Situation Awareness via Information Hovering in Post-disaster Communications

ABSTRACT

The majority of communication solutions for post-disaster areas rely on cellular infrastructure support. But disasters such as fires, floods, earthquakes, can disrupt communication network making its services unavailable.

New portable antenna towers, as point-to-point radio communications, are a way to mitigate the communication when cellular infrastructures are unavailable. But their deployment requires time decreasing the likelihood of finding people alive in the following of such events.

In this work, we present the architecture of Floater, a mobile application that implements opportunistic communication for context-aware services in post-disaster scenarios.

Floater is based on Floating Content, an opportunistic communications paradigm that geographically constrains message replication. We implement Floater using Google Nearby API that enables seamless nearby interactions without having to be connected to the Internet. Moreover, we provide the implementation notes and the comparison of the technologies adapted with the respect of the state-of-the-art.

1 INTRODUCTION

Communications are of paramount importance on the occurrence of a disaster. When a catastrophic event occurs such as a flood, an earthquake, or a fire, the exchange of information decreases individual exposure to various risks, mitigates material damages, and potentially saves lives [1].

Disasters such as fires, floods, earthquakes, disrupt communication network making its services unavailable. Such disruptions can be a direct consequence of the disaster event or operated to prevent further damages to the power lines —to reduce the risk of cascading effects. New portable antenna towers, as point-to-point radio communications, are a way to mitigate the communication when cellular infrastructures are destroyed, deprived of power supply, or congested. But the deployment of portable antenna towers requires crucial time decreasing the likelihood of finding people immediately following such events —the possibility of finding people alive aftermath an earthquake rapidly declines in the first hours reaching practically zero in two-three days.

Given the ubiquitous and the technology embedded, smartphones are of particular interest in these events. Smartphones can exchange information also with devices used for monitoring infrastructures such as dams, bridges, or power plants, empowering users with essential information. These devices enable opportunistic communications —direct communication between peers—- providing information exchange even in the unavailability of cellular infrastructures.

Opportunistic communications fit particularly well a disaster scenario [2]. Smartphones, as well as the Internet of Things (IoT) devices, create ad-hoc networks and eventually connect to the Internet in a delay-tolerant mode [3] enabling critical information exchange. Several solutions are available for smartphone-based communications in the form of WiFi Access Point or cellular access networks [4, 5]. These solutions aim at connecting two peers in a delay-tolerant mode allowing the exchange of text, video, or voice messages. Though of sure value for the affected population, such solutions are inefficient in terms of bandwidth, memory, and energy. In disaster scenarios with a massive exchange of information, with no control on the amount of information shared nor on its relevance, such solutions do not scale jeopardizing the communication channels.

In this paper, we present Floater architecture, the prototype of a mobile application that applies the concept of Floating Content in order to implement, in a distributed fashion, a situation awareness service for post-disaster scenarios. Floating Content or FC is an opportunistic communications paradigm [6]. FC geographically constrains message replication implementing infrastructure-less spatial information storage within an area denoted as Anchor Zone. FC allows Floater to create a tagged map of the disaster area shared among participating peers, in order to exchange critical information efficiently. We implement Floater using Google Nearby API that enables seamless nearby interactions without having to be connected to the Internet.

The rest of the paper is structured as follows: Section 2 illustrates the state-of-the-art of the opportunistic communication application focusing on post-disaster scenarios. Section 3 introduces the FC model implemented by Floater app in Section 4. Section 5 describes Floater architecture, implementation and challenges. Finally, Section 6 concludes the paper.

2 RELATED WORK

The first 72 hours post-disaster, known as the *golden relief time*, are the most critical ones to search and rescue efforts, after which the probability in finding survivors drop drastically [7].

Authors in [8] present an evaluation of opportunistic communication in disaster-event, where forwarding information generated in the incident location (e.g., victims' medical data to a coordination point), is critical for a quick, accurate, and coordinate intervention. They compare the most significant opportunistic routing protocols through simulations in realistic disaster-event. To address the lack of communication and information support needed in this crucial

ACM SACe, March-April 2020, Brno, Czech Republic

time, authors in [9] introduce human-centric wireless sensor network: an infrastructure that supports the capture and delivery of shared information in the field.

