On the persistence of wireless advertising without infrastructure support

Noelia Perez Palma
IMDEA Networks Institute, Spain and
University Carlos III of Madrid, Spain
noelia.perez@imdea.org

Vincenzo Mancuso
IMDEA Networks Institute, Spain
vincenzo.mancuso@imdea.org

Marco Ajmone Marsan
Politecnico di Torino, Italy and
IMDEA Networks Institute, Spain
marco.ajmone@polito.it

***Abstract*—The number of devices connected to the Internet keeps growing, and the traffic they collectively generate grows even faster, creating congestion issues, especially in the network access segment. Some of the newer Internet applications can however be served with device-to-device (D2D) opportunistic communications, thus offloading a portion of the traffic from the network.**

This thesis studies and presents experiments on the possibility of providing content availability to end user terminals by exploiting D2D infrastructureless connectivity and opportunistic communication. In order to avoid unrealistic assumptions on the behavior of D2D communications, the work includes and builds upon the implementation of an Android application that supports infrastructureless distributed content sharing between wireless devices using Wi-Fi Direct. The collected experimental data permit to analyze the occurring events and to assess and compare the performance of the class of services under evaluation against existing communication protocols.

I. Introduction

Infrastructureless services are becoming important [15] since the society of today has already incorporated a huge number of mobile devices mainly carried by humans in their daily activity. In recent years, as envisaged by [8], protocols such as Bluetooth or Wi-Fi Direct have been developed to allow devices to communicate without requiring a wireless access point. Hence, most of our devices are already equipped with the hardware and software necessary to support the capabilities provided by these communication protocols.

Dynamic and personalized advertising by means of device-to-device (D2D) communication tools is imminent due to cellular traffic offloading requirements and the existence of environments with network infrastructure deficiencies [15]. This thesis studies the performance of direct communication between devices equipped with Wi-Fi Direct, with the aim of exchanging information without making use of any infrastructure. Such services relay instead on the mobility of devices to move information as well. Therefore, this work shows if and how Wi-Fi Direct D2D communications can be used for personalized and infrastructureless advertising, news spreading protocols with strong local geographical relevance, or alike services.

So far, existing approaches to the performance evaluation of such infrastructureless services have been based on models and simulation [13], [16], while experimental works [4], [7], [12] have used Wi-Fi Direct to represent the performance to be expected in real environments with a small subset of devices and in entirely controlled and static scenarios. Specific applications using Wi-Fi Direct have also been developed in order to achieve similar performance analysis to ours, but most of them pose lacks in certain schemes, which are considered to be essential from our perspective. For instance, in [20] the authors present a more efficient way of creating peer-to-peer (P2P) connections by modifying the discovery phase provided by Wi-Fi Direct standard in Android, but those experiments do not ensure a better overall performance of data dissemination in crowded scenarios. In [21] a chat system is implemented based on profiling matches to create connections. It also takes advantage of Over-The-Top (OTT) methodology which connects to a centralized server to determine users location and interests for improved peer selection. This system requires the user intervention for searching neighbor peers and for selecting who they want to exchange data with. Even though this proposal could have been useful for our aim, it can not be applied to all kind of scenarios regarding our main purposes which are based on automatic network creation for message spreading without user interaction and without infrastructure relying. In the system developed in [14], among their objectives, the authors propose a communication model called passive broadcast that includes Bluetooth and Wi-Fi Direct technologies to carry out local data dissemination and then characterize the parameters involved in the experiments to minimize the process time. The drawbacks of their experimental setup are that they rely on a fully-connected network, thus removing from the scenario the possible users mobility and the consequent formation and disruption of connections among devices. As a result, they are missing an essential part of performance time evaluation.

Moreover, so far only a few works have focused on assessing the feasibility of deployments relying on the existing wireless communication protocols together with the exploitation of actual human mobility. In [1], the application developed follows fairly accurately our proposal with a main handicap in the communication protocol. They built the system on top of the Bluetooth Application Programming Interface (API) in Android, which is known to be less efficient that Wi-Fi Direct in terms of bandwidth, time for peer and service detection and implementation incompatibilities among different device manufacturers [19].

This thesis' contribution supports the idea that the pervasive use of devices along with the relationships between humans will lead to a wide range of opportunities to create opportunistic networks and thus, benefit from information dissemination and acquisition. Specifically, the contribution of this work is threefold: (i) assess the feasibility of infrastructureless networks to provide connection establishment between wireless devices in an efficient manner, (ii) grant content availability to end users by exploiting D2D opportunistic communications, (iii) introduce a working scheme on top of which further researches can build specific services in pursuance of satisfying future needs.

The rest of the paper is organized as follows: Section II introduces Wi-Fi Direct technology and describes all the working phases of the protocol. A detailed explanation of Wi-Fi Direct API in Android is included in Section III along with the limitations encountered during the application development. A complete description of the android application implementation follows in Section IV. Section V illustrates the experimental setup and results of the thesis. A compendium on related work is presented in Section VI and finally, conclusions and future work are detailed in Section VII.

II. Wi-Fi Direct Overview

Wi-Fi Direct is a technology defined by the Wi-Fi Alliance to allow direct device to device communications between 802.11 devices. Wi-Fi Direct supports typical Wi-Fi speeds up to 250 Mbps and works with ranges up to 200 meters. Featuring better time and throughput characteristics than other D2D protocols, such as Bluetooth, Wi-Fi Direct has become the reference standard to support D2D communications, allowing group formation of one-to-one and one-to-many devices.

Additionally, simultaneous connection to a Wi-Fi Direct group and to an infrastructure network is an optional feature. However, nowadays typical smartphones have only one Wi-Fi adapter so they can only connect to a hot spot or to another device, not to both at the same time.

In the next subsections we describe the different elements of the operation of Wi-Fi Direct that are shown in Fig. 1 and Fig. 2.

A. Device and Service Discovery

First of all, when devices are willing to form a P2P group, they alternately listen and send probe requests with additional P2P information elements on the "social channels" 1, 6 and 11 in the 2.4GHz band. At this point, available P2P devices receiving a probe request reply with probe response frames including P2P information elements describing the group characteristics.

An optional feature during this process is the so called service discovery. Once a peer device has been discovered, it can be asked to describe the services it provides. Thus, the device asking for a service may choose to connect to the discovered peer based on whether it provides the required service. This is carried out by a higher layer protocol, e.g. UPnP or Bonjour [11].

B. Group Formation

After the device discovery phase, P2P devices can start forming groups. There are three different ways in which devices can form a group, described in what follows.

1) Standard: This is the main procedure to form a group. It consists in the previous phase of devices discovery and a negotiation of roles through a three-way handshake. In a P2P group any P2P device can take the role of P2P Group Owner (GO), which acts as an access point, or P2P Client which associates to the GO. So, this negotiation determines which of the two peers becomes the GO, as depicted in Fig. 1. During this negotiation also the operating channel, the Wi-Fi Protected Setup (WPS) configuration method, and whether the group is a persistent group, are set. A device can become group owner by choice using the GO Intent attribute of the GO Negotiation request/response frames. The final frame exchange is a GO Negotiation confirmation.

2) Autonomous: In the autonomous group formation type, a device decides to create a P2P group by its own in which it will take the role of GO.

3) Persistent: A connection between two devices can be established as a persistent group. In this way, the session information will be stored in both devices involved in the connection. Therefore, they will remember each other in the next connection attempts. This procedure is convenient for a later re-instantiation of a P2P group, i.e., if a device wants to establish a group with an already known peer, they will avoid the GO negotiation phase and will only do a two-way handshake called Invitation Procedure to re-establish the session stored, as exemplified in Fig. 2.

After any type of group is formed, the GO will start announcing the group by means of beaconing with the negotiated SSID. In this way, other P2P devices can discover the group and send an invitation request.

C. Provisioning and Encryption

In the provisioning phase both the client and the GO exchange credentials using the WPS protocol to support a secure connection. Usually, WPS requires user interaction to enter a PIN code or push a button on the device to allow the connection to be instantiated.

Second, since WPS is based on WPA-2 security, there is an exchange of encryption keys using a 4-way handshake. During this procedure the GO assumes the role of authenticator and the client is the supplicant. Additionally, the GO conducts a Dynamic Host Configuration Protocol exchange to provide an IPv4 address to both devices. Finally, the connection between the devices is established and secured and the data exchange can take place.

It is worth noting that re-instantiating a persistent group is profitable, because, apart from replacing the GO Negotiation by the invitation exchange as mentioned before, the provisioning phase also benefits by the reuse of the stored network credentials. In this way, devices can automatically re-connect when required, avoiding the interaction of the user. Hence, the frames and time invested in the entire process are reduced.

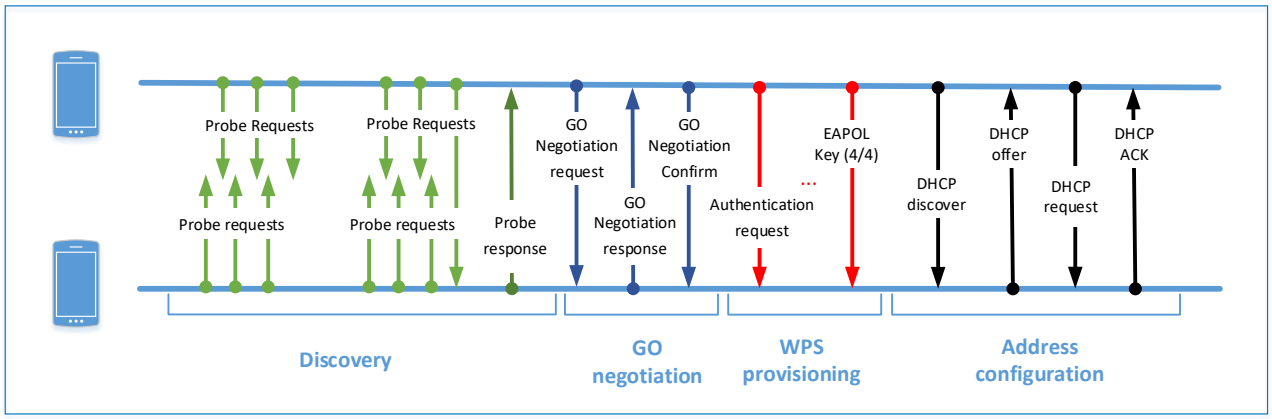


Fig. 1: Wi-Fi Direct standard group formation.

