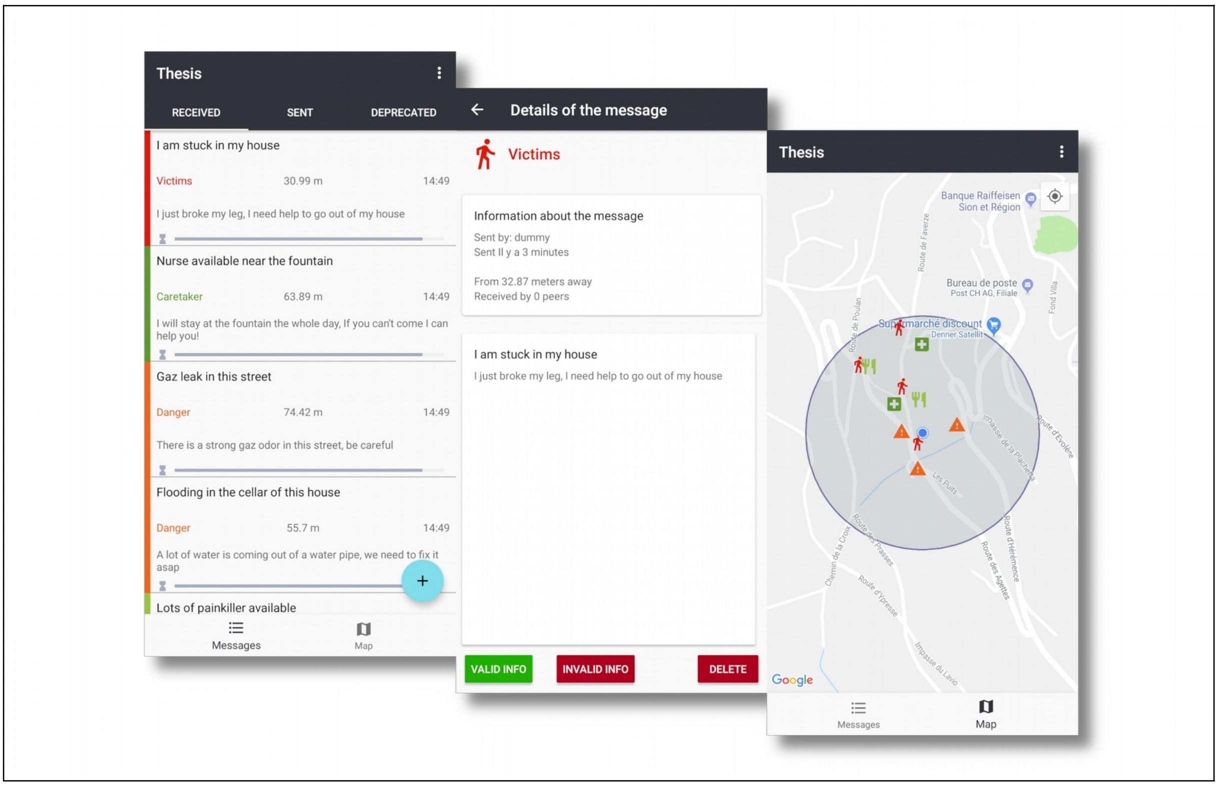
Hes·so// VALAIS WALLIS

Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Bachelor thesis 2018

Analysis, design, and implementation of an infrastructure-less situation awareness application



I am stuck in my house
Victims 30.99 m 14:49
I just broke my leg, I need help to go out of my house

Nurse available near the fountain
Caretaker 63.89 m 14:49
I will stay at the fountain the whole day, If you can't come I can help you!

Gaz leak in this street
Danger 74.42 m 14:49
There is a strong gaz odor in this street, be careful

Flooding in the cellar of this house
Danger 55.7 m 14:49
A lot of water is coming out of a water pipe, we need to fix it asap

Lots of painkiller available

Victims

Information about the message
Sent by: dummy
Sent Il y a 3 minutes
From 32.87 meters away
Received by 0 peers

I am stuck in my house
I just broke my leg, I need help to go out of my house

Student : Flavien Bonvin

Professor : Yann Bocchi

Submitted : 30.07.2018

www.hevs.ch

ABSTRACT

This work aims to implement an application based on opportunistic communication. It must be operational in a disaster situation when the cellular network is not available.

During this work, a first part was consecrated to the study of the several types of ad hoc networks and the various technologies related to them. This has served as a foundation for the study of ad hoc networks in smartphones since an android application was developed.

In a second phase, we defined the main requirements and features of the application. After that, the analysis of the existing solution was effectuated. We took care to emphasize the differences between the market solution and our implementation of the use case. It was necessary for us to know if such an application had its place in the market and if it was the case, in which way it differs from the competition.

Once this was done, we developed the application. We have recorded the different reflections made during the implementation to show the different issues. Those issues can be related to either the back or the front end of the application. To do this, we had to project ourselves into the position of someone experiencing this kind of event.

Finally, we have devoted a part of the document to the analysis of the results to know if an application of this kind, relying on ad hoc networking, can be used during a disaster.

Keywords

Ad hoc networks, Google Nearby, opportunistic communications

FOREWORD

This work was done in the context of a bachelor's thesis in Business Information Technology at the HES-SO Valais. It was proposed by the professor Yann Bocchi and Gianluca Rizzo. We received the task to develop a tool that will explore the potential of opportunistic communication offered by the Android system.

Theoretical research on ad hoc communications had been conducted in the past, but because of software and hardware limitations, it was difficult to implement them. Modern technologies such as Google Nearby and Apple iBeacon offer an abstraction layer and can provide a solution to some of the problems past studies had to cope with.

This thesis has multiple objectives; in the first instance, we will explain and give an overview of the solutions offered by the different actors of this sector. Some technologies unused during the implementation will be analyzed as well, for example Apple iBeacon or the fifth generation of wireless systems (5G).

After this research stage, we will implement an application capable of exchanging messages in ad hoc mode. The original idea was to implement a vehicular service such as traffic warning notification. However, the use case is flexible and can deviate from the initial concept.

At last, we will detail the actual limitations of the technologies we found during the implementation. On top of that, we will provide some recommendations that could help the future development of opportunistic communications. Those guidelines will provide advice on which technology to choose depending on the needs of the project.

ACKNOWLEDGMENT

We would like to sincerely thank all the people who have enabled us to carry out this bachelor's work under the best conditions. Special thanks to:

Mr. Yann Bocchi, our teacher in charge, for having supervised and followed us throughout this work. We are very grateful for his advice and precious feedback.

Mr. Gianluca Rizzo for providing valuable technical advice through his expertise and for his feedback on the different documents made in the thesis.

Finally, thanks to all the people who reviewed and corrected this work.

Table of Contents

Illustration Index

Index of Tables

LIST OF ABBREVIATIONS

AODV	Ad hoc On-Demand Distance Vector
APK	Android Package Kit
CDMA	Code Division Multiple Access
CRC	Cyclic Redundancy Check
DC-DC	Direct D2D communication with device controlled link establishment
DC-OC	Direct D2D communication with operator controlled link establishment
DR	Device Relaying
DR-DC	Device relaying with device controlled link establishment
DR-OC	Device relaying with operator controlled link establishment
DSDV	Destination Distance Vector
DSR	Dynamic Source Routing
DSRC	Dedicated Short Range Communication
EPFL	École Polytechnique fédérale de Lausanne
EVDO	Evolution-Data Optimized
FEC	Forward Error Correction
GPS	Global Positioning System
GSM	Global System for Mobile Communication
GUID	Globally Unique Identifier
HSPA	High-Speed Packet Access
IMT	International Mobile Telecommunication
IoT	Internet of Things
ITS	Intelligent Transportation Systems
ITS JPO	Intelligent Transportation System Joint Program Office
ITU	International Telecommunication Union
L2CAP	Logical link control and adaptation protocol
LAN	Local Area Network
LAR	Location-Aided Routing
LE	Low Energy
LTE	Long-Term Evolution
LTE	Long Term Evolution

MANETs	Mobile ad hoc networks
MCU	Micro-controller Unit
MIMO	Multiple-Input Multiple-Output
MMDA	The Metro Manila Development Authority
NFC	Near-Field Communication
NGO	Non-governmental organization
OLSR	Optimal Link State Routine
PAN	Personal Area Network
RFCOMM	Radio frequency communication
SDK	Software Development Kit
SDP	Service Discovery Protocol
SIG	Bluetooth Special Interest Group
SoC	System on a chip
SPANs	Smartphone ad hoc networks
TCP	Transmission Control Protocol
UHF	Ultra High Frequencies
UUID	Universally unique identifier
V2I	Vehicles to infrastructures
V2V	Vehicles to Vehicles
VANETs	Vehicular ad hoc networks
VHF	Very High Frequencies
WARP	Wireless Ad Hoc Routing Protocol
Wi-Fi	Wireless fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
ZRP	Zone Routing Protocol

INTRODUCTION

During a disaster, victims are left to themselves until rescue teams arrive. These first hours or days are vital and can save lives if those affected had a way to communicate efficiently. Mobile phones are an option; unfortunately, in many cases, the network infrastructure is also destroyed, making these devices useless.

This is why an application that allows communication without using the traditional telephone network, based on technologies such as Bluetooth or Wi-Fi, is justified.

The objective of this work is, therefore, to know if such an application can see the light of day with existing technologies. That is the reason why an Android application is developed. However, the compatibility with Apple's ecosystem must be taken into account.

In the first part of the work, we will analyze the different types of existing ad hoc networks as well as the underlying technologies. This analysis will provide us with a better understanding of the issues involved in such networks. After that, the definitions of the objectives that our application must meet will be established. This will allow us to know if similar solutions already exist on the market.

In a second phase, we will develop the application and we will detail the reflections made during this last part. The reflections will relate as much to the functioning of this type of network as to the interface of the application. These are points on which we have reflected and for which we have analyzed several options to select the best.

Finally, an analysis of the results obtained by the application will be carried out to determine if current technologies allow the implementation of this type of communication. Possible future developments will also be mentioned in this chapter.

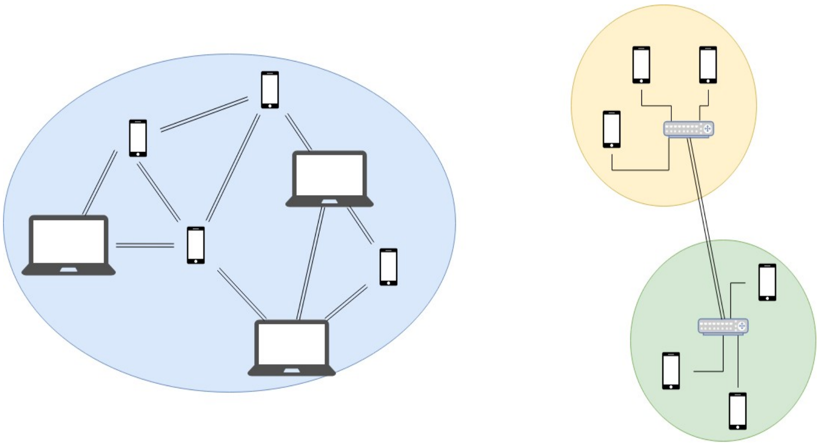
1. What Is an Ad Hoc Network

The Cambridge Dictionary defines ad hoc as something that is made or happens only for a particular purpose or need, not planned before it occurs (Cambridge Dictionary, n.d.). Ad hoc is used on multiple occasions, such as ad hoc organizations which are created especially for a given task or project. Ad hoc is also utilized in the military where an ad hoc unit is quickly deployed to help other units or regroup survivors into one entity.

In this work, we will focus on ad hoc networks and the technologies surrounding them. Those types of networks are infrastructure less, meaning that they can be operated without any central control. Ad hoc networks can be set anywhere, whenever, due to their dynamic and self-configuring nature.

Because no network specific hardware is needed, data travel through the network from device to device called "nodes". Each of those nodes forward packets onward until it reaches its destination. This type of exchange is called multi-hop networking; this involves two or more hops to send information from the source to the destination.

1.1. Ad Hoc and Traditional Networks

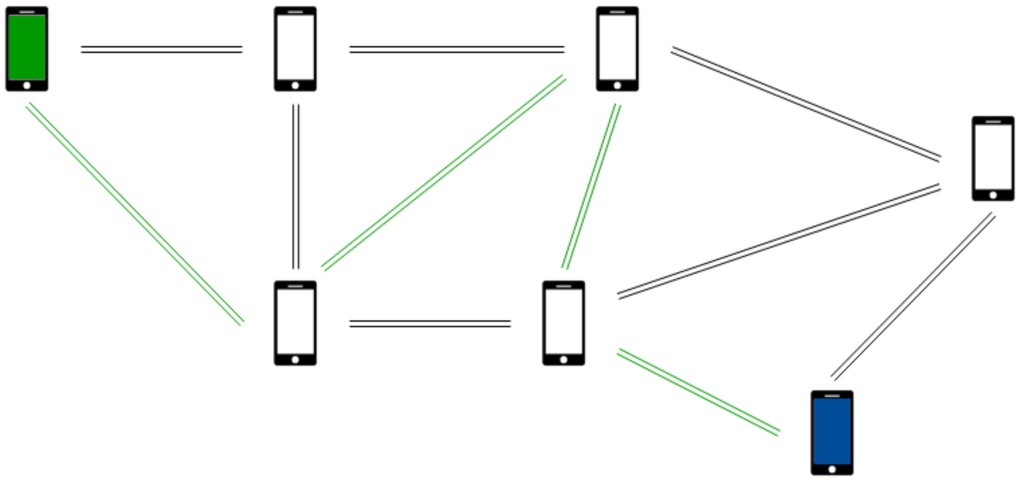


1. Illustration: Divergences between ad hoc and traditional networks. From the authors

The illustration above shows the differences between ad hoc on the left and traditional networks on the right. As shown in the illustration, no central hardware is needed in ad hoc mode, each device connects one to another regardless the type of machine, assuming they both implement the same protocol and network radio, creating one large grid. On the other side, the more classic approach needs routers to establish a connection between the two networks; the example can be two Local Area Networks (LAN) connected.

1.2. Multi-Hop Communication

The following illustration demonstrates what a multi-hop communication is. The path data will take change depending on the routing protocol used and on the routing table established by the network. In our example, there are seven smartphones constituting a smartphone ad hoc network (SPANs).



2. Illustration: Multi-hop demonstration. From the authors

The source of the message is transmitted from the green phone, passes through three other phones and reaches the destination in blue.

1.3. Advantages and Drawbacks of Ad Hoc Networks

What ad hoc networks offer is overwhelming and can change the way information is exchanged. However, this involves many technologies and challenges. We tried to establish a brief list of what the benefits are:

- Works where no service is available. In case of a war or a catastrophic event where existing infrastructure is destructed or disabled, ad hoc communication still works and can help people in this area to communicate.
- Setting up and transferring information are swift because of its absence of infrastructure to configure.
- Since there is no hardware to install (no antenna, relay, router) the costs are minimal and can be a significant advantage in developing areas.
- Communication is still possible in case of the unavailability or the destruction of a node in an ad hoc network as long as the devices remain connected by one link.

As said before, there are numerous challenges to face when we want to implement ad hoc networks, here are some:

- Each device is a router; this means that data has to transit frequently. This results in a higher battery drain than normal usage.
- Devices cannot be switched off depending on the number of nodes in a network. This can cause the whole network to collapse and make communications unusable.
- The dynamic nature of such networks makes the routing rough. Routing tables have to be continuously updated to ensure that each device is still present in the grid.
- If devices are moving, a wide bit rate variation can be expected.

1.4. Application of Ad Hoc Networks

All types of network we will analyze can be regrouped as Mobile ad hoc network (MANET) because they all consist of devices that can move such as smartphones or vehicles. We will not explain what a MANET is but instead focus on details of the specific types of networks such as SPANs, vehicular ad hoc networks (VANETs) and Disaster rescue ad hoc networks. We will not describe all types of ad hoc networks, only those studied while we were defining the use case we implemented. More information about use cases can be found later in this document, chapter 3 – Use Case Analysis (p. 28).

The following part is a quick overview of what are those types of networks and what kinds of technologies they use. Technical details can be found later in this document, chapter 2 – Ad hoc Networks in Smartphones (p. 10).

1.4.1. Smartphone Ad Hoc Networks (SPANs)

Smartphones are portable devices with network capabilities using both licensed frequencies operated by cellular carriers and unlicensed ones used by Bluetooth and Wireless Fidelity (Wi-Fi). Those use the unlicensed ISM band ranging from 2.4 GHz to 2.5 GHz (Shoemake Matthew, 2001, p. 4).

Usage of These Types of Networks

Several usages of these types of networks have already been made during protestations where government entities either monitored or disabled the existing network infrastructure. This was the case during the Tunisian's Arab spring (Hern Alex, 2014), where 40,000 Iraqis downloaded the application called FireChat allowing them to communicate. The same application was used at two other occasions, during the Hong Kong Protest in 2014 (Cohen Noam, 2014) and the same year in

Russia when anti-government leaders recommended their followers to install the application (Milian Mark, 2014).

1.4.2. Vehicular Ad Hoc Networks (VANETs)

Vehicles are becoming smarter generation after generation; they are loaded with numerous sensors and probes which make them aware of their environment. In the current state, data is processed and kept in the car but is not shared among other vehicles.

VANETs objective is to address this issue by allowing Vehicles to Vehicles (V2V) and Vehicles to Infrastructures (V2I) communications; it will provide a safer and more efficient driving experience. They are part of the Intelligent Transportation Systems (ITS) framework.

Intelligent Transportation System

ITS goal is to provide drivers a safer, more coordinate use of transport networks. Governments are developing a range of tools that will help them to achieve their objectives, V2V and V2I are a crucial element of it. Both European and American governments created special commissions dedicated to the development of ITS. They provide both financial and legislative help to support innovative ITS projects (European Commission, n.d.).

The American administration created the Intelligent Transportation System Joint Program Office (ITS JPO). In their latest report they established a strategic plan to contain multiple categories (Barbaresso Jim et al., 2014, p. V) such as:

- Connected vehicles: research and implementation on V2V and V2I solutions.
- Automation: research on automated V2I systems that will transfer some control from the driver to the vehicle.
- Interoperability: ensure that data will be compatible from one system to another.

Applications of VANETs

There are several applications that could make the traffic safer and more efficient.

- Car platooning: reducing the inter-vehicular distance, down to a few centimeters, this saves fuel for the following vehicles and allows a faster and safer highway driving. Acceleration and breaking will be synchronized across vehicles.
- Intelligent traffic system: reducing the time vehicles spend stopped, at traffic lights or crossroads, can significantly reduce pollution noise and help relieve traffic congestion.

- On the road services: marketing opportunities such as shop discovery and gas station advertising can be created with VANETs.

Technologies Used

Short-range radio technologies are an obvious requirement for VANETs; a paper from University of Reading in England covered the existing technologies. They presented 14 different existing wireless standards, and they discussed the limitation and support of safety and non-safety applications for each of them.

During the paper, they defined two types of applications. Safety applications where data is processed in real time and is aimed at minimizing road accidents. Non-Safety applications that enable the passenger to access to a service such as the Internet or Voice over IP (VoIP) (Anwer Shahid & Guy Chris, 2014, p. 661). The summary of their research is in the appendix of this paper.

They came up with the conclusion that Dedicated Short Range Communication (DSRC) is the protocol that might suit most of these applications. It works on the 5.9 GHz frequency bands and can provide service for V2V and V2I (Anwer Shahid & Guy Chris, 2014, p. 667). However, they only analyzed existing technologies, some advancement promised by the incoming 5G network might be suited for V2V and V2I communications.

Well-known technologies such as Wi-Fi and Bluetooth have been reviewed by the authors of the study but are not recommended for VANETs.

Wi-Fi has many different standards such as 802.11 a, 802.11 ac, 802.11 b, 802.11 e, 802.11 g, and 802.11 n. The most appealing standard for VANETs is 802.11 ac because of its high data rate, up to 1Gbps, and its high-operating frequency, 5 GHz. However, it is not recommended by the authors of the paper because of the decrease of performance as the number of vehicles in the network increases. Other Wi-Fi standards are prone to interference because they operate on the unlicensed 2.4 GHz band and are therefore not recommended.

Bluetooth is used to create Personal Area Network (PAN) and can be used for V2V and V2I applications. Moreover, it cannot be used for safety applications because of its short range, slow transfer rate and air interference vulnerability (Anwer Shahid & Guy Chris, 2014, p. 665).

1.4.3. Disaster Rescue Ad Hoc Networks

When a catastrophic event occurs, like a flood, an earthquake or a fire, communication can save many lives. Frequently, radio towers are either destroyed by the event or congested; meaning that civilians touched by the accident are left alone and cannot ask for help because no service is available.

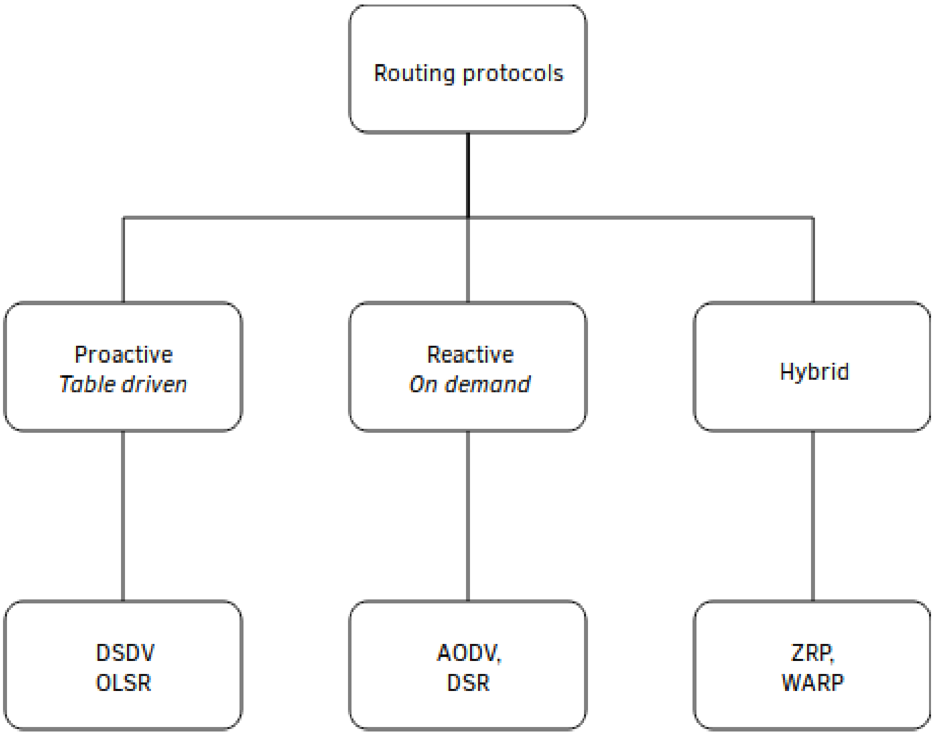
Rescue teams use radios and can quickly deploy antennas to communicate in the affected areas. Unfortunately, this does not mean for civilians that the service is back, since they do not operate on the same frequencies as the one used by smartphones. In addition, those types of infrastructure are set up only when the catastrophe occurs, which, in the meantime, leaves people alone. Disaster rescue ad hoc networks want to empower the victims of those events during the first crucial days.