TeamPhone, introduced in [10], is a system which provides smartphones the capabilities of communications in disaster recovery. TeamPhone enables connections among rescue workers in an energy-efficient manner. A similar approach is proposed in [11] producing the prototype of relay by smartphone, which delivers emergency messages from disconnected areas as well as information sharing among people in evacuation centers. COPE is an energy-aware cooperative opportunistic alert diffusion scheme presented in [12]. COPE supports rescue operations and maintains mobile devices functionalities for maximum network coverage.

Authors in [13] introduce a cooperative opportunistic alert diffusion scheme for trapped survivors during disaster scenarios to support rescue operations.

Drones and vehicles support opportunistic communications increasing the number of peers within the communication area. Authors in [14, 15] present a vision for future unmanned aerial vehicles assisted disaster management.

The above approaches and communication prototypes in disaster-events do not implement any specific opportunistic communication model leaving open the network optimization. Given the inefficient network management, such systems may cause network overhead delivering unreliable services.

Follow, we introduce Floater: a multi-platform application, based on the Floating Content model, that limits the opportunistic communication preventing network overhead.

3 FLOATING CONTENT BASICS

Floating Content or FC is an infrastructure-less communication model that enables context-aware message exchanges in an area denoted as Anchor Zone (AZ) and over a limited amount of time [6].

Users engaging in sharing a piece of content, denoted as seeders node, initialize an AZ based on the application requirement (see the blue node in Figure 1a). Within the AZ, the opportunistic exchange takes place enabling probabilistic content storing. Every time a node with the content comes in contact with a node without it, within the AZ, the content is exchanged as shown in Figure 1b. Without loss of generality, the Floating Content paradigm assumes that the time taken to replicate the content is negligible with respect to contact time. However, any other communication model can be applied.

Nodes entering the AZ do not possess a copy of the content, and those exiting the AZ discard their copy as shown in Figure 1c. The typical behavior of a node consists in entering the AZ with no content, receiving the content from another node, distributing the content to nodes with no content met within the AZ, leaving the AZ and dropping the content.

As a result of such opportunistic exchange, the content *floats* (i.e., it persists probabilistically in the AZ even after the seeder(s) left the AZ). In this way, the content is made available to nodes traversing the AZ for the whole duration of its floating lifetime without infrastructure support.

A first Floating Content performance parameter is content *availability* at a given time —the ratio between the number of nodes with

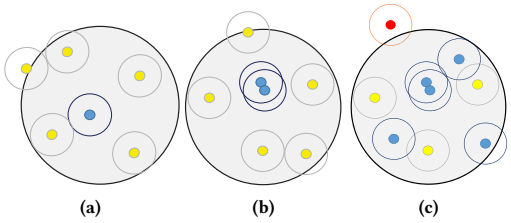


Figure 1: Basic operation of Floating Content. 1a) Seeder (blue) defines the AZ. 1b) Opportunistic message exchange between nodes. 1c) Nodes going out of the AZ (red) discard the content.

content over the total amount of nodes inside the AZ at that time. A high value of availability is correlated with a low likelihood of content disappearance from the AZ and a high likelihood of getting the content for a node entering the AZ. This last feature is captured by the *success probability*, the probability that a node entering the AZ receives a copy of the content. The success probability is the primary performance indicator, and the related to the performance of applications and services relying on FC. The success probability is a direct indicator of the effectiveness with which the *floating content* is delivered and made available to nodes traversing the AZ. Another relevant performance parameter is the time to get the content from entering the AZ, which is suitable for safety applications requiring that the message reaches nodes as soon as possible.