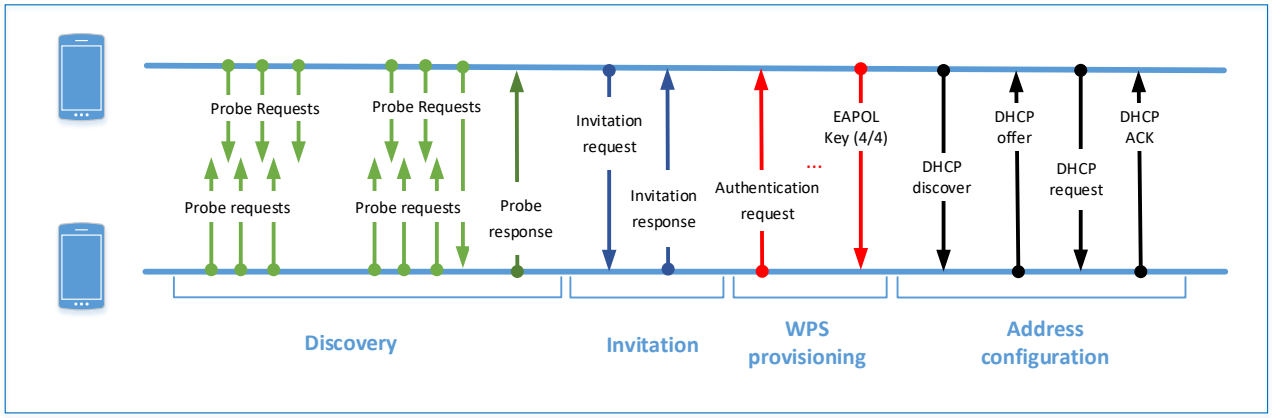


Fig. 2: Wi-Fi Direct persistent group formation.

III. Wi-Fi Direct in Android

Devices running on Android 4.0 (API level 14) or later are being manufactured with the appropriate hardware and software features to connect directly to each other via Wi-Fi without an intermediate access point. For this aim, Android OS complying with the Wi-Fi Alliance's Wi-Fi Direct certification program, has developed a Wi-Fi Direct API for mobile applications, also know as Wi-Fi P2P [9].

A. Android Wi-Fi P2P API

The API implemented by the Android developers is a tool that enables programmers to manage the Wi-Fi Direct functionalities provided by the smartphones manufacturers. The components involving the Wi-Fi P2P API allow devices supporting this feature to discover, connect and communicate to others. Mobile applications that share data among users, such as a multi-player game or a content sharing application can be built on top of services implementing Wi-Fi P2P API functionalities and take advantage of speedy connections across distances much longer than the ones offered by other D2D communication protocols.

In the first place, Wi-Fi P2P API components consist of a class called WifiP2pManager that includes methods to allow a device to discover, request, and connect to peers.

On the other hand, there are interfaces called Event Listeners that are registered to specific events. These listeners contain callback methods that are triggered when the specific events happen, notifying the success or failure of the previous class, WifiP2pManager, method calls.

Finally, objects from the class Intent are used to alert about particular events detected by the Wi-Fi P2P framework. They signal, for instance, dropped connections or newly discovered peers.

Here, we have mentioned the main parts involving Wi-Fi P2P API. In the next subsections we present the limitations encountered in Android Wi-Fi P2P, and later we detail how the methods explained before can be implemented in order to build a service to achieve the desired behavior.

B. Limitations

Even though Android Wi-Fi P2P provides useful and open source means to enable D2D communication between devices, it also presents a number of limitations when applied to different scenarios, mainly because the Wi-Fi Direct protocol was designed to fulfill specific requirements and meet security constraints.

We have proved that some of these limitations can be avoided by rooting the devices. Specifically, with the help of Xposed Framework we initially replaced the methods inside the Android Java Class WifiP2pService to avoid user authentication. Of course, this feature implied a breach of the security restrictions required by the Wi-Fi Direct standard. However, as mentioned before, the Wi-Fi Direct prescribes an authentication with WPS. Thus, from the Lollipop version onwards, the Android designers decided to modify the Wi-Fi Direct API and make it no longer possible to avoid user identification. Therefore, taking advantage of rooting in newest Android based smartphones became an impractical task and so the implementation finally used in this thesis does not use any rooted device, but only simple commercial-off-the shelf devices with their original operating system.

This choice has some practical and important implications since, as shown in [5], with non-rooted devices some scenarios are not feasible, e.g., it is not possible to have devices acting as a GO and Client at the same time in different groups, or devices taking the role of a GO in more than one group and devices playing the role of a Client in more than one group. However, this choice makes the work presented in this thesis more realistic for the deployment of real services, with the real limitation and security constraints of existing and commercial operating systems. In this work, we had to deal with these limitations and find *legal* workarounds to implement D2D infrastructureless services, i.e., we had to use only functionalities that an Android app can run without having root user's privileges. The way in which we tackle the implementation of our approach is explained next in Section IV.

IV. An Android App for infrastructureless communications

We aimed at creating an ad-hoc network with similar techniques as proposed in [5] and [12]. The main difference to previous works is that their deployments were set for real experiments but in non-realistic scenarios. Basically, they present static deployments where devices are able to communicate according to different proposed architectures and benefit from already set topologies, thus avoiding mobility limitations which differs completely from our purpose where the creation of dynamic opportunistic networks is a key concept.

A. Dissemination within a Geofence with one-to-one Groups

In this work, we have developed an Android application that uses Wi-Fi Direct protocol to spread messages within an area marked by means of a *geofence* and within a fixed time horizon. Geofencing is a suitable mechanism to virtually represent the perimeter of a geographical region [10]. In this way, a geofence is dynamically located based on the coordinates of the area of interest and a given radius.

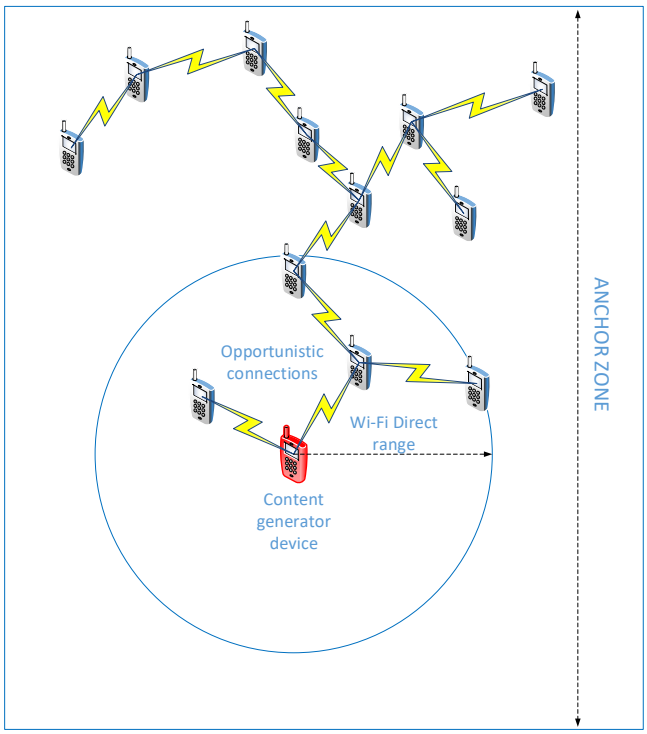


Fig. 3: Ad-hoc network between devices.

The application is used to advertise events and share contents without requiring cellular or hot spot connectivity only allowing one-to-one group formations between devices, as shown in Fig. 3. The goal of the application is to spread a piece of information among all the interested users within the area marked by the geofence, assuming as well the actual mobility patterns present in human environments.

In this way, if people carrying their devices are moving around, devices will have more chances of spreading some content if they manage to connect, at least, to one of the peers around. In fact, most probably they will move apart soon. If a couple of already connected devices waited for more peers to join the group, the actual mobility of certain environments could cause connection failures. Indeed, maintaining the group would become an arduous task. As a result, the data transfer would practically never happen. It is important to note here that whenever a GO leaves the group, the whole group will be dropped, so these were enough factors to choose one-to-one connections only for the application developed for this thesis.

B. State Machine of the Application

In our scenario, the application will act in an epidemic manner trying to connect to as many other devices as possible and infect them by forwarding the content. This procedure is done in loops following the states shown in the state machine

of Fig. 4. We describe next the main characteristic of every state and how the transitions take place.

When a device enters the marked geofence it will start scanning the social channels looking for potential peer devices. This is the `scanning` state in the state machine of Fig. 4. While the scanning is running, any other device can send an invitation request to the first device so it will pass to the `connected` state directly. A second option is to keep scanning during the time period set and connect to one of the peers discovered when the scanning time expires. At this point, the device will send an invitation request to the chosen peer device, which will imply going through the `connecting` state and then to `connected` state when the peer accepts the invitation. There is also the possibility that no peers are discovered by the device so the scanning phase will begin again. It is in `connected` state when a device pair exchange data with each other and right after both of them enter the `disconnecting` state. Next state, `disconnected`, will take place in the exact moment that the application is notified with a disconnection callback triggered from the previous `disconnecting` state.

Due to the nature of human mobility, frequent link disruptions can happen during the connection attempt. In the first place, two devices can have established a connection and, before the data transfer has finished, the connection can be dropped. On the other hand we can find other types of failures before establishing a connection, that is, during the connecting phase. If the local system is busy, it will fail in the connection attempt, as well as if the peer device is busy for a long period of time it will never accept the invitation request, so that the device attempting to connect will fail after a timeout. In both cases there will be a transition to the `failed` state.