Existing Technologies

We have analyzed many existing technologies for those types of ad hoc networks, they are detailed in the chapter 3.1 – Disaster Assistance Use Cases (p. 29).

1.5. Type of Routing Protocols

Routing data from one device to another in an ad hoc network is laborious because of its highly dynamic nature. Three types of routing protocols are used. We are not going to detail exactly how each of the following protocols work, the objective is to show the differences between proactive, reactive, on-demand and position-based protocols.



3. Illustration: Sorting of routing protocols (Deepshikha and Durga Prasad, 2016)

1.5.1. Proactive - Table Driven

Table-driven protocols maintain a routing table of the whole network; this table is periodically updated and thus generate additional traffic on the grid. Speed is an advantage of this type of protocol because each route is known before data is transmitted. There is no discovery time as it can be the case with reactive protocols. However, if data comes and the route is unknown, the message is put in a queue until the route is discovered. These types of protocols are not power efficient and are hard to recommend for IoT devices that want to be in a sleep state as much as possible.

The two most popular protocols are Optimal Link State Routing (OLSR) and Destination Distance Vector (DSDV) (Deepshikha Bhatia, 2016, p. 3387).

1.5.2. Reactive - On demand

Reactive protocols create routes on demand when an unknown route is required. If this is the case, the entire network is flooded by route request. All nodes on the network initiate a discovery process until a working route has been found.

If the transmission is a critical factor, on-demand protocols are not recommended. The route discovery can take time and increases with the number of nodes in the network.

Specific reactive protocols are ad hoc On-Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) (Deepshikha Bhatia, 2016, p. 3388). AODV is one of the protocols used in ZigBee.

1.5.3. Hybrid

Hybrid protocols combine both proactive and reactive methods. Usually, those protocols start by creating a routing table (proactive) and then serve an unknown destination with a reactive protocol. Zone Routing Protocol (ZRP) for example, creates zones around a device, where all possible routes inside that zone are known. If data has to exit the area, the protocol will query other zones to find the correct destination and send the data.

Hybrids are ZRP or Wireless Ad Hoc Routing Protocol (WARP) (Moltchanov Dimitri, n.d.).

1.5.4. Position-based Protocols

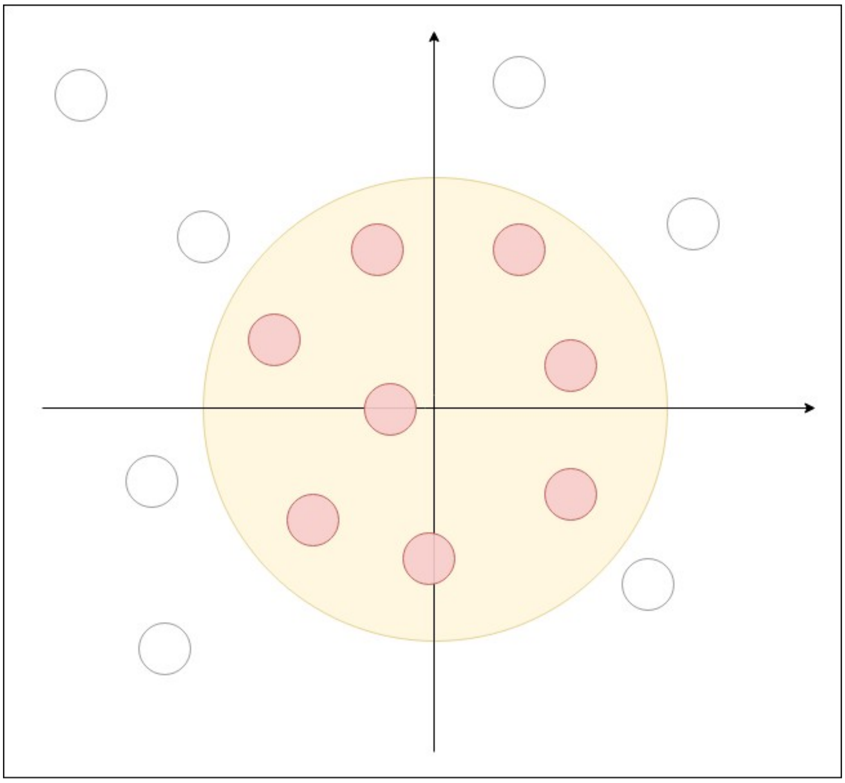
Because a lot of portable devices have a GPS, it is possible to use it to determine what route a data packet should take. Location-Aided Routing (LAR) is one of those types of protocols.

Floating Content

Floating content is a concept of data sharing in a restricted area and are a type of position-based protocols. Messages inside this area are exchanged to every peer. Once a peer exits the zone, it loses its access to the message, which are either destroyed or hidden. When a peer enters the sector, it will receive all the information contained in the sub-mentioned area.

This kind of content delivery system is challenging. Since they rely on the presence of peers inside an area, it is hard to ensure the persistence of information. Indeed, if every peer leaves a zone, the message will be lost permanently. However, a base station inside the zone can be a solution to that problem; it will secure data on a stationary device.

The following illustration shows how floating content works. The yellow area is the sector where messages are exchanged. All peers inside, represented as circles, hold the information, whereas those outside do not.



4. Illustration: Floating content representation

2. Ad Hoc Networks in Smartphones

We will focus now on the technologies embedded in smartphones as we will explore ad hoc networks through an Android application. However, we will not only focus on the Google platform but will also analyze what other companies are developing. Lastly, we will make a quick overview of future technologies that might help the development of ad hoc applications such as the latest Bluetooth iteration or the 5G mobile network.

2.1. Google Android

Google offers multiple ways to implement ad hoc networks; developers can either use directly the interface they want to use, Bluetooth or Wi-Fi, with the API provided by the company or use an abstraction layer that takes care of everything in the background called Google Nearby. In the following part, we will focus on each of those technologies.

2.1.1. Bluetooth API

Two classes are needed to create a peer-to-peer Bluetooth connection:

- “BluetoothDevice”: represents a remote Bluetooth device. It lets the developer create a connection with the respective device or query information. This class will be used to create the peer-to-peer connection.
- “BluetoothAdapter”: represents the local Bluetooth adapter. It lets the developer perform fundamental Bluetooth tasks. This class will be used in our case to listen to an available connection.

The peer-to-peer connection methods developed by Google uses the Radio Frequency Communication (RFCOMM) protocol.

RFCOMM Protocol

The RFCOMM protocol is a transport layer protocol that emulates an RS-232 serial port. It is made on top of the Logical Link Control and Adaptation Protocol (L2CAP). This protocol is used across a broad set of devices because it helps manufacturers to implement Bluetooth capabilities on devices having a serial port. Because of its reliability, this protocol can be compared to the Transmission Control Protocol (TCP). One significant difference is the number of supported ports, TCP handles up to 65,535 and RFCOMM only 30 ("Choosing a transport protocol", 2008).

Google developed two different ways to create peer-to-peer Bluetooth connections, one is secure, and the other is not. If confidential information is exchanged, the secure connection is recommended because the other is prone to man-in-the-middle attacks.

Secure Method

To create a secure connection, two methods are used:

```
createRfcommSocketToServiceRecord() //from the BluetoothDevice class
listenUsingRfcommWithServiceRecord() //from the BluetoothAdapter class
```

This method was implemented at API Level 5; it uses the Service Discovery Protocol (SDP) and lookup Universally Unique Identifier (UUID). Explanation of both SDP and UUID can be found below. When connecting to another Android device, generating a UUID for the application is recommended.

Insecure Method

To create an insecure connection, two methods are used:

```
createInsecureRfcommSocketToServiceRecord() //from the BluetoothDevice class
listenUsingInsecureRfcommWithServiceRecord() //from the BluetoothAdapter class.
```

This method was implemented later on the Android development, at API Level 10, it is very similar to the one explained above. However, the data exchanged are not secured, and it is not recommended in scenarios where data are sensitive.

SDP and UUID

SDP allows one device to know what services are supported by the devices it tries to connect to. Those services are identified using a UUID. Every Bluetooth device implements this protocol (Poole, n.d.).

UUID, also called Globally Unique Identifier (GUID) is a unique 128 bit identifier; they are used to identify services and information. They do not rely on a central database to ensure their uniqueness; it implies that it is possible to have two similar UUIDs, but the probability is nearly zero and thereby making it negligible. They are displayed in 5 groups formatted as follows 8-4-4-4-12 making a total of 32 hexadecimal digits. Here is one randomly generated UUID:

```
9763b1fc-befb-4c22-9979-cdad9a4b4375
```

Range of Connection

There are two major classes of Bluetooth devices, other exist but are not used as much:

- Class 1: maximum permitted power 20dBm (100 mW), range up to 100 meters.
- Class 2: maximum permitted power 4dBm (2.5 mW), range up to 30 meters.

Smartphones are class 2 devices and can only be paired with devices closer than 30 meters in the ideal condition. However, manufacturers advertise a more conservative 5 to 10 meters range (Bluair, n.d.).

The Bluetooth Special Interest Group (SIG) states that a 350 meters Bluetooth 4 connection is possible. However, the test was made between a standard smartphone, and a Bluetooth Low Energy (LE) Microcontroller Unit (MCU). Only a notification was exchanged and no information about bit rate or stability was given in the article (Bluetooth SIG, 2016, p. 11).

2.1.2. Wi-Fi Direct API

The Wi-Fi peer-to-peer is available on Android since API Level 14 (Android 4.0). The API is fully compliant with the Wi-Fi Alliance's Wi-Fi Direct specifications. The main advantage of Wi-Fi over Bluetooth is a more extended range.

Wi-Fi Direct

Wi-Fi Direct is a standard developed from the Wi-Fi Alliance, a nonprofit organization that promotes and maintains Wi-Fi standards, which allows devices to connect one to another without the need for a central access point and an Internet connection.

The objective of Wi-Fi Direct is to straighten some problems that Bluetooth has, such as slow transfer rates and lengthy pairing processes. On top of that, since Wi-Fi Direct uses the same technologies used by classic Wi-Fi, only one device needs to be Wi-Fi Direct compliant, not every machine in the network.

Wi-Fi Direct Support on Android

Several classes handle Wi-Fi Direct, they are all part of "android.net.wifi.p2p". They allow to discover and connect to other devices. Contrary to the Wi-Fi Direct specification, each device on Android needs to support the protocol (Android Developers, 2018).

We will explain the classes needed to implement Wi-Fi Direct in an application. Nevertheless, we will not give full technical details about them.

The most important class is called “WifiP2pManager” with this class it is possible to:

- Initialize connection
- Discover nearby devices
- Start P2P connection

Sockets handle data transfer. Two classes are required to send data correctly:

- “ServerSocket”, that will wait for a client connection in the background.
- “Socket” that will use the IP address and port of the “ServerSocket”.

Range of Connection

The Wi-Fi Alliance states that a 200-meter Wi-Fi Direct connection is possible (Wi-Fi Alliance, n.d.). However, they do not mention what devices are used or the transmission power of the latter.

Furthermore, we found many different regulations in the world concerning the maximum power emitted by Wi-Fi appliances. Nonetheless, we were not able to find maximal output power for home devices such as a router. In addition, no information was found regarding the emitting power of smartphones.

We concluded that smartphone manufacturers have some leeway in the implementation of the protocol leading to a wide disparity between devices, making it difficult to provide range numbers.

2.1.3. Google Nearby

As explained earlier, Google Nearby offers an abstraction layer allowing the programmer to focus on the coding of the application instead of spending time on network interface study. Everything is managed in the background by the operating system depending on the needs, network availability or the strength of the signal.

The API is available for both Google Android and Apple iOS. Because Google Nearby is integrated into the Google Play Service, it is available for all Android devices running Android 2.3 or higher. We consecrated a chapter later in this document on Google Nearby on iOS devices, chapter 2.2 – Apple iOS (p. 17).

To facilitate development, Google Nearby is divided into three different APIs:

- Message API: allows the exchange of small binary payload between devices. They need to have an Internet connection but do not necessarily need to be on the same network. The API uses Bluetooth, Bluetooth LE, and near-ultrasonic audio to communicate a pairing code between devices. Google Nearby Message is recommended for multiplayer gaming, sharing content or broadcasting a resource (Google Developers, 2018). More information about this API can be found in the following part of this paper.
- Connection API: creates a fast, reliable, secure connection between two or more devices in an entirely offline manner. A combination of Bluetooth, Bluetooth LE and Wi-Fi hotspots is idealized. Uses cases for this type of API are a collaborative whiteboard, create a multi-screen gaming experience where a smartphone becomes remote, and the game is displayed on a TV (Google Developers, 2018). More information about this API can be found in the following part of this paper.
- Notification API: allows the developer to create an interactive notification with nearby devices. Actions such as opening an HTTPS URL or trigger an application intent are possible. This API is developed for Bluetooth LE beacons. There are several usages for this API, but they all rely on the same principle of creating a notification on nearby devices.

All the code and methods are within the same class Google developed, called “Nearby”. In the following part, we will give more details about the Message and Connection API. We will not explain the Notification API because it does not suit our need for the application we will develop.

Message API in Detail

As said before, Nearby Message is used for the exchange of a pairing code between devices; this means that the service is not meant to exchange large data such as images or videos, it is a means of connection instead. An active Internet connection is needed because the pairing code received is sent to a Google Nearby Message server for validation.

The typical steps involved in the connection are the following (Google Developers, 2018):

1. The publisher application requests a message and a pairing code (token). The Nearby Message server makes an association between the payload and the token.
2. When the application is launched, the device will use Bluetooth, Bluetooth LE, Wi-Fi and ultrasounds to broadcast the message and make it detectable by other smartphones.
3. The subscribing device uses the same technologies. The device will send its token to the publisher and detect the publisher’s token.

4. Once both devices detect themselves, they send a report to the server. It will validate the tokens.
5. After the report, the server will handle the message exchange between devices.

We are not going to detail all the methods and how to implement this API in an application. We will show the basic methods that allow the publishing and listening of pairing codes. Because Nearby Message is a publish-subscribe API, there is a dedicated method for both of those actions:

```
Nearby.getMessageClient(activity).publish(message);
Nearby.getMessageClient(activity).subscribe(messageListener);
```

The message that will be broadcast can weigh up to 100 KB, but Google does not recommend anything bigger than 3 KB to ensure fast transfers (Google Developers, 2018).

Battery Usage

Because of the heavy use of radios and sensors, the battery consumption is increased by a 2.5 to a 3.5 rate with Google Nearby activated. To reduce the battery drain when using Communication API, Google made some recommendations:

```
Nearby.getMessagesClient(Activity).unpublish();
Nearby.getMessagesClient(Activity).unsubscribe();
```

Connection API in Detail

Connection API enables a true peer-to-peer communication between multiple devices. This API was created to help developers with the implementation of such networks. Since there is a wide range of different hardware and Android versions, it is hard to create a reliable, fast, and efficient ad hoc network using the Bluetooth or Wi-Fi API of Google. The connection API makes advertising, discovering and connecting between many devices fast and straightforward.

Only two phases are involved with the API:

1. Pre-connection where publishers will advertise themselves and discoverers will scan surroundings and send a connection request to any publisher it will encounter.
2. Post-connection when devices are connected, data can be exchanged in this phase.

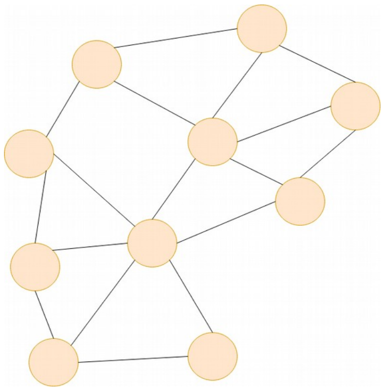
Once devices are connected, there are no differences between publishers and subscribers; all the devices can send data. There are three different types of payload (Google Developers, 2018):

- BYTES, byte array limited to 32k.
- FILE, file of any type is supported.
- STREAM can be used for files that we do not know the size beforehand, this can be recorded audio, for example.

Nearby Connection API supports two types of strategies; there is no strategy better than the other, their use depends on the use case. The topology is set during the creation of the advertising object:

```
Nearby.getConnectionsClient(this)
   .startAdvertising(
      mConnectionLifecycleCallback,

        new AdvertisingOptions.Builder().setStrategy(Strategy.P2P_STAR).build());
```

Connection API Strategy P2P_CLUSTER

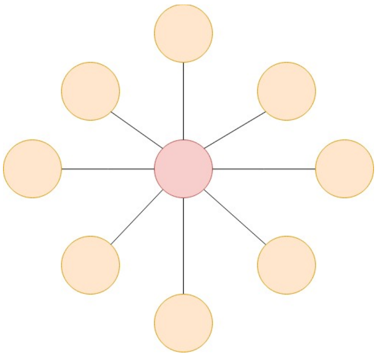


5. Illustration: P2P_CLUSTER strategy. From the authors

The P2P_CLUSTER strategy creates a mesh network; it is the default strategy used by the API. Devices up to 100 meters apart can be connected by using this strategy. This topology is flexible. However, the P2P_START topology offers a larger bandwidth.

This is an M-to-N topology; this means that any device in the network can initiate an outgoing connection to M other devices, they can also accept incoming connections from N other devices (Google Developers, 2018).

Connection API Strategy P2P_STAR



6. Illustration: P2P_START strategy. From the authors

This star-shaped topology works well in situations where one device is the master of the network. Fewer devices transmit data simultaneously, this results in a better bandwidth and would allow, for example, video sharing across a group of smartphones. This topology is stricter than the P2P_CLUSTER topology.

2.2. Apple iOS

2.2.1. MultipeerConnectivity

Apple developed a proprietary framework that allows devices running iOS, macOS, and tvOS to share files, messages, and streams. The portable devices, iPhone, iPad, and iPod, use a combination of infrastructure such as Wi-Fi, peer-to-peer Wi-Fi, and Bluetooth; on the other hand, laptops and computers use infrastructure Wi-Fi, peer-to-peer Wi-Fi and Ethernet (Apple Developers, 2018).

The framework can work on both infrastructure and ad hoc mode making it versatile. As it is the case for Google Nearby, Apple does not allow developers to choose the interface they want to use. Everything is handled by the operating system depending on an algorithm developed by the company.

There are two phases involved with this framework: discovery and session. In the discovery phase, the application will scan nearby devices and advertises itself as a new available device. The application advertises itself to allow other users to send a session invitation.

A session invitation is sent to the selected devices and users can accept or decline it. Once the invitation is accepted, applications can share data. A callback is fired when a peer joins or leaves the session.

Once the application moves into the background, any open sessions will close, and the application will stop advertising and browsing. If the application is reopened, browsing and advertising automatically restart but closed sessions need to be handled by the developer.

2.2.2. Google Nearby

Google made Google Nearby available on iOS; this makes Google Nearby and MeshKit from Open Garden the universal solutions since they are available on Apple's iOS and Google's Android.

To add Google Nearby to any iOS application, Google used the open source dependency manager CocoaPods. Developers have to create a file called "Podfile" in the project directory and insert the following code.

```
source 'https://github.com/CocoaPods/Specs.git'
platform: ios, '7.0'
pod 'NearbyMessages'
```

2.3. Other Implementations

2.3.1. Bluetooth 5

The newest Bluetooth 5 promises twice the speed and four times the range depending on the mode. In reality, the Bluetooth SIG created two new physical layer variants: LE2M, which provide twice the speed and LE Coded, who provide four times the range at the expense of the speed (Woolley Martin, 2017). It is not possible to have both speed and range at the same time.

Bluetooth 5 Range Increase

By using the Bluetooth 5 LE Coded the range is significantly improved. The Bluetooth SIG defined that the maximal Bit Error Rate (BER) to 0.1%. With Bluetooth 5 the SIG wanted to increase the range without increasing the transmitting power while keeping the BER at 0.1%. Bluetooth 5 uses two technologies to deal with errors (Bluetooth SIG, 2016, p. 12):

- **Error detection:** already used in Bluetooth 4.x, the Bluetooth protocol uses a Cyclic Redundancy Check (CRC) to ensure that the packet received is correct. If there is an error, it will not acknowledge it, and the transmitter will send the packet again.
- **Error correction:** this is a new addition to Bluetooth 5. On top of error detection, the protocol will try to correct errors up to a certain extent. Bluetooth 5 uses Forward Error Correction (FEC), no further details will be given in this document.

As for the Bluetooth 4 tests, a company, Nordic Semiconductor, tested a connection between two Bluetooth 5 MCU. They used the nRF52840 System on a chip (SoC), it has an 8dBm maximum output power (Nordic Semiconductor, 2018, p. 326). They were able to achieve a distance of 750 meters (Nordic Semiconductors, 2016). However, the test was only a connection test; no bit rate was given.

Summary Table of Bluetooth 5 Changes

The following part summarizes some of the differences between Bluetooth 4 and Bluetooth 5.

Feature	*Bluetooth 4.x*	*Bluetooth 5*
Radio Frequency (MHz)	2400 to 2483.5	2400 to 2483.5
Distance/Range (m)	Up to 100	Up to 200
Nominal Data Rate (Mbps)	1	2
Latency (ms)	<6	<3
Network topology	Star-bus, Mesh	Star-bus, Mesh
Multi-hop Solution	Yes	Yes
Message size (byte)	31	255

1. Table: Difference Bluetooth 4 and 5 (Collotta Mario, Pau Giovanni, Talty Timothy & Tonguz Ozan K., n.d.)

Bluetooth 5 Ad Hoc Improvements

There is no improvement for ad hoc networks implemented in the new Bluetooth 5 standard. The Bluetooth SIG made no statement regarding significant development in this field.

With the development of Google Nearby and MultipeerConnectivity by Apple, Bluetooth is becoming an underlying technology. This means that these services proposed by Google and Apple will benefit from the improvement made by the new Bluetooth standard. Both speed and range will be increased as a result of this new version.

An application that would implement the Bluetooth API from Google will also gain both range and speed depending on the mode used. Android supports this new standard since the Oreo version released in February 2018.