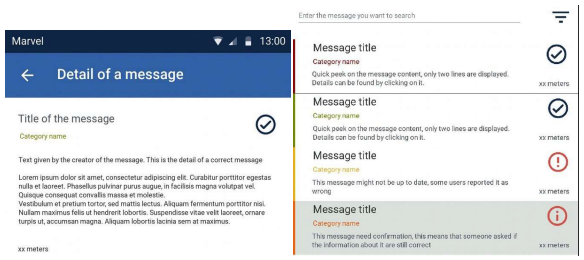
FC studies consider a setup where infrastructure support to FC is ubiquitous [16]. A data-based approach for dynamic management of FC in vehicular scenarios that minimizes the overall use of bandwidth and user storage space was proposed in [17]. Different from existing results, [18] applies to arbitrary mobility scenarios and spatio-temporal patterns of node density distributions by proactively modulating the content replication and storage strategies over time. The numerical assessment on a set of realistic scenarios in [18] shows very high levels of accuracy, adapting content availability over time in a QoS aware manner. The above studies enable us to balance the opportunistic replications and to rely on FC for network and energy resources management.

4 FLOATER APP

Floater enables communications between peers in a common area without requiring the support of a cellular network. The application is available on Android and iOS [19].

The Floater app enables a user to send, receive, and manage messages in the local user area or Anchor Zone —according to the Floating Content communication model previously described. The user that sends a new message can label it as follows:

- Victims, a user requiring help such as in a car breakdown situation;
- Danger, a user spotting a particular danger such as a rough road surface;
- Resource, a user localizing a new resource such as a restaurant;
- Caretaker, a user localizing health-point such as an ambulance.



(a) (b)

Figure 2: Floater message category: (a) Mock-up message details. (b) Mock-up message list.

The above categories identify the main application use cases. For instance, a shop owner can advertise exclusive deals by floating a *Resource* message nearby the interested shop; a pedestrian can exchange a *Danger* message that spots a malfunction in the cross traffic lights; or a skier can request help by sending a *Victim* message aftermath an avalanche.

After selecting the category as shown in Figure 2a, the user can add title, description, and pictures to the message. Floater enables seamless nearby interactions using Bluetooth, WiFi, or ultrasound network interfaces. The transmission radius is based on the message category. User can select the receiver radius in the application setting —displayed on the app map page— discarding messages far away from the actual user position.

Every user in the interested area is allowed to validate a message (as shown in Figure 2b), as a way to confirm the validity of the information carried by the message. In this way, the freshness and accuracy of the information spread via Floating Content is checked. The app automatically eliminates a message once it has been confirmed as wrong multiple times or out of date. The following list summarizes the main actions enabled by the Floater App:

- send messages with a title, category, position, and an image;
- receive every message in a the configured zone. For instance, a driver can lunch Floater to receive traffic information in the local area avoiding congested roads.
- Ask about the validity of a message. The receiver can validate or not the message. The automatic mechanism implemented in Floater deletes rejected messages avoiding spam and overhead.
- Filter the messages by categories, distance, date;
- see the messages on a map labeled with different markers;
- see deleted messages in the bin.

4.1 Use Cases

Floater provides connections between peers without using infrastructure communications that may be jeopardized by disaster events such as a flood, an earthquake or a wildfire. The difficulties in maintaining infrastructure base communications, in such disasters, motivate the ad hoc communication. Applications proving offline ad hoc communication such as Floater are the only solution to do not let alone a population, a community, or a single human in an emergency state. Despite disaster events, Floater allows the evaluation of a speech by the peers present in the audience, enables inter-vehicular communications in Vehicular Ad-hoc Networks, and provides connectivity when communication infrastructures are overloaded such as in disaster events. Figure **??** illustrates (from left) the available message list, the message creation, and Anchor Zone coverage according to the victim's position.

4.2 Floater via Google Nearby

We implement Floater using Google Nearby to establish direct communication channels with other devices without data connections (i.e., infrastructure support). Google Nearby increases the possibility of exchanging information between two nodes as it is based on the use of Bluetooth, WiFi, and ultrasounds interfaces. In addition, whenever cellular communications are available, they are used. This allows taking advantage of the surviving infrastructure available in a post-disaster scenario. Figure 4 shows Google Nearby connection procedure:

- Advertise and discovery procedures, respectively *StartAdvertising* and *StartDiscovery* methods, are called as soon as the application starts.
- *ConnectionLifecycle* method handles incoming connection.
- As soon as an advertiser has been found, *âĂIJEndpointDiscovery* method requires a connection by sending the name of the device and an identifier. Only applications with the same identifier can connect each other —no unwanted third-party peers.
- *Payload* method fires when a new payload is received.

