.4c shows the number of WiFi scans per user. Figure 5.4d shows the average number of WiFi APs per WiFi scan for each user. Figure 5.4e shows the average number of WiFi APs per hour for each user.

These figures give us an idea if our users are moving or stationary, they also can reflect on the moving rate of our users. For example, we see that the user with the identifier 20 was stationary, as the device contributed a lot of hours (the $95th$ percentile) (figure 5.4a), but with only 57 unique WiFi scans (the $40th$ percentile)(figure 5.4b) at an average of 0.10 of WiFi AP per hour (the $4th$ percentile) (figure 5.4e). The number of 949 WiFi scans for this user (the $59th$ percentile) does not really reflect the number of contributed hours (the $95th$ percentile); this is due basically to the device entering the doze mode. Android devices tend to enter the doze mode whenever they are not being used to optimize the battery consumption, thus, limiting the number of WiFi scans.

Synthesis. To answer **RQ2**, we can clearly see that by using WiFi scans we can identify if the user is in moving state, as the device receives different WiFi scans, or in a stationary state, as the device keeps receiving the similar WiFi scans. Also, our WiFi place recognition algorithm uses the same technique to tell if the device is in a moving state, in a known place or in a new place.

5.7.3 Evaluation metrics

In this section we present, for each experiment variant, how we calculate the metrics, on which we base our results.

- **Overall stay**, represents the overall number of hours of places recognized by our algorithm for all of our users.
- **Max Stay**, represents the maximum number of hours of places recognized by our algorithm one user can get.
- **Max Occurrences**, represents the maximum number of occurrences of one place one user can get.

Table 5.1 Experiment metrics

Variant	Overall stay	Max Stay	Max Occurrences	Places with > 1 hour	Max Places detected
Jaccard	2077	524	3684	58	22
Simhash	2053	517	4072	55	47
Jaccard-expansion	1799	524	3684	67	315
Simhash-expansion	1685	505	3842	62	259
Shared places	2022	524	3684	59	23
10 APs	2001	524	3772	59	30
5 APs	1958	524	2192	62	41
1 AP	1847	524	1776	61	233

- **Places with > 1 hour**, represents the number of places recognized by our algorithm for more than one hour.
- **Max Places detected**, represents the maximum number of distinct places one place one user can get.

5.7.4 Shared places

In this section, we run our place recognition algorithm with each variants. Table 5.1 report on the results of each variant:

1. **Jaccard**. This is the default variant. In the Vanilla variant, we use the Jaccard similarity, we do not apply the place expansion property and we use all the WiFi APs in the WiFi scans as input.
2. **Simhash**. We can see the results when using the Simhash algorithm. We do not provide plain text MAC addresses, but only the Simhash of the list of MAC addresses as explained in Figure 5.2. We can see that we have similar performances compared to the Jaccard variant. The only different metric is the *Max Places detected* for one user, which differs because the Simhash used similarity is less sensitive than the *Jaccard* similarity, so it can generate more places.
3. **Shared places**. In this variant we share the detected places between users, and instead of using a local database for known places, we use a cloud aggregator (Section 5.4). We can see that sharing the other users' places does neither impact nor improve the performances of the place recognition algorithm. But it can help identify the same places between users which can be used to share other metrics between users.

Synthesis. To answer **RQ1**, regarding the utility part, we can see that the *Simhash* variant has similar performances compared to the default variant (with *Jaccard* similarity), which means in terms of performances we did not lose much utility due to the privacy preserving of *Simhash* use. Eventually, we have dismissed some data (like the MAC addresses), but these data are not important to contextualize the user. We can also see, that sharing the WiFi places between users do not change the performances of the WiFi place recognition algorithm, but can help share other important metrics between them.

5.7.5 WiFi scan size

In this section we consider 3 variants; each one is designed to provide us with analysis regarding the size of the WiFi scan we give as input to our WiFi place recognition algorithm. Table 5.1 shows the metrics for each variant:

1. **10 APs**. In this variant we only take the first 10 APs from the WiFi scan. We can see that we do not lose much in terms of performances.
2. **5 APs**. In this variant we only take the first 5 APs from the WiFi scan. We can see that we are starting to lose some performances. In particular, the number of *Max Occurrences*, which drops to 2192 compared to the default variant 3684. This can result of some places not being recognized.
3. **1 AP**. In this variant we only take the first AP from the WiFi scan. We can see that wre are starting to lose some performances. In particular, the number of *Max Occurrences*, which drops to 1776 compared to the default variant 3684. This can result of more than half of the user places not being recognized. At the same time, we can see that the most important places (*Places with > 1 hour*) are still recognized with almost the same as the default variant.

Synthesis. To answer the second part of **RQ1** regarding the data utility of WiFi places, we can say that our place recognition algorithm can use less WiFi data as input and it can keep almost the same utility. The performances can degrade a bit, but the user's most important places (*Places with >1 hour of stay*) are still collected.

5.7.6 Place expansion

Table 5.1 contains the metrics collected for each of the following variants.

1. **Jaccard-expansion**. In this variant, we use the place expansion property. Algorithm 4 considers a place expansion whenever the similarity is less than 0.30 and a new place expansion is created. We can see that the *Max Places detected* metric is very high and this is due to the number of expansion each place can have.

2. **Simhash-expansion**. In this variant, we use the Simhash variant with the place expansion property. We can see that the *Max Places detected* metric is also very high, while other metrics are almost like the default variant (Jaccard). This confirms that the similarity of the Simhash algorithm can accept the place expansion property as the generated similarity, as it can vary between 0.30 for the nearby places.

Synthesis. To answer **RQ3**, we can see that we have the distance property for both variants, the default one (**Jaccard-expansion**) and the **Simhash-expansion** variant using the privacy preserving solution. While the similarity distance is valid between the data of the same place (building), which is ensured by our algorithm, there is no similarity between different places, as their WiFi scans and Simhash differ. Finally, we can say that our WiFi place recognition algorithm preserves the distance between the data of the same place, but not for different places.

5.7.7 Discussion

We can see from the results in Table 5.1 that our algorithm accept different variants of parameters without losing much in performances. Each variant can be useful for some use cases. The Simhash variants preserves the privacy of the users. The expansion variants can be used when we want a similarity distance between nearby places. The shared places variant provide a way to share places between our users. The first *n* APs can be used when we do not have access to all the WiFi scans. For example, in iOS, we can only get the current WiFi network. While we lose some details, we still can recognize most of the users' important places.

5.8 Conclusion

In this chapter, we provide an automatic real-time user contextualization based on the WiFi places. We reported on the creation the WiFi Places, on its usefulness. We motivated its usage with examples of real scenarios that can be captured by this place. We have studied the properties of this WiFi place and we have elaborated on what makes this WiFi places so useful in capturing the user place. The WiFi places can be used to be shared between the users of the same app while preserving their privacy.