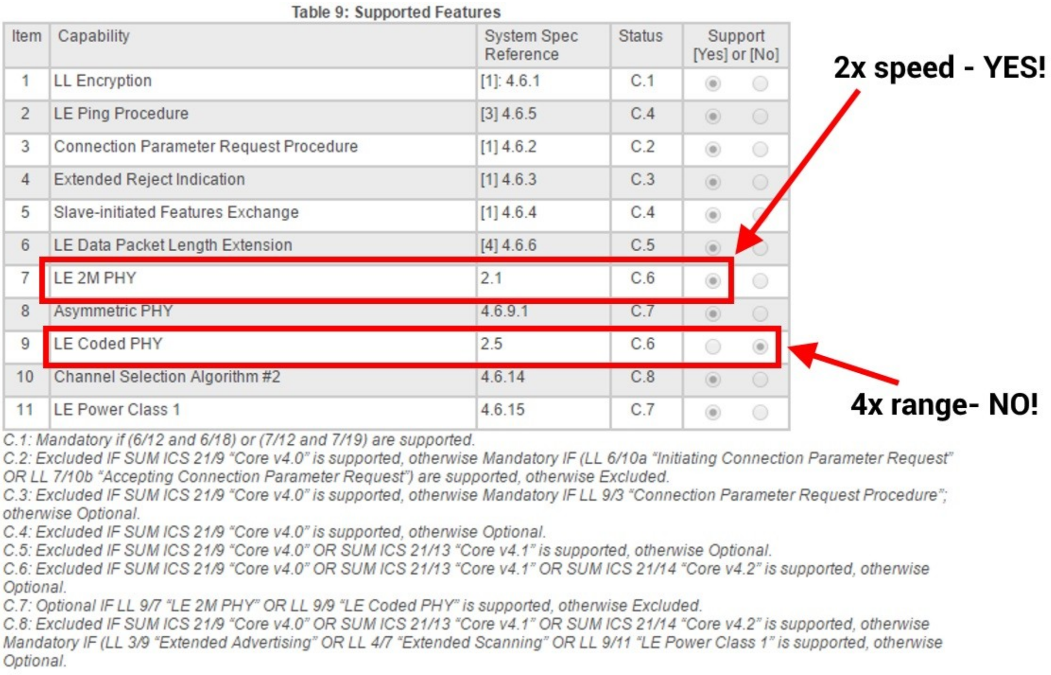


Table 9: Supported Features

Item	Capability	System Spec Reference	Status	Support [Yes] or [No]
1	LL Encryption	[1]: 4.6.1	C.1	● ○
2	LE Ping Procedure	[3] 4.6.5	C.4	● ○
3	Connection Parameter Request Procedure	[1] 4.6.2	C.2	● ○
4	Extended Reject Indication	[1] 4.6.3	C.3	● ○
5	Slave-initiated Features Exchange	[1] 4.6.4	C.4	● ○
6	LE Data Packet Length Extension	[4] 4.6.6	C.5	● ○
7	LE 2M PHY	2.1	C.6	● ○
8	Asymmetric PHY	4.6.9.1	C.7	● ○
9	LE Coded PHY	2.5	C.6	○ ●
10	Channel Selection Algorithm #2	4.6.14	C.8	● ○
11	LE Power Class 1	4.6.15	C.7	● ○

C.1: Mandatory if (6/12 and 6/18) or (7/12 and 7/19) are supported.
C.2: Excluded IF SUM ICS 21/9 "Core v4.0" is supported, otherwise Mandatory IF (LL 6/10a "Initiating Connection Parameter Request" OR LL 7/10b "Accepting Connection Parameter Request") are supported, otherwise Excluded.
C.3: Excluded IF SUM ICS 21/9 "Core v4.0" is supported, otherwise Mandatory IF LL 9/3 "Connection Parameter Request Procedure"; otherwise Optional.
C.4: Excluded IF SUM ICS 21/9 "Core v4.0" is supported, otherwise Optional.
C.5: Excluded IF SUM ICS 21/9 "Core v4.0" OR SUM ICS 21/13 "Core v4.1" is supported, otherwise Optional.
C.6: Excluded IF SUM ICS 21/9 "Core v4.0" OR SUM ICS 21/13 "Core v4.1" OR SUM ICS 21/14 "Core v4.2" is supported, otherwise Optional.
C.7: Optional IF LL 9/7 "LE 2M PHY" OR LL 9/9 "LE Coded PHY" is supported, otherwise Excluded.
C.8: Excluded IF SUM ICS 21/9 "Core v4.0" OR SUM ICS 21/13 "Core v4.1" OR SUM ICS 21/14 "Core v4.2" is supported, otherwise Mandatory IF (LL 3/9 "Extended Advertising" OR LL 4/7 "Extended Scanning" OR LL 9/11 "LE Power Class 1" is supported, otherwise Optional.

2x speed - YES!

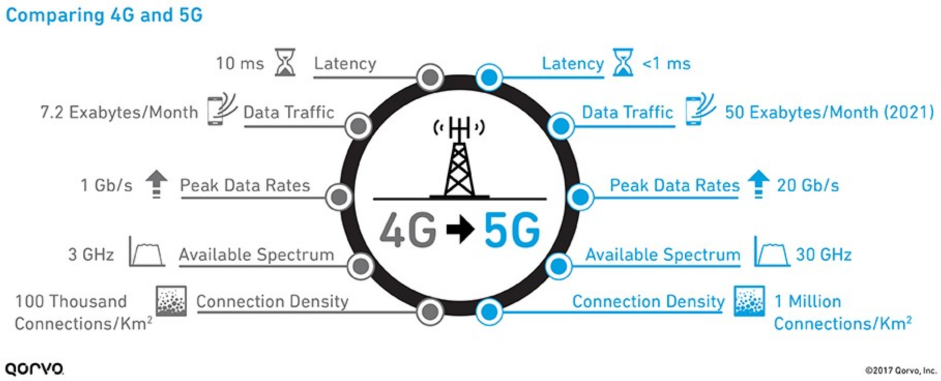
4x range- NO!

7. Illustration: Bluetooth 5 support of the Samsung Galaxy S8 (Sims Gary, 2017)

Regretfully, manufacturers are free to choose the features that are supported by the device they produce. The Samsung Galaxy S8 officially supports the last iteration of the Bluetooth, yet the Bluetooth LE Coded PHY is not (Sims Gary, 2017). This means that the device will be able to have twice the speed but not four times the range.

2.3.2. Fifth Generation Mobile Network

Still in a development stage, the 5G promises some considerable improvement over the actual cellular networks. In the coming years, the number of mobile devices connected to the Internet will skyrocket (Cisco, 2017) making the current infrastructure obsolete. The following illustration compares the differences between what is currently supported by the 4G and what the 5G needs to implement.



8. Illustration: Comparing 4G and 5G (Qorvo, 2017)

Even if the standard is not established yet, it is part of the International Telecommunication Union (ITU) International Mobile Telecommunication (IMT)-2020 standards, the ITU IMT-2020. In their 2017 article, Nordrum Amy, Clark Kristen, and IEEE Spectrum Staff listed technologies that will be part of the new standard.

Millimeter Wave

Mobile operators have been using the same frequencies for a long time; this result in crammed frequencies were adding traffic became complicated. The frequency band spectrum used by the 4G (700 MHz to 2600 MHz) remains, however, higher and less used frequencies are going to be added.

Those high frequencies also called millimeter waves because of their sub-centimeter size, range from 30 to 300 GHz (Rogerson James, n.d.). Not the whole millimeter wave spectrum will be used, Ofcom, the office of communication of the United Kingdom, and the European spectrum regulators identify three key bands, 700 MHz, 3.4-3.8 GHz and 24.25-27.5 GHz.

Smart Cells

Adding new frequencies to the cellular network will not be sufficient to achieve 5G requirements. This is why smart cells are necessary: instead of creating a massive cell tower providing a cellular network for a large area, carriers will develop smaller and less powerful antennas that will be installed throughout cities.

One objective of this antenna increase is to take advantage of the massive Multiple-Input Multiple-Output (MIMO) another critical feature of the 5G.

Massive MIMO

MIMO is already used in many wireless technologies such as Wi-Fi (IEEE 802.11n, IEEE 802.11 ac), 3G (HSPA+), 4G with Worldwide Interoperability for Microwave Access (WiMAX) and 4G Long Term Evolution (LTE). The technology uses multiple reviving and emitting antennas for the same base station.

Until now, a base station might rely on fewer than ten antennas, whereas a massive MIMO will use dozens of antenna that will allow better data. This much increase of antenna will increase interferences; this is why beam forming is another base technology of the 5G

Beam Forming

With the addition of cells and the massive MIMO interference will become a problem. Beam forming wants to address this issue by directing the signal to the user thereby reducing interference for nearby users.

Millimeter waves can transmit much more data than the actual spectrum used because of the higher frequency. Nonetheless, they cannot penetrate building, cars, trees as easily. This problem can be minimized with beams forming the signal in one direction.

Full Duplex

Actual technologies use two different frequency bands for receiving and sending data. With full duplex, it will be possible to use the same frequency for transmitting and receive information. This will theoretically double the capacity of a frequency band.

Full duplex generates more interference when an antenna transmits and emits at the same time on the same frequency. Also, an echo can be heard. An excellent echo-canceling protocol is required to make the full-duplex possible.

5G Ad Hoc Improvements

With the multiplication of mobile devices such as smartphones, laptops, tablets and the rise of new appliances like autonomous cars, beacons, and IoT devices, the need to have device-to-device (D2D) communication has become crucial. In their paper, Tehrani Mohsen Nader, Uysa Murat, and Yanikomeroglu Halim (2014) identified two different types of D2D communication.

Device relaying (DR) is a system where devices with weak signal coverage relay their information through another device to communicate to the base station of the operator. Direct D2D, consists of two devices or more communicating one to another without sending data to the base station.

With that in mind, they dressed a list of four different types of device-tier communications (Tehrani Mohsen Nader, Uysa Murat, & Yanikomeroglu Halim, 2014):

- Device relaying with operator controlled link establishment (DR-OC): D2D communication with the help of the operator to establish the link between devices.
- Direct D2D communication with operator controlled link establishment (DC-OC): devices communicate one to another without the need of a base station. However, the link establishment is handled by the operator.
- Device relaying with device controlled link establishment (DR-DC): devices themselves handle the link establishment procedure.
- Direct D2D communication with device controlled link establishment (DC-DC): everything handled by the devices, no operator control.

Those types communication help to reduce the network congestion that might appear. Nonetheless, interference can be caused by this type of communication. In the case of DR-OC and DC-OC, this can be handled by the base station that will allocate resources depending on the availability. In DR-DC and DC-DC, this will be handled by devices and great smart interference management is mandatory.

Final Words About the 5G

The 5G wants to improve cellular networks in many fields, and there is a real need for ad hoc communication as the number of devices increases. However, it is still difficult to see how it will be implemented because of the non-existing standardization yet.

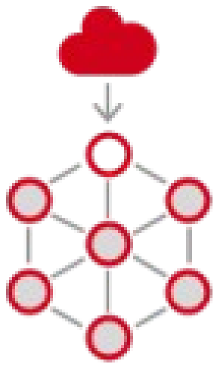
Many fields would benefit from ad hoc communication with the 5G network. Autonomous cars require fast, reliable ways to communicate one to another, the <1 ms latency and 20 Gb/s data peak promised by the future standard is precisely what is needed. Location-aware beacons or IoT will become more and more present in the future. By using ad hoc communication, they will not overload the network.

In this paper, we presented alternative technologies to a cellular network such as Wi-Fi and Bluetooth. However, all of them operate in unlicensed frequencies; this means that interferences are uncontrolled. If an excellent interference management software is developed with the 5G network, ad hoc communication should greatly benefit from this technology.

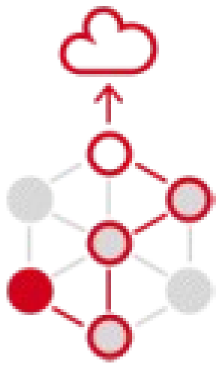
2.3.3. MeshKit

When Open Garden developed FireChat, Google Nearby was not released. The Californian company had to create its framework. The software is called MeshKit and is compatible with both Google's Android and Apple's iOS.

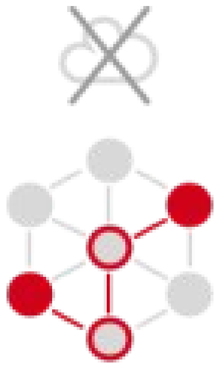
The MeshKit Software Development Kit (SDK) allows users to share content without the Internet (Open Garden, 2018). The API uses a combination of Bluetooth, Bluetooth LE, Wi-Fi Direct, ANT and other wireless protocols. The result is a complete infrastructure-less communication mean, as explained below, the framework is split in three different parts:



9. Illustration: Cloud to mesh (Open Garden 2018)



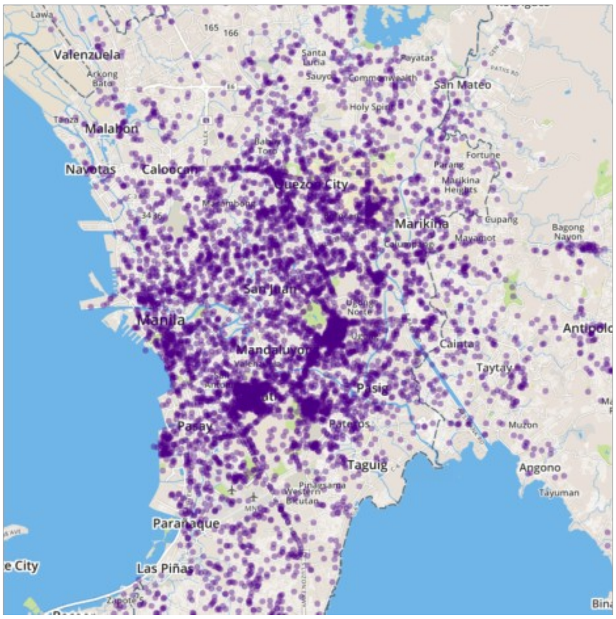
10. Illustration: Mesh to cloud (Open Gaden, 2018)



11. Illustration: Peer to peer (Open Garden, 2018)

- Cloud to mesh: phones that are connected to the Internet share news and content to the disconnected one.
- Mesh to cloud: nodes without an Internet connection can send data to the Internet through other nodes
- Peer to peer: offline data sharing among peers. The report can be sent back to a server if needed.

MeshKit Implementation



12. Illustration: Peer to peer connection (Open Garden, 2018)

Metro Manila, one of the densest urban areas in the world, simulated a 7.2 magnitude earthquake. The drill test was conducted by The Metro Manila Development Authority (MMDA), in collaboration with non-governmental organizations (NGO), private and public organizations. They chose MeshKit to broadcast information because cell towers were either destructed or congested by traffic. A map of the network created during the drill showed that the mesh imitated by FireChat covered the vast majority of the city. The company states that 80 percent of the population were able to receive the messages delivered by the local authorities (Open Garden, 2016).

We were not able to find the content of the messages exchanged during this event, neither were we able to know if the population used FireChat during the drill.

MeshKit Availability

MeshKit is not publicly available, we tried to contact the company to know what the conditions are and if it was possible to test the technology. Unfortunately, we did not receive any answer, nor do we know how much MeshKit costs.

Final Words About MeshKit

Open Garden has developed a complete framework that can create mesh in multiple ways depending on the use case. The solution is different from Google Nearby. As it does not provide the same services, the Google Nearby Notification API is similar to the Peer to Peer part of MeshKit.

MeshKit stores messages when connection is lost and will forward them as soon as connectivity is back. This is particularly interesting in scenarios where only a few peers are close to the device, or in case the mesh would be defective.

One of the major problems we had with the framework is the lack of documentation offered by Open Garden. Everything we found was summarized in a single web page and we were not able to have technical information such as code snippets, best practices or references. We concluded that Open Garden sells the framework and those pieces of information are then available.

2.3.4. The Serval Project

The Serval Project is an Australian project that wants to create an alternative to infrastructure-based communications. The main benefits are the following (The Serval Project, 2017):

- Communication anytime: when infrastructure fails information exchanges are not possible, the Several Mesh allows nearby phones to communicate.
- Communication anywhere: not all countries are well covered with the cellular network. In Australia, for example, around 75% of does not have access to any connection. Ad hoc nets developed by Serval can help phones communicate directly with each other.
- Communication privately: end-to-end encryption (256-bit ECC) is activated by default on all exchanges using the Serval app.
- Communication with people: Several wants to connect everyone on earth regardless of the circumstances.

They developed an open-source Android application called “Serval Mesh”. It allows phone to calls, send messages and share files. The current version of the application is the 0.93 and was released in May 2016.

The wiki page of the application is confusing, and it is hard to understand how it works. They state that the application does not require any infrastructure. However, they suggest three ways of connecting and exchanging information (The Serval Project, 2015):

- Using an access point and exchanging data over the Internet. This is not infrastructure less since an access point is required.
- Creating a portable hotspot, one device create a hotspot (from the Android settings), and other devices connect to it.

- Using ad hoc mode, they do not give many details on how this works and which technologies are used. However, we concluded that it was not Google Nearby or any other classic API developed by Google as it requires root access.

We tested the application and its features. Let aside the old and tough to understand user interface; we were not convinced by the product. We were able to exchange files and messages, but it always needed either an access point or a portable hotspot created by one of the phones. This is not an ad hoc communication as a central appliance is needed.

We were not able to test the most interesting feature as we do not have root access to any of the phones we used. There are no explanations on how the ad hoc mode works and what technologies are used

Mesh Extender

The Serval Project team developed a hardware device called the Mesh Extender. It creates a Wi-Fi network that allows people nearby to connect and use the Serval Mesh application.

Final Words About the Serval Project

In the end, we found only a few information about the project; we are not able to know the team behind it, the date of creation or other crucial information. The wiki they created is complete but confusing. It seems that they are working on a lot of different projects such as mobile applications, hardware devices or protocols, but none of them seems to be finished.

3. Use Case Analysis

This thesis has several objectives; one of them is the implementation of an application that would present the current possibilities offered by ad hoc networks. Moreover, the application must demonstrate that ad hoc networks are a viable means of communication. For those reasons did we established the following list of requirements:

- Implementation of offline ad hoc communications exclusively
- Respond to a real-world problem and provide an answer to it
- Uniqueness in the way it implements the solution
- Presentable in a salon where there is not much room

Following those requirements, we suggested the following potential use cases:

- Peer reviewing: aimed at orators who want to improve their presentation. This application can provide a platform that allows the evaluation of a speech by the peers present in the audience. A dynamic list of criteria is established beforehand, and people can give their feedback once the presentation is finished. This use case is justified in scenarios where Wi-Fi infrastructure are mediocre and where people have trouble connecting to the Internet. Despite this, it was not retained because most of the time the available networks are sufficient making ad hoc communications not legitimate.

- Offline decentralized chat: targeted to the public of music festivals where some people do not have unlimited cellular data plan and have trouble communicating with friends because of non-existent or lousy Wi-Fi networks. The application has implemented private and public chatrooms. It has not been selected because some other application, like FireChat, provide the same services.

- Inter-vehicular communications: an experiment that would test the feasibility of communication between vehicles using smartphones. Ad hoc communications are mandatory in those types of scenarios because they can reduce the latency created by infrastructure architecture. Data do not need to go back and forth between the vehicle and the access point. It is a research implementation of ad hoc communications. Because of its complexity and its difficulty to be presented in a salon, this use cases have been rejected.

In the next part we will explain the application we decided to implement and analyze the alternatives currently existing on the market. In the final part, we will summarize what the competition offers and where our use case is different.

3.1. Disaster Assistance Use Case

In case of a disaster such as an earthquake, a tsunami, an avalanche or even a war, infrastructure base communications are difficult or impossible to maintain. The population is then left alone in an emergency state until the rescue teams arrive. During those first crucial hours or even days, communication is more than mandatory, this can save lives.

This is for those reasons that we decided to implement an application that can be used in such type of situation. The application will be developed on Android and will neither require an account nor an Internet connection. All communication will be handled in ad hoc mode and a mesh network of peers will be dynamically created as they join the area.

We identified two distinct roles in our application, victims and helpers. Victims send messages asking for help. Helpers send messages and consult other people's messages.

A message, composed of a title, a description, a category and the position, will be sent to nearby devices and users can read them. Since not all messages will be received, users will have to set a radius around them in order to receive messages inside this area, a predefined radius will be set in the application.

Users in the area will be able to validate a message, this ensures the freshness of information. We also imagined an automatic message deletion once a message has been confirmed three to five times as wrong.

As explained before, several features will be implemented. The following list is not exhaustive but here are the most important features sorted by priority:

- Send messages with a title, category and position
- Receive every message in a configured zone
- Ask for confirmation about a message
- Filter the messages by categories, distance, date. The filter may vary during the implementation.
- See the messages on a map, each message will have a marker
- See deleted messages that are in a bin
- Adding images to a message to provide a better understanding of it

Other functionalities might be implemented. The full detail of them can be found in the product backlog of the implementation.

There will be several requirements. Because we do not know what technology will be used, we established the following table that summarizes the requirements of the technologies analyzed before. All of the technologies support basic unicast but no information was found about multicast and broadcast support.

Feature	Bluetooth API	Wi-Fi Direct API	Google Nearby	MeshKit
Minimal Android Version	2.0	4.0	2.3	—
Network interfaces	Bluetooth	Wi-Fi	Bluetooth — Wi-Fi — Ultrasounds	Bluetooth – Wi-Fi
Need infrastructure	No	No	No	No
iOS compatible	No	No	Yes	Yes

2. Table: Comparison of Available Technologies

The application will provide a quick and easy service to use for people in desperate need. It can help people organize themselves when rescue teams did not arrive, which can save lives or aid people share vital resources. Using ad hoc networks allows the exchange of messages, we analyzed similar solutions in the following part of this document but none of them respond to this particular problem.

However, the application will be only be developed on Android, there is no Apple iOS support yet. Despite the fact that it is important to keep in mind that an iOS application might be a future evolution and the technology choice need to take that in account. On top of that, this type of network heavily relies on people nearby: if there are not enough people using the application in a given area it will be impossible to exchange information.

In the following parts, we will explore what solutions exist on the market and how our application differs in terms of functionalities and requirements. After that, we will summarize the key differences of all the previous mentioned solutions.

3.1.1. goTenna

13. Illustration: goTenna Logo (goTenna, n.d.)

goTenna is a relatively new company, founded in New York, whose startup wants to create decentralized communications. The idea came after the hurricane Sandy that hit the United States in 2012. Daniela Perdomo saw the flaws of a centralized network when cell towers went down leaving millions of people alone (Rhodes Margaret, 2014).

At that time, cell phones were not able to establish a Bluetooth connection with other devices that were more than 10 meters away. An external contraption was required, and one week later Daniela already drew a prototype (Rhodes Margaret, 2014).

Since then, the company launched two portable devices, one aimed at professionals, the goTenna Pro and one simpler for the mass market, the goTenna Mesh. They both provide the same service; the professional model has some additional functionalities that are not relevant to the public. The mesh and the Pro are both simple Ultra High Frequencies (UHF) radios; however, the pro can emit on the Very High Frequencies (VHF) spectrum.

14. Illustration: goTenna Mesh (goTenna, n.d.)

15. Illustration: goTenna Pro (goTenna, n.d.)

Difference Between goTenna Mesh and goTenna Pro

As said before, the two products are not aimed for the same public. They share many similarities but are technically very different. The following chart will highlight the main variations.

	goTenna Mesh	goTenna Pro
Output power	1W	5W
Frequencies	UHF	142-175 MHz (VHF) 445-480 MHz (UHF)
Battery life	24 hours	40 hours
Antenna	Non-removable	Removable, SMA connector
Protection	No protection	IP68 + MIL SPEC
Range	6.4 km in open area 0.8 km in a congested area	—
Encryption	End-to-end 384-bit elliptic curve encryption	End-to-end 384-bit elliptic curve encryption

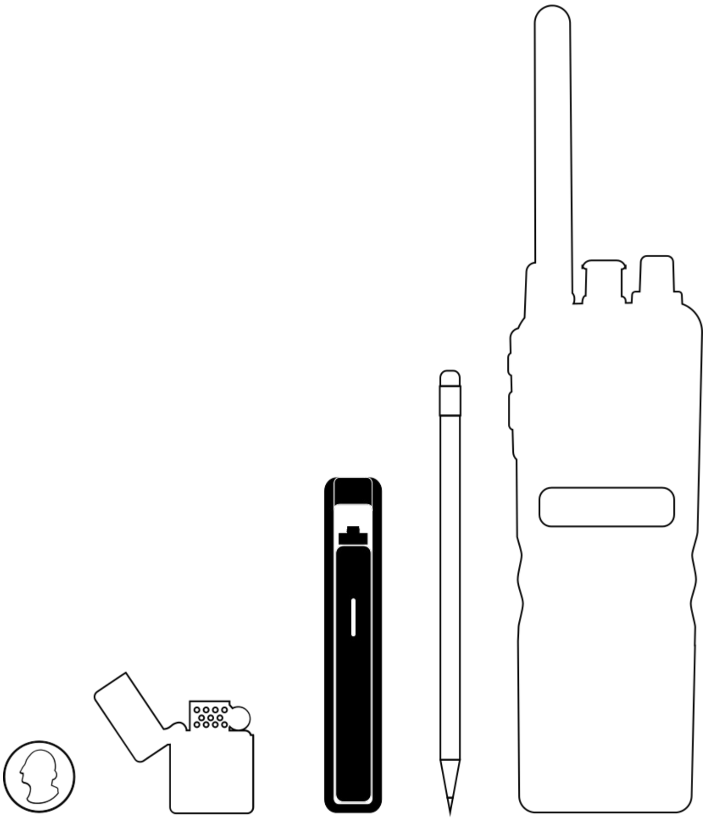
3. Table: Comparison of goTenna Mesh and Pro (goTenna, n.d.)