The choice of the communication technologies used for exchanging content between two nearby nodes is managed by the Google Nearby Interface in a way which is transparent to Floater, and based on considerations on the quality of the channel (and hence also on interference levels) and on the amount of information to be exchanged. Table 1 lists the fundamental differences between Floater (implemented with Google API) and the existing solutions on the market. We can see that Floater does not have special requirements, and supports the most message types.

	Feature	Floater	goTenna	Fire	Nearby	Zello
	Instant Message	✗	✓	✓	✓	✓
	Require specific hardware	✗	✓	✗	✗	✗
	Infrastructure-based	✗	✗	✗	✗	✓
	Require the Internet	✗	✗	✗	✓	✗
Message supported	Text	✓	✓	✓	✓	✗
Message supported	Voice	✗	✗	✗	✗	✓
Message supported	Position	✓	✓	✗	✗	✗
Message supported	Images	✓	✗	✓	✓	✗
	Require an account	✗	✓	✓	✓	✓
Routing type	Text	✗	✓	✓	✗	✓
Routing type	Voice	✗	✓	✓	✗	✓
Routing type	Position	✓	✓	✓	✓	✗

Table 1: Floater vs. existing communication solutions

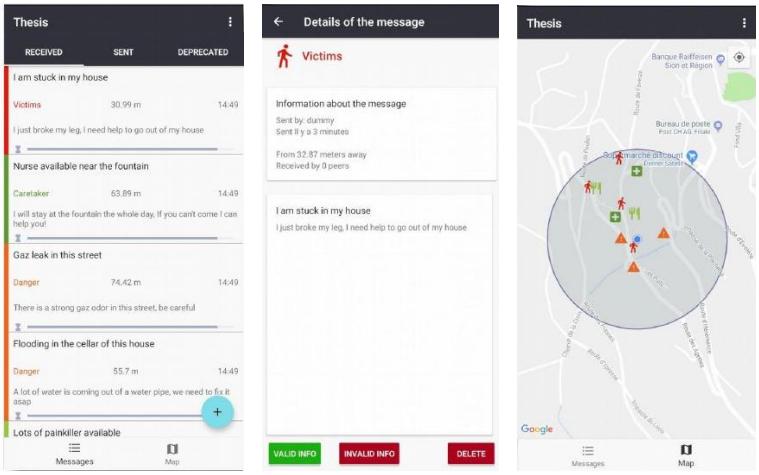


Figure 3: Floater disaster event use case.

Google Proximity Beacon enables Google Nearby to react to the user's context through beaconing, to monitor the status of the environment and the diagnostics endpoint, and to leverage the physical Web using Bluetooth Low Energy (BLE) [20]. BLE is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have stricter power requirements.

5 IMPLEMENTATION NOTES

In the following section, we highlight the motivation behind our solution. We start with network management and the related problems concerning the application back end. Then, we discuss user experience challenges focusing on problems such as error handling. Finally, we conclude with user interface challenges and present the designed mock-ups in this phase.

5.1 Application Back End

In a catastrophic event, message updated dissemination is an essential task. But the high dynamic of these disaster events may cause inaccurate information dissemination —the high frequency of the number of victims is an example of information hard to keep up to date.

In Floater app, every message has a default lifetime. Users support this mechanism by providing information feedback: positive feedback contents extend message lifetime, whereas negative feedback contents shorten it. Such feedback mechanism enables the control on the number of messages allowing only relevant and up to date information into the system.

Information trustfulness is another crucial feature of Floater app. Since incorrect information may have critical consequences, users can rate each message associated with a place. Such rating assists users in evaluating information reliability and in ranking users according to their trustfulness. This rating value, which includes reliability and trustfulness, is attached to the new users' messages for ranking it.

