Infrastructureless Pervasive Information Sharing with COTS Devices and Software

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*—Information sharing is becoming a relevant issue for mobile broadband operators, due to the increasing popularity of social networks, to the increasing volumes of shared information, and to the steady increase in the number and capabilities of mobile devices connected to the Internet. Offloading information sharing services from the cellular infrastructure to device-to-device (D2D) communications can offer a welcome reduction of traffic. This paper discusses experiments with a smartphone information sharing application that can be used on commercial-off-the-shelf devices, with no need to root the device's software. In order to avoid unrealistic assumptions on the behavior of D2D communications, this work includes and builds upon the implementation of an Android application that supports infrastructureless distributed content sharing among wireless devices using Wi-Fi Direct. The collected experimental data permit a detailed analysis of the occurring events, and a careful assessment of the performance of pervasive information sharing services. Our experiments reveal that many assumptions commonly used in the literature do not hold in real settings. We conclude that delay-tolerant services can be supported, albeit we also show that high densities of devices can (somewhat counter-intuitively) impair performance.**

I. Introduction

Infrastructureless information sharing services are becoming important [1], also thanks to the availability of technologies such as Bluetooth or Wi-Fi Direct that have been developed to allow devices to communicate without requiring a fixed wireless access point [2]. Most of today's portable devices not only can access mobile broadband (MBB) networks, but they are also equipped with the hardware and software necessary to support device-to-device (D2D) communications [3].

So far, most of the approaches to the performance evaluation of such infrastructureless services were based on models and simulation [4], [5]. Experimental works like [6], [7], [8] have either used Bluetooth and Wi-Fi Direct to predict the performance to be expected in real environments with very small numbers of devices in entirely controlled and static scenarios, or simply evaluated the performance of D2D protocols rather than of the services that can be built on top of them. Application programs for mobile devices (*apps* for short) based on Wi-Fi Direct and/or Bluetooth have been developed in order to efficiently create peer-to-peer (P2P) connections to support direct message exchange, although relaying on a centralized server to determine users location and interests for improved peer selection [9], or to carry out local data dissemination while minimizing the process time, in absence of mobility [10]. Indeed, 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 [11] and [12], the authors present an app similar to the one we will describe later, although based on Bluetooth rather than Wi-Fi Direct, which also incurs severe issues with bandwidth limitation, time for peering, service detection and implementation incompatibilities among different device manufacturers, as unveiled by the authors of [13].

978-1-5386-4725-7/18/$31.00 2018 IEEE

The work we report in this paper concerns an experimental study of the performance of information sharing services that do not require infrastructure support from MBB operators, while aiming to pervasively spread messages among groups of devices that subscribe to one or more classes of information (e.g., commercial advertisements, local news, traffic warnings, etc.). Since infrastructureless pervasive mechanisms strongly depend and rely on the mobility of devices, this work shows if and how state-of-the-art technologies like Wi-Fi Direct, which is available in commercial-off-the-shelf (COTS) devices, can be used for—and impose limits to—information sharing services accessed by real users. Our work's contribution supports the idea that the pervasive use of devices, along with human mobility, will lead to a wide range of opportunities to create opportunistic networks, and thus benefit information dissemination and acquisition. However, we show that using WLAN-like coverage for D2D, although devices reach much shorter distances than advertised, can have a detrimental impact on information dissemination speeds, depending on the user density. In general, technological limitations are still huge. Specifically, the contribution of this work is fourfold: (i) assess the feasibility of infrastructureless pervasive data sharing networks to provide connection establishment between Wi-Fi Direct devices, (ii) discuss the limitations imposed by Wi-Fi Direct and by COTS devices and operating systems, and (iii) understand the key mechanisms that affect the performance of the class of services under evaluation.