We believe that if a more broad experiment is launched with more users and more data, it can further prove our results. Other types of data can be also collected to infer more information about the user WiFi places, and to confirm the utility of our framework.

In the next chapter, we conclude our thesis, we summarize our contributions and present our perspectives in the short and the long term.

Chapter 6

Conclusions and Perspective

With the progression of mobile devices' capabilities and the continuous need for real crowd-sourced data to fuel the artificial intelligence solutions and to provide the best possible user experience, users remains kept out of data acquisition process, possibly impacting their privacy. Furthermore, privacy policies are just imposed on the users who are in the first place the owners of these data.

In this thesis, we address this issue by proposing a solution that puts the user in the center of crowdsourced data acquisition solutions. While the P2P privacy-preserving proposed solutions lack in most of the time real deployment or extensive testing, we propose a testing framework for our P2P privacy-preserving solution, we evaluate and generalize our testing solution beyond our case study to give other developers and scientists a framework to test other mobile P2P-based solutions.

This chapter summarizes the main contributions of this thesis and gives some hints on our future directions and perspectives.

6.1 Contributions

The main goal of our thesis is to help the scientists collect anonymous data about the user, we want to put the user in the center of privacy control. We want to provide a robust mobile peer-to-peer privacy preserving solution and a robust and reproducible testing environment for it, thus, testing all the P2P mobile apps. Our main contributions are:

A user location privacy preserving solution in mobile crowdsourcing platforms. FOUGERE helps to preserve and increase the user's privacy by exploring the proximity of these users in order to confuse the adversary on the server side. We give the user the control over her privacy, she can control which privacy guarantees her collected data have to maintain. The user will disseminate anonymized data from her mobile device using nearby P2P communications, we apply LPPMs on each dissemination. Our adversary model suppose that, once the data leaves the user device, it can be intercepted by any adversary. We presented how FOUGERE can outperform the famous location privacy attacks with little impact on the data utility.

A WiFi places data collection framework. To complement FOUGERE, in indoor environments, we propose to collect WiFi places to better capture the user context. We study the creation and the sharing of the WiFi places in a way that preserves user privacy and puts the user in control of which access level she wants to grant to the app. We presented how our algorithm can be tuned to meet certain types of requirements and use cases. As input, our algorithm can use few WiFi APs, this proves that our algorithm can work in poor WiFi APs covered places and with OSs that only expose the current WiFi AP that the mobile device is connected to. To preserve the user privacy, our algorithm uses a hashed version of the WiFi *place_id* to increase privacy when sharing the WiFi places with another user.

A nearby P2P testing framework. We also deliver a nearby P2P testing framework for mobile apps and crowdsourcing platforms that supports nearby P2P communications (*e.g.*, WiFi, Bluetooth). While our testing framework helps to test these apps at scale, our framework can also guide the tuning of the parameters of P2P privacy-preserving solutions before deploying them in the field.

Our solution therefore ease the maintenance of existing mobile apps and reduce the cost of creating new nearby P2P mobile apps. We extend the syntax based on existing black box testing frameworks, like Calabash, to describe P2P scenarios like out of band data exchanges. Our testing framework helps the developers to detect bugs *a priori* and to eventually tune P2P parameters when scaling the deployment up to 500 emulated devices.

To conclude, we believe that our contributions can help the developers, the data collection campaign managers and the research community. In particular, our contributions:

- Improve the user privacy, while maintaining the data utility,
- Give the users more control over their privacy settings,
- Help the scientists to collect a useful user context with WiFi places,
- Preserve the user privacy when sharing the WiFi places between them,
- Detect if the user is moving between WiFi places or not,
- Provide a distance between the data of the same WiFi place,
- Improve the quality of P2P mobile apps,

- Reduce the deployment time by providing a testing environment for P2P parameters tuning.

6.2 Perspectives

6.2.1 Short-term Perspectives

In this section we will present the immediate venues for extending our work.

Include and adapt more LPPMs into FOUGERE. In FOUGERE, we have adapted some LPPMs (cf. Section 4.3). While new LPPMs are proposed along years, we need to keep our LPPMs up to date with the recent privacy attacks and adapt new LPPMs when possible. We plan maintain an overview of state-of-the-art LPPMs to include them in FOUGERE. As part of this process, we intend to investigate how to give the user more privacy control through the potential parameters of the LPPMs by working on the simplification of such parameters, whose implications tend to be hard to interpret for end-users.

Deploy FOUGERE in a collection campaign. So far, we have designed the PEERFLEET framework to assess the privacy performance of FOUGERE. The next step is to deploy FOUGERE in real-life usage, and study its performance with real users. This will help:

- To provide an understanding of the user control on her privacy,
- To evaluate the battery consumption and overhead when using FOUGERE,
- To enforce the privacy of participating users,
- To motivate the use of FOUGERE and encourage other scientists and mobile apps to do the same.

Extend FOUGERE to *Location-Based Services* (LBSs). For the moment, FOUGERE can be used with crowdsourcing campaigns, we can use the same logic as in Section 2.1.1, where we presented some of the state-of-the-art works that propose to use collaboration between users to hide from the *Location-Based Services* (LBS) server. We plan to use the same mechanism and extend FOUGERE to the LBSs. This will help to keep the user in the center of privacy handling, the user can decide on how much data she is willing to share with the LBS server and how much data utility she is willing to lose in favor of her privacy.

Collect more data for WiFi places. In our work, We run our collection campaign for only 2 months, we want to find incentives in order to find new users and keep existing users engaged and participating to the data collection. Collecting more data will give us more data to analyse and infer from. This can further prove our results and improve our algorithms. The next step will be to identify correlations between the WiFi places and other user's activities. For example, can we find a dependency between the WiFi places and the user mood? The social presence and the apps the user is using? A dependency between the WiFi places and other device sensors will help prove our results and help improve our place recognition algorithm.

Use the same place recognition algorithm on different types of sensors. While WiFi data are available most of the time, sometimes, the user can be in places where there is no WiFi APs. To deal with that, we can use the GSM *cell_ids* instead of WiFi APs MAC addresses, this will give us a user coarse place, which reflects on the rural areas where we do not really need a good precision. Another type of data would be the GPS location, where we can use these GPS locations to generate the *place_id*, which can help keep the same data type and allow us to apply all of the algorithms we have built on top of WiFi places. We plan to study the utility of these additional data sources, and compare them in terms of utility and cost. In particular, we intend to see if the generated places from these data sources can reflect the user daily activities or not.