How It Works

In this section we are going to see the default feature that the Mesh and the Pro have in common, where another chapter will be dedicated to the goTenna Pro additional features. By default, the Mesh and Pro are just antennas, to use them a connection with a smartphone is needed. The device automatically broadcasts information and creates a decentralized mesh network. Once the device is online, users can send a message and share their position in either a private conversation or a group chat. All communications are encrypted and cannot be tempered.

The goTenna Mesh and Pro have an excellent point-to-point communication range but can be widely increased by hopping through other goTenna units in the area. Even if not paired with a phone, a goTenna device will act as a relay node when switched on.

What Else Does the goTenna Pro Offer



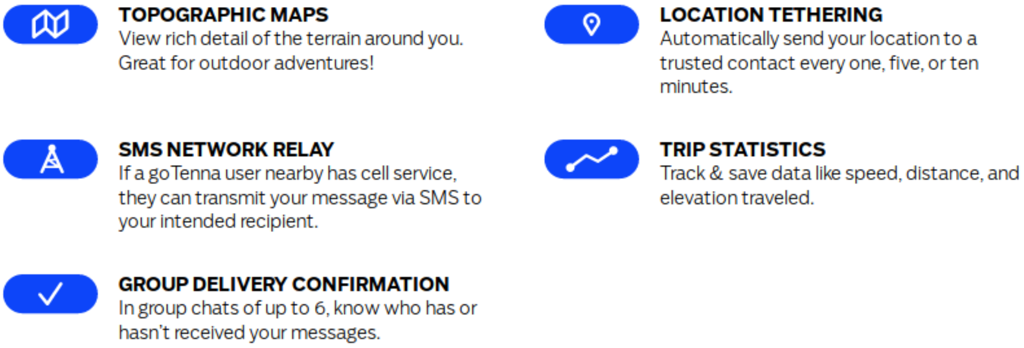
16. Illustration: goTenna Pro size (goTenna, n.d.)

There are several additional features in the goTenna Pro, such as a higher emission power, 5 W instead of 1 W, sturdier build quality with an IP68, dust tight and protected against longtime immersion, and MIL-SPEC certification. On top of those elements, the goTenna Pro offer professionals more flexibility. The device can be programmed and can work on both VHF 142-175 MHz and UHF 445–480 MHz spectrum. This allows a goTenna device to join an existing infrastructure if needed. goTenna advertises its goTenna Pro as ten times lighter and ten times smaller than legacy radio. The device weighs only 80 grams and has a volume of only 67 cubic centimeters without an antenna. On the illustration above, goTenna compares its Pro model with everyday objects.

goTenna also offers a deployment kit that stores up to 30 goTenna Pro. The deployment kit offers a wide range of features such as a local Ubuntu server or even the charge, transport and updates of the Pro models stored inside. The case is rugged and weights only 11 kilos making it easily transportable.

17. Illustration: goTenna deployment kit (goTenna, n.d.)

goTenna plus is another service they provide. It is a subscription-based application designed for people that need more functionalities. The additional features users get with the goTenna plus are the following:



TOPOGRAPHIC MAPS
View rich detail of the terrain around you. Great for outdoor adventures!

LOCATION TETHERING
Automatically send your location to a trusted contact every one, five, or ten minutes.

SMS NETWORK RELAY
If a goTenna user nearby has cell service, they can transmit your message via SMS to your intended recipient.

TRIP STATISTICS
Track & save data like speed, distance, and elevation traveled.

GROUP DELIVERY CONFIRMATION
In group chats of up to 6, know who has or hasn't received your messages.

18. Illustration: goTenna Plus features (goTenna, n.d.)

Online Reviews of goTenna Products

We did not order goTenna products to test them; it was not useful for our use case. We searched online to see what the general opinion of those devices is. Because the goTenna Mesh is the commercial product, we mainly found reviews of this model and none about the Pro exemplary. Here are the main advantages:

- The device is easy to use, and the setup is a breeze. Once the smartphone and the Mesh are paired there is nothing else to do.
- It is an excellent price to service ratio; there are other more expensive solutions, yet they, that do not offer all the functionalities the goTenna Mesh offers.
- Some people adhere to the decentralized and free network the Mesh creates. This ideology is found in multiple reviews or article.

Even if there are many things to love about the products, some review highlighted the following flaws:

- Some people had problems with the Android version of the application; the device would keep disconnecting. This issue might come from the Android platform
- Some customers outside the United States had problems with the country code of their phone number, and had to make configuration changes in the application.
- The value added by goTenna plus was not sufficient for some reviewers either. The topographic maps were not as functional as expected.

We went on multiple websites to found different reviews. We took Amazon product review (Amazon, n.d.), a review from More Than Survival (Xavier Thomas, 2017) and the website of Jen Eric Ramblings (Ramblings Eric, 2017).

Pricing of the Products

The price is taken from the official website and ordered from the United States to avoid custom and delivery charges; a goTenna Mesh pair costs $179 and a goTenna pro $499 per unit. The goTenna deployment kit costs $20,000 and is delivered with 20 goTenna Pro.

There were little to no complaints about the price of the products. goTenna advertises its Pro model as 20 to 40 times cheaper than a legacy radio system, but they give no details on how they achieve such a difference and what they include in the price calculation.

Advantages of goTenna Products

Even if we did not order goTenna products to test them, we were able to identify what is right about goTenna products:

- The service they provide is well implemented and works flawlessly for most of their customers. The application is user-friendly and is regularly updated (Ramblings Eric, 2017).
- The Mesh, as well as the Pro model, is well designed, the look and the form factor is appealing
- We think that the price is correctly balanced for both models.
- The devices not only work in point-to-point mode but can also create a mesh network that will tremendously increase the range.

Drawbacks of goTenna Products

- Because of its lower price, the Mesh is not waterproof; which is shameful for a device aimed at hikers.
- Even if the application is great, users can only share text messages. There are no possibilities to call someone. This is not a concern for the Mesh, but it might be one for the Pro: goTenna advertises its products as a replacement for legacy radio. However, they do not offer voice communication which is mandatory in some situations.
- The devices are excellent and well built. However, this is a hardware device. Information sends from a Mesh can only be received by others if they use a goTenna product.
- We found online that some customers had problems with the Android application, random disconnections and country code issues. However, this might have been corrected with an update of both the Mesh and the application.

Difference With Our Use Case

Even if we want to develop an application, we share some features with goTenna's products. Where we stand out is in the ease of access and the target audience.

Downloading a new application is swift: a simple Internet connection is needed. This makes our solution more comfortable to use. Many people have a smartphone and an installing an application is a common task.

We do not target our application to hikers, it is not the audience we want to reach. As said before, our application is meant to be used in case of a catastrophic event, when there is no other communication mean left, and it is hard to find survivors. Our application will not rely on an external hardware device, everything is contained in the smartphone of the user.

3.1.2. FireChat

19. Illustration: FireChat logo (Open Garden, 2018)

FireChat is a proprietary mobile application developed by Open Garden, a San Francisco-Based company. It was created in 2015 and use MeshKit a proprietary framework that adds peer-to-peer mesh connectivity to any application, chapter 2.3.3 – MeshKit (p. 24). The application was used many times in the past.

How It Works

We already covered the details of MeshKit and how it works. In this chapter, we will focus on how the application works, what are the steps needed to make it work. There will be no technical explanation; they can be found in chapter 2.3.3 – MeshKit (p. 24).

An account is needed to use the application. FireChat does not need anything other than an email address to connect since it sends an email with a code. If privacy is a concern, FireChat needs a valid email; it is the only way to reconnect to an account.

Once the account is created the user has several methods of communication:

- Public chatrooms: those chatrooms can be accessed by anybody in the world, regardless of the network availability.
- Private messages: fully encrypted classic messages, which are sent to only to users in the chatroom. Those chatrooms can have up to 50 people.

On top of those two types of chatrooms, users can follow other users, block users, send a photo that only the following users can see and have a favorite chatroom.

Pricing of FireChat

FireChat is free; there is no paid tier. We could not find in the privacy policy if FireChat and Open Garden reserved the right of selling data to third parties.

Advantages of FireChat

FireChat is, without a doubt the best-known mesh networking application; it has been quoted in the press on numerous occasions. Here are some advantages we found:

- The setup is quick; only an email is required to allow users to exchange messages.
- The public chatroom is a great way to exchange about a subject. They are easily created and can host an unlimited number of people.
- The application looks nice and is not cluttered with settings.

Flaws of FireChat

- In case of a catastrophic event with no Internet connection available, users will not be able to validate an email. We do not understand why the application needs a valid email address to connect.
- The only information that can be exchanged by the application is text and pictures/videos. There is no possibility way to attach a location to a message, this makes it hard to use in case of a disaster other than simply broadcasting information across an area.
- Even if the application looks nice, it can be confusing sometimes; there are four tab and no explanation related to their purpose. We had to look at the online documentation to understand how the application works.
- The application is not privacy-friendly oriented and the fact that the company does not clearly state its privacy policy has raised doubts.

Difference with Our Use Case

Even if we share many similarities with FireChat, our application is not addressed to the same public and does not solve the same problem.

We will focus our application on catastrophic events, FireChat has been used in those kinds of scenarios, but it is not its main purpose. Because of our direction, some social features will not be implemented. On top of that there is no possibility to send a position to a chat or a private conversation. Meaning that floating content is not an option with FireChat.

Since we will focus on disaster rescue help, we want to attach the position to every message. There will also be some categories for a message. Ideally, a map will be implemented in the application and all the message in the area will be displayed on top of it. All those functionalities are not implemented in FireChat which set the two applications apart.

Furthermore, our application will not require an account creation; users will exchange messages as soon as the application is installed on the phone. We will ask for a name at the first application launch, as this is a vital information when searching for survivors in a devastated area.

The creation of private conversation is useful, but it is not the essential functionality we want to have. Public chatrooms might be implemented in the application, but are not the priority.

Because we had problems finding some information on the application we want to make it as simple as possible. We can easily imagine that people in a distress situation will not be able to understand a complex application, so the simpler the better. There will also be a user's guide available in the application.

3.1.3. Nearby Chat

20. Illustration: Nearby Chat icon (Swain Daniel, 2017)

Nearby Chat is a beta application showcasing the Google Nearby API. No technical documentation is available on this application; we believe that Google Nearby Message API is used because the application does not work without the Internet. The application is a side project of a single developer who wanted to experiment with Nearby.

How It Works

The application is effortless, a pseudonym is the only information to enter, and users can exchange messages with people nearby (within 30 meters). The messages are ephemeral as they are deleted as soon as the user closes the application or is out of range.

Pricing of Nearby Chat

The application is free.

Advantages of Nearby Chat

There is not a lot to say about the application. The advantages are the swiftness of use, nothing is required apart for a name.

Flaws of Nearby Chat

We cannot consider the usage of Google Nearby Message API as a flaw. It is an experiment from where the developer wanted to explore the API. For our use case, it would be a disadvantage because users would not have access to the Internet.

Differences With Our Use Case

Our application will be very similar to Nearby Chat; we want a quick process that allows the user to be connected within seconds. However, we need to use an API that does not require the Internet, otherwise our application will become quickly useless.

Private messages might also be a future evolution of our application; Nearby Chat does not support this.

3.1.4. Zello

21. Illustration: Zello logo (Zello, 2018)

Zello is an application available on Google Android, Apple iOS, and Microsoft Windows. It simulates a walkies-talkies push to talk experience. People can communicate by voice all around the world using the cellular network.

Even if the application is infrastructure based, we decided to insert it in this paper since it has been used during disasters.

The application hit the news multiple times because it allowed people to communicate in a challenging environment. It was the case during the Turkish protest of 2013 (Arthur Charles, 2013). During the anti-government protest of 2014 in Venezuela, Zello was also used by protesters to organize demonstrations (Zhang Sarah, 2014). Another usage of the application was during the hurricane Irma that hit the United States between August and September 2017 where rescue teams used the app to save people trapped by the flood (Holley Peter, 2017).

How It Works

Zello simulates two-way radios; this allows users from all over the world to talk one to another. Once an account is created, which is mandatory, it is possible to send a voice message to either one person or to a channel where people can freely join; there are no user limits.

Channels are not necessarily open, the application offers multiple options:

- Open: anybody can talk
- Zelect: channel owner, moderators and authorized users can talk
- Zelect+: channel owner, moderators and authorized users can talk and listen
- Zellocast: channel owner and moderators can talk

Two other options are available; password protection for the channel and paid access. In case of paid access, Zello will take a part of the access cost and send the channel's owner what is left monthly via PayPal.

Pricing of Zello

Personal usage of Zello is free; anyone can access to channels and talk without limits. Zello developed a solution aimed at companies called ZelloWork. Because it is not aimed at our target audience, we will not detail the conditions of ZelloWork in this paper.

Advantages of Zello

- No limits are set to public channels, up to 450,000 people joined the same channel during the Venezuelan protests (Zhang Sarah, 2014). No other solution on the market offers such an easy way to connect that many people together.
- The fact that Zello has been used in disaster-stricken area proves that it is suited for those scenarios. It shows that it is capable of providing the help people need at that time.
- There are several ways to join channels making it easy for users to search for help.

Flaws of Zello

- Voice is the only ways to communicate inside Zello. Users cannot send a text on channels, making the work for rescue teams harder if the person does not talk the local language or has a strong accent.
- Zello cannot be used without the Internet. If no active connection is available, the users will not be able to connect and ask for help.
- An account is mandatory: this is not a problem if people have the time to prepare themselves for a disaster, but if they are surprised they might not be able to set the

application up. During the hurricane Irma, the heat map of account creation followed the forecast of the hurricane path (Holley Peter, 2017).

Difference With Our Use Case

Because it has been used during a real disaster, Zello proved that walkie-talkie like communication could help both rescue team and victims. We did not think of voice as an option within our application, but we might consider adding this feature if the technology we used allows it.

The main difference between Zello and our use case other than the means of communication used is the required Internet access. Our application will not require the Internet to work; this will make our application suited for an area that is profoundly affected by a catastrophe or a war.

We also want to develop an app that does not require the creation of an account. The only information it will require is a name, which is, in our opinion, mandatory to help the rescue team identify survivors.

3.2. Comparison of the Existing Solutions

We have analyzed four competing alternatives to our application, each of them provides an excellent service in their category. They all address a specific problem, but none of them are suitable as support during a disaster. The main functionality they lack is the position management. None of them support floating content, except goTenna that allows users to share their position.

One significant difference between our application and the competitors is the way messages are treated; they have all an instant message solution whereas our does not. The messages are displayed on every phone in an area, but users cannot comment or react to them. The only interaction possible with messages is the confirmation or invalidation of the information.

We do not discard the option of an instant messaging part for our application. This can be useful in some scenarios such as search organization or updates on resources available at a specific place. However, this is not the main functionality of the application.

This concept also involves that our application will only work in broadcast mode because no one-to-one or one-to-many communications are planned. This can also be as excellent features to implement but it is not a priority for us.

The following table summarizes the fundamental differences between our implementation and the existing solutions on the market.

Feature		**Our app**	**goTenna**	**FireChat**	**Nearby chat**	**Zello**
Instant message based		✗	✓	✓	✓	✓
Require specific hardware		✗	✓	✗	✗	✗
Infrastructure-based		✗	✗	✗	✗	✓
Require the Internet		✗	✗	✗	✓	✗
Message supported	*Text*	✓	✓	✓	✓	✗
	Voice	✗	✗	✗	✗	✓
	Position	✓	✓	✗	✗	✗
	Images	~	✗	✓	✓	✗
Require an account		✗	✓	✓	✓	✓
Routing type	*Unicast*	✗	✓	✓	✗	✓
	Multicast	✗	✓	✓	✗	✓
	Broadcast	✓	✓	✓	✓	✗

4. Table: Comparison of our use case with the existing solutions

4. Use Case Implementation Reflection

Because we developed an application that will be used during a disaster, we had to project ourselves in this kind of situation to see the problems might occur. Once they had been found, multiple solutions were searched and analyzed. The resolution which seemed to us the most relevant was then selected.

We will detail in the following part all our reflections and explain the reasons why we chose one solution over another. This part is separated into three distinct chapters; the first one is devoted to the network management, and the problems related to the application back end. The second is dedicated to the user experience challenges, where problems such as error handling are discussed. Finally, the last chapter will discuss the user interface challenges we faced. In this last part, we will present the mock-ups we designed during this reflection phase.

This chapter focus on the reflections made before the development phase of the project. All the technical details and the methods used in the development can be found in the chapter 5 - General Implementation (p. 56) and 6 – Details of the Implementation (p. 67).

4.1. Challenges — Application Back End

4.1.1. Information Freshness

During a catastrophic event, situations can evolve rapidly. Therefore we need to keep the messages as updated as possible and since information might be wrong, it has to be handled by the system. We found two ways of handling this issue.

Creating a Fixed Expiration Date for Messages

This is the most basic implementation. Messages will be automatically deleted after a programmed time ensuring that all messages are fresh and up to date. However, this comes with its limitations. Some messages need to be kept longer than others, for example, a gas leak in the street is crucial information that cannot be deleted.

People will be under pressure, and we do not want them to change a critical aspect of the application, this is why this setting will be hard-coded and not modifiable. More information on this choice in the chapter 4.1.7 Application Settings (p. 47).

Asking for Updates About Uncertain Messages

A dynamic expiration date is a better alternative; nearby users need to give feedback on messages. A positive confirmation will postpone the expiration date whereas negative will reduce it.

This is a better choice because urgent messages will be kept. However, this relies on user participation and it raises two problems: the most obvious is user participation; if people are busy, messages will be deleted and lost. The other issue may occur in a situation where there are many messages in the same area, confirming them can become time-consuming and users might stop doing it.

Selected Solution

We decided for our application to develop a simple dynamic expiration date because it provides the best solution to the information freshness in our opinion.

4.1.2. Information Trustfulness

A wrong message can lead to severe time and resources loss. Therefore, two separate options were evaluated to guarantee the accuracy of the information.

Scoring Users

One of the first ideas we had was to create a score for every user: their score will be increased at each positive confirmation of their message and reduced if someone reports it as wrong. If a user has too many wrong reports, he will not be able to send new messages, or his messages will not be considered trustworthy.

Rely on People's Honesty

According to the situation, it is unlikely that people would intentionally create fake messages. In this case, no trust system will be implemented, and all messages will be considered reliable. Errors are still possible and they are handled differently; we detail the way we take care of them in the point 4.2.1 – Error Handling (p. 49).

Selected Solution

We decided, for ease of implementation, to rely on people's honesty. This choice has been motivated by the fact that people in a distress situation are less likely to deceive others.

4.1.3. Floating Content or Geo-Fencing

Floating content is an essential part of our application; it allows the exchange of messages in a restricted area configurable by the user to avoid information overflow. We imagined two different strategies for the implementation of this feature.

Ignore Strategy

This strategy is the strictest. Each message received is tested: if it is within the area of the user, its configured radius, the message is saved and displayed to the user. This design is easier to implement because it only requires a simple test during the reception.

The potential data loss is the biggest problem with this handling method. If there are only a handful of peers connected in an area and one of them leaves, all the messages saved will be deleted as soon as the device exits the zone. We already discussed this problem in the chapter 1.5.4 – Position-Based Protocols (p. 9).

Save and Hide Strategy

A safer strategy would be to save all received messages and to display only the one that are inside the user's proximity. This would correct the problem of potential data loss but at the same time it would result in a more significant quantity of data.

Selected Solution

We decided to opt for the save and hide strategy as it provides a safer and more consistent experience for the users. The quantity of data increase will not be a problem in our opinion because the application only exchanges text messages.

4.1.4. Duplicate Data

Duplicate messages can appear in areas with much activity. This situation can create problems in some case. To avoid issues, it has to be handled automatically by the system. We did not find multiple strategies to implement this kind of functionality in the application.

The system we established uses the user's positions to check if a message with the same category already exists in the vicinity of the latter. If that is the case, a popup message will ask the user if the message he wants to send is the same as the one nearby.

4.1.5. New Peer Connection

When a new peer connects, he needs to receive all the exchanged messages of the area. However, a user cannot receive a message he already has, because this would create duplicate data. This type of situation occurs when a user leaves the zone or switches his phone off. Indeed, all the messages already received are stored in the phone memory, and they need to be ignored at the reception. We did not find multiple ways to solve this problem other than having a strict data check when new messages appear.

4.1.6. Message Saving

When using the application, exchanged messages are saved on the application's cache. Once the application is closed, the cache is flushed, and at the next launch, no message will appear. This is why permanent storage is required. Two compatible options with our need are available on Android: shared preferences and internal file storage. We do not need to create an SQLite database as we only store messages and the creation of a database is not justified.

Shared Preferences

Shared preferences are a key-value association. They are saved across sessions and they can store any primitive types such as Boolean, float, long, string. They are recommended in scenarios where not many data are saved. For more massive data sets, the internal storage is recommended.

Internal File Storage

Each Android application has a private folder that is not accessible for users unless they have root access. It can store all types of files and it does not have a space limitation.

Selected Solution

We decided to use a different shared preferences file for each type of messages as it provides enough functionalities for our needs. In the future, if storage becomes an issue, it would be wise to create internal files. However, after the tests we conducted, shared preferences are enough.

4.1.7. Application Settings

We want to let users have some control over the application by allowing the change of some parameters such as the username and the radius for the floating content. However, the user should not, in our opinion, be able to change fundamental settings such as the ones related to the network management. According to our thinking, the critical settings are related to the application back end. They can be, for example, the expiration time for messages or advertising, and discovery settings. When changing those settings, the operation of the application changes and it can lead to information loss.

4.1.8. Offline Challenges

Google Nearby Connection API does not require any Internet connection. This means that our application can be used regardless of the network availability. However, this does not mean that our application does not require the Internet at all. Several points can block the use of the application if they are not handled.

Installation of the Application

Users need to have the application installed before the disaster occurs. If that is not the case, they will not be able to communicate. To address this issue, we imagined a sharing feature in the application where users can send the installer one to another. This would allow unprepared users to access the mesh network created and exchange information.