Both `disconnected` and `failed` states will eventually lead back to the `scanning` state to repeat the process steadily. In this way, there will be no possibility of getting stuck in any state. Thanks to this aggressive operation mode the application will get to a situation where it has to share content with a peer as fast as possible or, in case of failure, try with next one. Resulting in the creation of an opportunistic and dynamic ad-hoc network among the devices in the area.

C. Wi-Fi P2P Implementation

When it comes to implementing the procedure described above into an Android application we have to take into account certain structures and method calls that we mention below.

To begin with, we will describe the Android Class WifiP2pManager that provides the API for managing Wi-Fi peer-to-peer operation in order to let the application exploit the connectivity features of Wi-Fi P2P.

As mentioned in Section III-A, the API works in an asynchronous way which means that every frame exchanged as a response to any request is received on the listener callbacks. These listeners are WifiP2pManager.ActionListener objects and have two main callbacks, named onSuccess() and onFailure(int), which indicate whether the initiation of some request results on success or not. Upon failure, a callback returns

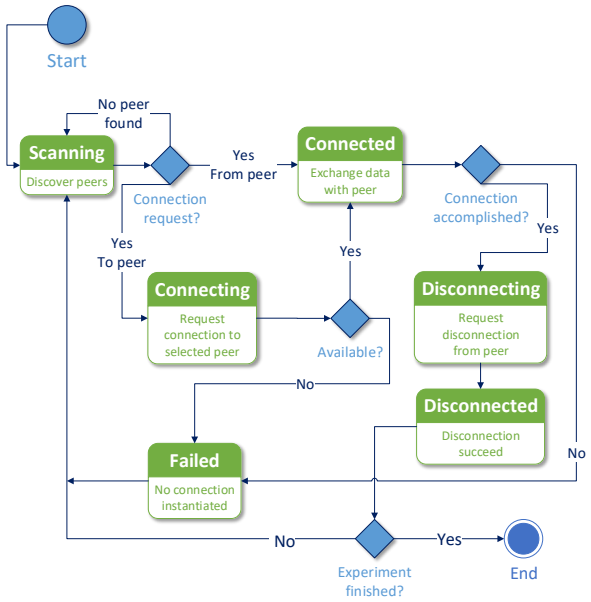


Fig. 4: State Machine of the application.

an integer that represents the reason of the failure, being P2P_UNSUPPORTED, ERROR or BUSY the three possible outcomes. We found that, in most of the cases, the main failure reason returned by this callback was BUSY, indicating that the operation failed because the local framework was busy and unable to serve the request.

1) Discovery of peers: The discovery of available peers in our implementation is defined by a state called `scanning`, as it is commonly referred to in the research community. For this aim, the API provides the function discoverPeers(). However, this is not useful for our purpose, since we do not attempt to connect to every peer around, but only to the peers that are running the same service. In this way, a device can discern which peer to establish a connection with and avoid spamming users that are not interested in our content. To filter the peers we use pre-association service discovery to first advertise which services a device is running before the connection setup. DNS based service discovery (Bonjour) is the higher layer protocol implemented in our application. We describe next the service discovery in detail.

In order to advertise a Bonjour service we first need to add the local service itself by calling addLocalService() in the application. After that, the framework automatically notifies the neighboring peers about the service before establishing a connection.

Here is the main difference to the previous mentioned process for the discovery of peers. Instead, in our implementation of the scanning phase, devices fetching peers with the desired service will make a call to addServiceRequest() and then start running the method discoverServices(). The actual device is notified of discovered peers through the following listener callback for Bonjour protocol: setDnsSdResponseListeners().

It is important to note that the discovery of services is a state that remains active until the device moves to the `connecting` or `connected` states to setup a connection to a peer.

2) Connection: In the next step, after the discovery of peers, the application can initiate a connection to a chosen peer from the available ones through the connect() request. The way our service selects a peer providing the service is pseudorandom, uniformly distributed, avoiding to connect to the same device twice in a row unless there is only one device in the list of discovered peers. This forwarding technique, compared to a purely random one, showed a 10% improvement in the percentage of messages reaching all phones.

After a successful connection, the function requestConnectionInfo() is called to gather the group information. This information is stored in an instance of WifiP2pInfo class that contains the address of the group owner (groupOwnerAddress) and a flag (isGroupOwner) to indicate if the current device is the p2p group owner. From this moment, both devices are able to exchange content through a socket connection.

3) Data exchange: During the connection state, devices have the opportunity to exchange the content they have stored in their databases with the peer device they are connected to. The content handled in our experiments consists of basic strings of characters representing the message itself and related metadata, such as the message's universally unique identifier (UUID), the ID of the device that generated that message, the time when it was generated, time to live and the path it followed while being disseminated among the devices.

As it can be deduced from the attributes in a message frame, the message size can vary from one device to another, being the reachable average about 300 KB. Messages can increase size due to the path field because every time the content is transferred to a new device this information is stored in the path and so it grows. It is important to bear in mind that this field will not be included in practical scenarios, since the information it reports is only necessary for research purposes. The rest of each message is standard content with the same length which can be changed in practical scenarios depending on the advertisement subject, probably enlarged. These factors must be taken into account for the development of application dependent services, meaning that, if messages are to be very long it will be required to play with the TTL attribute in order to keep the list of messages within an acceptable size for future transfers. The simple yet efficient way in which devices transfer their data is explained in what follows.

For data exchange, Android devices will open a socket instance from the Socket Class in Android to perform the endpoints for communication between them. The device acting as a GO will open a server socket and the device acting as a client will open a client socket. Once the connection is established, as exemplified in Fig. 5, devices will send to each other a list with the messages ID they own. Upon reception of this list, devices check which of theses messages, based on the IDs, they are missing. In this way, they reply to the first device with a reduced list containing only the IDs of the missing messages. With this information, the peer device replies with a new list containing the complete messages requested before. A set of flags are also involved to track whether the devices have exchanged any data with the connected peer. When flags are set to 1, the data transfer is finished and the devices can close the ongoing connection.

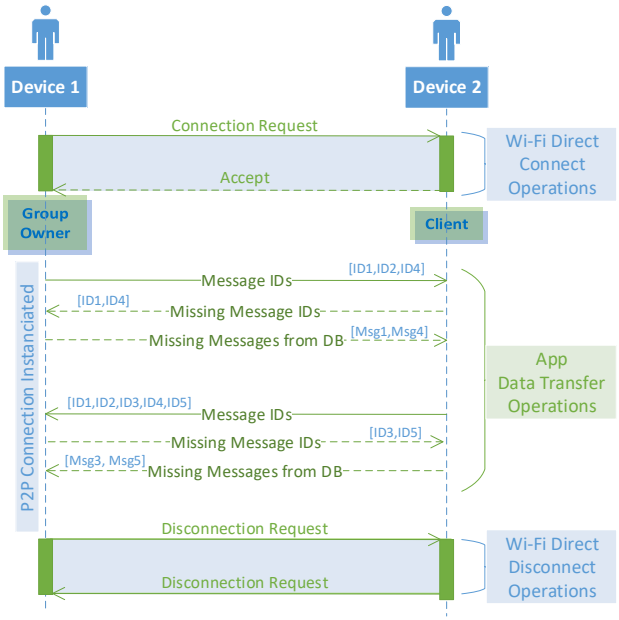


Fig. 5: Sequence Diagram of the service workflow.

4) Disconnection: It is noteworthy that if the whole process explained above takes longer than a certain time period previously set on the application (for API operational reasons), it will be dropped not to bias the application performance. WiFi P2P Android Classes do not contain a switching option, so that, we have to call the P2P API method removeGroup() to close the current connection. In this way the formed group is broken, and so, the connection is closed.

5) Failures: During the connection process a failure may occur. Failures are handled in different situations:

- Failure due to busy state of local system: This failure is detected right away the connection attempt if the system is not available to start a new connection.
- Failure due to timeout: This failure is determined after certain period of time attempting to connect to another device. If in the timeout interval the connection is not setup the connection attempt will be dropped. Thus, the process will start again.
- Failure within an established connection: This failure is detected when two devices establish a connection and for any reason, such as mobility, it breaks. A timeout process will check the state of the connection steadily, if the flags tracking the data transfer are not set to 1 at the moment of verification, the device will drop the connection assuming failure.

D. Database Implementation

It is important to bear in mind that, in wireless networks, links between sources and destinations vary frequently with the additional probability of link disruptions. Collisions are also of great concern since devices transferring data cannot get to know whether its dispatch experienced a collision. Due to these impairments, it arises the need to ensure the availability of contents from all devices by making a local copy of all messages exchanged or generated by them. Furthermore, if a device crashes or it is switched off, the application will continue storing the messages when it is turned back on.

For the implementation of a database to store the content of the application we used the Android SQLiteDatabase Class. This Class provides general methods to create and manage an SQL database.

E. Geofence Implementation

Based on our experiments and works like [1], real user mobility often results in the formation of clusters. To take advantage of this clusterization we direct our target to ensure content survival during a certain period of time and within a certain area of interest marked by means of a geofence. Thus, we set up a time to live (TTL) for every content generated and an Android geofence within which these data can be distributed.

It is important to note that these characteristics are application dependent so the application had to be made location-aware. In such a way, to create a geofence the application needs first to specify the latitude and longitude of a geographical region and a radius, then it will regularly update the user location to check his proximity to that region. Google Play services offers developers a library of APIs, mainly used to provide Android apps with core functionalities such as authentication services, privacy services, push notification services and, specially location services, from which we take advantage in our work. With the combination of Google Play services location APIs and creating geofence objects, we can periodically determine whether the user has entered or left the area, or is dwelling inside. Requests to the location class methods will return notifications informing about these events, know as *geofence transitions*. Upon receiving the geofence transitions notifications, some actions are executed by the application regarding the event occurred:

- Enter: If a user has entered the area of interest he is now flagged as available to establish connections with other users and exchange content.
- Exit: The service in the user's device will be no longer operating and it will also drop all the content related to the visited area, as commonly adopted in the literature (see, e.g., [1] and references therein). Indeed, messages are dropped once the device leaves the area to ensure the availability of content only where it is relevant.