The rest of this paper is organized as follows: We describe the design of our Android app in Section II. Section III illustrates the experimental setup and results of our research. A brief compendium of related work is presented in Section IV

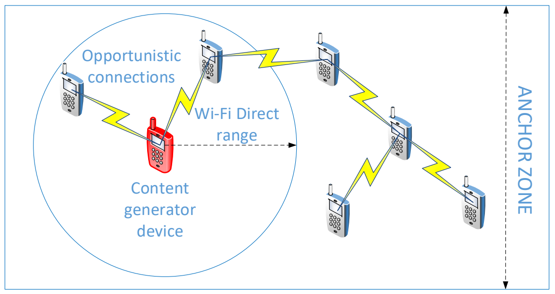


Fig. 1: Infrastructureless pervasive information sharing.

and, finally, conclusions are presented in Section V.

II. An App for Pervasive Information Sharing

We aim at creating a pervasive information sharing framework in which devices carry and exchange messages without the help of a network infrastructure. The relevance of messages is limited in space and time, i.e., information has to be spread within a limited region and before a deadline. These are parameters to be selected according to the nature of the information carried by the devices. Here we limit our study to the service *template*, rather than exploring the performance of specific services that could be built on top of such template. The typical operational scenario is depicted in Fig. 1, in which devices running the information sharing app generate messages to be delivered to any other device running the same app within a delimited region called "anchor zone".

Our work uses similar techniques as proposed in [8] and [14], although, differently from those works, we do not rely on already set topologies and architectures, and do not limit the scope of our work to static deployments. We built instead a pure dynamic opportunistic network [15], [16].

We adopted Wi-Fi Direct on COTS Android devices, without using root user's privileges to run any software. This choice makes our work realistic for the deployment of real services, with the real limitations and security constraints of existing and commercial operating systems.

A. Design of App and Information Sharing System

We have used the *geofence* mechanism [17] to delimit the anchor zone in shape of a circle around a programmable point. In addition, we set up a *time to live* (TTL) for every message generated, which is carried by the message itself.

Given the limitations of Wi-Fi Direct in Android, and the fact that user mobility makes groups unstable, we decided to enforce one-to-one links only, with a *Group Owner* (GO) and a *Group Member*, i.e., a client, in each group (this has a cost in terms of performance as we show later, but is necessary in high mobility scenarios). Groups are formed using the *persistent* procedure [18].

Our app operates in an *epidemic* manner [16] trying to connect to as many other devices as possible and *infect* them by forwarding all the messages it stores locally (either self-generated or previously received).

As represented in the state machine of Fig. 2, when a device enters the marked geofence it starts scanning the

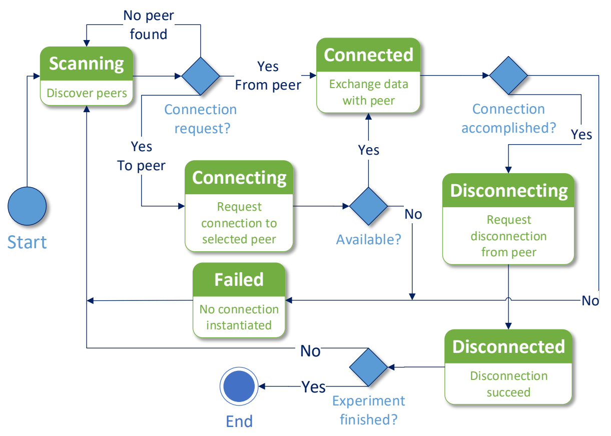


Fig. 2: State Machine of our app.

social channels looking for potential peer devices. This is the `Scanning` state. While the scan is running, any other device can send an invitation request to the first device, so the latter can move to the `Connected` state directly. A second option is to keep scanning, and connect to one of the peers discovered when the scanning time expires. At this point, the device sends an invitation request to the chosen peer, which implies going through the `Connecting` state and then to the `Connected` state when and if the peer accepts the invitation. There is also the possibility that no peers are discovered by the device, so that a new scanning phase begins. When in `Connected` state, paired devices exchange data with each other, and, right after, both of them enter the `Disconnecting` state. The next state, `Disconnected`, is reached in the exact moment that the app is notified with a disconnection callback triggered by 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 fails in the connection attempt. If the peer device is busy for a long period of time, it never accepts the invitation request, so that the device attempting to connect fails after a timeout. In both cases there is a transition to the `Failed` state.