However, this type of communication is hard to implement since an application needs to act as a server. To make the exchange easier, we imagined a workaround: during the installation, the installer, an Android Package Kit (APK), will be placed in the download folder of the device. To transfer the APK to other devices, users only need to send it via Bluetooth. Although this provides a solution, it requires absolute mastery of the phone, and we do not believe it is within everyone's reach.

This best solution would be, as explained above, a button to click on the device that would send the APK to anyone nearby.

Maps

Floating contents requires a map when an Internet connection is available the application will download the Google Map required. However, if there is no connection, it will not be possible to see the map unless it has been downloaded previously.

Google Maps does not provide this kind of feature by default outside the official application. It is possible, however, to generate URL depending on the position of the user to get the image of the map. Once this link is generated, we can download the images and display them instead of the one from the Internet.

Alternatives, such as Open Street Map are more flexible with the offline use of their map.

4.2. Challenges — User Experience

4.2.1. Errors handling

Users might make mistakes while sending a message because of the pressure they will be under. This is entirely normal, but error handling feature needs to be implemented.

Delay Before the Actual Transmission of the Message

A small delay, around 10 to 30 seconds, between the press of the send button and the real transmission of the message can help prevent some mistakes. After the sent button has been pressed, the message is placed in a queue, where it is still editable. It is sent to all connected peers once the delay has been reached.

Update a Message

Another approach of the problem would be to immediately send the message but it would allow users to update their messages afterward. However, this strategy will create additional traffic for every modification made by the user as a message will be sent another time.

Summary of the Message Before the Transmission

Proofreading a message is also a feasible option; once the sent button has been pressed, a new view containing a summary of the message will be displayed. The user will then confirm that no error is present. Once the confirmation is given, the missive will be transmitted to nearby peers.

Selected Solution

We concluded that message reviewing was the best solution because it is easier to implement than a delay and it does not generate additional traffic. We still want users to be able to update messages in the application; if there are still errors, there will be an option to correct them.

4.2.2. Reassure the Victims

Because victims can be in a situation of extreme distress, they need to be reassured by the application. We will display the number of peers that received the message. This information is displayed in all the messages, which is, at the same time, an excellent indicator of the network's health.

4.2.3. Messages Identification

Since our application relies on message exchange, a message identification system is compulsory. This system will allow nearby users to differentiate messages and enable quick identification of the message type. Naturally, each category needs a name; we approved, therefore, the following classification: “Victim”, “Danger”, “Resources” and “Caretaker”, they represent the type of data users are likely to exchange.

However, we assumed that text categories are not enough. We need to improve the category distinguishing even further. Color-coded categories were our first thought, which would vastly help the recognition but it may not be enough for some people. This is why, in a second reflection, we decided to use icons and colors for each category. Details about the colors and the icons used can be found in the chapter 4.3.2 – Colors (p. 54) and 4.3.3 – Iconography (p. 54).

4.3. Challenges — User Interface

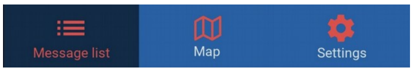
4.3.1. Application Layout

Before starting the application implementation, we decided to make mock-ups of the future user interface; this helped us find vital components of the application. The following screenshots come from the mock-ups previously mentioned. There are color and design differences between the picture presented below and the final application design.

Colors have been changed soon after the creation of the mock-up because they did not provide enough contrast between elements present on the screen. Explanations and the presentation of the color palette are in section 4.3.2 – Colors (p. 54). Details regarding the design change can be found in 6.3 – Application User Interface (p. 76).

Bottom Navigation Menu

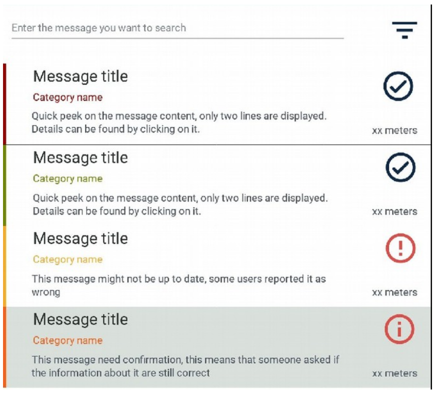
As previously mentioned, the application will be used in a stressful environment and needs, therefore, to be as simple as possible; we decided to use a bottom menu that will contain all the different menus required. It will contain the three fundamental views of the application. The first one contains a list of messages, the second one the map where all messages are displayed and lastly the application settings.



22. Illustration: Mock-up bottom navigation menu

Message List

The message list will contain all the messages near the user. It needs to have as much information as possible so that users can have all the information related to a message in a fraction of a second.



Enter the message you want to search

Message title
Category name
Quick peek on the message content, only two lines are displayed. Details can be found by clicking on it.
xx meters

Message title
Category name
Quick peek on the message content, only two lines are displayed. Details can be found by clicking on it.
xx meters

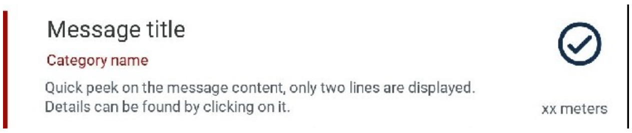
Message title
Category name
This message might not be up to date, some users reported it as wrong
xx meters

Message title
Category name
This message need confirmation, this means that someone asked if the information about it are still correct
xx meters

23. Illustration: Mock-up message list

Users can see for each message present on the list, its title, description, category name, and distance. On top of that, the color band on the left shows the category's color and the icon on the right shows how reliable the message is.

We placed this icon to inform users if the message is still up to date or not; the icon becomes a red exclamation point once the information might be outdated.



Message title

Category name

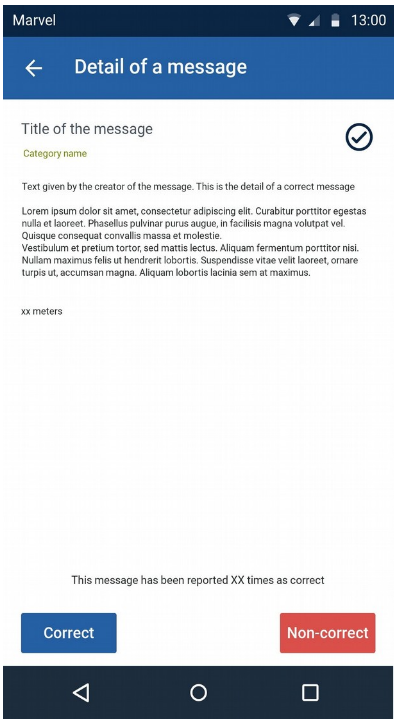
Quick peek on the message content, only two lines are displayed.
Details can be found by clicking on it.

xx meters

24. Illustration: Mock-up detail of one message

Message Details

When a message has been selected, a view containing all the details is presented. Two buttons are present at the bottom of the screen; they are used to confirm or reject the message state. This is used for the dynamic expiration date.



Marvel 13:00

Detail of a message

Title of the message

Category name

Text given by the creator of the message. This is the detail of a correct message

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur porttitor egestas nulla et laoreet. Phasellus pulvinar purus augue, in facilisis magna volutpat vel. Quisque consequat convallis massa et molestie.
Vestibulum et pretium tortor, sed mattis lectus. Aliquam fermentum porttitor nisi. Nullam maximus felis ut hendrerit lobortis. Suspendisse vitae velit laoreet, ornare turpis ut, accumsan magna. Aliquam lobortis lacinia sem at maximus.

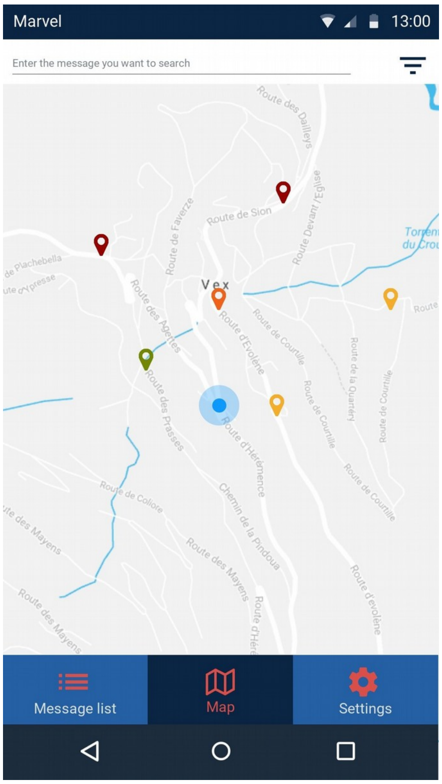
xx meters

This message has been reported XX times as correct

25. Illustration: Mock-up messages details

Map

We decided to display the information of the messages on a map, this is more visual and it provides an overview of the situation of an area. In the beginning, we did not think of icons to represent the various categories of a message, which is why there are only colored pins on the following picture.



26. Illustration: Mock-up messages map

4.3.2. Colors

Our application color changed between the initial mock-up and the final implementation. This was due to difficulties to identify the different elements on the screen. To help us find the correct color palette, we divided them into two main categories. One part is related to the design of the application, and another is for the different categories a message can have. A third category with less critical colors have been established too; we call this group utility colors.

Application Colors

We wanted to have sober colors for the user interface. By doing so, users will have less trouble to find relevant information on the screen as they will have brighter colors and they will, therefore, be more visible. With the material design used on the Android platform, applications have two types of colors at their disposal.

- Primary colors: which are the most prominent as they will be present in many elements on the screen, such as the action bar on the top or the buttons.
- Secondary colors: which help users identify selected parts of the user interface. This is also called the accent color.

Each color type can have light and dark variations; this provides a contrast between on-screen elements. Lastly, text color is also selected for both categories and will be used throughout the application to ensure consistency.

The following table shows what the primary colors and its variants are. We chose a tuna color as it will not interfere with any other element. Tuna is a sober color and message category colors will be readily identifiable as they are brighter.

⬛	⬛	⬛	
Primary color	Primary light	Primary dark	Primary text
#33333d	#5d5c67	#0c0c17	#ffffff

5. Table: Application Primary Colors

To avoid confusion between the user interface element and the categories color, we chose an accent color that is not present anywhere else. This is why this light blue and its variations were selected.

🟦	🟦	🟦	⬛
Secondary color	Secondary light	Secondary dark	Secondary text
#80deea	#b4ffff	#4bacb8	#000000

6. Table: Application secondary colors

Message Category Colors

Category colors are crucial for our application; they must be easy to differentiate and provide information of the message type as soon as seen. As said before, we approved four different categories type “Victim”, “Danger”, “Resource”, and “Caretaker”. The first two types were related to a hazard and they needed to have a reddish color. The other two are concerning help and they must have a color in green tones.

The tint selected is bright and easily differentiate from any other color present in the user interface.

🟥	🟧	🟩	🟩
Category victim	Category danger	Category Resource	Category Caretaker
#e3170a	#f26419	#9bc53d	#60992d

7. Table: Message categories color

Utility Colors

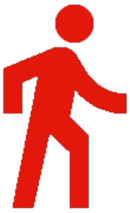
The last two colors present on the application provide a visual feedback for users, as they are used to show the status of elements. The selected tones are easy to differentiate from the colors selected for the categories, which is our goal as we do not want to confuse the users.

🟥	🟩
Error color	OK color
#b00020	#20b000

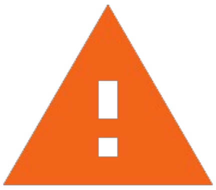
8. Table: Application utility colors

4.3.3. Iconography

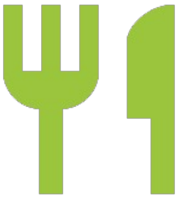
Icons were also required in our application since they provide a better message category recognition. They need to be unique and easy to distinguish. The selected icons for each category in their corresponding colors are the following.



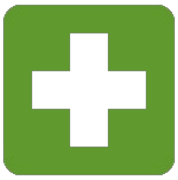
27. Illustration: Victim icon



28. Illustration: Danger icon



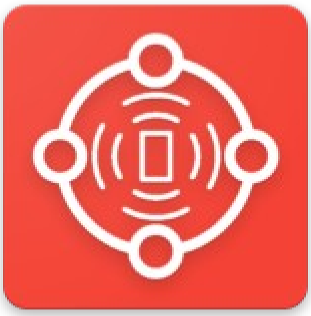
29. Illustration: Resource icon



30. Illustration: Caretaker icon

4.3.4. Application Icon

We wanted to create a simple, yet effective application icon. The objective was to have a bright background and a simple illustration on top. The final design is the following:



31. Illustration: Default application icon

32. Illustration: Round application icon

The illustration represents several aspects of the application. The outer circle symbolizes the mesh network connecting all peers together as well as four people holding their hands. The inner part shows a smartphone sending information to the network. Moreover, the round version looks like a lifebuoy to us.

5. General Implementation

We decided to split the implementation into two parts. The first one will cover the general implementation; it includes the technology and the programming language choice, for example.

The second will detail some of the aspects evoked during the use case reflection; we will demonstrate how fundamental elements of the application have been handled. The result of those implementations will be analyzed in the "Discussion" chapter (p. 79).

5.1. Technology Choice

Because an Android application will showcase the use case we chose, we have to select a technology available on Google's operating system. However, we saw a portage on the Apple's ecosystem as a potential future evolution. This is why we want to select a technology that is already well supported and available on both platforms.

During our initial research, we analyzed many different technologies. Some of them were quickly dismissed because of their lack of support by the major operating systems and their insufficient functionalities; this is the case of the Serval Project and the Bluetooth 5.

Using directly the Bluetooth and Wi-Fi direct API developed by Google might be an option, this provides a greater flexibility than tools like Google Nearby or MeshKit. However, this choice is made at the expense of ease of development because everything needs to be implemented by the developers. Besides, this would have made the application hard or even impossible to develop for iOS devices since Apple does not give the same accesses as Google does.

Google Nearby Connection API is therefore the best option for our use case. The API is a secure choice for the future because Google has a dedicated team devoted to the development and support of it. On top of that, using Google Nearby ensures an easier maintainability of the application over time because new technologies can be added to the one already supported by the framework.

5.2. Programming Language

Android supports both Java and Kotlin, the latter have been supported since the release of Android Studio 3.0 and is inter-operable with Java. Although Kotlin brings some improvement such as less code, about 20% less, to achieve the same result, more readable code and performance improvement (Sommerhoff Peter, 2018). Because we had no experience with it, we decided to develop our application with the more traditional Java.

If in the future, if a conversion is necessary, it can be done with Android Studio which provides a Java to Kotlin translation; some adjustment might be required to make the application work as intended. However, since both languages can be executed at the same time, the conversion can be divided into multiple steps.

5.3. Project Architecture

The project is split into several packages to support the work of the developer. It is divided as follows:

- Adapter: contains the custom adapters created for the several lists present in the application.
- Model: the model of objects used in the application, contains the Message and Endpoint class.
- Utils: mostly consisted of abstract methods. This is where the configuration of Google Nearby or the communication is handled, for example.
- View: all the codes related to the views present in the application. We created sub-packages to help the organization of the numerous files.

5.4. Gradle Dependencies

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.

The following dependencies are required for the implementation of the user interface.

```
implementation 'com.android.support:appcompat-v7:27.1.1'
implementation 'com.android.support:design:27.1.1'
implementation 'com.android.support:cardview-v7:27.1.1'
implementation 'com.android.support.constraint:constraint-layout:1.1.2'
```

The ones below are related to the application back end. They enable the usage of Google Nearby and GPS. The GSON dependency is used to translate any object into a JSON string; this is used for transferring data across devices.

```
implementation 'com.google.android.gms:play-services-nearby:15.0.1'
implementation 'com.google.android.gms:play-services-maps:15.0.1'
implementation 'com.google.android.gms:play-services-location:15.0.1'
implementation 'com.google.code.gson:gson:2.8.5'
```

5.5. Constant of the Application

Many variables are required in multiple locations, this is why we created a file containing all those constants. In the following part, we will cover the most important constants created and the reason for their presence in this file.

5.5.1. Permission Related Constants

Some permissions are mandatory for the application to run. Without them, some functionalities might not work, which would make the application useless. This is for that reason that a string array of permission was created.

Because it is required by the Android system, we also created the code for the mandatory permission.

```
public static final String[] MANDATORY_PERMISSION =
        new String[]{
                Manifest.permission.BLUETOOTH,
                Manifest.permission.BLUETOOTH_ADMIN,
                Manifest.permission.ACCESS_WIFI_STATE,
                Manifest.permission.CHANGE_WIFI_STATE,
                Manifest.permission.ACCESS_FINE_LOCATION

        };
public static final int CODE_MANDATORY_PERMISSIONS = 1;
```

5.5.2. Message Related Constants

All the messages present in the application are stored in a distinct ArrayList depending on their origin. Because those lists are used in different views and in some of the utils classes, we must store them in an easily accessible place.

```
public static ArrayList<Message> MESSAGES_RECEIVED;
public static ArrayList<Message> MESSAGE_SENT;
public static ArrayList<Message> MESSAGE_QUEUE;
public static ArrayList<Message> MESSAGE_QUEUE_DELETED;
public static ArrayList<Message> MESSAGE_QUEUE_LOCATION;

public static ArrayList<Message> MESSAGES_DISPLAYED;
```

We created the following list of messages:

- Messages received: all the messages the device received
- Message sent: all the messages sent from the device

- Message queue: messages waiting for a peer to connect to be sent. This is used when the user wants to send a message, but there is nobody around.
- Message queue deletion: messages waiting for a peer to connect to be sent. This list is used when a user wants to delete a message he wrote, but nobody was around at the moment he clicked the delete button.
- Message queue location: messages awaiting a location to be sent. If a user wants to send a message, but the application did not retrieve a location yet, the message is stored until the GPS gets a location. Once it is done the message is sent to all connected peers.

Because the messages exchanged can be new, deleted or updated, we needed a way to differentiate those states. This is why we created constants that guarantee that no error is made during the development.

The string is used before the message is sent, the status is added as a field of the message and is checked at the message reception. More information about the process of receiving a message can be found in the chapter 5.7.2 – Message Flow (p. 63).

```
public static final string MESSAGE_STATUS_NEW = “new”;
public static final String MESSAGE_STATUS_DELETE = “delete”;
public static final String MESSAGE_STATUS_UPDATE = “update”;
```

5.5.3. Google Nearby Related Constants

Because peers connect and disconnect automatically, we need to keep track of them. HashMaps are used to allow an easier search and management of these previously mentioned peers. As for the message list, those maps are accessed from multiple locations, this is why they are in the constant file. Details about the “Endpoint” object can be found in the chapter 5.8 – Endpoint Object (p. 66).

The “NearbyManagement” object is the instance used throughout the application to access the abstract class stored in the utils package. This class handles all the configuration, connection, and transmission of data across devices.

```
public static NearbyManagement NEARBY_MANAGEMENT;

public static final Map<String, Endpoint> DISCOVERED_ENDPOINTS = new HashMap<>();
public static final Map<String, Endpoint> CONNECTING_ENDPOINTS = new HashMap<>();
public static final Map<String, Endpoint> ESTABLISHED_ENDPOINTS = new HashMap<>();
```

We keep track of the following devices:

- Discovered endpoints: all the endpoints the device discovered.
- Connecting endpoints: all endpoints the device is currently trying to connect. This was used to avoid duplicate connection attempts at the same time.
- Established endpoints: all the endpoints the device is connected to. Because Google Nearby is built that way, device will always try to connect one to another, this map avoids this behavior.

5.5.4. Preference Related Constants

Shared preferences are files containing key-value pairs; they also require a name. An application can have several preferences files. This is what we decided to use to store application settings, and the messages exchanged. Each message list explained above have its shared preference file and the corresponding key.

We could have created one file containing all messages and differentiate them with the key only, but for clarity reason, we decided to separate them. The implementation details of the saving and retrieving of those preferences are in the chapter 6.1.6 - Shared Preferences Message Use (p. 72).

```
public static final String PREF_NAME_MESSAGE_RECEIVED =
                            “ch.hevs.fbonvin.message.received”;
public static final String PREF_NAME_MESSAGE_SENT =
                            “ch.hevs.fbonvin.message.sent”;
public static final String PREF_NAME_MESSAGE_QUEUE =
                            “ch.hevs.fbonvin.message.queue”;
public static final String PREF_NAME_MESSAGE_QUEUE_DELETED =
                            “ch.hevs.fbonvin.message.queue.deleted”;
public static final String PREF_NAME_MESSAGE_QUEUE_LOCATION =
                            “ch.hevs.fbonvin.message.queue.location”;

public static final String PREF_KEY_MESSAGE_RECEIVED =
                            “message_received”;
public static final String PREF_KEY_MESSAGE_SENT =
                            “message_sent”;
public static final String PREF_KEY_MESSAGE_QUEUE =
                            “message_queue”;
public static final String PREF_KEY_MESSAGE_QUEUE_DELETED =
                            “message_queue_deleted”;
public static final String PREF_KEY_MESSAGE_QUEUE_LOCATION =
                            “message_queue_location”;
```

The application requires other preferences. The value of those variables is saved in the constant file. They are the following:

- App ID: stores the UUID generated during the installation of the application, this is used to identify devices.
- Username: the user can change its username, it is mandatory as it serves as an identifier for peers around.
- Radius Geo fencing: define the area of message reception. The default value is 100 meters.

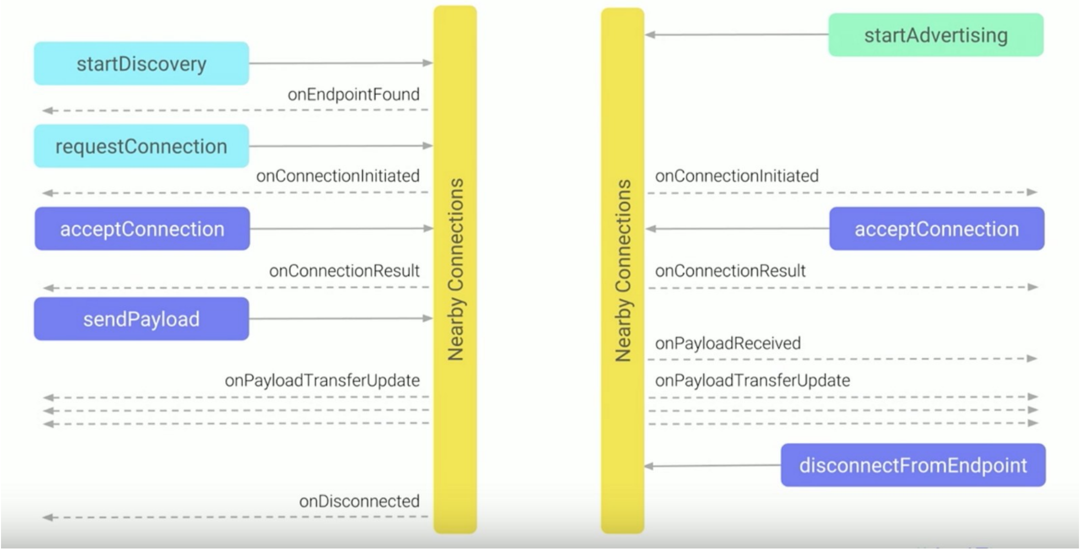
```
public static string VALUE_PREF_APPID;
public static String VALUE_PREF_USERNAME;
public static String VALUE_PREF_RADIUS_GEO_FENCING;
```

5.6. Google Nearby Configuration

5.6.1. Connection Procedure

The connection procedure of Google Nearby consists of two distinct roles. The advertisers advertise themselves and the discoverers search for advertisers. Once one is found they will try to establish a connection.