5.2 Message Flow

We illustrate the sending-receiving message flow in Figures 5 and 6. The respective methods are defined in the âĂIJCommunication ManagementâĂİ class, which acts as a network router. When a user clicks the send button in the Floater app, the message will be sent only if the two following conditions are satisfied:

- connected devices are on the map. Otherwise, the message will be placed in the message queue until a new device is connected.
- The device has a current location. Otherwise, the message will be placed in the location queue until the GPS receives a new location.

Each message has a header that allows identifying the content and checking its updates.

The receiving procedure needs to process the received message. We designed the reception of the payload this way to make the addition of a new type of data more accessible. On message received, the following conditions are checked:

- payload, for identifying message type (e.g., BYTE, FILE, or STREAM).
- Header, for checking message updates.
- Status, for calling the corresponding method.

5.2.1 Duplicate Data. In the creation of a new message, Floater uses user positions to check if a message of the same category already exists in the user local area. If that is the case, a popup message informs the user of presumable duplication.

5.2.2 New Peer Connection. When a new peer connects, he needs to receive all the exchanged messages of the local area. To avoid duplicate messages in the app content list, Floater executes a strict data check when new messages appear —existing messages are ignored at reception.

5.2.3 Application Settings. Floater allows users to change some parameters such as the username and the transmission radius. But the fundamental settings, such as message lifetime and the number

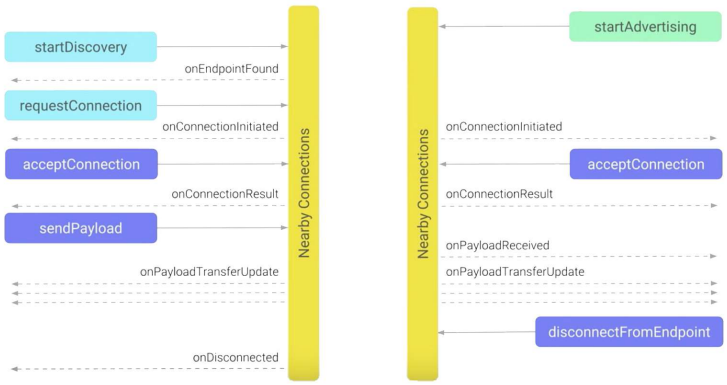


Figure 4: Google Nearby connection procedure (Android Developers, 2017).

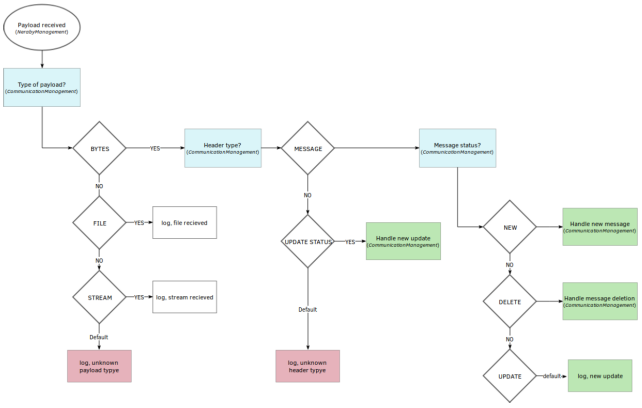


Figure 5: Sending message flow

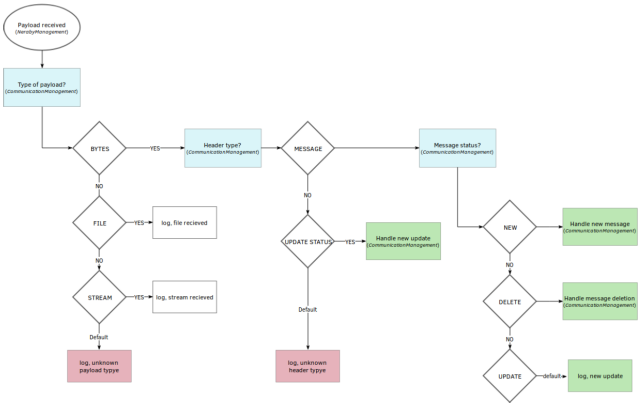


Figure 6: Receiving message flow

of peers within the local area, are out of user control. Such critical settings are related to the application back end.