Both `Disconnected` and `Failed` states will eventually lead back to the `Scanning` state to repeat the process.

B. Implementation Highlights

Here we describe the key features of the app we have developed using the Wi-Fi P2P API and other Android APIs.

1) Discovery of peers: The discovery of available peers in our implementation is operated in the state called `Scanning`. Since our app does not attempt to connect to every physical device in our experiment, but only to the peers that are running the same service, we use pre-association service discovery (Bonjour, which is DNS-based) to advertise the services a device is running before the connection setup.

2) Connection: After the discovery of peers, the app initiates a connection to a peer chosen at random, excluding the last paired device, unless it is the only one in the list. Hence, we balance out link selection attempts, with no need to keep track of the full history of connections. Note that, beside being not practical, keeping track of connections is not useful neither in mobile environments nor when devices keep generating new messages to spread, like it occurs in our case and is common for advertisement and other messaging applications based on the social networking paradigm.

3) Data exchange: The content handled in our experiments consists of basic strings of up to 300 B, including some metadata used to analyze the service performance, such as the universally unique identifier (UUID) of the message, the ID of the device that generated that message, the time when it was generated, the time to live and the path it followed while being disseminated among the devices. Once the connection is established, devices send to each other a list with the UUIDs of the messages they have, and only missing messages are exchanged.

4) Failures: In the `Connecting` and `Connected` states, failures may occur. A failure due to *busy state* of the local system is detected right after the connection attempt if the system is not available to start a new connection. A failure due to *timeout* is determined after a certain period of time attempting to connect to another device. A *data transfer* failure within an established connection is detected by means of another timeout that limits the connection time.

5) Geofence: The app specifies latitude and longitude of a geographical region and a radius. Thanks to Google Play services location APIs, we can periodically determine whether the device has entered or left the anchor zone, or if it is dwelling inside. If a device leaves the anchor zone, the service will be no longer operating and it will also drop all the information collected in the anchor zone, as suggested in the literature (see, e.g., [11] and references therein).

6) Logs and data analysis: Each device generates an incremental file including the events occurred in a local SQL database, using standard Android features and services. After the experiments are finished and the logs are collected at a central server, we retrieve them to analyze the behavior of the service. For this purpose, we developed a parser in Python.

III. Experiments

In order to test the behavior and the performance of our information sharing app based on Wi-Fi Direct, we need to mimic some realistic mobile communication scenarios. The most typical environments consist in users carrying their own devices, and either moving in a given area, or remaining at a fixed position for some time. Examples of these behaviors are provided by different kinds of social gatherings, such as work spaces, scientific conferences, shopping malls, sport events, and busy metropolitan areas.

Our experiments are based on a work space environment, where 22 smartphones are carried by colleagues during their normal daily work activities. We used reliable yet inexpensive 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 app for wireless information sharing, as described in Section II. Each phone generates every 10 minutes a new piece of information to be shared, with all devices participating in the experiment being destination of all data.

TABLE I: Connections and success rate during 5 experiments of 1 hour each

# of devices	Connection attempts	Success rate
3	2551	37%
5	5884	13%
10	10425	10%
20	23318	6%

A. Static Experiments

To begin with, we emulated static gatherings of one hour duration, similar to work meetings, or conference sessions. These experiments allow us to analyze the feasibility of the information sharing service in terms of connection establishment delay and data transfer duration. From the total amount of available phones, we selected progressively larger subsets of active devices, to study the behavior and the effectiveness of our app for varying number of users.