The following chart demonstrates all the methods, colored rectangles, required during the entire process from the beginning of the advertising to the disconnection of the devices. All the arrows are callbacks that are used to handle different events during the procedure.



33. Illustration: Google nearby connection procedure (Android Developers, 2017)

5.6.2. Implementation

Because it is the spinal cord of our project, we spent extra time developing and testing the “NearbyManagement” class. In the following chapter, we will explain the overall operation of Google Nearby. However, not every aspect of the class will be detailed.

Both methods “StartDiscovery” and “StartAdvertising” are called as soon as the application has started. To let the user know when everything is running, we created an interface, “InearbyActivity”, that displays a message on the application.

Several callbacks are required to make the application work:

- “ConnectionLifecycle”: used throughout the procedure, from the initialization of the connection to the handling of peer disconnection. It will accept incoming connection and handle the result of it.
- “EndpointDiscovery”: exclusively used to handle the discovery of a new endpoint. As soon as an advertiser has been found, this callback will request a connection by sending the name of the device and an ID we created. This ID ensures that only applications with the same identifier can connect one to another, avoiding unwanted third-party peers.
- “Payload”: this callback is fired when a new payload is received. At this point, our “CommunicationManagement” class takes control of the handling of new data. We did not take advantage of the transfer update callback since only small text messages are exchanged.

5.7. Messages Object

Messages are the only piece of information shared between phones, which is why they must contain all the mandatory data. In the following part we will explain how the messages are constructed and what kinds of variables and field they contain. After that, we will describe what flow the message will take when being sent and received. Lastly, details will be given on how the propagation of messages is handled by the application and how we avoid duplicate data.

5.7.1. Message Class Structure

There are two distinct types of variables stored in a message: some are related to the message itself, others to the network information. All messages contain all those fields because of the decentralized nature of the network. Without the use of a central device handling the exchange and routing of information, messages need to keep track of where they have been or of their expiration time, for example.

A typical message is constituted of the following fields. On the left, there are the message related variables and on the right, the network related one.

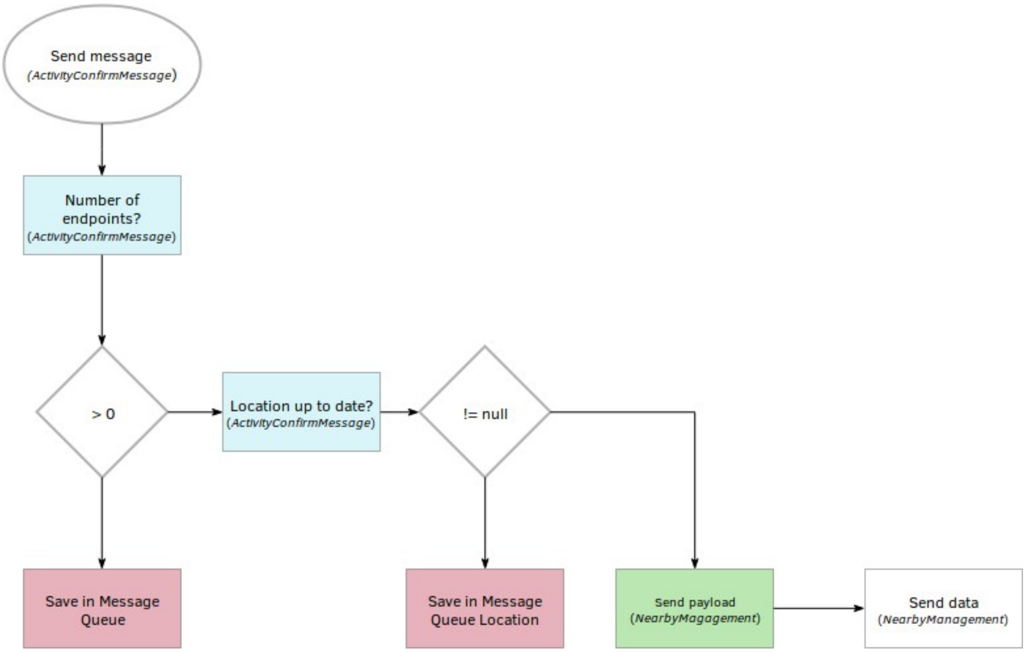
```
private String dateCreatedMillis;
private String dateCreatedString;
private String creatorAppId;
private String senderAppID;
private String creatorUserName;
private String title;
private String category;
private String description;
private Double messageLatitude;
private Double messageLongitude;
```

```
private String dateExpirationMillis;
private int progress;

private String messageStatus;

private ArrayList<String>
    mMessageSentTo;
```

5.7.2. Message Flow

There are three main paths a message can take: one when sent, another when received, and lastly one when propagated, this occurs when we send all the messages saved to a new peer. We will explain each of them with the help of a flow diagram. All those methods are in the “CommunicationManagement” class which acts like a router in a traditional network.

Sending



34. Illustration: Message flow—Sending

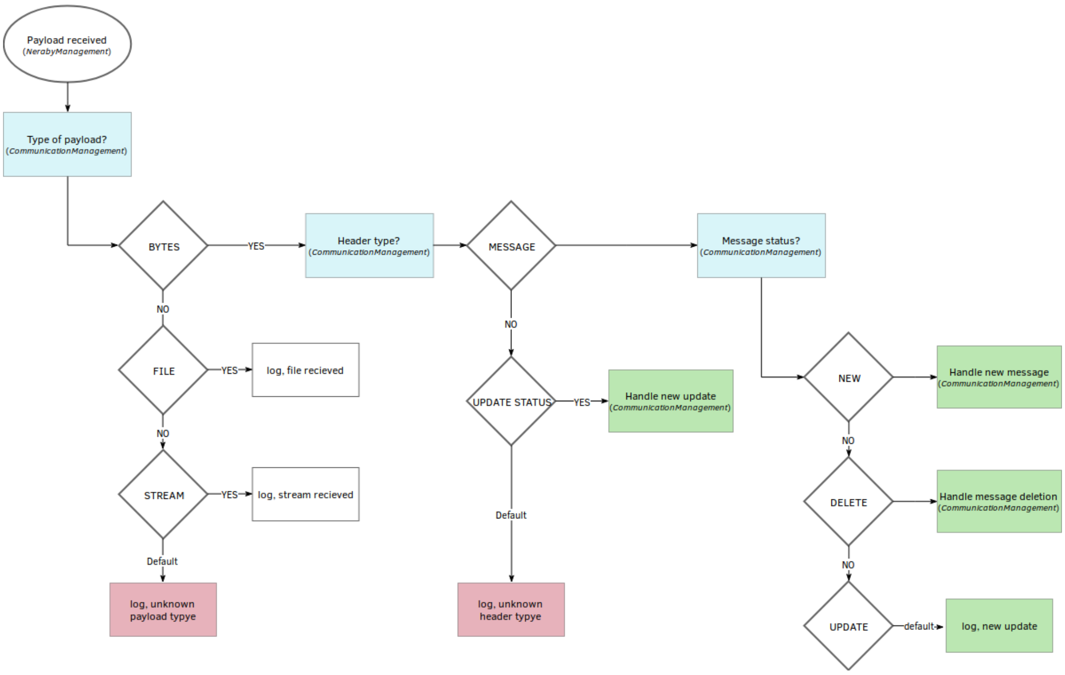
The process of sending a message is pretty straightforward. Once a user has clicked the send button present in the confirmation screen, the message will be sent only if the two following conditions are satisfied:

- There are connected devices, but if that is not the case, the message will be placed in a queue (“MESSAGE_QUEUE”) until a new device is connected.
- The device has an up-to-date location, but if that is not the case, the message will be placed in a queue (“MESSAGE_QUEUE_LOCATION”) until the GPS receives a new location and there are connected peers.

To each message that is about the be sent, there is a header attached to it. This information, a simple string, allows the distinction between all types of data we are sending. This was required because we often send payloads as bytes. To differentiate a new message to a message update, for example, this header was required. They are saved in the constant file to ensure a smooth implementation.

```
String[] content = new String[]{HEADER_MESSAGE, message.toString()};
NEARBY_MANAGEMENT.sendDataAsByteListRecipient(sendTo, gson.toJson(content));
```

Receiving

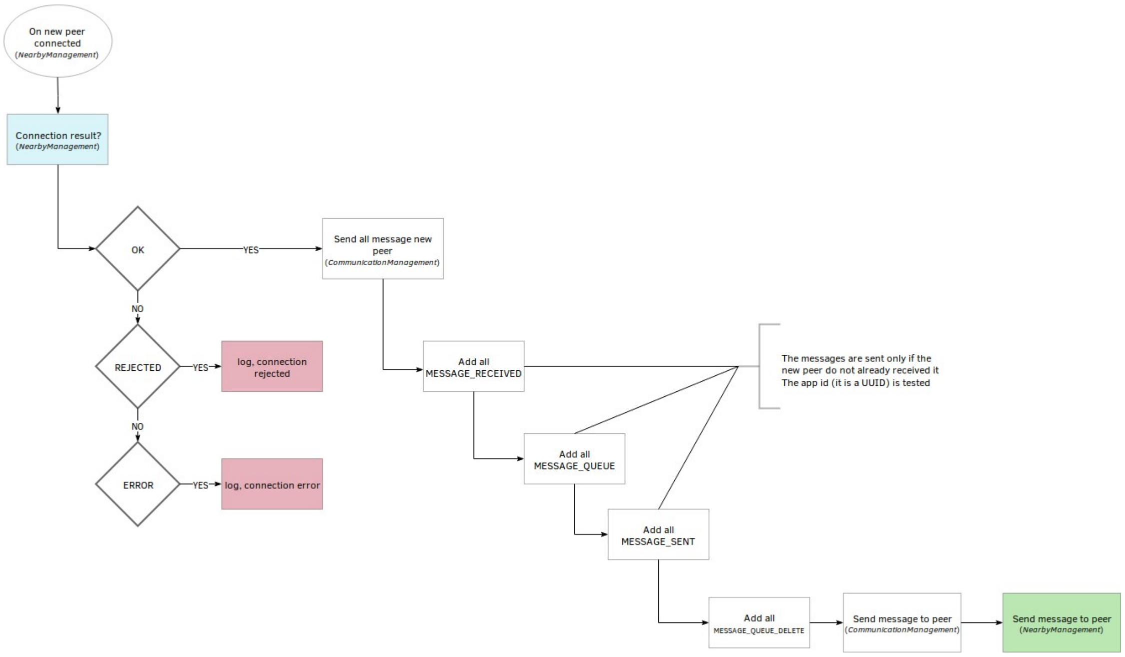

Payload received (NerabyManagement)
Type of payload? (CommunicationManagement)
BYTES
YES
Header type? (CommunicationManagement)
MESSAGE
Message status? (CommunicationManagement)
NO
FILE
YES
log, file recieved
NO
STREAM
YES
log, stream recieved
Default
log, unknown payload typye
NO
UPDATE STATUS
YES
Handle new update (CommunicationManagement)
Default
log, unknown header typye
NEW
Handle new message (CommunicationManagement)
NO
DELETE
Handle message deletion (CommunicationManagement)
NO
UPDATE
default
log, new update

35. Illustration: Message flow—Receiving

The receiving procedure seems more complicated since the application needs to control more conditions to process a message. We designed the reception of the payload this way to make the addition of a new type of data easier because everything is modular. If we receive a message, the following conditions will be controlled:

- The type of payload: test of the type of payload is received. It can be a BYTE, a FILE or a STREAM. In the case of a message, the type is a BYTE.
- The type of header: as explained above, we use headers to differentiate BYTE payloads. The header for a message is “message”.
- The type of status: once we have received a message, we will test what are the actions to do. According to its status, which can either be “new”, “delete”, or “upgrade”, it will be redirected to the corresponding method.

Propagation


On new peer connected (NearbyManagement)
Connection result? (NearbyManagement)
OK
YES
Send all message new peer (CommunicationManagement)
NO
REJECTED
YES
log, connection rejected
NO
ERROR
YES
log, connection error
Add all MESSAGE_RECEIVED
Add all MESSAGE_QUEUE
Add all MESSAGE_SENT
Add all MESSAGE_QUEUE_DELETE
Send message to peer (CommunicationManagement)
Send message to peer (NearbyManagement)
The messages are sent only if the new peer do not already received it
The app id (it is a UUID) is tested

36. Illustration: Message flow — Propagation

The messages need to be shared among all the devices nearby. If a device disconnects from the network and connects back later, all new messages sent, in the meantime, will be transmitted. To ensure that a device does not receive information twice, we store in every message the list of the devices which received it; the device ID is used as a unique identifier.

As soon as a peer connected, the device already connected will check if it already received the information. The message from the “MESSAGE_RECEIVED”, “MESSAGE_QUEUE”, and “MESSAGE_SENT” will be tested.

5.8. Endpoint Object

The endpoint is used to store information about the peers we connect to. It contains two fields, both save a different identifier.

- ID: this Google Nearby generates this ID at the connection, which is a four-letter string.
- Name: the application ID of the device; it generated during the first installation of the application. This identifier is a standard UUID.

We created this object because the “endpointID” generated by Google Nearby is not sufficient to identify a device. At each connection, a new ID will be created making the identification of a particular device impossible. We added therefore the UUID of the device since it does not change unless the user deletes all the data and caches of the application.

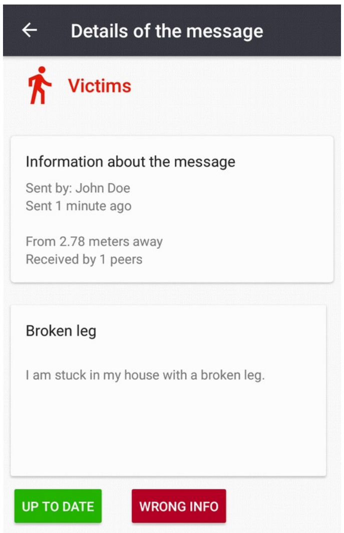
6. Details of the Implementation

6.1. Application Back End

6.1.1. Dynamic Expiration Date

User interface Addition

To ensure the information freshness, we decided to create a dynamic expiration date where each message has a time limit. Once this limit has been reached, the message will automatically be placed in a separated list where user can retrieve them. To avoid message deletion, nearby users should check it. Two buttons are present on the bottom of the message details view as shown below.



Details of the message

Victims

Information about the message

Sent by: John Doe
Sent 1 minute ago

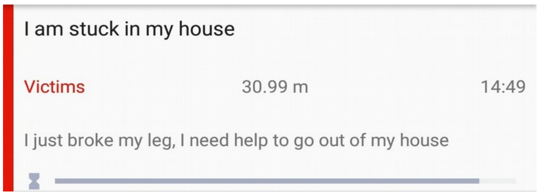
From 2.78 meters away
Received by 1 peers

Broken leg

I am stuck in my house with a broken leg.

37. Illustration: Dynamic expiration date implementation

A progress bar has also been added to the message list to see more clearly the time left for every message. This progress bar will reduce until the message has reached its time limit. The details of the display update are explained in the chapter 6.1.8 – Display Update (p. 73). The following screenshot shows how this progress bar is displayed to users.



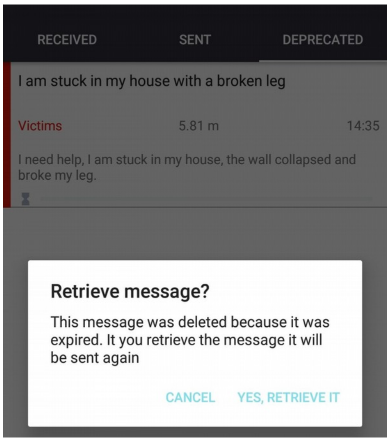
I am stuck in my house

Victims 30.99 m 14:49

I just broke my leg, I need help to go out of my house

38. Illustration: Progress bar in message list

Once the time limit has been reached, the message is placed in the deprecated list. All those messages are listed in the third tab present on the home screen. When clicking any of the messages present in this list the following dialog will ask the user if he wants to retrieve it.



I am stuck in my house with a broken leg

Victims 5.81 m 14:35

I need help, I am stuck in my house, the wall collapsed and broke my leg.

Retrieve message?

This message was deleted because it was expired. It you retrieve the message it will be sent again

39. Illustration: Restore deprecated messages

If the user wants to retrieve the message, the standard message creation screen is open with all the fields filled with the information of the message. The user can then edit some text and send it back to every connected peer.

Back End of the Dynamic Expiration Date

We created a variable in the constant file that represents the expiration time, which is expressed in milliseconds and is used when the message is created. At the creation, we get the current timestamp and we add the delay defined in the constants. Once it is done, we save both of these values.

```
long timeMillis = System.currentTimeMillis();
long dateExpiration = timeMillis + MESSAGE_EXPIRATION_DELAY;

dateCreatedMillis = String.valueOf(timeMillis);
dateExpirationMillis = String.valueOf(dateExpiration);
```

Every time the display updates, we recalculate the remaining time for the messages. This progress, expressed in percentage, is decreasing until zero is reached. After research, we established that the buttons have the following effects on the remaining time:

- When a message is confirmed ad up to date, the current expiration time is extended by the delay defined in the constant file.
- When the message is not up to date, the remaining time, this is the expiration time minus the current time, is divided by two.

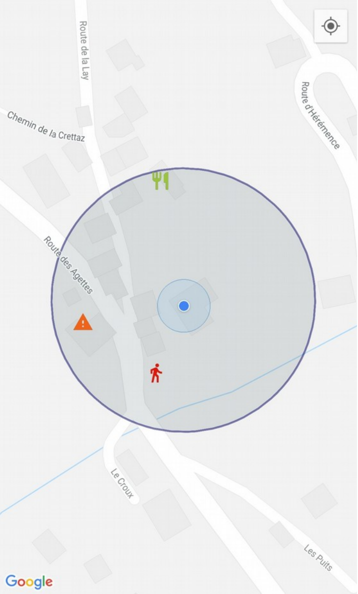
Those methods are implemented in the message class to make it readily available.

6.1.2. Floating Content

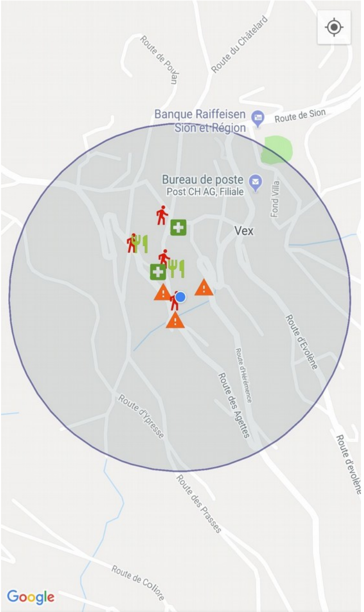
To avoid information overflow, we decided to implement the concept of floating content in the application. This means that users can only see messages sent by devices that are near them. This distance is configurable in the settings. Details about settings are in chapter 6.1.7 – Application Settings (p. 73).

User Interface Addition

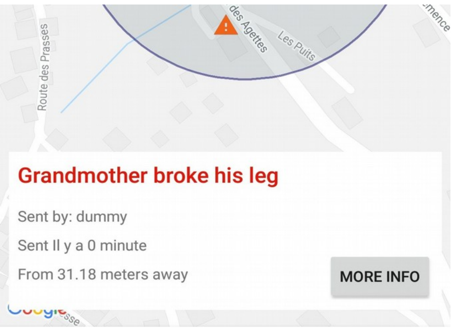
Not many additions were required to implement floating contents. We simply added a circle around the user's position. This provides a quick and easy overview of messages in the respective area of the user. The following pictures were taken in the application and at the same location yet with different radius settings.



40. Illustration: Floating content configured at 50 meters



41. Illustration: Illustration: Floating content configured at 400 meters



Grandmother broke his leg

Sent by: dummy

Sent Il y a 0 minute

From 31.18 meters away

42. Illustration: Detail of a message in the map

Each marker presented on the map has a custom-made icon; it represents the category of the message. On click, a panel containing the most essential information related to the message appears. The button "More Info" redirects the user to the detail page of the message.

Back End Modifications

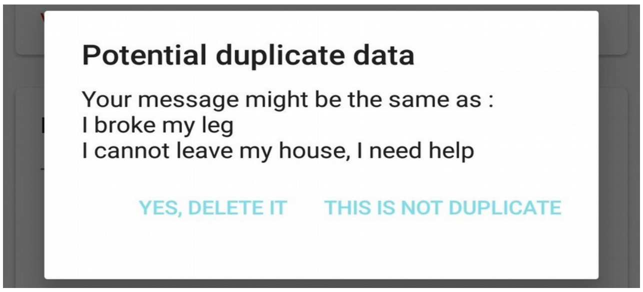
We created a class called "LocationManagement" that handles all the methods related to the floating content. Android integrates a method which calculates the distance between two points. Since we store the message and user location, it is easy for us to determine if a message is inside the area or not. When the display is updated, all the distances are recalculated.

Since the display updates involve several aspects of the application, they are detailed in the chapter 6.1.8 – Display Update (p. 73).

6.1.3. Avoid Duplicate Message Creation

We added a verification step when sending a message to avoid the creation of duplicate information. During this step, all the messages contained in the "MESSAGE_RECEIVE", "MESSAGE_SENT" and "MESSAGE_QUEUE" will be tested.

To be considered as duplicate, two conditions must be met. The message must be closer than a minimal distance configured in the constant file, by default 10 meters, and have the same category. If this is the case, a popup will alert the user that his or her message might already exist.



Potential duplicate data

Your message might be the same as :
I broke my leg
I cannot leave my house, I need help

43. Illustration: Potential duplicate message dialog

The user has the choice to ignore the dialog by clicking "This is not duplicate", which will send the message, or delete the message and go back to the list of messages.

6.1.4. Detect Duplicate Messages at Reception

The lack of a central unit that manages the network means that there is no way to know if a device has already received a message. We created therefore the "mMessageSentTo" ArrayList in the Message class. The list keeps track of all the application ID the message was sent to.

This is why at each message reception, we check if the “MESSAGES_RECEIVED” list already contains the message. It will also check that the message was not created by the device, because created messages are saved in the “MESSAGE_CREATED” list.

```
private static void handleNewMessages(Message m){

    boolean flagAlreadyReceived = true;


    if(!MESSAGES_RECEIVED.contains(m) && !
          m.getCreatorAppId().equals(VALUE_PREF_APPID)){

      flagAlreadyReceived = false;

        m.getMessageSentTo().add(VALUE_PREF_APPID);
    }

    if(!flagAlreadyReceived){
        FRAG_MESSAGE_LIST.updateDisplay(m);
    }
}
```

If the message has not been received, the application ID of the device is added to the message, and the display is updated.

6.1.5. Message Propagation

The transmission of messages to a freshly connected peer is mandatory. Without this system, only peers connected at the moment of the broadcast would receive the message and any peer connecting after that would not see them.