5.2.4 Offline Challenges. Google Nearby Connection does not require any Internet connection 4. Floater app can be used regardless of network availability. But it needs cellular connection for the here listed operations.

- Application Installation: users need to have the application installed before the disaster occurs. If that is not the case, victims are unable to communicate via Floater. To address this issue, we envision a sharing feature in the application where users can send the installer. This would allow unprepared users to access the mesh network creating and replicating information. To make the installer exchange easier, we envision a workaround: during the installation, the installer—an Android Package Kit (APK)— is placed in the download folder of the device. To transfer the APK to other devices, users only need to send it via Bluetooth. This solution requires absolute mastery of the device.
- Floating Content Maps: when an Internet connection is available, the application downloads the map required by the FC model. But if no connection is available, the system needs to download the map periodically in order to use it in offline mode.
- Network Status: the number of peers that received the message is an excellent indicator of the network health that requires an internet connection for retrieving network data.
- Messages Classification: Since our application relies on message exchange, a message identification system is compulsory. This system allows nearby users to differentiate messages and enables quick identification of the message type by ranking both, users and messages.

5.3 User Experience Challenges

Error handling feature needs to be implemented to avoid inaccurate information dissemination. Only for infotainment messages, we propose the following approaches:

- a delay from 10 to 30 seconds between the sending requester and the message transmission can prevent these mistakes. The sending message is editable for the whole delay time. Finally, It is sent to all connected peers once the delay has been expired.
- Another approach would be to immediately send the message and allow users to update it afterward. But this strategy

may cause network overhead and inaccurate information dissemination.
- A third option that avoids additional delay and network overhead, is a *double message acknowledge*. A summary of the message before the transmission adds additional control of the message. The user reconfirms if the message must be sent, or edits it.

For emergency messages, the priority is to send the message as soon as possible. Therefore, error handling feature is applied only after the message has been sent.

5.4 User Interface Challenges

Concerning the application layout, we implement a mock-ups of the future user interface; this supports us in finding vital component issues of the application. For instance, application colors have been changed after the creation of the mock-up since they have not provided adequate contrast of the elements on the screen.

The application is user-friendly; we use a bottom menu that contains all the essential requirements such as the list of messages, the map of all messages, and the application settings. The message list contains all the messages in the user local area. Each message has a title, a description, a category, and the sender position. On top of that, the color band displays the message category and the message fidelity (according to the ranks of the message and the sender). This icon informs users about the freshness of the message; the icon becomes a red exclamation point once the information is outdated.

When a message has been selected, two buttons are present at the bottom of the screen used to confirm or reject the message. This mechanism supports the message expiration date. The application map provides an overview of the user area showing the connected peers and resources available.

5.5 Project Architecture

We implement Floater using Android Studio 3.0, which supports both Java and Kotlin. We split the project as follows:

- Adapter, which contains the custom adapters for the lists in the application.
- Model, which contains the Message and Endpoint classes.
- Utils, mostly consisted of abstract methods. This is where to configure Google Nearby.
- View, all the codes related to the views present in the application. We created sub-packages to help the organization of the files.

All the dependencies present in the project are from Google; we did not want to use an external framework as they might be unsupported in the future.

6 CONCLUSIONS AND OPEN ISSUES

Floater provides a fast, effective, and reliable communication to people affected by a disaster. Floater do not rely on existing network infrastructure such as the cellular network or the Internet. We illustrate how ad hoc networks can be used during disasters events to provide the necessary support to victims left unprepared for the arrival of relief. We focused our research on technologies embedded on smartphones, given its ubiquitous. Google Nearby Connection API is the technology we decided to use due to its robustness and reliability during our tests and implementation. However, simulations can be carried out on a larger scale and with a greater diversity of equipment testing Google Nearby limits. An optimization concerning the case of offline maps needs to be provided.