Since scenarios where users gather are common, we believe that user's proximity to certain locations will lead in similar user's profiles. Thus, we aim at taking advantage of the content generated in a specific area in order to disseminate it locally to interested users by opportunistic communications.

F. Logs Reporting

Regarding the posterior application performance analysis, the system generates an incremental file including the events occurred up to that moment and stores it in the local database. Every minute the application will open a SCP connection to a server located in our research institute and upload the log file to that server. Therefore, collecting information logs requires Internet connection

This type of incremental storage is very useful specially when the Internet is down and logs cannot be sent to the server at a specific time. Considering that all the information about the performance of the application is locally stored, it can be uploaded, without missing any piece of the file, when the infrastructure is available or even retrieved directly from the devices if no Internet connection could be established.

Additionally, a synchronization system was required in order to fetch events in different devices, e.g., two devices reporting a connection to each other must happen at the same time, so that, we confirm that the logs information is trustworthy. To keep devices synchronized, we use an external application, called *Time Calibrator*, that connects to an atomic clock server in Switzerland to provide exact local time.

After the experiments are finished and the logs are stored at the server, we retrieve them to analyze the behavior of the service. For this purpose, we developed a parser in Python. In Fig. 6, the complete system procedure and the entities involved for data collecting, parsing and results analyzing are shown.

V. Experiment Evaluation

In our aim at producing realistic mobile scenarios, the more representative setups that come to mind are environments in which users move around carrying their own devices. One can find this type of settings not only in different kinds of social gatherings, such as shopping malls, football matches or concerts but also in busy metropolitan areas. For that reason, our framework for the experimental stage of the thesis consists in 22 smartphones carried by colleagues at our research institute. We used reliable yet not expensive mid-range devices, namely the Alcatel Pixi 4 (5) with Android 6.0 Marshmallow Operating System and a 2000 mAh Li-ion battery, providing Wi-Fi direct connectivity and running our application for wireless advertisement service, as described in Section IV.

From the total amount of devices we selected subgroups for the initial experimental phases, consisting of sets of 3, 5, 10 and 20 phones, having content generated every 10 minutes by each phone and assuming that all devices want to receive all messages, i.e., there is no profile discrimination. In this way, we can characterize the differences in the effectiveness of our service performance regarding the number of users. Besides varying the number of users involved in an experiment, it was also significant to collect results by testing time-dependent experiments, such as static gatherings or mobile encounters.

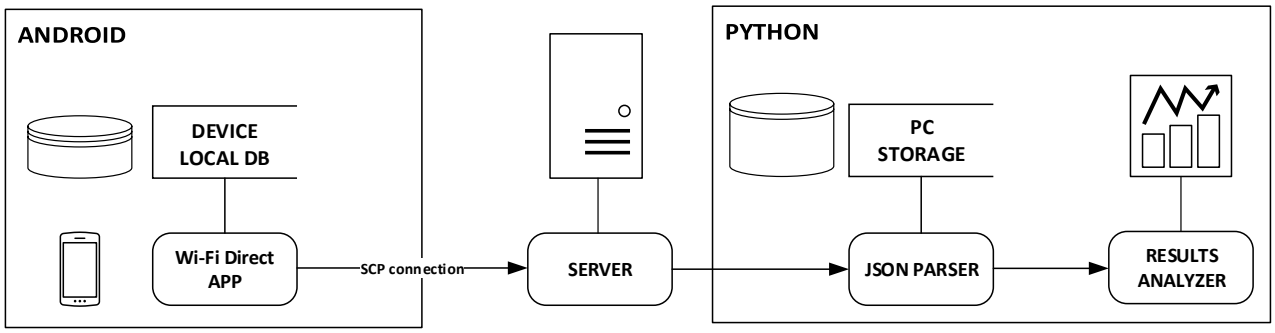


Fig. 6: Full system workflow diagram.

Static and dynamic human behaviors might depend on the time of the day. In next subsections we explain every particular scenario scheme, discuss the results and argue the reasoning.

A. Static Experiments

To begin with, we performed static gatherings of 1 hour duration represented by work meetings, lunch time or conference sessions. These common scenarios are useful to conduct experiments that will serve us to analyze the feasibility of the service in terms of connection establishment delay and data transfer duration.

In TABLE I, general first information about the experiments is depicted. There are three groups of observation divided by 3, 5, 10 and 20 devices according to the number of phones involved in the experiments. For every group we run 5 experiments of 1 hour each. As it can be observed, the number of connections attempts increases with the number of devices as it happens with the failed connections attempts, but not so with the number of successful connections, which stays in a close range even when varying the amount of devices. This situation is explained by the fact that the connection attempts are performed every 15 seconds by every device, which leads to a higher number of collisions in the band due to a higher access to the medium when the number of devices grows.

In spite of the dramatic growth experienced in the percentage of connection failures when adding more devices to the scenario, it has been proven by further experiments that a close to optimal loop length is 15 seconds. By reducing this parameter, as it can be interpreted from our results, a higher number of spectrum interferences occur. While increasing the length of the loop incurs in a loss in terms of makespan of messages. The concept of makespan refers to the amount of time needed for every message to reach all the devices involved in an experiment. Thus, it has been necessary to reach a trade-off by assessing the behavior of the service to benefit the end user satisfaction.

To assess the system in greater depth, we analyze the most relevant operational phases of our service separately, so that we can understand which parts of the process are susceptible to performance study and improvement.

TABLE I: Number of connections during 5 experiments of 1 hour each for different number of devices (with number of connections succeed and failed, and respective rates)

Dev	Conn. Attempts	Successes	Failures	S. Rate	F. Rate
3	2551	942	1609	37%	63%
5	5884	785	5099	13%	87%
10	10425	1057	9368	10%	90%
20	23318	1448	21870	6%	94%

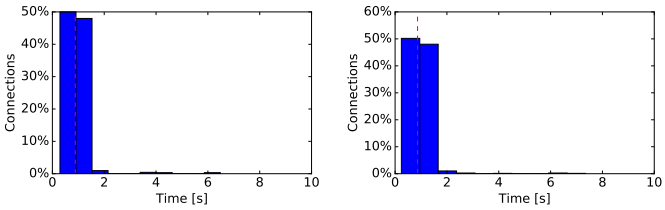
TABLE II describes the time spent in the `connected` state, which is the interval between the times when two devices have formed a connection, and when they disconnected. This phase performance is tightly related to the time needed to transfer data between a pair of devices. Being our system a prototype, as defined in Section IV-C3, messages sizes do not exceed 1 KB. However, in this specific scenario the amount of messages transfered in every contact grows with time, which means that at the end of the experiment with 10 phones generating messages every 10 minutes, a total amount of 420 KB could be transmitted in one contact. Bearing this in mind, we take a look at the results in which we are considering the successful connections where data transfers happened. As it is shown, the average time spent in data exchanges is less than 1 second.

TABLE II: Average duration between the `connected` and `disconnected` states for a specific number of successfully completed connections

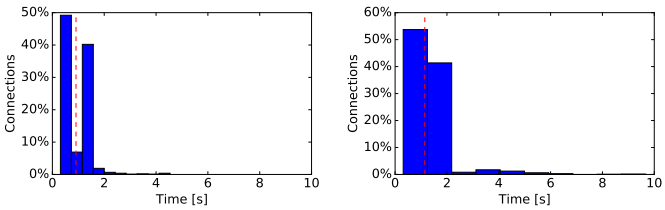
Dev	Connections	Average [s]	Standard Dev [s]
3	942	0.88	0.62
5	785	0.87	0.60
10	1057	0.91	0.54
20	1448	1.14	0.92

Moreover, Fig. 7 illustrates the distribution of the time spent during data transfers. From these results we can highlight that the performance offered by Wi-Fi Direct technology is suitable for our purpose at a first glance.

Nevertheless, data transfer is not the only critical part of the process. Indeed, many wireless technology standards for

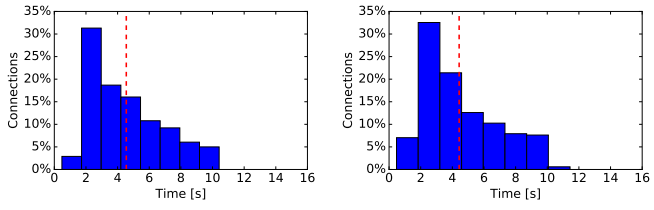


(a) 3 devices (b) 5 devices

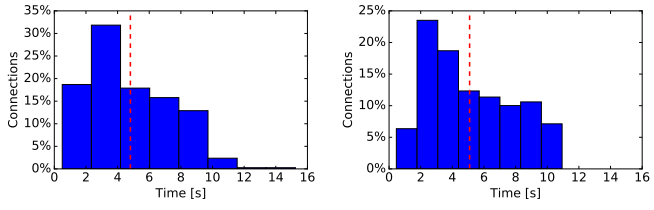


(c) 10 devices (d) 20 devices

Fig. 7: Histograms representing the time spent from the moment two devices have established a connection up to the moment they drop it.



(a) 3 devices (b) 5 devices



(c) 10 devices (d) 20 devices

Fig. 8: Histograms representing the time spent from the moment one device decides to connect to other device up to the moment they establish a connection.

exchanging data over short distances suffer from high delays when trying to establish a connection to a chosen peer device. Hence, the next step was to measure the average time spent in the `connecting` state, this is, from the moment a device chooses a peer to connect, to the moment the connection is established. It is noteworthy that, from the total amount of successful connections, only the subgroup of connections in which the device is starting the attempt can be considered in this measurement. Peer devices only receive a notification of invitation to connect, so that, they do not reveal useful information for this step.

TABLE III: Average duration of the `connecting` state in case of successful connection establishment

Dev	Connections	Average [s]	Standard Dev [s]
3	380	4.55	2.34
5	341	4.43	2.43
10	380	4.81	2.62
20	519	5.09	2.71

In TABLE III, it can be observed that the average time to instantiate a pairwise connection is around 4.5 seconds. This result largely outperforms the one offered by other protocols, such as Bluetooth that spends, together with data transfers, up to 12 seconds [1]. The time distribution given by the four different scenarios is represented in Fig. 8. From these graphics, the worst cases that can be found when reaching a peer device require about 10 seconds.