In TABLE I, we report general information about the experiments. These first results concern connection attempts and success rate for 3, 5, 10 and 20 phones involved in the experiments. For every group size we run 5 experiments of one hour each. As it can be observed, the number of connection attempts increases with the number of devices. However, the success rate fades off, as expected by using well known results on the capacity of wireless networks as a function of the population density [19], [20]. In our case, the presence of multiple devices willing to setup one-to-one D2D links causes link establishment collisions. The probability to have a success is of the order of $1/n$, where n is the number of devices present in the scenario, so that the time between two consecutive successes for a given device is of the order of nT_s, where T_s is the duration of a scanning plus the time to detect a failure in a connection attempt. Therefore it is clear that mobility helps not just because mobile devices route messages opportunistically, but also because it causes the partitioning of n devices into groups, in each of which the probability to have a successful connection is higher than in a scenario with all devices under each other's coverage. This is similar to what shown in [20] for the beneficial impact of mobility on the collisions suffered in IEEE 802.11 systems. These considerations lead to conclude that using Wi-Fi Direct might be not the optimal choice if its coverage leads to high values of n. Besides, other key factors are the duration of scanning intervals and the time to detect connection failures.

While detecting failures is typically very fast (cfr. Fig. 3) and depends on the hardware/software platform, the scanning time is a design parameter we can tune. For what commented before, scanning should be as short as possible. However, the scanning should also be long enough to allow for a

TABLE II: Average and standard deviation of the connection duration (time between entering into the `Connected` and `Disconnected` states) for a specific number of devices, and the associated number of successfully completed connections

# of devices	Connections	Average [s]	Std. deviation [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

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

# of devices	Connections	Average [s]	Std. deviation [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

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

# of devices	Connections	Average [s]	Std. deviation [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

connection to complete, otherwise it would be not effective in detecting neighbor's state changes. Since we have observed that connecting takes no more than 10 seconds and transferring data takes no more than 2 seconds with very high probability, while early failure detection is almost immediate (cfr. Fig. 3), we use a scanning interval of 15 seconds, which is slightly longer than the cycle that goes from connection attempt to data transfer or failure.

To assess the system behavior in greater depth, we next analyze the most relevant operational phases of our service separately, to understand which parts of the process can represent performance bottlenecks.

TABLE II reports the average and the standard deviation of the time spent in the `Connected` state, which is the interval between the instants when two devices formed a connection, and when they disconnected. The duration of this phase is tightly related to the time needed to transfer data between a pair of devices. The amount of messages that can be transferred at every contact grows with time. This means that at the end of the experiment, in the case of 10 phones generating messages every 10 minutes, a total amount of 420 kB could be transmitted in one contact. The average connection duration is of the order of 1 second, with standard deviation growing with the number of devices. Fig. 3, in plots (a) and (b), shows the histograms of the time spent for data transfers in the cases of 3 and 20 devices. We can see that connections lasting longer than 2 seconds are rare.

However, data transfer is not the only relevant part of the information sharing process. Indeed, many wireless technologies for exchanging data over short distances suffer from high delays when trying to establish a connection to a chosen peer

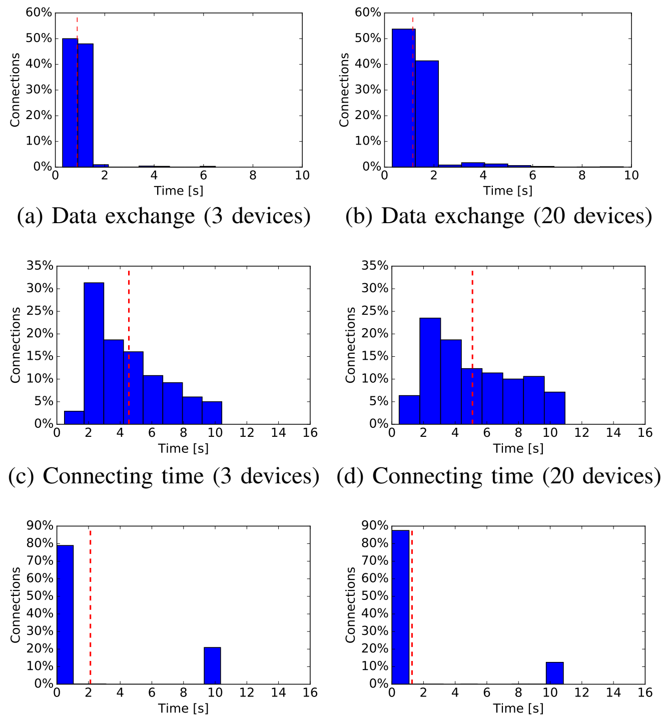
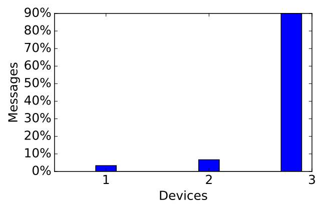


Fig. 3: Histograms representing the time spent for data exchange (plots (a) and (b)), the time spent to setup a connection (plots (c) and (d)) and the time spent to detect a failure while setting up a connection (plots (e) and (f)).