Infer more data from the WiFi places. The collected WiFi places can give us a lot of knowledge about the user activities. To this data, we can also add some important metrics: air quality, social presence, Internet quality and other important metrics. As discussed in the motivation (cf. Section 5.2), these metrics can help provide useful services that make the user experience better. The WiFi places can be used to aggregate these data with the user natural activities. We plan to see if there are any correlations between the user context as defined by the WiFi places and other collected metrics. If these correlations exist, then, this will help us to understand more about the user activities and her environment. This also means that, if we can predict the next WiFi places —given that the metric correlates with the WiFi places—we can also predict that metric. For example, if we find a correlation between the user mood and the WiFi places, then, we can predict the next user mood based on the next WiFi place.

Provide documentation and guidelines for developers to test their P2P mobile apps. During our evaluation of the PEERFLEET framework, we tested some mobile apps in order to find if our framework can leverage bug detection or not.

The next step in this area will be to capture the knowledge we obtained about these bugs to document them. Also, scaling these apps can present some technical and performance issues. In the OS APIs documentation, developers cannot find any guidelines on what are the best practices and issues to care about when developing such apps. As we have observed in Table 3.1, developers of such apps are still having troubles shipping working apps. In our contribution, we provided a testing framework that helps developers test their apps before deploying them. We believe that developers still need some help from the research community. In particular, documenting scalability bugs and issues to provide some guidelines and best practices in handling peer discovery, connection management and resource allocation can be of valuable help for them. Performance bottlenecks can also be documented, for example, we plan to compare the results of two different executions in terms of performance and trace back the source of performance degradation, abd then, cite this performance issue as caveats to the developers using the API.

6.2.2 Long-term Perspectives

Study the user UX comprehension of the privacy configurations. In FOUGERE, we put the user in the center of our privacy preserving collection system. With a simple configuration UI (cf. Figure 4.2), the users can choose and tune their privacy parameters with what better suits their comfort. Some research work have been done to find easier configuration for the LPPMs based on the utility and privacy [182] and to adapt the LPPMs configuration to each single user [68]. Some research questions that remains open are:

- Does the user understand these privacy settings?
- Will the user put all of parameters to the extreme values in order to preserve her privacy?
- What kind of incentives would help the user to share more data with stakeholders?
- What are the values that suits the best the user privacy and the data utility?

Some gamification methods can be used to engage more participation, but all of these methods are based on user incentives. These incentives can be of different kinds, like monetary, privacy guarantees (the anonymization solutions) or service-based incentives. Not to mention, what kind of incentives would be ethically correct? At least, can we make sure that the participating user understands the consequences of her settings on her privacy when she is sharing her personal data? And, once the user had understood these consequences, would we expect her to understand the protection mechanism and trust our anonymization solutions?

Predict the user next WiFi places. While contextualizing the user using WiFi places delivers a good user context information that can be used for different kind of applications, adding the next place prediction would be a great addition to the user context. In particular, we believe that this would help to bring the user quality of experience to another level. This can also help to reduce the resource consumption and manage the user streaming capabilities. In Section 5.2, we explained how a streaming service can benefit from the user next WiFi places prediction. In particular, when using WiFi places, an app gets the next WiFi place of the user, this app can prepare its content and services to adapt to the settings of the new environment beforehand. For example, if the user is entering a place with no Internet access, the streaming app can start buffering as much data as it could just to provide the best user experience, and this Internet connectivity issue would go unnoticed for the user. Next place prediction can also improve smart home solutions, by knowing that the user is going to be in her house in advance. This can help reduce resource consumption, for example, the house can only be heated when needed, the lights will be turned on only when the user is approaching the house. Along with reducing the resource consumption, the user will get a better user experience in getting a lot of tasks automated.

Extend P2P testing to other devices. Because the mobile nearby P2P communication technologies are standard protocols, they can work with a variety of devices that support wireless communications, like: the car stereo, wireless printers and *Internet of Things* (IoT) devices. We plan to include these devices to be tested in our nearby P2P testing framework. This will help provide a robust mobile apps that most of the time fail to perform a simple pairing or connection task. When the developers are willing to test new communication protocols to improve the user quality of experience, they can just use our nearby P2P testing framework, this should help them improve the existing wireless solutions and propose new solutions, especially with the emerging of IoT devices and smart cities.

References

[1] Wenjuan Tang, Kuan Zhang, Ju Ren, Yaoxue Zhang, and Xuemin Sherman Shen. Privacy-preserving task recommendation with win-win incentives for mobile crowdsourcing. *Information Sciences*, 2019.

[2] Sonia Ben Mokhtar, Antoine Boutet, Louafi Bouzouina, Patrick Bonnel, Olivier Brette, Lionel Brunie, Mathieu Cunche, Stephane D 'alu, Vincent Primault, Patrice Raveneau, Herve Rivano, and Razvan Stanica. PRIVA'MOV: Analysing Human Mobility Through Multi-Sensor Datasets. In *NetMob 2017*, Milan, Italy, April 2017.

[3] Rachit Agarwal, Shaan Chopra, Vassilis Christophides, Nikolaos Georgantas, and Valérie Issarny. Inferring Context of Mobile Data Crowdsensed in the Wild. NetMob 2019 - Conference on the scientific analysis of mobile phone datasets, July 2019. Poster.

[4] Mahesh K Marina, Valentin Radu, and Konstantinos Balampekos. Impact of indoor-outdoor context on crowdsourcing based mobile coverage analysis. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 45–50. ACM, 2015.

[5] Yung-Ju Chang, Gaurav Paruthi, Hsin-Ying Wu, Hsin-Yu Lin, and Mark W Newman. An investigation of using mobile and situated crowdsourcing to collect annotated travel activity data in real-word settings. *International Journal of Human-Computer Studies*, 102:81–102, 2017.

[6] Eric B Blancaflor, Jay Mark T Butalon, Patrick Eugene S Pascual, Bryan Angelo U Yaneza, and Mary Jane C Samonte. Parkpal: a park sharing and crowdsource park monitoring mobile application. In *Proceedings of the 10th International Conference on E-Education, E-Business, E-Management and E-Learning*, pages 383–388. ACM, 2019.

[7] Saroj Kaushik, Sunita Tiwari, Chhavi Agarwal, and Aakash Goel. Ubiquitous crowdsourcing model for location recommender system. *JCP*, 11(6):463–471, 2016.

[8] Sandra Servia-Rodríguez, Kiran K Rachuri, Cecilia Mascolo, Peter J Rentfrow, Neal Lathia, and Gillian M Sandstrom. Mobile sensing at the service of mental well-being: a large-scale longitudinal study. In *Proceedings of the 26th International Conference on World Wide Web*, pages 103–112. International World Wide Web Conferences Steering Committee, 2017.

[9] Joaquín Torres-Sospedra, Antonio Jiménez, Stefan Knauth, Adriano Moreira, Yair Beer, Toni Fetzer, Viet-Cuong Ta, Raul Montoliu, Fernando Seco, Germán Mendoza-Silva, et al. The smartphone-based offline indoor location competition at ipin 2016: Analysis and future work. *Sensors*, 17(3):557, 2017.