In this work, every stage of the proposed state machine has been deeply studied. Therefore, in addition to the main phases already described, we recognized failed connection as a possible factor that could bias the whole process if not taken into consideration. As explained in Section IV-C5, failures can be due to three different situations. Failures as a result of broken connections or timeout are detected every 15 seconds, just as a new loop takes place by default. On the other hand, failures caused by a busy state of the local system are instantly notified to the service, meaning that a new attempt of content spreading can be initiated without incurring any relevant negative impact in the dissemination process, in terms of delay.

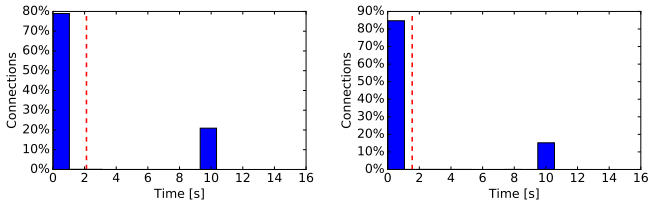
TABLE IV: Average duration of `connecting` state in case of connection failure

Dev	Connections	Average [s]	Standard Dev [s]
3	1602	2.12	4.06
5	5031	1.55	3.59
10	9294	1.50	3.57
20	21739	1.27	3.30

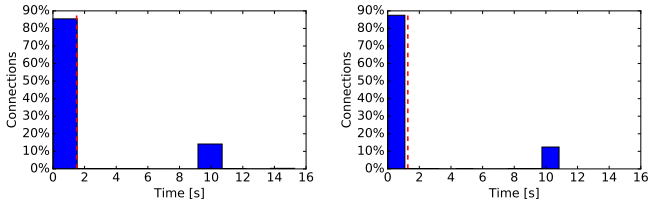
According to our previous analysis, the graphics in Fig. 9 illustrate very positive results in view of the fact that most of the failed connections are due to the busy state of the local system and thus they are promptly detected. More specifically in TABLE IV, an average time from connecting to failure is proved to be less than 2 seconds for most of the scenarios.

The last analysis of this section is related to the spreading of messages, their dissemination among devices and the average time it takes to accomplish the task in controlled and static scenarios. This results will be later compared to the performance obtained in dynamic environments.

TABLE V shows the following information based on the number of devices in the experiment: total number of messages generated during the experiment, number and percentage of messages that reached all devices, average time needed for the

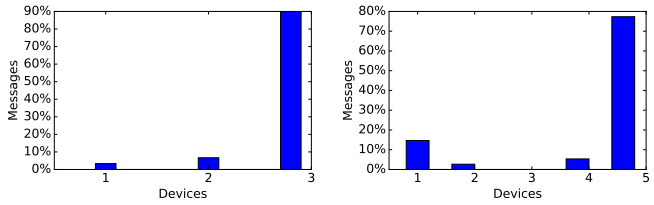


(a) 3 devices (b) 5 devices

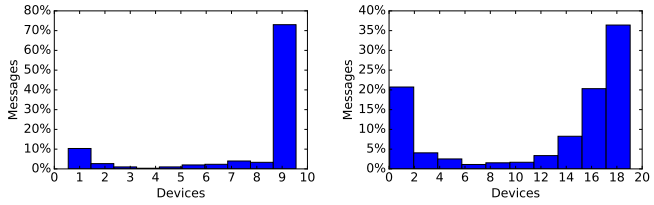


(c) 10 devices (d) 20 devices

Fig. 9: Histogram representing the time spent from the moment one device decides to connect to other device up to the moment it discovers to have failed on the attempt.



(a) 3 devices (b) 5 devices



(c) 10 devices (d) 20 devices

Fig. 10: Histogram representing the number of devices that were reached by every message during the experiment.

previous messages to reach all devices, known as makespan, and the standard deviation.

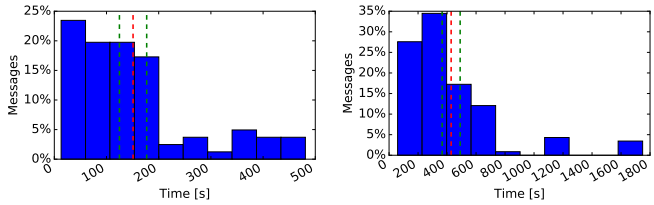
TABLE V: Average time needed for a message to reach all the devices

Dev	Msgs.	Reach.	Reach. %	Avg [s]	SD [s]
3	90	81	90%	150.86	117.38
5	150	116	77%	428.25	336.10
10	300	219	73%	952.40	438.31
20	600	114	19%	2217.70	474.72

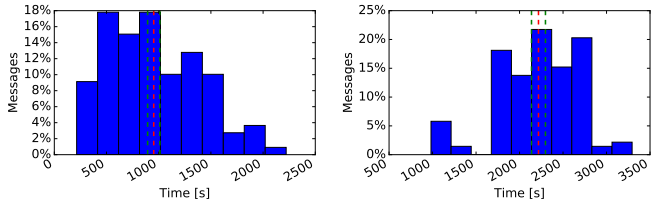
In static scenarios, as we can see in Fig. 10, the percentage of messages that reached all phones is fairly high, which means that the feasibility of the service for our main purposes has been confirmed. Additionally, in this first proof of concept, the service reaches a more than acceptable performance, leading to a good satisfaction of the end user.

In the graphics of Fig. 11, the average makespan of the messages that reached all the devices is illustrated. This result not only depends on the number of devices involved in the experiment but also on the number of messages generated, as presented before in TABLE V. Since every device generates content every 10 minutes, the higher the number of devices, the higher the number of messages.

In settings with a fixed number of messages and variable number of devices, the makespan is tested to exhibit an almost linear growth which can be preferable and beneficial in scenarios where not all users are allowed to contribute to the generation of content. For instance, in shopping malls where the only users interested in generating content probably related to offers are shops, a fixed number of messages can be generated every morning and remain available during working



(a) 3 devices (b) 5 devices



(c) 10 devices (d) 20 devices

Fig. 11: Histogram representing the distribution of time spent only by the messages that reached the maximum number of devices.

hours. To prove this point, in next section, we evaluate mobile experiments lasting an entire working day.

Finally, as per our analysis of the system, the Wi-Fi Direct protocol does not allow peer devices to know the state of the devices they discover around, meaning whether peers are currently attached to other connections so they are not available at that moment. In this way, we could not discriminate among peers in order to select the best one according to the requirements of the environment. Lacking the necessary information there is not an optimal choice for message forwarding, so that, among the possible options studied, a modified random

forwarding scheme has been the technique employed in our experiments, as detailed in Section IV.

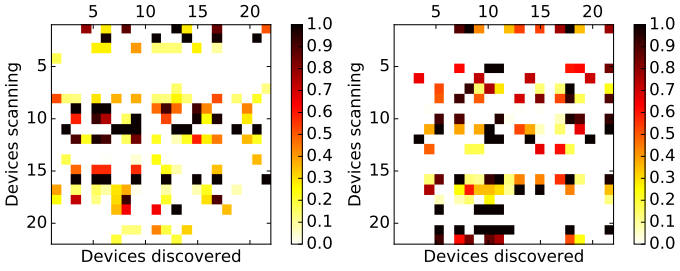
Given this election and keeping in mind the short period in which devices try to re-instantiate new connections, it was very likely that they would incur in failures in many of their connection attempts, as shown in previous results. For that reason, we strongly believe that a future update of the wireless protocol, with the required information, will provide the opportunity to make better choices regarding forwarding strategies which will derive in a reduced makespan time.

B. Mobile Experiments

In a posterior experimental stage, the 22 devices were distributed to people in a large office setting in order for the service to start operating in a small but practical environment. To give a better picture of the scenario, it consists of a three-story building with some open areas, labs and offices per floor where professors and students co-work. Results are based on the mobility and service performance of the devices during working hours (from 10 AM to 6 PM) along 5 days.

To get started, it is interesting to take a look to Wi-Fi Direct protocol efficiency regarding the scanning phase, in order to list the peers that can be found around every device. A common reasoning would be to expect a symmetric matrix including both sides of the scene, i.e., devices which are able to discover certain peers while scanning, can also be discovered from the other side. But it was proven that this is not the case.

During two different periods of time, from noon to 1:30 PM comprising lunch time, and from 4 PM to 5:30 PM which is not supposed to be a busy time slot, we compared what we call the visibility matrix of all devices in Fig. 12.

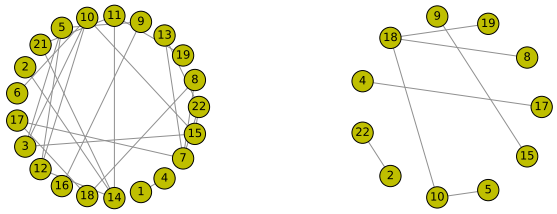


(a) 12 PM to 1:30 PM. (b) 4 PM to 5:30 PM.

Fig. 12: Visibility Matrices representing devices discovering each other.

The term visibility refers to the discovery of peers by every device, thus if a device discovers a number of peers these are said to be visible to the first one. Since the phase of scanning is carried out very often due to the loop process, each interval analyzed includes a large amount of samples, and the matrix displays warmer colors as the frequency of seen peers increases. In this way, we can find that even if the functioning of the protocol is far from perfect according to the symmetry of the discoveries, many devices reported new discovered peers during gathering hours as represented by the lighter colors on the left side matrix. Similarly, the matrix on the right displays the results for the same evaluation at later hours when people usually spend time working on their own. In this case, mostly darker colors can be appreciated since workers stay at the same place for longer, seeing the same peers with higher frequency. Of course, lighter colors may happen due to spontaneous meetings.