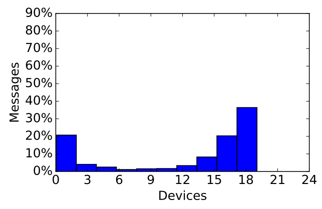
device. Hence, it is important to measure the average time spent in the `Connecting` state, from the moment a device chooses a peer to connect, to the moment the connection is established. Fig. 3, in plots (c) and (d), shows the histograms of these times in the cases of 3 and 20 devices. We can see that times up to 10 seconds are common. Note that these values only refer to successful connections.

Finally, Fig. 3, in plots (e) and (f), shows the histograms of the times from the moment one device decides to connect to another device up to the moment it discovers to have failed the attempt, in the cases of 3 and 20 devices. Also in this case, durations can be as large as 10 seconds, but averages are of the order of 1 or 2 seconds. As explained in Section II-B4, 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. The histograms in Fig. 3 (e) and (f) show 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.

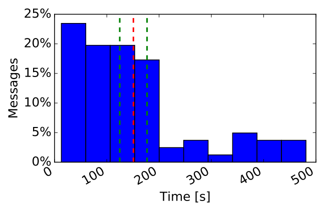
The average time to instantiate a connection is around 4.5



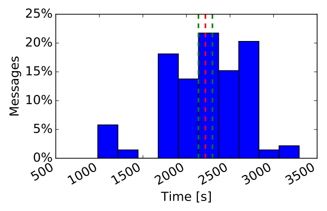
(a) Number of reached devices (with 3 devices)



(b) Number of reached devices (with 20 devices)



(c) Makespan (with 3 devices)



(d) Makespan (with 20 devices)

Fig. 4: Histograms representing the number of devices that were reached by every message during the experiment and the distribution of time spent only by the messages that reached the maximum number of devices (makespan).

TABLE V: Messages that reached all devices, normalized to the total number of generated messages, with statistics of their makespan

# of devices	Messages	Average [s]	Std. deviation [s]
3	90%	150.86	117.38
5	77%	428.25	336.10
10	73%	952.40	438.31
20	19%	2217.70	474.72

seconds, with a standard deviation close to 2.5 seconds, as can be seen in TABLE III. This result is of the same order of magnitude as for Bluetooth, that spends, together with data transfers, up to 12 seconds [11].

The next result concerns the spreading of messages, their dissemination among devices and the average time it takes to accomplish the task in static scenarios. These results will be later compared to the performance obtained in dynamic environments. TABLE V shows statistics for the messages that reach all devices and the time they need to complete such process from generation time (namely, the *makespan*).

In static scenarios, as we can see in Fig. 4 (a) and (b), the percentage of messages that reach most phones is fairly high, however, the time necessary to reach all devices is quite long (see plots (c) and (d)), up to about one hour, for 20 devices.

B. Mobile Experiments

In mobile experiments, 22 devices were distributed to researchers sharing an office building, which has three levels, with open areas, labs and offices at each floor. The results that we report were collected during working hours (from 10 AM to 6 PM) along 5 days.

One of the most interesting aspects of our information sharing app is how fast the information spreads after generation. In Fig. 5, we show an envelope chart (gray area) with the cumulated number of devices reached as a function of the time since generation, for all messages; this means that all curves referring to individual messages lie in the plotted gray area. In addition, we plot three curves, reporting the average over all messages, and the average plus/minus standard deviation. We can observe in Fig. 5 that the dynamic of information sharing is not very fast, and that the average number of devices reached by the information to be shared, even after 3 hours, does not grow much beyond 18 out of the 22 devices.