[10] Byoungjip Kim, Seungwoo Kang, Jin-Young Ha, and Junehwa Song. Visitsense: Sensing place visit patterns from ambient radio on smartphones for targeted mobile ads in shopping malls. *Sensors*, 15(7):17274–17299, 2015.

[11] Jordan Naftolin. Adjustable mobile phone settings based on environmental conditions, March 22 2016. US Patent 9,294,612.

[12] Ruiyun Yu, Xingyou Xia, Shiyang Liao, and Xingwei Wang. A location prediction algorithm with daily routines in location-based participatory sensing systems. *International Journal of Distributed Sensor Networks*, 11(10):481705, 2015.

[13] Christian Heipke. Crowdsourcing geospatial data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):550–557, 2010.

[14] Waze. https://www.waze.com. Accessed: 2019-08-12.

[15] Lakhdar Meftah, Maria Gomez, Romain Rouvoy, and Isabelle Chrisment. AndroFleet: Testing WiFi Peer-to-Peer Mobile Apps in the Large. In *32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*, 2017.

[16] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. Testing nearby peer-to-peer mobile apps at large. In *Proceedings of the 6th International Conference on Mobile Software Engineering and Systems*, pages 1–11. IEEE Press, 2019.

[17] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. Fougere: User-centric location privacy in mobile crowdsourcing apps. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 116–132. Springer, 2019.

[18] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. Improving user-centric privacy in data-intensive mobile apps. *Journal of Parallel and Distributed Computing*, 2020.

[19] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. Leveraging wifi places for anonymous location data collection. In *To be defined*, pages 1–10. To be defined, 2020.

[20] Fougere: Anonymous location data collection middleware for mobile apps. https://github.com/m3ftah/fougere. Accessed: 2019-08-12.

[21] Androfleet: A tool to test wifi peer-to-peer mobile apps in the large. https://github.com/m3ftah/androfleet. Accessed: 2019-08-12.

[22] Testing p2p mobile apps. https://github.com/m3ftah/NearbyTest. Accessed: 2019-08-12.

[23] Automatic user contextualization based on wifi fingerprints. https://github.com/m3ftah/wifiContext. Accessed: 2019-08-12.

[24] An android crowdsourcing mobile app using the wifi places framework. https://www.daykeeper.app/. Accessed: 2019-08-12.

[25] Rajesh Krishna Balan, Archan Misra, and Youngki Lee. LiveLabs: Building An In-Situ Real-Time Mobile Experimentation Testbed. In *ACM HotMobile*, pages 0–5, New York, New York, USA, 2014. ACM Press.

[26] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5), sep 2012.

[27] Junxian Huang, Cheng Chen, Yutong Pei, Zhaoguang Wang, Zhiyun Qian, Feng Qian, Birjodh Tiwana, Qiang Xu, Z Mao, Ming Zhang, and Others. Mobiperf: Mobile network measurement system. *Technical Report. University of Michigan and Microsoft Research*, 2011.

[28] Catia Prandi, Paola Salomoni, and Silvia Mirri. mPASS: Integrating People Sensing and Crowdsourcing to Map Urban Accessibility. *Proc. of CCNC'14*, 2014.

[29] Niels Brouwers and Koen Langendoen. Pogo, a Middleware for Mobile Phone Sensing. *Proceedings of the 13th International Middleware Conference*, pages 21–40, 2012.

[30] Haksoo Choi, Supriyo Chakraborty, Zainul M. Charbiwala, and Mani B. Srivastava. SensorSafe: A framework for privacy-preserving management of personal sensory information. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6933 LNCS, pages 85–100. Springer, Berlin, Heidelberg, 2011.

[31] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. Anonysense: privacy-aware people-centric sensing. *Mobisys08 Proceedings of the Sixth International Conference on Mobile Systems Applications and Services*, pages 211–224, 2008.

[32] Nicolas Haderer, Romain Rouvoy, and Lionel Seinturier. A preliminary investigation of user incentives to leverage crowdsensing activities. *2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2013*, pages 199–204, 2013.

[33] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez Del Prado Cortez. Next place prediction using mobility Markov chains. In *Proceedings of the First Workshop on Measurement Privacy and Mobility MPM 2012*, pages 1–6, New York, New York, USA, 2012. ACM Press.

[34] Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sonia Ben Mokhtar, and Sara Bouchenak. A control-theoretic approach for location privacy in mobile applications. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1488–1493. IEEE, 2018.

[35] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. Hmc: Robust privacy protection of mobility data against multiple re-identification attacks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):124, 2018.

[36] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting Privacy Against Location-based Personal Identification. In *Proc. of VLDB'05*, 2005.

[37] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. A classification of location privacy attacks and approaches. *Personal Ubiquitous Comput.*, 18(1), January 2014.

[38] Alastair R. Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Proc. of Percom'04*, 2004.

[39] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of MobiSys'03*, 2003.

[40] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proc. of ACM SIGSPATIAL'09*, 2009.

[41] Mohamed F. Mokbel. Privacy in location-based services: State-of-the-art and research directions. In *Proc. of MDM'07*, 2007.

[42] Derya Birant and Alp Kut. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, 60(1), 2007.

[43] Michael Chau, Reynold Cheng, Ben Kao, and Jackey Ng. Uncertain data mining: An example in clustering location data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006.

[44] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 33(1), 2004.

[45] Reza Shokri, George Theodorakopoulos, Jean Yves Le Boudec, and Jean Pierre Hubaux. Quantifying location privacy. In *Proc. of S&P'11*, may 2011.

[46] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. Ap-attack: a novel user re-identification attack on mobility datasets. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 48–57. ACM, 2017.

[47] Vincent Primault, Mohamed Maouche, Antoine Boutet, Sonia Ben Mokhtar, Sara Bouchenak, and Lionel Brunie. Accio: How to make location privacy experimentation open and easy. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 896–906. IEEE, 2018.

[48] Sophie Cerf, Sonia Ben Mokhtar, Sara Bouchenak, Nicolas Marchand, and Bogdan Robu. Dynamic modeling of location privacy protection mechanisms. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 26–39. Springer, 2018.

[49] Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Robert Birke, Sara Bouchenak, Lydia Y. Chen, Nicolas Marchand, and Bogdan Robu. PULP: Achieving Privacy and Utility Trade-off in User Mobility Data. In *Proc. of SRDS'17*, September 2017.

[50] Daniel Kifer. l-Diversity : Privacy Beyond k -Anonymity. *Proceedings of the 22nd International Conference on Data Engineering*, 1(1):1–36, mar 2006.