Furthermore, Fig. 13 depicts the contacts with data exchange between devices during the time slots described above. In Fig. 13a, all devices have eventually come into play and exchange data with a peer device, matching the visibility matrix at that time. Per contra, in Fig. 13b only a reduced group of devices contacted each other. It is worth noting that, in theses graphs only contacts with data transfer are represented and also only once, which means that the nodes in both circular graphs may have contacted other nodes whose edge is not drawn because they had nothing new to exchange. Besides, subgroups inside this reduced group can happen, meaning that the nodes can be placed in different areas, for instance, users carrying devices 5, 10,18 and 19 are located in the same area, while 2 and 22 or 9 and 15 are in two separated zones. We can argue the same for the left circular graph, where subgroups could have been formed for different groups of workers.



(a) 12 PM to 1:30 PM. (b) 4 PM to 5:30 PM.

Fig. 13: Groups of nodes exchanging data.

In the following results, we assess some of the parameters as we did for static scenarios and evaluate the main differences introduced by human mobility when using this kind of services.

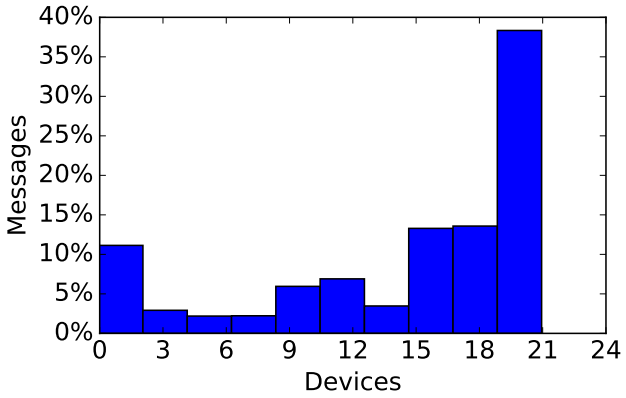


Fig. 14: Spreading of messages among the devices during 5 days of mobile experiments.

A histogram with the percentage of messages that have been spread among devices during the 5 days experiment is depicted in Fig. 14. As for the static experiments, the highest percentage of messages reached the total number of devices or a close amount of them, between 18 and 22 phones.

In Fig. 15, the cumulated number of devices reached by every message during the experiment is represented. In the x axis we have plotted the actual generation time of every message to give a big picture of the message performance during a whole day experiment.

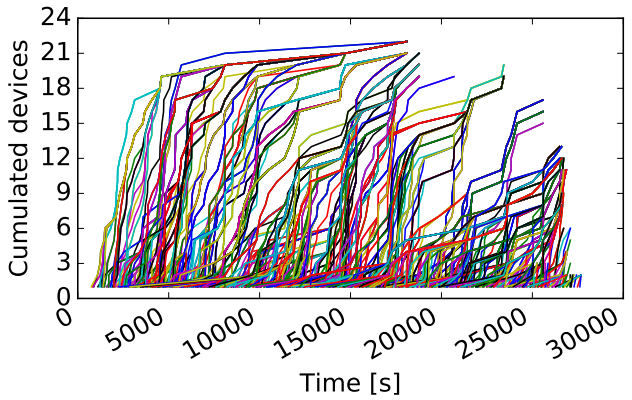


Fig. 15: Number of devices reached by every message during mobile experiments.

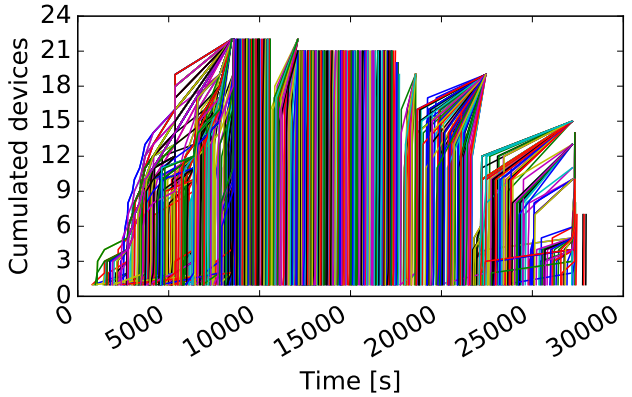
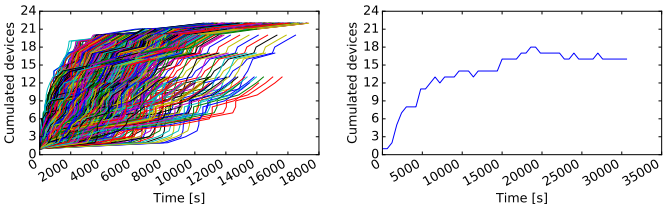


Fig. 16: Number of devices reached by every message during ideal case experiments

As a comparison, Fig. 16 illustrates the dissemination of messages generated during the whole day in an ideal environment, with the aim of comparing our results to frequently used ideal assumptions. As commonly done in the literature, we have considered an ideal case where connection instantiation and data transfer are instantaneous, based on the device visibility, i.e., each device is assumed to be able to transfer its entire database content to every peer discovered during the scanning phase, with no delay. In Fig. 16 we can observe that, specially when approaching lunch time, when device carriers gather, devices can discover most of the existing peers. Thus, messages are spread right away to the maximum number of devices. Being this situation far from the results obtained in the real experiment, we would like to remark the significant contribution of our experimental analysis.

From the previous plots, it is straightforward to think that messages generated in the first period of a daily experiment would travel further than those generated later on. The latter messages could influence the graphics in such a way that the reader has the wrong impression that there exists some messages stuck in a small number of phones and will never accomplish its dissemination task. In the following analysis, we focus on the behavior of the messages spreading during the mobile experiments in pursuance of a demonstration for the previous statement. In Fig. 17a, each message is plotted as a line whose generation time has been shifted to t=0. In such a way, we can appreciate the evolution of the spreading between devices as every message ages. For a clearer representation, we appended Fig. 17b with the average of devices reached by the contents within 10 minute slots. In these graphs, we observe a not very rapid growth which, according to our assumptions, is due to the continuous generation of new messages.



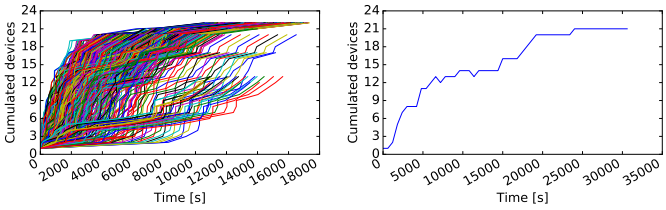
(a) Whole day experiment. (b) Average whole day experiment.

Fig. 17: Cumulated number of reached devices since the time of generation for each content and the average during a whole day experiment.

To assess this case, we differentiated among the generation time of the messages in order to portray their dissemination performance during the morning and the afternoon periods. In Fig. 18 and Fig. 19, it is clear that messages generated during the morning reach the maximum number of devices at the last stages of the experiment while messages generated during the afternoon only approach half of the total amount of devices. Thus, as shown in Fig. 20, all the earliest messages, except for a negligible percentage, reached a significant amount of devices during the experiments. This evidence leads us to support our previous assumptions made during the static experiments evaluation section (V-A). Indeed, we expected that for certain scenarios, such as shopping centers, messages generated in the morning would obtain a greater availability achievement which is now confirmed.

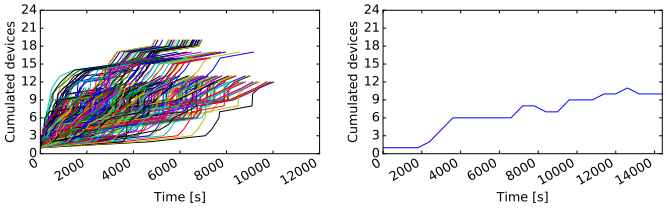
It is also important to point out that, as mentioned before, morning periods, in which there are more gatherings, present greater message mobility than the latest hours of the day, when people work on their own. This can be seen in Fig. 18b, when the morning period finishes around midday (14400 s in the x-axis of the figure), the number of phones reached is higher

than at the end of Fig. 19b, (25200 s). In this way, we also demonstrate that human mobility has a positive effect in the dissemination of messages.



(a) 10 AM to 2 PM. (b) Average morning experiment.

Fig. 18: Cumulated number of reached devices since the time of generation for each content and the average during the morning period of a whole day experiment.



(a) 2 PM to 6 PM. (b) Average afternoon experiment.

Fig. 19: Cumulated number of reached devices since the time of generation for each content and the average during the afternoon period of a whole day experiment.

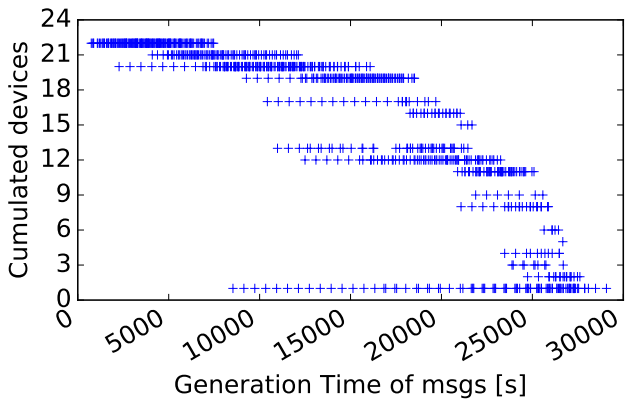


Fig. 20: Correlation between generation time and number of reached phones.

As a closure of the experimental evaluation section, in Fig. 21 we can observe the distribution of the makespan for messages that reached the highest numbers of devices, from 18 to 22 phones. To achieve this goal, the average time needed was less than 3 hours and a half. We consider this average to be large and, as explained in the analysis for the static experiments in Section V-A, despite the fact that makespan is one of the parameters that could be greatly improved by benefiting from more accurate forwarding strategies, still allows for a class of advertisement with loose timing requirements or even for spreading non-immediate content, such as offers or news.

Consequently, from dynamic experiments we can deduce that human mobility can cease to be seen as a gap or limitation and start to be considered a big asset in the spreading of relevant content to users located by a point of interest or surrounding areas.