The procedure requires the creation of a list of messages that the new peer did not receive. The recipient list “mMessageSentTo” is checked for each message; if it does not contain the new device’s application ID, the message will be transmitted. Otherwise it will be ignored. All the messages of the “MESSAGE_RECEIVED”, “MESSAGE_QUEUE”, and “MESSAGE_SENT” are checked. Once the list is completed, we order the messages by distance so that the nearest messages will be sent first. This ensures that, even if the connection time is short, closest messages will be exchanged.

```
public static void sendAllMessagesNewPeer(Endpoint endpoint){

    ArrayList<Message> listMessage = new ArrayList<>();

    listMessage.addAll(checkRecipientMessages(MESSAGES_RECEIVED, endpoint));
    listMessage.addAll(checkRecipientMessages(MESSAGE_QUEUE, endpoint));
    listMessage.addAll(checkRecipientMessages(MESSAGE_SENT, endpoint));
    listMessage.addAll(MESSAGE_QUEUE_DELETED);

    MessageManagement.OrderByDistance(listMessage);

    for (Message m : listMessage){
        if(MESSAGE_QUEUE.contains(m)){

            MESSAGE_QUEUE.remove(m);
            FRAG_MESSAGES_SENT.updateDisplay(m);
        }
        if(MESSAGE_QUEUE_DELETED.contains(m)){

            MESSAGE_QUEUE_DELETED.remove(m);
        }
        sendMessageUniqueRecipient(endpoint.getId(), m);
    }
}
```

6.1.6. Shared Preferences Message Usage

As explained before, we decided to use shared preferences to permanently store messages. The procedure used to store the messages is the same as for any other application. Since we are storing ArrayList of messages, we used the GSON library, developed by Google, to translate the list into a format that the preference can write. The following snippet shows the operations required to save one message list.

```
public static void saveMessages(Activity activity){

   SharedPreferences prefsReceived =
       activity.getSharedPreferences(PREF_NAME_MESSAGE_RECEIVED,
               Context.MODE_PRIVATE);

    Gson gson = new Gson();

    String json = gson.toJson(MESSAGES_RECEIVED);
    prefsReceived.edit().putString(PREF_KEY_MESSAGE_RECEIVED, json).apply();
}
```

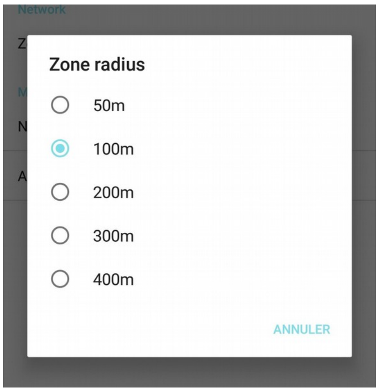
To retrieve messages from the preferences file, we created the following code. The GSON library object is used to translate the string saved back to an ArrayList of messages. After the translation, we control that the list of messages is not empty and we save the retrieved messages in the corresponding constant.

```
public static void retrieveMessages(Activity activity){
    SharedPreferences prefsReceived =
      activity.getSharedPreferences(PREF_NAME_MESSAGE_RECEIVED,
              Context.MODE_PRIVATE);

    Gson gson = new Gson();
    String json = prefsReceived.getString(PREF_KEY_MESSAGE_RECEIVED, "");
    Type typeReceived = new TypeToken<ArrayList<Message>>(){}.getType();

    ArrayList<Message> tempReceived = gson.fromJson(json, typeReceived);

    if(tempReceived != null && tempReceived.size() > 0) {
        MESSAGES_RECEIVED.addAll(tempReceived);
    }
}
```

6.1.7. Application Settings

Two variables are saved to the settings: the username and the floating content radius. We did not create settings that might change some functionalities of the application or reduce network performances.

The floating content radius has five possible distances, 50, 100, 200, 300 and 400 meters. We limited the distance to 400 meters as more length was not necessary in our opinion. Moreover, we did not want to let users have an unlimited distance too, as it makes floating content useless.



Zone radius

○ 50m

● 100m

○ 200m

○ 300m

○ 400m

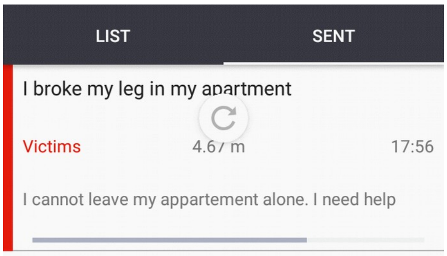
44. Illustration: Radius settings

6.1.8. Display Update

As explained before, display updates change several elements of the application. During those updates, the following components are modified:

- The distance between the device and all messages are recalculated.
- All displayed messages are deleted, the messages that are in the range of the zone radius and that did not expire are added back in the list.
- All messages with progress lower than zero are deleted placed in the deprecated list.

Updates can occur at two different times; when a new location is received or when the user pulls the message list up.


LIST SENT

I broke my leg in my apartment

Victims 4.67 m 17:56

I cannot leave my appartement alone. I need help

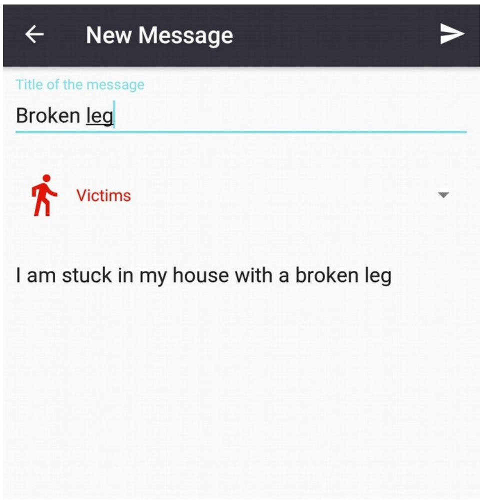
45. Illustration: Pull to refresh

6.2. Application User Experience

6.2.1. Error Handling

The confirmation screen was added to avoid some errors the user might make. The pictures below show how the message is handled.

First, the user enters the title, category and the description. After that, the user press the send button and the summary of what was entered is displayed. If an error is present, the user can come back and correct it.

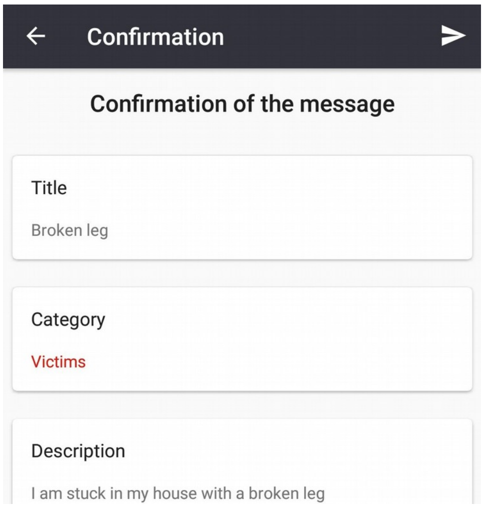

New Message

Title of the message: Broken leg

Victims

I am stuck in my house with a broken leg

46. Illustration: Message creation


Confirmation

Confirmation of the message

Title: Broken leg

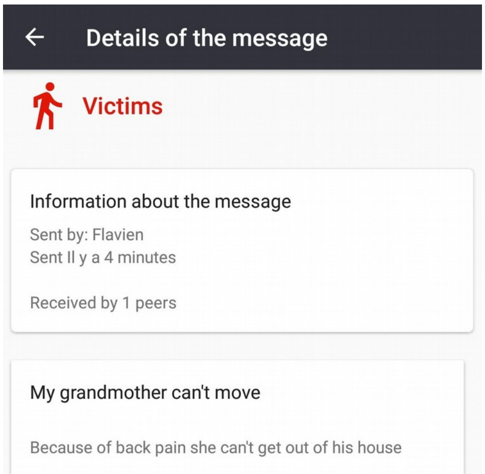
Category: Victims

Description: I am stuck in my house with a broken leg

47. Illustration: Message confirmation

6.2.2. Recipient Count

Reassuring victims is important; this can support them in stressful situations. To count the number of persons that received a message, we need to count the unique entries contained in the “mMessageSentTo” list. This information is displayed in the message details.



Details of the message

Victims

Information about the message

Sent by: Flavien
Sent Il y a 4 minutes

Received by 1 peers

My grandmother can't move

Because of back pain she can't get out of his house

48. Illustration: Recipient count in message details

The method used to count the number of recipients is in the “Message” class.

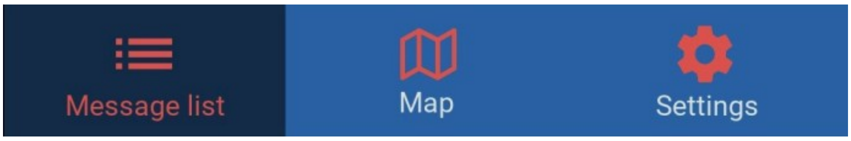
```
public int retrieveMessageRecipient(){

    ArrayList<String> uniqueValues = new ArrayList<>();

    for (String s : mMessageSentTo){
        if(!uniqueValues.contains(s) && !s.equals(creatorAppId)){
            uniqueValues.add(s);
        }
    }

    return uniqueValues.size();
}
```

6.3. Application User Interface

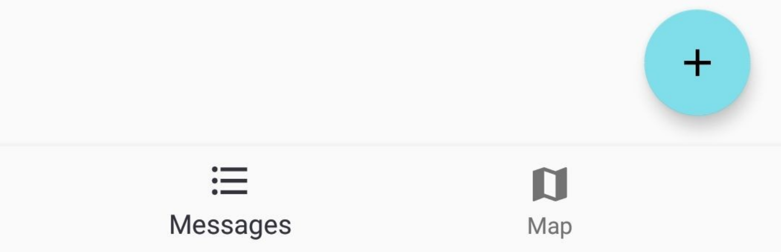
In this chapter, we will discuss the changes operated between the initial mock-ups and the final application implementation and the reasons for these changes. We will only cover the elements of the user interface that were modified or added.

6.3.1. Bottom Navigation

The bottom navigation slightly changed as the setting option was removed from it. We made this choice facilitate the use of the application. Indeed, having only the message list and the map makes navigation more natural. On top of that, default settings do not need to be changed. Therefore, we did not want to make them as accessible as there were in the initial mock-up.

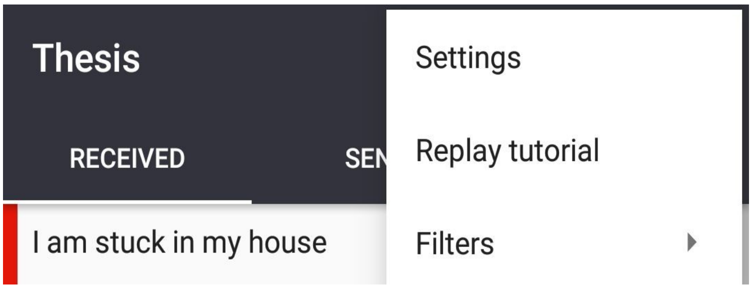


49. Illustration: Initial bottom navigation menu



50. Illustration: Final bottom navigation menu

To access the settings, users must click the top button present in the message list and map view.



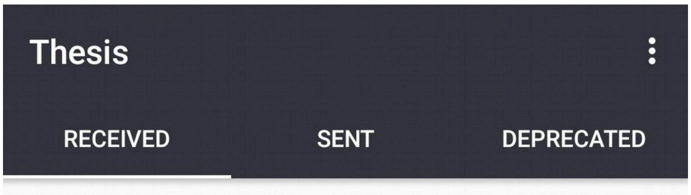
I am stuck in my house

51. Illustration: New settings menu location

6.3.2. Message List

The message list is split in three separated tabs all containing a specific type of message:

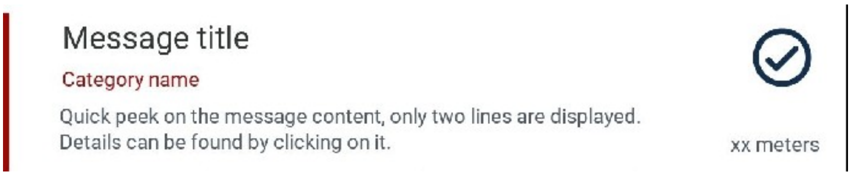
- Received, all the message the device received.
- Sent, all the messages sent from the device
- Deprecated, all the messages that reached the expiration date are placed there, users can then retrieve messages from this place and send them back.



52. Illustration: Tabs present in the home screen

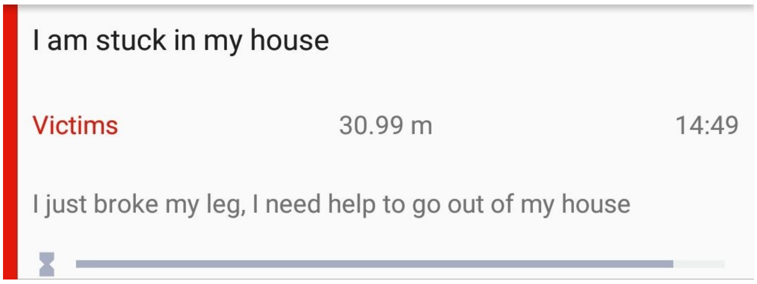
The message list had some change because of the way messages are handled. In the beginning, we imagined a scoring mechanic where users would vote a message up or down. Since the dynamic expiration date was chosen, the user interface needed modification to accommodate with this implementation.

Apart from some position change for elements present in the message list, the most significant change is the replacement of the status icon with a progress bar. The latter representing the time remaining before the message deletion.



Message title
Category name
Quick peek on the message content, only two lines are displayed.
Details can be found by clicking on it.
xx meters

53. Illustration: Initial message in the message list



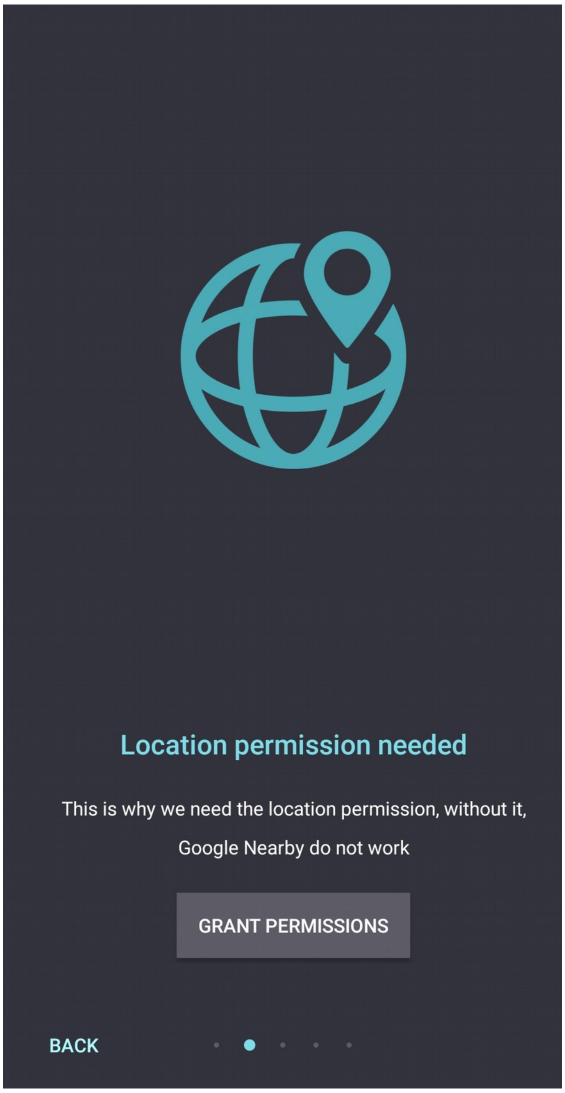
I am stuck in my house
Victims 30.99 m 14:49
I just broke my leg, I need help to go out of my house

54. Illustration: New message in the message list

6.3.3. Application Tutorial

A tutorial at the first application start was not planned at the beginning of the development and was only added to the product backlog later on. This is why we did not create mock ups of it.

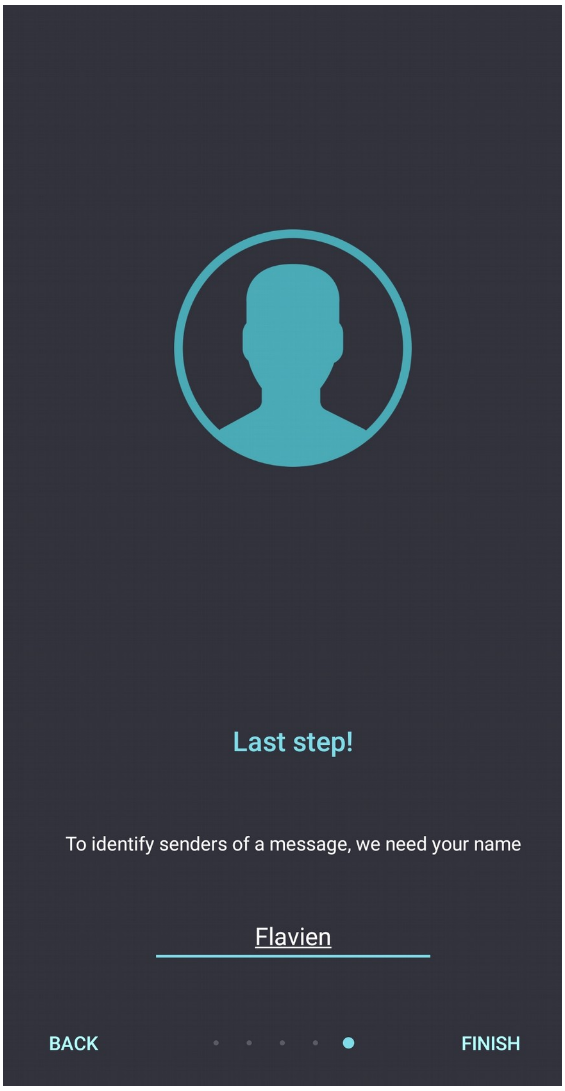
This tutorial provides some basic explanation on how the application works. In the meantime, we ask for the location permission required by Google nearby and explain why it is mandatory. We also prompt the user to choose a username.



Location permission needed

This is why we need the location permission, without it, Google Nearby do not work

55. Illustration: Requesting mandatory permission



Last step!

To identify senders of a message, we need your name

Flavien

56. Illustration: Asking for username

7. Discussion

During the development, we made some choices that were justified at the time, but that might have to be improved. In the following chapter, we will discuss those points and detail how they can be improved in our opinion.

We will also take some time to analyze the implementation of Google Nearby and how the process to integrate it into the application went.

Lastly, we will discuss the future improvement of the application. There are features we planned, but time was too short to implement them or new improvement that are not in the product backlog.

7.1. Application Back End

We spent more time on the application back end than on the front since we wanted to make our application as stable as possible. This time was required because we created a basic routing protocol and we wanted to cover most of the potential scenarios. Some of the planned features were not developed because of time restriction; we will discuss them in the chapter 7.4 – Future Improvements (p. 85).

The reflection made before the implementation of the application highlights some of the significant aspects to handle when implementing ad hoc networks. Those characteristics are fundamental and they need to be managed in the most feasible way since the whole application relies on those features.

We believe that the back end we developed is sufficient enough for our application since we covered most of the scenarios where problems could arise.

In the following part, we will analyze the crucial parts of the application back end, and we will evaluate their success or limitation. We will not discuss Google Nearby in this chapter since we have a section dedicated to this technology below

7.1.1. Dynamic Expiration Date

Avoiding outdated and possibly wrong information is mandatory. We therefore developed a simple system of peer validation: when a user confirms a message as valid, we add the delay set in the constant file to the expiration time. On the contrary, if a user judges a message as invalid, we divide the remaining time by two. This is a simple implementation that can be vastly improved.

Possible Improvements

There are several aspects of the dynamic expiration date that can be improved. Firstly, we do not think that the method of the dynamic expiration date explained above is sufficient. A better approach would be to count both valid and invalid votes, log the time of the feedback and establish a strategy of delay increase or reduction.

Another problem with our implementation is voting limitations. No system prevents a user to validate a message more than once. Since we rely on people's honesty, we did not think that was an issue during the implementation. However, on second thought, we concluded that this is an important feature.

Lastly, the delay we used during our testing was only two minutes; we were not able to find what the correct timing would be in a real-world application. We left this delay for the deployment of the application in the Google Play Store.

7.1.2. Duplicate Data and Propagation Handling

Duplication of data must be avoided at all costs and it can occur on two occasions:

- If a user writes an already existing message
- When a new peer connects and receives the messages exchanged in an area

Both of those cases have been handled by the application and it does now provide sufficient protection in our opinion.

To avoid duplicate data at creation, we control that no message with the same category has been created near the user. The distance is set in the constant file and cannot be changed in the application. By default, this distance is 10 meters. This is a straightforward implementation and it can be improved, more information in the following chapter.

The whole procedure involved when new peers connect is robust and avoids unnecessary transmission.

Possible Improvements

As said before, our duplicate message detection system only relies on distance and category. We do not think that this is sufficient in an area with more activity. To further improve the detection, title and description should be checked to control if the message already exists.

7.1.3. Isolated Device

The application must handle cases when a device is not connected to any peer. If that is not the case, some users might have a frustrating experience, which is something we want to avoid. Hence, we created several lists of messages regarding this issue. Those lists are the following:

- Message queue, messages waiting until a peer connects to the network to be received.
- Message queue delete: saves all the messages a user wanted to delete, even though no other device was connected.

Both lists are used as soon as a peer connects. Indeed, the messages contained are transmitted in the background without any user interaction. These lists are also saved when the application is closed.

Another message waiting list has been created for scenarios where the application does not know the device’s location. This list stores messages until the GPS update the device’s position. Once it is done, all the messages are transmitted with the new location.

Possible Improvements

Not all types of communication are saved in a queue. This is the case, for example, for dynamic updates. In scenarios where a device is isolated, this is something that should be improved to ensure that each communication is saved.

7.1.4. Application Distribution and Offline Maps

As discussed during our reflection, we need to handle offline application distribution and maps download when the Internet is available. However, because of the time limitation, we were not able to implement any of those features.

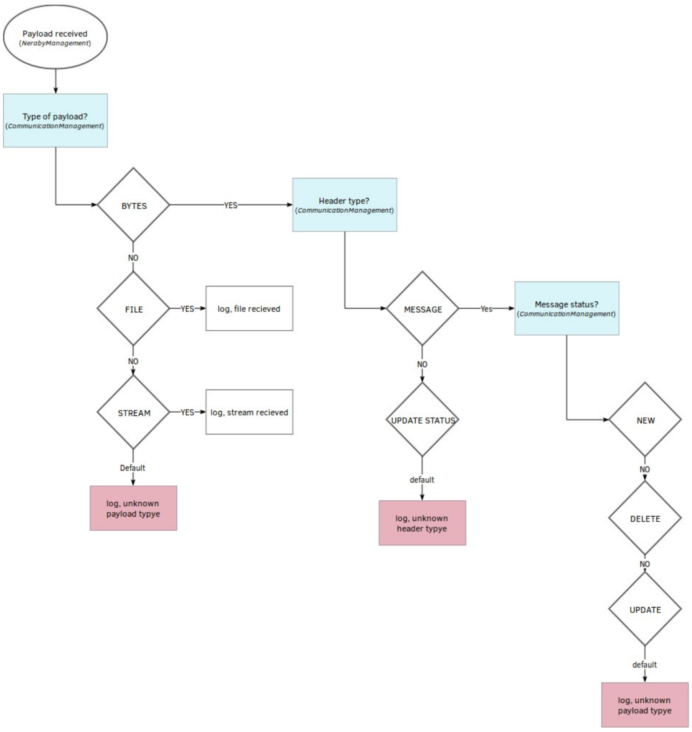
We did some research on the best practice to download an offline map and even developed a small code snippet that converts a GPS location to corresponding tiles used by Google Map. This code can be found in the “OfflineMapManager” class. However, the code only generates the correct URL.