[51] Li Ninghui, Li Tiancheng, and Suresh Venkatasubramanian. t-Closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings - International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[52] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. *Proceedings - IEEE International Conference on Mobile Data Management*, pages 65–72, 2008.

[53] Rui Chen, Benjamin C M Fung, Noman Mohammed, Bipin C. Desai, and Ke Wang. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, 231, 2013.

[54] Bin Zan, Marco Gruteser, and Xuegang Ban. Linking Anonymous Location Traces Through Driving Charasteristics. *Codaspy*, 2013.

[55] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. The long road to computational location privacy: A survey. *IEEE Communications Surveys & Tutorials*, 2018.

[56] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Nunez del Prado Cortez. GEPETO: A GEoPrivacy-Enhancing TOolkit. In *Proc. of AINA Workshops'10*, 2010.

[57] Chris Y.T. Ma, David K.Y. Yau, Nung Kwan Yip, and Nageswara S.V. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proc. of MobiCom'10*, 2010.

[58] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Achieving Guaranteed Anonymity in GPS Traces via Uncertainty-Aware Path Cloaking. *IEEE Transactions on Mobile Computing*, 9(8):1089–1107, aug 2010.

[59] Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.

[60] Delphine Christin, Daniel M. Bub, Andrey Moerov, and Saffija Kasem-Madani. A distributed privacy-preserving mechanism for mobile urban sensing applications. In *Proc of. ISSNIP'15*, apr 2015.

[61] Hayam Mousa, Sonia Ben Mokhtar, Omar Hasan, Osama Younes, Mohiy Hadhoud, and Lionel Brunie. Trust management and reputation systems in mobile participatory sensing applications: A survey. *Computer Networks*, 90:49–73, 2015.

[62] Hayam Mousa, Sonia Benmokhtar, Omar Hasan, Lionel Brunie, Osama Younes, and Mohiy Hadhoud. A reputation system resilient against colluding and malicious adversaries in mobile participatory sensing applications. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 829–834. IEEE, 2017.

[63] Hayam Mousa, Sonia Ben Mokhtar, Omar Hasan, Lionel Brunie, Osama Younes, and Mohiy Hadhoud. Privasense: Privacy-preserving and reputation-aware mobile participatory sensing. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 38–47. ACM, 2017.

[64] Chi Yin Chow, Mohamed F. Mokbel, and Xuan Liu. Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, 15(2), apr 2011.

[65] Reza Shokri, Panos Papadimitratos, George Theodorakopoulos, and Jean Pierre Hubaux. Collaborative location privacy. In *Proceedings - 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS 2011*, pages 500–509. IEEE, oct 2011.

[66] Reza Shokri, George Theodorakopoulos, Panos Papadimitratos, Ehsan Kazemi, and Jean Pierre Hubaux. Hiding in the mobile crowd: Location privacy through collaboration. *IEEE Transactions on Dependable and Secure Computing*, 11(3):266–279, may 2014.

[67] Tao Peng, Qin Liu, Dacheng Meng, and Guojun Wang. Collaborative trajectory privacy preserving scheme in location-based services. *Information Sciences*, 387, 2017.

[68] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. Adaptive location privacy with alp. In *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, pages 269–278. IEEE, 2016.

[69] Adrien Luxey, Yérom-David Bromberg, Fábio M Costa, Vinícius Lima, Ricardo CA da Rocha, and François Taïani. Sprinkler: A probabilistic dissemination protocol to provide fluid user interaction in multi-device ecosystems. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[70] Yérom-David Bromberg, Adrien Luxey, and François Taïani. Cascade: Reliable distributed session handoff for continuous interaction across devices. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 244–254. IEEE, 2018.

[71] Paarijaat Aditya, Viktor Erdélyi, Matthew Lentz, Elaine Shi, Bobby Bhattacharjee, and Peter Druschel. Encore: Private, context-based communication for mobile social apps. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 135–148. ACM, 2014.

[72] Lillian Tsai, Roberta De Viti, Matthew Lentz, Stefan Saroiu, Bobby Bhattacharjee, and Peter Druschel. enclosure: Group communication via encounter closures. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 353–365. ACM, 2019.

[73] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5):36–44, 2012.

[74] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. Worker types and personality traits in crowdsourcing relevance labels. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1941–1944. ACM, 2011.

[75] Matthias Stevens and Ellie D'Hondt. Crowdsourcing of pollution data using smartphones. In *Workshop on Ubiquitous Crowdsourcing*, pages 1–4, 2010.

[76] Irene Garcia Martí, Luis E Rodríguez, Mauricia Benedito, Sergi Trilles, Arturo Beltrán, Laura Díaz, and Joaquín Huerta. Mobile application for noise pollution monitoring through gamification techniques. In *International Conference on Entertainment Computing*, pages 562–571. Springer, 2012.

[77] Enrico Gregori, Luciano Lenzini, Valerio Luconi, and Alessio Vecchio. Sensing the internet through crowdsourcing. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 248–254. IEEE, 2013.

[78] Benjamin L Ranard, Yoonhee P Ha, Zachary F Meisel, David A Asch, Shawndra S Hill, Lance B Becker, Anne K Seymour, and Raina M Merchant. Crowdsourcing—harnessing the masses to advance health and medicine, a systematic review. *Journal of general internal medicine*, 29(1):187–203, 2014.

[79] Nicolas Haderer, Romain Rouvoy, and Lionel Seinturier. Dynamic deployment of sensing experiments in the wild using smartphones. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7891 LNCS, pages 43–56. Springer, 2013.

[80] Sanae Rosen, Hongyi Yao, Ashkan Nikravesh, Yunhan Jia, David Choffnes, and Z Morley Mao. Mapping global mobile performance trends with mobilyzer and mobiperf. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 353–353. ACM, 2014.

[81] Min (Ucla) Mun, Sasank (Ucla) Reddy, Katie Shilton, and Nathan Yau. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. *MobiSys*, 2009.

[82] Tathagata Das, Prashanth Mohan, Venkata N Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. PRISM: Platform for Remote Sensing using Smartphones. *Proc. of MobiSys'10*, 2010.

[83] Ling Hu and Cyrus Shahabi. Privacy assurance in mobile sensing networks: Go beyond trusted servers. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2010*, pages 613–619. IEEE, mar 2010.

[84] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. SPPEAR: Security & Privacy- Preserving Architecture for Participatory-Sensing Applications. In *Proc of. WiSec'14*, 2014.

[85] Ioannis Boutsis and Vana Kalogeraki. Location privacy for crowdsourcing applications. In *Proc of. UbiComp'16*, 2016.

[86] Sheng Gao, Jianfeng Ma, Weisong Shi, Guoxing Zhan, and Cong Sun. TrPF: A trajectory privacy-preserving framework for participatory sensing. *IEEE Transactions on Information Forensics and Security*, 8(6), jun 2013.