TABLE VI: Average duration for the messages to reach between 18 and 22 devices during the mobile experiments

Dev	Msgs.	Reach.	Reach. %	Avg [s]	SD [s]
22	2739	1641	60%	12170.61	4267.56

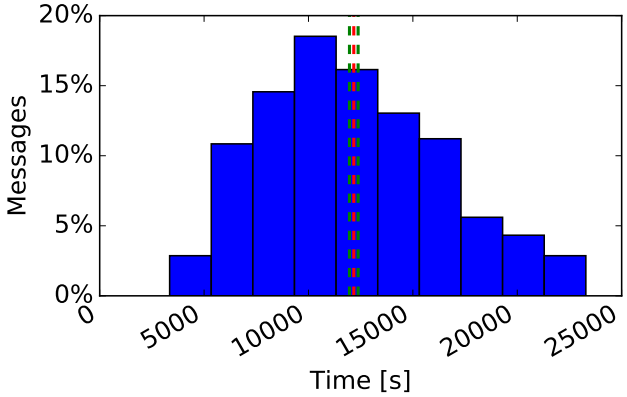


Fig. 21: Distribution of the makespan of messages that reached between 18 and 22 devices during the mobile experiments.

C. Battery consumption

Along with the analysis above, we kept track of every device battery consumption during the dynamic experiments. It is noteworthy that, in the course of the experiments, no other application has been running but our service. In such a way, we could quantify the battery draining of the devices due to the recurrent use of Wi-Fi Direct protocol operations.

Surprisingly, this continuous execution is not as consuming as we expected. In Fig. 22, we can observe a plot of the average battery consumption for 8 devices during the last 2 hours of a daily experiment. Quantifying their battery life over time, we obtained an average value of 9% of battery consumption per hour which leads to a final drain, during a whole day experiment, of approximately 60%.

To obtain a detailed analysis about the previous percentages, we accessed the smartphone settings menu which provides us with the specific breakdown information about battery consumption. This information gives us the battery consumption statistics of the phone from the last full charge, so we took a couple of phones and run the experiment for one hour approximately right after unplugging them. Fig. 23 depicts a graph for each phone with the energy consumed during

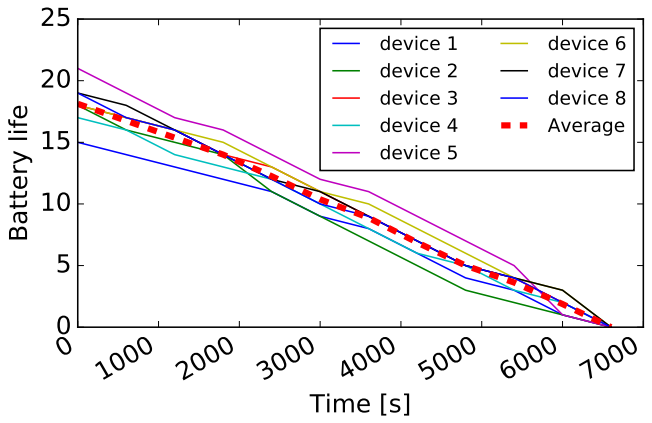
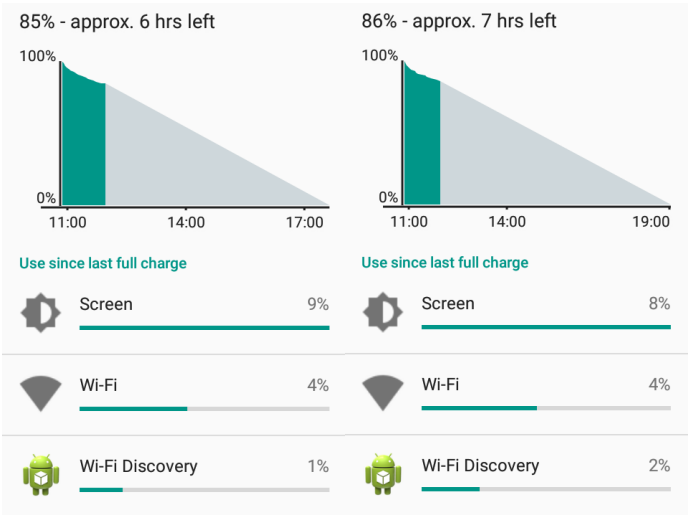


Fig. 22: Average battery life for 8 phones during 2 hours in a mobile experiment.

the experiment and expected battery life, plus the breakdown percentages that corresponds to every operational entity.



(a) Device 1 (b) Device 2

Fig. 23: Battery usage by different smartphone applications.

From the total battery usage in one hour, we can appreciate that, only 1-2% of that amount is due to our service, Wi-Fi Discovery, corresponding to a computed power used of 42 mAh and 57 mAh for these specific cases. Besides, 4% (115 mAh) of the usage is due to Wi-Fi connectivity which includes connections to the server over Wi-Fi and D2D connections over Wi-Fi Direct. As stated earlier, Wi-Fi connections to the server are only needed to upload the event logs for its posterior evaluation, which implies that this feature will be removed from the actual application when integrated in practical scenarios, avoiding then a significant part of the Wi-Fi battery usage.

Finally, around 60% of the total power used by the devices deals with the use of the screen and can be pulled apart from our service achieving reasonably fair power consumption values. Our conclusion from theses measurements is that, admitting clear limitations regarding our realistic paradigm, the Wi-Fi Direct standard itself offers very efficient operations in terms of power consumption.

VI. Related work

D2D communications introduce a large number of opportunities for delay-tolerant networks formation that can enhance a wide range of research areas, such as content delivery together with location-aware systems and/or context-aware computing, sensor systems, failure handling techniques and even vehicular ad-hoc networks (VANETs). For this reason, a number of researchers have already recognized some of the fields where this paradigm can be applied. In next sections we chronologically present the evolution of the related researches and provide a precise description of their contributions and distinctions to our work.

A. Early stages

Building services without the need of an infrastructure has attracted the interest of researchers since almost a decade. In the first place, one of the most common topics that takes advantage of opportunistic networks is known as *floating content*, also referenced in some works as *content sharing*. The idea is basically the same, the distribution of pieces of content among a number of nodes located in certain areas, named anchor zones. In the early stages of the field research, theoretical works as in [13] studied the viability of this concept, focused on the expected lifetime of the content, by means of analytical models under some predefined criteria. The authors provided mainly results related to node encounter rates and mean contact times in a system entirely dependent on the mobility of devices and opportunistic encounters between them. These analytical results were applied to different fundamental mobility patterns varying also other parameters, such as the communication range, and finally validated by simulation experiments.

In [16], the authors conducted a similar work to the previous paper adding an urban scenario evaluation in the city of Helsinki. While characterizing the floating content performance, analytical results proved that urban environments were profitable for the implementation of this type of services even with different nodes densities. They showed that, with some geographical limitations, the content sharing concept was still feasible. They also presented evaluations on synthetic mobility models arguing that factors like real interaction patterns in human mobility or social context would not influence their findings. This argument could not be refuted since, a major limitation back to the time of previous works was the emerging technological development of short range wireless protocols, which only allowed researchers to assess their theoretical frameworks in simulated environments. First standards, such as Bluetooth, reached 90 percent penetration in all mobile phones by 2014.

As a backbone for related work to be further investigated and deployed, we consider that the authors in [8] have looked

at the need for opportunistic and delay-tolerant networks with a very appropriate and precise approach that can be applied to a large number of D2D communication scopes. That work claims that opportunistic networks provide the required communication support for future application scenarios that will motivate opportunistic computing challenges leading to an enhancement of user experience. They addressed not only opportunistic computing tasks, but also the exploitation of user's social behavior to disseminate information and distribute resources. It was also pointed out the striking increase of user-generated content (UGC) which, from our perspective, will leave aside the old client-server structures because, as stated in the paper, all users will become information producers and consumers. Part of the motivation for our own research comes from ideas also identified in that work. Scrutinizing their outlook, we converged our objectives to overcome the well-known challenges posed by user's mobility and enforce the opportunities offered by human encounters. Despite that for the time being it is not easy to implement, we plan to follow user-centric approaches to further exploit the nodes mobility in order to find optimal content forwarding strategies. Lastly, we would like to remind that, even thought the call for protocols related to service discovery and information dissemination in a disconnected scenario is being accomplished by different standards, tasks related to information and resources management depending on the application are still under development. To our understanding, a way to tackle this gap lies in the implementation of specific upper layer applications that follow the basis of a secure connectivity and exploits context information and preferences from the users. In this way, the communication protocol can be isolated and eventually accessed, in a seamless manner, to any kind of application-level service that requires its functionality.

During the above mentioned period, there were also some researchers that tried to sidestep the most popular standards, like Bluetooth, IEEE 802.11 (in ad-hoc mode) and Wi-Fi Direct, relying on the argument that these technologies presented severe limitations at that time and that they were not going to be improved or solved in a coming future. In [19], authors proposed an alternative opportunistic communication system, named WiFi-Opp, including the use of stationary and mobile APs. They evaluate the performance of the setup by simulating content spreading on real-world contact traces. Results of WiFi-Opp and ad-hoc 802.11 dissemination performance were compared, based on the amount of data each node distributes, as well as the energy efficiency for both approaches. That work also presented an implementation of the service as a proof of concept but no further investigation on it has been shown.

B. Services built on top of specific technologies

Given that ad-hoc 802.11 has stopped attracting attention and Wi-Fi Direct has begun to be more commonly integrated into some of the users mobile devices, works like [7] and [4] started exploring the viability of opportunistic networks creation on top of Wi-Fi Direct protocol. Both works evaluated the protocol performance in terms of discovery, group formation and connection delays and tried to characterize accurate configurations for small and static scenarios, but they did not assess this behavior including realistic scenarios mobility. For that reason and, in sight of the expected penetration of Wi-Fi Direct, these studies' results and their experimental setups involved a motivation for later development of new services.