Nevertheless, connectivity plays an important role, as we can prove by looking at the number of peers reached by messages generated at different times. The earlier messages travel farther than those generated later on, since the opportunities to connect are more. Results are shown in Fig. 6, where for each message we plot one marker with abscissa equal to the message generation time and ordinate equal to the number of devices reached by the end of the experiment. We clearly see that, with few exceptions, early messages reach a larger number of devices during the experiments with respect to messages generated later in the day. Note that the trend in Fig. 6 is not linear, and messages generated in the middle of the day travel faster, due to the characteristics of user mobility.

As a final experimental result, in Fig. 7 we present the histogram of the makespan for messages that reach most devices (at least 18). To achieve this goal, the average time needed is close to 3.5 hours. This appears as a severe limitation on the feasibility of effective information sharing services when targeting large audiences and quick spreading of information.

C. Results discussion

The results we obtained compare somehow negatively with those in [12], which reported that with an app using Bluetooth on 12 smartphones, in a research institute, almost 90% of the devices are reached within a few minutes, although messages often do not reach all devices within a full working day. Our results are better than those obtained in [11], with an app using Bluetooth in a university campus with $\sim$50 to $\sim$70 mobile devices, where it took more than 4 hours to distribute a message to 70% of the devices. While message transfer times are comparable with the two technologies, in our experiments the wider coverage of Wi-Fi did not help, because of the huge fraction of failed connection attempts due to the large number of devices in transmission range. Indeed, a key parameter for the performance of the information sharing app is the average number of devices within transmission range, which should be ideally 1, as indicated by previous studies about the intrinsic performance of opportunistic communications [19], [20].

The analysis of WhatsApp carried out in [21] shows that multimedia flows have a typical size of a few hundreds of kilobytes, and that the throughput experienced by users is of the order of 1 Mb/s in both uplink and downlink, thus resulting in transfer times of the order of a few seconds. Therefore, connection durations we have observed are comparable with the ones of infrastructure-based communications. Hence, we conclude that D2D does not help much in speeding data transfer. However, we remark that infrastructureless services

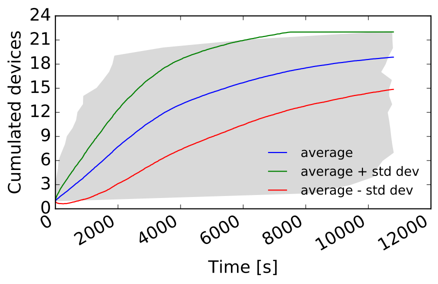


Fig. 5: Cumulated number of reached peers vs. time since message generation, in mobile experiments with 22 peers.

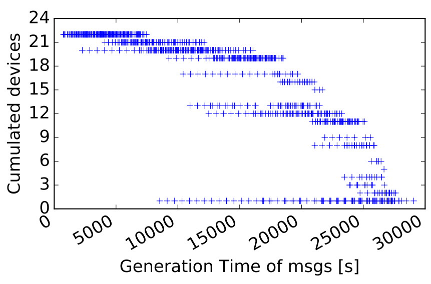


Fig. 6: Correlation between generation time and number of reached devices.

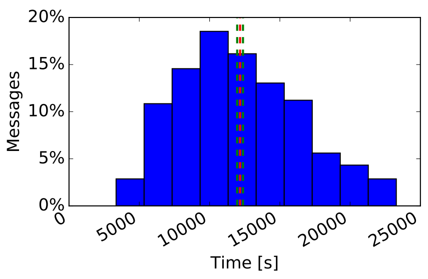


Fig. 7: Distribution of the makespan of messages that reached between 18 and 22 devices during the mobile experiments.

like ours do not suffer for the presence of network and server bottlenecks, and are available under undesirable circumstances like natural disasters, hacker attacks and censorship attempts.

Moreover, the use of a networked messaging app relieves the terminal from the discovery of peers and the establishment of connections, which, as we have seen, is prone to multiple collisions due to the fact that a device can only join one group under current Android limitations. This effect exacerbates with growing terminal density, as observed in our experiments.