[87] Suining He and S-H Gary Chan. Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys & Tutorials*, 18(1):466–490, 2015.

[88] John Krumm and Eric Horvitz. Locadio: Inferring motion and location from wi-fi signal strengths. In *mobiquitous*, pages 4–13, 2004.

[89] Teemu Pulkkinen, Johannes Verwijnen, and Petteri Nurmi. WiFi positioning with propagation-based calibration. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks - IPSN '15*, pages 366–367, New York, New York, USA, 2015. ACM Press.

[90] Simon Yiu, Marzieh Dashti, Holger Claussen, and Fernando Perez-Cruz. Wireless RSSI fingerprinting localization, feb 2017.

[91] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-markie: Indoor pathway mapping made easy. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 85–98, 2013.

[92] Chengwen Luo, Hande Hong, and Mun Choon Chan. Piloc: A self-calibrating participatory indoor localization system. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pages 143–153. IEEE, 2014.

[93] Zan Li, Xiaohui Zhao, Fengye Hu, Zhongliang Zhao, Jose Luis Carrera, and Torsten Braun. Soicp: A seamless outdoor-indoor crowdsensing positioning system. *IEEE internet of things journal*, 2019.

[94] Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. Push the limit of WiFi based localization for smartphones. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 305, New York, New York, USA, 2012. ACM Press.

[95] Konrad Górski, Mateusz Groth, and Łukasz Kulas. A multi-building WiFi-based indoor positioning system. In *2014 20th International Conference on Microwaves, Radar and Wireless Communications, MIKON 2014*, pages 1–4. IEEE, jun 2014.

[96] Abby Stephanie Salazar, Leocundo Aguilar, and Manuel Castañón-Puga. *Localization of Indoor Areas Using Wi-Fi Signals , Type-2 Fuzzy Inference Systems and Data Mining*, volume I. 2014.

[97] Hong Li, Limin Sun, Haojin Zhu, Xiang Lu, and Xiuzhen Cheng. Achieving privacy preservation in WiFi fingerprint-based localization. In *Proceedings - IEEE INFOCOM*, pages 2337–2345. IEEE, apr 2014.

[98] Myungjun Jin, Bonhyun Koo, Sangwoo Lee, Chansik Park, Min Joon Lee, and Sunwoo Kim. IMU-assisted nearest neighbor selection for real-time WiFi fingerprinting positioning. In *IPIN 2014 - 2014 International Conference on Indoor Positioning and Indoor Navigation*, pages 745–748. IEEE, oct 2014.

[99] Ahmed H. Salamah, Mohamed Tamazin, Maha A. Sharkas, and Mohamed Khedr. An enhanced WiFi indoor localization System based on machine learning. In *2016 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016*, pages 1–8. IEEE, oct 2016.

[100] Simon Yiu and Kai Yang. Gaussian Process Assisted Fingerprinting Localization. *IEEE Internet of Things Journal*, 3(5):683–690, oct 2016.

[101] Nicholas Capurso, Tianyi Song, Wei Cheng, Jiguo Yu, and Xiuzhen Cheng. An Android-Based Mechanism for Energy Efficient Localization Depending on Indoor/Outdoor Context. *IEEE Internet of Things Journal*, 4(2):299–307, apr 2017.

[102] Afaz Uddin Ahmed, Neil W. Bergmann, Reza Arablouei, Branislav Kusy, Frank De Hoog, and Raja Jurdak. Poster Abstract: Fast Indoor Localization Using WiFi Channel State Information. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 120–121. IEEE, apr 2018.

[103] Giuseppe Caso, Luca De Nardis, Filip Lemic, Vlado Handziski, Adam Wolisz, and Maria-Gabriella Di Benedetto. Vifi: Virtual fingerprinting wifi-based indoor positioning via multi-wall multi-floor propagation model. *IEEE Transactions on Mobile Computing*, 2019.

[104] Paul A Zandbergen. Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning. *Transactions in GIS*, 13:5–25, 2009.

[105] Thanh-Binh Nguyen, Thuong Nguyen, Wei Luo, Svetha Venkatesh, and Dinh Phung. Unsupervised inference of significant locations from wifi data for understanding human dynamics. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia*, MUM ’14, pages 232–235, New York, NY, USA, 2014. ACM.

[106] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic, and Alex Sandy Pentland. Predicting personality using novel mobile phone-based metrics. In *International conference on social computing, behavioral-cultural modeling, and prediction*, pages 48–55. Springer, 2013.

[107] Jun Rekimoto, Takashi Miyaki, and Takaaki Ishizawa. Lifetag: Wifi-based continuous location logging for life pattern analysis. In *LoCA*, volume 2007, pages 35–49, 2007.

[108] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[109] Haipeng Zhang, Zhixian Yan, Jun Yang, Emmanuel Munguia Tapia, and David J Crandall. Mfingerprint: Privacy-preserving user modeling with multimodal mobile device footprints. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 195–203. Springer, 2014.

[110] Arkadiusz Stopczynski, Vedran Sekara, Piotr Sapiezynski, Andrea Cuttone, Mette My Madsen, Jakob Eg Larsen, and Sune Lehmann. Measuring large-scale social networks with high resolution. *PloS one*, 9(4):e95978, 2014.

[111] Xuan Xie, Hongze Xu, Guang Yang, Zhi Hong Mao, Wenyan Jia, and Mingui Sun. Reuse of WiFi information for indoor monitoring of the elderly. In *Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016*, pages 261–264. IEEE, jul 2016.

[112] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. Offline behaviors of online friends. *arXiv preprint arXiv:1811.03153*, 2018.

[113] Antonio del Corte-Valiente, José Manuel Gómez-Pulido, and Oscar Gutiérrez-Blanco. Efficient techniques and algorithms for improving indoor localization precision on wlan networks applications. *International Journal of Communications, Network and System Sciences*, 2(07):645, 2009.

[114] Christian Beder and Martin Klepal. Fingerprinting based localisation revisited: A rigorous approach for comparing rssi measurements coping with missed access points and differing antenna attenuations. In *2012 international conference on indoor positioning and indoor navigation (IPIN)*, pages 1–7. IEEE, 2012.

[115] Dimitrios Lymberopoulos and Jie Liu. The microsoft indoor localization competition: Experiences and lessons learned. *IEEE Signal Processing Magazine*, 34(5):125–140, 2017.

[116] Nan Zhang and Jianhua Feng. Polaris: A fingerprint-based localization system over wireless networks. In *International Conference on Web-Age Information Management*, pages 58–70. Springer, 2012.

[117] Muhammad N. Sakib, Junaed Bin Halim, and Chin Tser Huang. Determining location and movement pattern using anonymized WiFi access point BSSID. In *Proceedings - 7th International Conference on Security Technology, SecTech 2014*, pages 11–14. IEEE, dec 2015.