REFERENCES

[1] Gerard O'Reilly, Ahmad Jrad, Ramesh Nagarajan, Theresa Brown, and Stephen Conrad. Critical infrastructure analysis of telecom for natural disasters. 12 2006.

[2] Ramon MartÃŋ, A Crowcroft, J Yoneki, and E MartÃŋ. Evaluating opportunistic networks in disaster scenarios. *Journal of Network and Computer Applications*, 36, 11 2012.

[3] Mary R. Schurgot, Cristina Comaniciu, and Katia Jaffres-Runser. Beyond traditional dtn routing: Social networks for opportunistic communication. *IEEE Communications Magazine - IEEE Commun. Mag.*, 50, 10 2011. doi: 10.1109/MCOM.2012.6231292.

[4] Andreas Miaoudakis, Nikolaos Petroulakis, and Ioannis Askoxylakis. Communications in emergency and crisis situations. 06 2014.

[5] Daniel GutiÃĺrrez, J.M. LeÃşn-Coca, M Askalani, S.L. Toral, Federico Barrero, E Asimakopoulou, Stelios Sotiriadis, and Nik Bessis. A survey on ad hoc networks for disaster scenarios. *Proceedings - 2014 International Conference on Intelligent Networking and Collaborative Systems, IEEE INCoS 2014*, pages 433–438, 03 2015.

[6] Esa Hyytiä, Jorma Virtamo, Pasi Lassila, Jussi Kangasharju, and Jörg Ott. When does content float? Characterizing availability of anchored information in opportunistic content sharing. IEEE Conference on Computer Communication (INFOCOM), 2011.

[7] Sergio F. Ochoa, AndrÃĺs Neyem, JosÃĺ A. Pino, and Marcos R.S. Borges. Supporting group decision making and coordination in urban disasters relief. *Journal of Decision Systems*, 2007.

[8] Abraham MartÃŋn-Campillo, Jon Crowcroft, Eiko Yoneki, and Ramon MartÃŋ. Evaluating opportunistic networks in disaster scenarios. *Journal of Network and Computer Applications*, 2013.

[9] Sergio F. Ochoa and Rodrigo Santos. Human-centric wireless sensor networks to improve information availability during urban search and rescue activities. *Information Fusion*, 2015.

[10] Z. Lu, G. Cao, and T. La Porta. Teamphone: Networking smartphones for disaster recovery. *IEEE Transactions on Mobile Computing*, Dec 2017.

[11] H. Nishiyama, M. Ito, and N. Kato. Relay-by-smartphone: realizing multihop device-to-device communications. *IEEE Communications Magazine*, April 2014.

[12] F. Mezghani and N. Mitton. Opportunistic alert diffusion in disaster scenario âĂŤ stay alive longer ! In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, Oct 2017. doi: 10.1109/PIMRC.2017.8292230.

[13] Farouk Mezghani. An android application for opportunistic alert diffusion in disaster scenario. 2017.

[14] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, Jan 2017.

[15] Milan Erdelj and Enrico Natalizio. Drones, smartphones and sensors to face natural disasters. In *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, DroNet'18, pages 75–86, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5839-2. doi: 10.1145/3213526.3213541. URL http://doi.acm.org/10.1145/3213526.3213541.

[16] G. Manzo, R. Soua, A. Di Maio, T. Engel, M. R. Palattella, and G. Rizzo. Coordination mechanisms for floating content in realistic vehicular scenario. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017.

[17] Analytical models of floating content in a vehicular urban environment. *Ad Hoc Networks*, 88, 2019.

[18] Gaetano Manzo, Juan OtÃąlora Montenegro, Marco Marsan, Torsten Braun, Gianluca Rizzo, and Hung Nguyen. Deepfloat: Resource-efficient dynamic management of vehicular floating content. 08 2019.

[19] Flavien Bonvin. Floater. URL https://play.google.com/store/apps/floater.

[20] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty. Neighbor discovery latency in ble-like protocols. *IEEE Transactions on Mobile Computing*, 17(3):617–631, March 2018. doi: 10.1109/TMC.2017.2737008.