Indeed, as previously remarked, in our work we have found that Wi-Fi Direct on Android suffers some serious limitations. From the viewpoint of an infrastructureless service designer, the most critical limitation consists in the fact that a device cannot be part of multiple groups. This makes connection management a hard task when users move and groups have to be continuously teared down and re-established as soon as the GO leaves. This is one of the main reasons why we have used one-to-one groups in our work, thus limiting the management complexity and connection re-establishment overheads.

Our results also show that connection opportunities are much less frequent than expected, due to scan technology limitations. Therefore, more research is needed to understand how to smartly select peers when more than one option is available. For instance, one could modify the content of Wi-Fi Direct probes to invite neighbors to connect based on a short description of the database available at the node sending the probe (e.g., a short hash). Such changes are not in the scope of the paper because here we addressed the suitability of plain COTS devices and SW, whereas the proposed improvements would request at least OS updates (e.g., API updates).

IV. Related work

Building services without the need of an infrastructure has attracted the interest of researchers since almost a decade. In the early stages of the research, theoretical works as in [4] studied the viability of the *floating content* concept, focusing on the expected lifetime of the content, by means of analytical models. In [5], the authors conducted a similar work, adding an urban scenario evaluation in the city of Helsinki. While characterizing the floating content performance, analytical results proved that urban environments were apt for the implementation of this type of services even with different node densities. Authors in [2] addressed not only opportunistic computing tasks, but also the exploitation of user's social behavior to disseminate information and distribute resources.

Given that Wi-Fi Direct started to be more commonly integrated into mobile devices, works like [6] and [7] 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 mobility.

In [22] and [23] the authors present Liberouter and PodNet, two communication architectures based on WLAN to enable neighborhood networking. These frameworks allow diverse types of devices to exchange information in an opportunistic fashion. However, Liberouter does not provide support for any lower layer communication technology while PodNet only does for Bluetooth and 802.11 in ad hoc mode. On the other hand, specific apps using Wi-Fi Direct were also developed in contexts similar to ours, but most of them followed different paths. For instance, in [24] the authors present an efficient way of creating P2P connections by modifying the discovery phase provided by the Wi-Fi Direct standard in Android, but those experiments do not ensure a better overall performance of data dissemination in crowded scenarios.

In the system developed in [10], 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 evaluation.

More recently, smartphones have incorporated enhanced versions of the short range standards. For this reason, a panoply of innovative testbeds and models have been devised in some studies with the objective of bringing new schemes that would fill the gaps of the D2D communication paradigm. In papers like [8] and [14], the authors proposed different network topologies in order to enable bidirectional inter-group communication. 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 COTS, *unrooted* devices.

Works even closer to our approach can be found in the field of D2D communications. In [11], complementary studies to our analysis are presented. Basically, the key concepts evaluated in that work are the efficiency with which content is shared by devices, accounting for users mobility and connectivity, as well as the survivability of that content. Based on a similar interest, we extended these analyses 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 the Wi-Fi Direct standard, the work in [11] is based on the Bluetooth communication protocol presenting greater limitations in terms of delay, pairing failures and battery consumption. These drawbacks, and the statistics presented in works like [6] and [14], where Wi-Fi Direct outperforms compared protocols, motivated our choice for that technology to conduct new experiments.

V. Conclusions

We have designed, implemented and tested in an office environment an Android app based on Wi-Fi Direct to provide infrastructureless pervasive information sharing services. We used standard versions of the operating system and standard APIs, so that our app can be installed on COTS Android devices without leveraging on root user's privileges. 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.

The experiments described in this paper reveal that Wi-Fi Direct can be used for delay-tolerant information sharing services. However, many key assumptions commonly adopted in analytical works do not hold with current technologies. In particular, typical contact intervals between devices and necessary scanning and connection establishment operations make the cycle between successful data exchanges quite long and far from the ideal case assumed in many models in which file transfer occurs instantaneously to all discovered neighbors. We have also discussed how wireless coverage and density of neighbors determine the performance of the service, just like it happens when evaluating the capacity of a Wi-Fi system.