[118] Piotr Sapiezynski, Arkadiusz Stopczynski, Radu Gatej, and Sune Lehmann. Tracking human mobility using WiFi signals. *PLoS ONE*, 10(7), 2015.

[119] David Kofoed Wind, Piotr Sapiezynski, Magdalena Anna Furman, and Sune Lehmann. Inferring stop-locations from WiFi. *PLoS ONE*, 11(2), 2016.

[120] Taehwa Choi, Yohan Chon, and Hojung Cha. Energy-efficient WiFi scanning for localization. *Pervasive and Mobile Computing*, 37:124–138, jun 2017.

[121] Ying Dar Lin, Jose F. Rojas, Edward T H Chu, and Yuan Cheng Lai. On the accuracy, efficiency, and reusability of automated test oracles for android devices. *IEEE Transactions on Software Engineering*, 40(10):957–970, oct 2014.

[122] Mattia Fazzini, Eduardo Noronha De A. Freitas, Shauvik Roy Choudhary, and Alessandro Orso. Barista: A Technique for Recording, Encoding, and Running Platform Independent Android Tests. In *Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017*, pages 149–160. IEEE, mar 2017.

[123] Atif M. Memon, Martha E. Pollack, and Mary Lou Soffa. Automated test oracles for GUIs. *ACM SIGSOFT Software Engineering Notes*, 25(6):30–39, 2000.

[124] Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana, and Nicola Amatucci. Considering context events in event-based testing of mobile applications. In *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2013*, pages 126–133. IEEE, mar 2013.

[125] Tim A. Majchrzak and Matthias Schulte. Context-Dependent Testing of Applications for Mobile Devices. *Open Journal of Web Technologies*, 2(1):27–39, 2015.

[126] Oum EI Kheir Aktouf, Tao Zhang, Jerry Gao, and Tadahiro Uehara. Testing location-based function services for mobile applications. In *Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015*, volume 30, pages 308–314. IEEE, mar 2015.

[127] Wei Song, Xiangxing Qian, and Jeff Huang. EHBDroid: Beyond GUI testing for Android applications. In *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 27–37. IEEE, oct 2017.

[128] Ismayle de Sousa Santos, Rossana Maria de Castro Andrade, Lincoln Souza Rocha, Santiago Matalonga, Káthia Marçal de Oliveira, and Guilherme Horta Travassos. Test case design for context-aware applications: Are we there yet? *Information and Software Technology*, 88:1–16, aug 2017.

[129] Casper S. Jensen, Mukul R. Prasad, and Anders Møller. Automated testing with targeted event sequence generation. *Proceedings of the 2013 International Symposium on Software Testing and Analysis - ISSTA 2013*, page 67, 2013.

[130] Haowen Zhu, Xiaojun Ye, Xiaojun Zhang, and Ke Shen. A Context-Aware Approach for Dynamic GUI Testing of Android Applications. In *Proceedings - International Computer Software and Applications Conference*, volume 2, pages 248–253. IEEE, jul 2015.

[131] Siena Yu and Shingo Takada. Mobile application test case generation focusing on external events. In *Proceedings of the 1st International Workshop on Mobile Development - Mobile! 2016*, pages 41–42, New York, New York, USA, 2016. ACM Press.

[132] Christoffer Quist and Anders Møller. Systematic Execution of Android Test Suites in Adverse Conditions. *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, pages 83–93, 2015.

[133] Bo Jiang, Xiang Long, and Xiaopeng Gao. MobileTest: A tool supporting automatic black box test for software on smart mobile devices. In *Proceedings - International Conference on Software Engineering*, pages 8–8. IEEE, may 2007.

[134] Ke Mao, Mark Harman, and Yue Jia. Robotic Testing of Mobile Apps for Truly Black-Box Automation. *IEEE Software*, 34(2):11–16, mar 2017.

[135] Tsuyoshi Yumoto, Toru Matsuodani, and Kazuhiko Tsuda. A test analysis method for black box testing using AUT and fault knowledge. In *Procedia Computer Science*, volume 22, pages 551–560, 2013.

[136] Marc Hesenius, Tobias Griebe, Stefan Gries, and Volker Gruhn. Automating UI tests for mobile applications with formal gesture descriptions. *MobileHCI 2014 - Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 213–222, 2014.

[137] Tobias Griebe and Volker Gruhn. A model-based approach to test automation for context-aware mobile applications. *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*, pages 420–427, 2014.

[138] Ke Mao, Mark Harman, and Yue Jia. Crowd intelligence enhances automated mobile testing. In *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 16–26. IEEE, oct 2017.

[139] Nariman Mirzaei, Joshua Garcia, Hamid Bagheri, Alireza Sadeghi, and Sam Malek. Reducing combinatorics in GUI testing of android applications. In *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, pages 559–570, New York, New York, USA, 2016. ACM Press.

[140] Li Lyna Zhang, Chieh Jan Mike Liang, Yunxin Liu, and Enhong Chen. Systematically testing background services of mobile apps. In *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 4–15. IEEE, oct 2017.

[141] Siena Yu and Shingo Takada. External Event-Based Test Cases for Mobile Application. *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, pages 148–149, 2008.

[142] Jerry Gao, Wei Tek Tsai, Ray Paul, Xiaoying Bai, and Tadahiro Uehara. Mobile testing-as-a-service (MTaaS) - Infrastructures, issues, solutions and needs. In *Proceedings - 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, HASE 2014*, pages 158–167, 2014.

[143] Chieh-Jan Mike Liang, Nicholas D. Lane, Niels Brouwers, Li Zhang, Börje F. Karlsson, Hao Liu, Yan Liu, Jun Tang, Xiang Shan, Ranveer Chandra, and Feng Zhao. Caiipa: Automated Large-scale Mobile App Testing through Contextual Fuzzing. In *MobiCom*, pages 519–530, New York, New York, USA, 2014. ACM Press.

[144] Isabel K Villanes Rojas, Silvia Meireles, and Arilo Claudio Dias-neto. Cloud-Based Mobile App Testing Framework : Architecture , Implementation and Execution. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST*, pages 1–10, New York, New York, USA, 2016. ACM Press.

[145] Hyungkeun Song, Seokmoon Ryoo, and Jin Hyung Kim. An integrated test automation framework for testing on heterogeneous mobile platforms. In *Proceedings - 1st ACIS International Symposium on Software and Network Engineering, SSNE 2011*, pages 141–145. IEEE, dec 2011.

[146] Tao Zhang, Jerry Gao, and Jing Cheng. Crowdsourced testing services for mobile apps. In *Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on*, pages 75–80. IEEE, 2017.