For instance, authors in [14] presented PASA, a local message dissemination system based on a new communication model called *passive broadcast*. The system combines Bluetooth and Wi-Fi Direct technologies to support two different content distribution algorithms analyzed in the paper, and takes advantage of devices location. They also performed small scale real experiments using Android smartphones and simulated experiments for larger number of devices. Similarly, mQual service [20] extended a method of Wi-Fi Direct protocol in Android to develop an application for efficient creation an maintenance of networks that better adapts to dynamic environments. They carried out experiments simulating dynamic scenarios with 5 phones and changing the network topology over time. Results showed that mQual algorithm outperformed original Wi-Fi Direct standard.

Further ahead, services like HYChat in [13] came out in an attempt to provide social networks in proximity. This work implements D2D with the aim of providing an interactive chatting system to users located in a nearby range, then switching to OTT to maintain this capability to users that move out of each other's radio range. Even thought that work gave us a good insight for future applications, their scope consists in allowing end users interactive communication in a pairwise manner which means that, unlike our approach, no opportunistic networks creation is being considered.

C. Architectures exploiting topologies and grouping techniques

More recently, smartphones have incorporated enhanced versions of the short range standards mentioned during this work. For this reason, a panoply of innovative testbeds and analysis have been devised in some studies with the objective of bringing new schemes that fill the gaps of D2D communication paradigm. In paper [5], the authors proposed a logical network topology in order to enable bidirectional inter-group communication. The deployment of this setup relies on a fully connected network for data transfers and is implemented in different Android devices. Likewise, in [12] the authors purpose is also to enable communication among multiple Wi-Fi Direct groups. Here, interesting solutions are proposed for static multi-hop ad hoc networks creation and the exploitation of them for the interconnection of these groups. In the performance evaluation, they quantify the time and energy employed for data transfers between two groups based on both proposed solutions, time sharing and simultaneous connections, and discuss the results for the best configurations achieved. Even so, higher implementation challenges were essential to go one step forward and enable the inclusion of these mechanisms as a standard. In contrast, we use only

standard tools and legacy operating systems, so that our app can run on top of commercial off-the-shelf, *unrooted* devices.

Increasingly, more relevant schemes to our approach are arising in the field of D2D communications. In work [1], complementary studies to our analysis can be found. Basically, the key concepts evaluated in that work are the efficiency with which content is accessed by devices involving users mobility and connectivity, as well as the survivability of that content. To that aim, the authors developed an Android application to be installed in participants devices for real experiments, in that way they could assess the performance of the application and characterize users mobility patterns. Based on a similar interest, we extended these analysis to provide a deeper insight of the feasibility of content persistence in opportunistic networks created without requiring current infrastructure. Unlike our service that is implemented on top of Wi-Fi Direct standard, the work referenced is based on Bluetooth communication protocol presenting greater limitations in terms of delay, pairing failures and battery consumption. These drawbacks, and statistics depicted in works like [4] and [5] where Wi-Fi Direct outperforms compared protocols, motivated our choice for that technology to conduct new experiments.

D. Applications of D2D to other research fields

Finally, some researches have been made for purposes other than those previously discussed in this section but incorporating similar mechanisms to achieve their objectives. For instance, a critical issue in the association of devices over Wi-Fi Direct is authentication to provide secure communications. With this aim, in [17], the authors analyzed potential security threads of Wi-Fi Direct and introduced a key agreement protocol. Another subject of controversy being studied is data dissemination for public safety. In this case, researchers hold that relying on Wi-Fi Direct capabilities it would be possible to distribute information in a sufficiently efficient way to ensure public safety in cases of unexpected events that would incur in a malfunctioning of the current infrastructure. In the past few years, there has been also a rise in studies not only about the comparison but also the aggregation of Wi-Fi Direct and LTE D2D. For the former case, works like [6] discuss the trade-offs in the performance of both protocols for operations related to management, data transfers latency and energy consumption over the radio spectrum. In the latter case, enhancement of cellular networks is conceived as a need, thus significant works like [3] devised novel protocols to introduce D2D communications on top of LTE cellular infrastructure. Lastly, a line that we consider will experience a great impact in the coming years is D2D in the context of VANETs. Relevant works already arose in this field, such as [18], where a direct D2D communication scheme between vehicles is introduced. Additionally to simulation with real datasets, the evaluation comprises a practical set up where the system was implemented for smartphones aboard 35 vehicles during 2 months experimental study. Similarly in [2], counting on a fleet of 370 taxi cabs, authors collected real traces from the city of Rome. Benefiting from this dataset, they run experiments to test data dissemination through possible VANET protocols leveraging on opportunistic communications.

In these works, researchers joined the necessity for future enhanced vehicular networks, the visible proliferation of smartphones and mobility-driven human encounters to pave the way for innovative D2D communication perceptions.

VII. Conclusion and future work

In this thesis, having identified the massive increase in the number of networked individuals in quotidian activities, we have developed and tested an Android application based on Wi-Fi Direct to support D2D infrastructureless communication services between mobile devices through opportunistic networks. By installing the application in smartphones and conducting a collection of real experiments with colleagues at our research center offices, we have quantified the service performance in static and dynamic environments in terms of connections establishment and data transfers delays, as well as energy consumption. The evaluation results revealed that content can be disseminated among devices in an efficient way, thus confirming the feasibility of implementing D2D services for diverse scenarios, both to assist crowds and to serve smaller settings enabling delay-tolerant networks.

We see this thesis work as an initial step toward further developments regarding different research fields. There are numerous mechanisms that we consider of interest to be included within this type of services, such as, profiling, groups discrimination, content labeling, as well as the incorporation of any context-aware solution already proposed. In this way, dissemination can be enhanced if content is to be directed to specific audiences. Regarding efficiency in content forwarding, even though models have been introduced to tackle this issue, from the protocol side there is a shortage of information exchange about devices status that we believe will provide the necessary knowledge to set up the proper forwarding strategies. Furthermore, as future work we plan to integrate an indoor localization system to the service in order to select better peers accordingly to distances among devices and their mobility. Finally, we would like to remark that, albeit our service has been proved to work on top of the existing Wi-Fi Direct technology, many limitations have been encountered during the development and evaluation processes which are detailed in Section III-B. Thus, we want to promote the development of standards that better satisfy the requirements of this paradigm to implement disruptive cooperative applications and service models.

References

[1] S. Ali, G. Rizzo, V. Mancuso, and M. Ajmone Marsan. Persistence and availability of floating content in a campus environment. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2326–2334. IEEE, 2015.

[2] R. Amici, M. Bonola, L. Bracciale, A. Rabuffi, P. Loreti, and G. Bianchi. Performance assessment of an epidemic protocol in vanet using real traces. *Procedia Computer Science*, 40:92–99, 2014.

[3] A. Asadi and V. Mancuso. Wifi direct and lte d2d in action. In *Wireless Days (WD), 2013 IFIP*, pages 1–8. IEEE, 2013.

[4] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE wireless communications*, 20(3):96–104, 2013.

[5] C. Casetti, C. F. Chiasserini, L. C. Pelle, C. Del Valle, Y. Duan, and P. Giaccone. Content-centric routing in wi-fi direct multi-group networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–9. IEEE, 2015.

[6] M. Condoluci, L. Militano, A. Orsino, J. Alonso-Zarate, and G. Araniti. Lte-direct vs. wifi-direct for machine-type communications over lte-a systems. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pages 2298–2302. IEEE, 2015.

[7] M. Conti, F. Delmastro, G. Minutiello, and R. Paris. Experimenting opportunistic networks with wifi direct. In *Wireless Days (WD), 2013 IFIP*, pages 1–6. IEEE, 2013.

[8] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer*, 43(1), 2010.

[9] Creative Commons Attribution 2.5. Android Wi-Fi Peer-to-Peer (available online at https://developer.android.com/guide/topics/connectivity/wifip2p.html), 2017.

[10] Creative Commons Attribution 3.0. Geofence (available online at https://developers.google.com/android/reference/com/google/android/gms/location/geofence), 2017.

[11] W. K. Edwards. Discovery systems in ubiquitous computing. *IEEE Pervasive Computing*, 5(2):70–77, 2006.

[12] C. Funai, C. Tapparello, and W. Heinzelman. Enabling multi-hop ad hoc networks through wifi direct multi-group networking. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*, pages 491–497. IEEE, 2017.

[13] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott. When does content float? characterizing availability of anchored information in opportunistic content sharing. In *INFOCOM, 2011 Proceedings IEEE*, pages 3137–3145. IEEE, 2011.

[14] Y. Mao, J. Wang, J. P. Cohen, and B. Sheng. Pasa: Passive broadcast for smartphone ad-hoc networks. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE, 2014.

[15] V. F. Mota, F. D. Cunha, D. F. Macedo, J. M. Nogueira, and A. A. Loureiro. Protocols, mobility models and tools in opportunistic networks: A survey. *Computer Communications*, 48:5–19, 2014.

[16] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju. Floating content: Information sharing in urban areas. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 136–146. IEEE, 2011.

[17] W. Shen, B. Yin, X. Cao, L. X. Cai, and Y. Cheng. Secure device-to-device communications over wifi direct. *IEEE Network*, 30(5):4–9, 2016.

[18] X. Sun, S. Hu, L. Su, T. F. Abdelzaher, P. Hui, W. Zheng, H. Liu, and J. A. Stankovic. Participatory sensing meets opportunistic sharing: automatic phone-to-phone communication in vehicles. *IEEE Transactions on Mobile Computing*, 15(10):2550–2563, 2016.

[19] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre. Wifi-opp: ad-hoc-less opportunistic networking. In *Proceedings of the 6th ACM workshop on Challenged networks*, pages 37–42. ACM, 2011.

[20] H. Zhang, Y. Wang, C. C. Tan, and Y. Zhang. mqual: A mobile peer-to-peer network framework supporting quality of service. In *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*, pages 754–755. IEEE, 2015.

[21] J. Zuo, Y. Wang, Q. Jin, and J. Ma. Hychat: A hybrid interactive chat system for mobile social networking in proximity. In *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*, pages 471–477. IEEE, 2015.