Acknowledgments

The work has been partially supported by the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919). The work of V. Mancuso has been supported by the Ramon y Cajal grant (ref: RYC-2014-16285) from the Spanish Ministry of Economy and Competitiveness.

References

[1] 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*, vol. 48, pp. 5–19, 2014.

[2] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, no. 1, 2010.

[3] A. Asadi, Q. Wang, and V. Mancuso, "A survey on Device-to-Device communication in cellular networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1801–1819, Fourthquarter 2014.

[4] 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 *proceedings of IEEE INFOCOM 2011*, 2011, pp. 3137–3145.

[5] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas," in *proceedings of IEEE PerCom 2011*. IEEE, 2011, pp. 136–146.

[6] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-Device communications with Wi-Fi Direct: Overview and experimentation," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96–104, 2013.

[7] M. Conti, F. Delmastro, G. Minutiello, and R. Paris, "Experimenting opportunistic networks with Wi-Fi Direct," in *proceedings of IFIP Wireless Days 2013*, 2013, pp. 1–6.

[8] C. Funai, C. Tapparello, and W. Heinzelman, "Enabling multi-hop ad-hoc networks through Wi-Fi Direct multi-group networking," in *proceedings of IEEE ICNC 2017*, 2017, pp. 491–497.

[9] J. Zuo, Y. Wang, Q. Jin, and J. Ma, "HYChat: A hybrid interactive chat system for mobile social networking in proximity," in *proceedigns of IEEE SmartCity 2015*, 2015, pp. 471–477.

[10] Y. Mao, J. Wang, J. P. Cohen, and B. Sheng, "Pasa: Passive broadcast for smartphone ad-hoc networks," in *proceedings of IEEE ICCCN 2014*, 2014, pp. 1–8.

[11] S. Ali, G. Rizzo, V. Mancuso, and M. Ajmone Marsan, "Persistence and availability of floating content in a campus environment," in *proceedings of IEEE INFOCOM 2015*, 2015, pp. 2326–2334.

[12] S. Ali, G. Rizzo, V. Mancuso, V. Cozzolino, and M. Ajmone Marsan, "Experimenting with floating content in an office setting," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 49–54, June 2014.

[13] 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*. ACM, 2011, pp. 37–42.

[14] 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 *proceedings of IEEE WoWMoM 2015*, 2015, pp. 1–9.

[15] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1671–1688, Fourth 2013.

[16] M. Garetto, W. Gong, and D. Towsley, "Modeling malware spreading dynamics," in *proceedings of IEEE INFOCOM 2003*, vol. 3, March 2003, pp. 1869–1879 vol.3.

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

[18] W.-F. Alliance, "wi-fi peer-to-peer(p2p) technical specification v1. 1," *www. wi-fi. org/wi-fi-peer-peer-p2p-specification-v11*, 2009.

[19] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/18.825799

[20] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of ad-hoc wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 477–486, Aug. 2002. [Online]. Available: http://dx.doi.org/10.1109/TNET.2002.801403

[21] P. Fiadino, M. Schiavone, and P. Casas, "Vivisecting WhatsApp through large-scale measurements in mobile networks," in *proceedings of ACM SIGCOMM'14*, New York, NY, USA, 2014, pp. 133–134. [Online]. Available: http://doi.acm.org/10.1145/2619239.2631461

[22] B. Distl, G. Csucs, S. Trifunovic, F. Legendre, and C. Anastasiades, "Extending the reach of online social networks to opportunistic networks with podnet," in *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*. ACM, 2010, pp. 179–181.

[23] T. Karkkainen and J. Ott, "Liberouter: Towards autonomous neighborhood networking," in *Wireless On-demand Network Systems and Services (WONS), 2014 11th Annual Conference on*. IEEE, 2014, pp. 162–169.

[24] H. Zhang, Y. Wang, C. C. Tan, and Y. Zhang, "mQual: A mobile peer-to-peer network framework supporting quality of service," in *proceedigns of IEEE ICDCS 2015*, 2015, pp. 754–755.