[147] Hsiang Lin Wen, Chia Hui Lin, Tzong Han Hsieh, and Cheng Zen Yang. PATS: A Parallel GUI Testing Framework for Android Applications. In *Proceedings - International Computer Software and Applications Conference*, volume 2, pages 210–215. IEEE, jul 2015.

[148] Doug Serfass and Kenji Yoshigoe. Wireless sensor networks using android virtual devices and near field communication peer-to-peer emulation. In *Conference Proceedings - IEEE SOUTHEASTCON*, pages 1–6. IEEE, apr 2013.

[149] Björn Richerzhagen, Dominik Stingl, Julius Rückert, and Ralf Steinmetz. Simonstrator: Simulation and Prototyping Platform for Distributed Mobile Applications. *EAI International Conference on Simulation Tools and Techniques (SIMUTOOLS)*, pages 1–6, 2015.

[150] Inês Coimbra Morgado, Ana C R Paiva, and João Pascoal Faria. Automated pattern-based testing of mobile applications. In *Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014*, pages 294–299. IEEE, sep 2014.

[151] Henry Muccini, Antonio Di Francesco, and Patrizio Esposito. Software testing of mobile applications: Challenges and future research directions. In *2012 7th International Workshop on Automation of Software Test (AST)*, pages 29–35. IEEE, jun 2012.

[152] B. Kirubakaran and V. Karthikeyani. Mobile application testing — Challenges and solution approach through automation. *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pages 79–84, feb 2013.

[153] Abel Méndez-porras, Christian Quesada-lópez, and Marcelo Jenkins. Automated Testing of Mobile Applications : A Systematic Map and Review. *CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering*, pages 195–208, 2015.

[154] Domenico Amalfitano, Nicola Amatucci, Porfirio Tramontana, Anna Rita Fasolino, and Atif M Memon. A General Framework for comparing Automatic Testing Techniques of Android Mobile Apps. *Journal of Systems and Software*, 125:322–343, 2016.

[155] Mario Linares-Vásquez, Kevin Moran, and Denys Poshyvanyk. Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing. *Proceedings - 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017*, pages 399–410, 2017.

[156] Eduardo Cunha de Almeida, Gerson Sunyé, Yves Le Traon, and Patrick Valduriez. Testing peer-to-peer systems. *Empirical Software Engineering*, 15(4):346–379, 2010.

[157] Zhizhi Zhou, Hao Wang, Jin Zhou, Li Tang, Kai Li, Weibo Zheng, and Meiqi Fang. Pigeon: A framework for testing peer-to-peer massively multiplayer online games over heterogeneous network. In *2006 3rd IEEE Consumer Communications and Networking Conference, CCNC 2006*, volume 2, pages 1028–1032, 2006.

[158] Gerson Sunyé, Eduardo Cunha De Almeida, Yves Le Traon, Benoit Baudry, and Jean Marc Jézéquel. Model-based testing of global properties on large-scale distributed systems. *Information and Software Technology*, 56(7):749–762, 2014.

[159] My El, Hassan Charaf, Mohammed Benattou, and Salma Azzouzi. A JESS AGENT Based Architecture for Testing Distributed Systems. *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, 30:1619–1634, 2014.

[160] A.a Marroquin, D.a Gonzalez, and S.b Maag. A novel distributed testing approach based on test cases dependencies for communication protocols. In *Proceeding of the 2015 Research in Adaptive and Convergent Systems, RACS 2015*, pages 497–504, New York, New York, USA, 2015. ACM Press.

[161] Bogdan Butnaru, Florin Dragan, Georges Gardarin, Ioana Manolescu, Benjamin Nguyen, Radu Pop, Nicoleta Preda, and Laurent Yeh. P2PTester: A tool for measuring P2P platform performance. In *Proceedings - International Conference on Data Engineering*, pages 1501–1502, 2007.

[162] J R Boldman, U W Parks, R L Peruvemba, K J Prakash, and J T Wheeler. System and method for testing peer-to-peer network applications, 2007.

[163] V. Gorodetsky, O. Karsaev, V. Samoylov, S. Serebryakov, S. Balandin, S. Leppanen, and M. Turunen. Virtual P2P environment for testing and evaluation of mobile P2P agents networks. In *Proceedings - The 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM 2008*, pages 422–429. IEEE, sep 2008.

[164] Wi-FI Alliance. Wi-Fi Direct | Wi-Fi Alliance, 2015.

[165] Google Developers. Issues, 2017.

[166] Dimitrios-Georgios Akestoridis. CRAWDAD dataset uoi/haggle (v. 2016-08-28): derived from cambridge/haggle (v. 2009-05-29). Downloaded from https://crawdad.org/uoi/haggle/20160828/one, March 2018. traceset: one.

[167] Jialiu Lin, Norman Sadeh, Shahriyar Amini, Janne Lindqvist, Jason I Hong, and Joy Zhang. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 501, 2012.

[168] Simson L Garfinkel. IRBs and security research: myths, facts and mission creep. *Proceedings of the 1st Conference on Usability, Psychology, and Security*, pages 1–5, 2008.

[169] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. {CRAWDAD} dataset epfl/mobility (v. 2009-02-24). Downloaded from \url{anonymous}, feb 2009.

[170] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. De-anonymization attack on geolocated data. *Journal of Computer and System Sciences*, 80(8):1597–1614, 2014.

[171] John Krumm. Inference Attacks on Location Tracks. *Pervasive Computing*, 10(Pervasive):127–143, 2007.

[172] Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *Proc. of CCS'13*, pages 901–914, 2013.

[173] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. *SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium*, 13:21, 2004.

[174] Kassem Fawaz and Kang G Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM, 2014.

[175] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An Effectiveness Study on Trajectory Similarity Measures. *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*, 137(February):13–22, 2013.

[176] Chi-Yin Chow, Mohamed F Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. *Proc. of ACM SIGSPATIAL*, 2006.

[177] Latanya Sweeney. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal on Uncertainty*, 10(5):557–570, 2002.

[178] A Brush and John Krumm. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *Proc. of Ubicomp'10*, 2010.

[179] Lakhdar Meftah, Maria Gomez, Romain Rouvoy, and Isabelle Chrisment. ANDROFLEET: Testing WiFi Peer-to-Peer Mobile Apps in the Large. *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 961–966, 2017.

[180] Reza Shokri, George Theodorakopoulos, George Danezis, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Quantifying location privacy: the case of sporadic location exposure. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 57–76. Springer, 2011.

[181] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[182] Sophie Cerf, Bogdan Robu, Nicolas Marchand, Antoine Boutet, Vincent Primault, Sonia Ben Mokhtar, and Sara Bouchenak. Toward an easy configuration of location privacy protection mechanisms. In *Proceedings of the Posters and Demos Session of the 17th International Middleware Conference*, pages 11–12. ACM, 2016.