Regarding the offline application, we did not have enough the time to research the solution for this problem. We decided that offline map downloading was a priority.

7.1.5. Communication Handling

The system of tests we developed when receiving data is excellent. It provides solid routing mechanics and allows the addition of a new type of data in a matter of minutes.

We wanted to allow future improvement from the beginning of the development. Therefore, the "CommunicationManagement" class has been developed this way. By using conditional routing, it is possible to redirect incoming payloads to the corresponding methods. This is already explained in the chapter 5.7.2 – Message Flow (p. 63), but the following illustration shows the tests realized for every incoming payload.



57. Illustration: Payload flow

We can see that adding anything new is as simple as adding a condition to any of those tests.

Possible Improvements

We do not think that any improvement can be made concerning this part of the development. The solution implemented is flexible and can support new data. The way how incoming data is managed, however, depends on the future development. We only provided a solid conditional routing mechanic.

Fortunately, the protocol we created can be improved. What we developed is minimal and might limit the application in an area with more activity. Indeed, we send every message that the new peer did not receive regardless of their age or category. We only order the message send by the distance to ensure that the nearest messages are sent first. This might not be sufficient in some scenarios and ordering the messages by date and distance should be a better solution.

On top of that, we send to the peer every message it did not receive. This can represent many messages in a scenario where the area covered by the application is several square kilometers large. A maximal distance should also be set during the propagation process.

7.2. Application Front End

7.2.1. Confirmation Screen

Errors must be avoided at all costs since they provoke time and energy loss. During our reflection, we decided to implement a confirmation screen before the transmission of a message. This view summarizes the content of the message allowing users to correct inevitable mistakes.

Possible Improvement

We think that our implementation is a step in the right direction but it might not be enough in real-world use. The confirmation screen must be present, but a way to update a message would be a nice addition. Because there is currently no way to provide adjustment to a piece of information, this feature would be useful to change the number of remaining resources or the status of a victim, for example.

7.2.2. User Interface

We wanted to develop an application with a layout as simple as possible to avoid confusion and make the experience as smooth as possible. We decided to move the settings menu from the bottom navigation menu to the top action bar because it was not a core functionality of the application. Our goal was to focus on the message exchange system by making less visible everything else.

Possible Improvement

There are, however, some potential improvements in the user interface. We did not conduct a user survey where they give feedback about on the application. Without that info, it is hard to improve the application.

7.3. Google Nearby

7.3.1. Ease of Implementation

The implementation of Google Nearby was challenging: the fundamental concepts were easy to understand, but the details of the connection process were more complex. We had to take extra time at the beginning of the implementation to master the API.

The documentation provided by Google is barely sufficient to truly understand the connection process and the online examples are not up to date. Since we used the version 2.0 of the API, we were not able to find any up-to-date examples, most of them use the previous version of the Connection API which does not support offline peer-to-peer connections or is only experimentation. Fortunately, Google created a GitHub containing an up-to-date example of the Google Nearby Connection API.

Once the basic connection process was implemented, everything else was easy to add and manage, only the beginning took extra time.

7.3.2. Service Quality

As said before, we had some problems implement basic elements. However, the quality of the service provided by Google Nearby surprised us. The connection process and transfers are fast on modern devices, and the stability is on point.

7.3.3. Test of Multiple Devices

Since we had access to a limited number of devices, four Motorola G second generation and one OnePlus 5t, we are not able to ensure the quality of service across all Android devices. However, Google Nearby is part of the Play Services, and we assume that the compatibility is efficient with most of the high end to mid-range phones.

7.3.4. Motorola Bug

The application had strange bugs with the Motorola devices we used. Once in a while, the devices asked to pair one to another like it is the case with classic Bluetooth. This bug is known and will be corrected in the future update of Google Nearby (StackOverFlow, 2018).

7.3.5. Possible Improvement

Google Nearby works great in our application, exchanges are swift and the whole connection handling is stable. However, we did not test the battery impact of the API, which would be interesting. If the application is energy-intensive, some changes are required. Indeed, as mentioned by the response of “Xylthe” (StackOverflow, 2018) the discovery process requires more energy, it suggests limiting the discovery to small burst.

On top of that, we have no logic of connection. Some intelligent connection management can be a better implementation. For example, the creation of a small cluster of devices where only one of the smartphones of the cluster will communicate to another cluster can vastly improve the performances.

7.4. Future Improvements

7.4.1. Download Map Offline

One of the most important features to add is the possibility to download a map offline. Without this element, an essential functionality of the application would be missing. As explained in the chapter 4.1.8 Offline Challenges (p. 47) we developed one small code that can retrieve the URL required to download the image.

On top of that, sharing maps feature would be great. Users could send the map to other peers around.

7.4.2. Offline Application Share

We did not have enough time to investigate a proper way to share the application installer to other devices nearby. This is a crucial feature to add since not everybody will have the application installed before the disaster.

7.4.3. Instant Messaging Support

One feature we placed in the product backlog was the ability to send instant messages to nearby devices. We knew that we would not have the time to implement it, but it was essential for us to add it to the backlog.

The use of instant messaging can help reassure the victims since they will be able to discuss directly with someone else. This can also improve the organization of resources, for example. We think that one public chatroom can be enough for a small disaster. However, if many peers are connected in the same area, the creation of groups might be relevant.

7.4.4. Test With Multiple Devices

Since we tested the application with a limited number of devices and peers, it is mandatory to conduct the testing of the application further. A test with more users can also be a great indicator of the potential flaws of the communication protocol we developed.

7.4.5. Create Notification

We did not implement any notification for the application; users are not able to know if a new message is received unless they open the application. Two types of notifications can be implemented in the application:

- New message notification, each new message will create a notification informing users of new activity on the network.
- User participation notification, when a user is near a message, a notification could prompt him to validate the state of the message. This can help improve the overall freshness of the networks.

8. Product Backlog

To help with the development of the application, we established a product backlog containing all the features that might be implemented. In the end, we created 32 different functionalities, 17 must, nine should, five could and one would.

All those features add up to a total of 127 story points, not all those components have been developed. The total story points made is 79, the total of story point rejected is 2. The following table summarizes what is the total of features and story points implemented for each feature priority.

Feature priority	Total features	Done	Rejected	SP total	SP Made
Must	17	16	1	58	56
Should	9	5	0	34	12
Could	5	3	0	27	11
Would	1	0	0	8	0

9. Table: Product backlog implemented features summary

8.1. Implementation Feedback

We are satisfied with the current state of the application; it provides a solid and robust experience for the users. We knew since the beginning that not every user story would be implemented due to time restriction. Instead of excluding some user stories in the backlog, we decided to add all the features we think of to show how we imagined the perfect implementation.

Despite the lack of some features, all the user stories prioritized as “must” have been developed. One was, however, rejected, but this is because of an element’s design. On top of that, most of the “should” and “could” functionalities have been developed as well.

9. Management of the Project

This part will be consecrated to the administrative part of the thesis. We will explain our planning throughout the project and provide an analysis of potential time differences between the planned and the executed time. At the end of this chapter, a list of the tools used will be dressed.

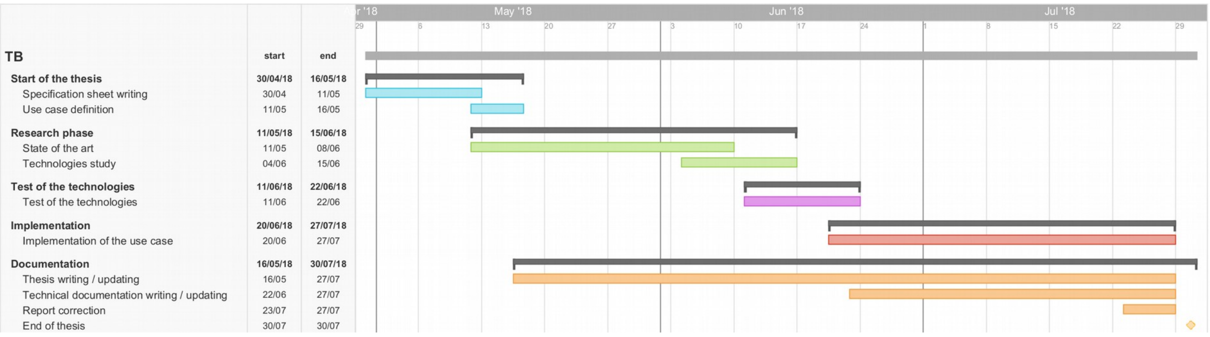
9.1. Planning of the Project

We established the following tasks during the creation of the initial planning:

- Specification sheet writing
- Use case definition
- State of the art
- Technology study
- Test of the technologies
- Implementation of the use case
- Thesis writing/updating
- Technical documentation writing/updating
- Report correction

9.1.1. Initial Planning

The initial planning was dressed during the first weeks of the thesis. We did not know what the use case would be or the different technologies existing.

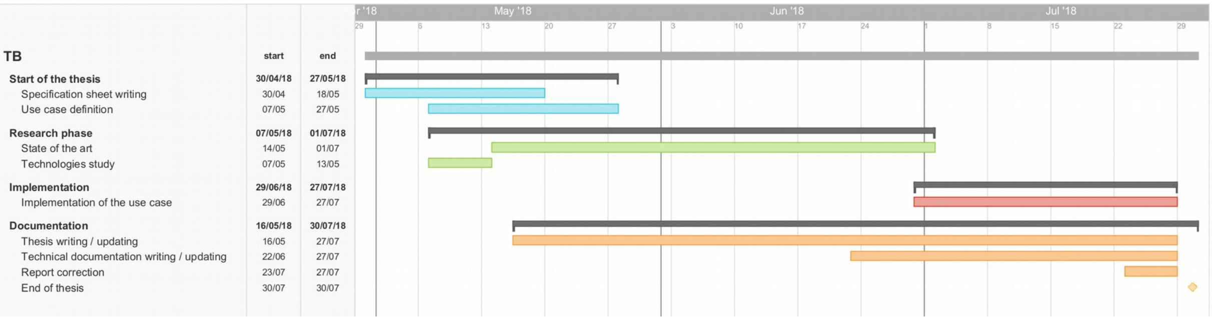


TB	start	end
Start of the thesis	**30/04/18**	**16/05/18**
Specification sheet writing	30/04	11/05
Use case definition	11/05	16/05
Research phase	**11/05/18**	**15/06/18**
State of the art	11/05	08/06
Technologies study	04/06	15/06
Test of the technologies	**11/06/18**	**22/06/18**
Test of the technologies	11/06	22/06
Implementation	**20/06/18**	**27/07/18**
Implementation of the use case	20/06	27/07
Documentation	**16/05/18**	**30/07/18**
Thesis writing / updating	16/05	27/07
Technical documentation writing / updating	22/06	27/07
Report correction	23/07	27/07
End of thesis	30/07	30/07

58. Illustration: Initial planning

9.1.2. Intermediate Planning

To adjust the planning, we decided to create an intermediate planning in the middle of the thesis. We will detail those changes in the next part of this chapter.



TB	start	end
Start of the thesis	**30/04/18**	**27/05/18**
Specification sheet writing	30/04	18/05
Use case definition	07/05	27/05
Research phase	**07/05/18**	**01/07/18**
State of the art	14/05	01/07
Technologies study	07/05	13/05
Implementation	**29/06/18**	**27/07/18**
Implementation of the use case	29/06	27/07
Documentation	**16/05/18**	**30/07/18**
Thesis writing / updating	16/05	27/07
Technical documentation writing / updating	22/06	27/07
Report correction	23/07	27/07
End of thesis	30/07	30/07

59. Illustration: Intermediate planning

9.1.3. Planning Change

Task name	**Initial planning (hours)**	**Intermediate planning (hours)**	**Delta (hours)**
Specification sheet	20	17	-3
Use case definition	10	5	-5
Technology study	35	8	-27
State of the art	60	100	+40
Test of the technologies	35	0	-35
Implementation of the use case	120	120	0
Thesis writing/updating	40	60	+20
Technical doc. writing/updating	20	30	+10
Report correction	20	20	0
Total	**360**	**360**	**0**

10. Table: Tasks planning

Some of the tasks were already done during the realization of the intermediate planning, which is the case for “Specification sheet”, “Use case definition” and “Technology study”. We simply entered the executed hours of those tasks.

There is a major time difference between the initial and the intermediate planning. The following tasks have the biggest delta.

Technology Study

We overestimated the time required to study technologies during the initial planning. We did not know at that time that only a few of technologies offered ad hoc functionalities. Indeed, we needed only 8 hours to study the different technologies.

State of the Art

The time we planned was utterly underestimated. We thought that 60 hours were sufficient, but we were wrong. We estimated that 40 more hours were required to complete the state of the art. The following factors can explain this massive time difference:

- We did not know how broad ad hoc networks were. This resulted in many times spent studying the different fields and possible applications of those networks although they are not related to the implemented application.
- On top of that, there are many theoretical papers exist, they are excellent but aimed at a formed audience. Moreover, there is not as much documentation for the beginners. This lack of easy-to-assimilate documentation resulted in time loss at the beginning of the work.
- Another reason is the time we took to define a use case. Because we did not know what the topic of the application was, we researched solutions in fields that were not related to the final implementation.
- We were in the army for two weeks at the beginning of the thesis. This disconnected us from the work although we had spent some time during these two weeks to work on the state of the art. This disconnection resulted in a loss of hours, required to resume the work where it was left.

Test of the Technologies

When realizing the intermediate planning, we already knew what technology we were going to use; making the test of other technologies irrelevant. We decided therefore to delete this task.

Thesis Writing/Updating

We decided to increase the time required to write the thesis because of the time needed to complete the state of the art. We did not want to run out of time at the end of the work; this is why we increased it by 20 hours.

9.1.4. Executed Hours

Task name	Intermediate planned hours	Executed hours	Delta hours
Specification sheet	17	17	0
Use case definition	5	5	0
Technology study	8	8	0
State of the art	100	111	+11
Test of the technologies	0	0	0
Implementation of the use case	120	122	+2
Thesis writing/updating	60	56	-4
Technical doc. writing/updating	30	8	-22
Report correction	20	22	+2
Total	**360**	**349**	**-11**

The actual time spent on each task is very close to the planned hours from the intermediate planning. We are quite satisfied with the correctness of the intermediate planning. However, some points need to be mentioned.

- The state of the art took longer than planned. Initially, we planned 60 hours and nearly the double was required, which is a vast difference. We completely underestimated the time required for this task.
- The technical report did not take as much time as planned. We saved a few hours because some of the text that is found in the technical documentation is already in this document.

In the end, we spent 11 hours less than planned. This is the cause of several factors. Indeed, two weeks of unexpected military service is the principal cause. Since we were not able to work as much as planned during those two weeks, we could only compensate during the weekend. We worked only 11 hours instead of the 50 planned during this time.

Since we did not have to test different technologies, we saw ourselves saving a certain number of hours. We reserved, in the initial planning, 35 hours for those tests.

9.2. Tools Used

9.2.1. Technical Tools

Development tools used are the standard in the industry, the code was written on Android Studio and the integrated Android emulator was used when needed. All the code was hosted on GitHub on two separate branches. The master was used to push improvement once they were sufficient enough and all the intermediate changes were pushed on a dev branch. This is not necessarily justified as only one person was developing it, but this is the way the developer usually works.

Marvel app was used to create the different mock-ups presented before the implementation of the application. The developer chose this website as an Android application allows the interaction with the created mock-ups and, for that reason, it makes it easier to evaluate them.

All the illustrations found in these documents were made on a website called draw.io. The service is totally free and the result can be saved in either PNG or XML files. The XML files contain all the information related to the illustrations, meaning that they can be re-uploaded on the website.

9.2.2. Administrative Tools

All the documentation, the backlog and the logbook were written on the LibreOffice suite. LibreOffice Writer was used as a word processor and LibreOffice Calc for spreadsheets. These programs were used because we worded on Linux and they are installed by default on our distribution. We will provide PDF files for every document we created, however, if modifications are made, we recommend using those program to avoid layout problems.

The planning was created on Teamgantt, the website was used only to create them, we did not use other features provided by the website. Our goal was to make a visually appealing planning, not to track precisely the tasks we had to do and their advancement.

Finally, we used a task manager called TickTick that helped us to track remaining tasks.

CONCLUSION

The purpose of the developed application in this work was to provide a fast, effective and reliable mean of communication to people affected by a disaster. However, this application should not rely on existing network infrastructure such as the cellular network or the Internet, which is often destroyed during these events.

Following an initial phase of analysis of the various types of ad hoc networks, their implications and limitations, we focused our research on technologies embedded on smartphones. This allowed us to discover Google Nearby Connection API which is the technology we decided to use.

This choice, although difficult to implement in the first time, proved its worth throughout the implementation. As a result of our development, we can confidently state that Google Nearby can be used in a crisis such as an earthquake. The technology has proven its robustness and reliability during our tests and in the implementation.

However, the work provided cannot be considered ready for production. Many points remain to be clarified to make this application ready for the public. This is particularly true for tests that can be carried out on a larger scale and with a greater diversity of equipment. This will allow to know the possible limits of Google Nearby which were not discovered during this development.

Besides, some features essential to the proper functioning of the application have not been implemented due to lack of time. This is notably the case of downloading offline maps and sharing the application with close users who would not have downloaded the tool before the disaster.

Despite these improvements, we have demonstrated that ad hoc networks can be used during disasters to provide the necessary support to victims left unprepared for the arrival of relief.

PERSONAL EXPERIENCE FEEDBACK

I enjoyed working on this project which allowed me to go deeper into a subject as I had never done before. The research phase, although going beyond the planned time, pleased me because I was able to deepen my knowledge of specific subjects that I did not know at all. I was able to choose Google Nearby with confidence because I could prove that the other technologies were not up to the task.

I also really appreciate the Android development I learned during my studies at the HES. Being able to make an Android application for a bachelor's degree made me realize that I wanted to work in this sector in the future.

Moreover, I have been using Linux for about a year. I challenged myself at the beginning of the project to only use this system exclusively during all the work, and it was successful. In addition to everything I learned about ad hoc networks and Android development this work allowed me to get acquainted with that I did not know such as LibreOffice or Gimp.

It is, therefore, an incredibly enriching experience that I lived through this bachelor and I am delighted with the work done.

REFERENCES

Amazon (n.d.). goTenna Mesh SMS & GPS Smartphone Booster Device (Blue/Green) goTenna Mesh SMS & GPS Smartphone Booster Device (Blue/Green). Retrieved June 13, 2018, from https://www.amazon.com/goTenna-Phone-Through-Others-Extend/product-reviews/B0721B8M3J/ref=cm_cr_arp_d_hist_3?ie=UTF8&filterByStar=three_star&reviewerType=all_reviews&pageNumber=1#reviews-filter-bar

Android Developers. (2017, May 19). How to Enable Contextual App Experiences (Google I/O '17) [Video file]. Retrieved July 28, 2018, from https://www.youtube.com/watch?v=1a0wlI96cpE

Android Developers. (2018). android.net.wifi.p2p. Retrieved June 23, 2018, from https://developer.android.com/reference/android/net/wifi/p2p/package-summary

Anwer Shahid, A. S., & Guy Chris, G. C. (2014). *A Survey of VANET Technologie* (ISSN 2079 - 8407). Retrieved from http://www.cisjournal.org/journalofcomputing/archive/vol5no9/vol5no9_1.pdf

Apple Developers. (2018). MultipeerConnectivity. Retrieved June 26, 2018, from https://developer.apple.com/documentation/multipeerconnectivity

Arthur Charles, A. C. (2013, June 4). Turkish protesters using encryption software to evade censors. *The Guardian*. Retrieved from https://www.theguardian.com/technology/2013/jun/04/turkish-protestors-encryption-software-evade-censors

Barbaresso Jim, B. J., Cordahi Gustave, C. G., Garcia Dominie, G. D., Hill Christopher, H. C., Wrigh Karissa, W. K., & Jendzejec Alex, J. A. (2014). *ITS 2015 -2019 STRATEGIC PLAN* (FHWA-JPO-14-145). Retrieved from https://www.its.dot.gov/strategicplan.pdf

Bluair. (n.d.). Bluetooth Range. Retrieved May 29, 2018, from http://www.bluair.pl/bluetooth-range

Bluetooth SIG. (2016). Bluetooth 5 Go Faster. Go Further.. Retrieved May 29, 2018, from https://www.bluetooth.com/bluetooth-technology/bluetooth5/bluetooth5-paper

Cambridge Dictonnary. (n.d.). ad hoc. Retrieved June 3, 2018, from https://dictionary.cambridge.org/dictionary/english/ad-hoc

Choosing a transport protocol. (2008). Retrieved May 28, 2018, from https://people.csail.mit.edu/albert/bluez-intro/x95.html#rfcomm-and-tcp

Cisco. (2017). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper* (1454457600805266). Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

Cohen Noam, C. N. (2014, October 5). Hong Kong Protests Propel FireChat Phone-to-Phone App. *The New York Times*. Retrieved from https://www.nytimes.com/2014/10/06/technology/hong-kong-protests-propel-a-phone-to-phone-app-.html

Collotta Mario, C. M., Pau Giovanni, P. G., Talty Timothy, T. T., & Tonguz Ozan K., T. O. K. (n.d.). *Bluetooth 5: a concrete step forward towards the IoT*. Retrieved from https://arxiv.org/pdf/1711.00257.pdf

Deepshikha Bhatia, D. B. (2016). *A Comparative Analysis of Proactive, Reactive and Hybrid Routing Protocols over open Source Network Simulator in Mobile Ad Hoc Network*. Retrieved from https://www.ripublication.com/ijaer16/ijaerv11n6_22.pdf

European Commission. (n.d.). Intelligent transport systems. Retrieved June 7, 2018, from https://ec.europa.eu/transport/themes/its_en

Google Developers. (2018). Nearby. Retrieved June 23, 2018, from https://developers.google.com/nearby/

Google Developers. (2018). Nearby Connection API Overview. Retrieved June 23, 2018, from https://developers.google.com/nearby/connections/overview

Google Developers. (2018). Nearby Message API Overview. Retrieved June 23, 2018, from https://developers.google.com/nearby/messages/overview

Google Developers. (2018). Nearby Message API Publish and Subscribe. Retrieved June 23, 2018, from https://developers.google.com/nearby/messages/android/pub-sub

Google Developers. (2018). Nearby Connection API Strategies. Retrieved June 23, 2018, from https://developers.google.com/nearby/connections/strategies

GoTenna. (n.d.). goTenna. Retrieved June 8, 2018, from https://www.gotenna.com/

Hern Alex, H. A. (2014, June 24). Firechat updates as 40,000 Iraqis download 'mesh' chat app in censored Baghdad. *The Guardian*. Retrieved from https://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad

Holley Peter, H. P. (2017, November 7). Hurricane Irma just made a digital walkie-talkie the No. 1 app online. *The Washington Post*. Retrieved from https://www.washingtonpost.com/news/innovations/wp/2017/09/06/hurricane-irma-just-made-a-digital-walkie-talkie-the-no-1-app-online/?noredirect=on&utm_term=.3e393acaa59c

Milian Mark, M. M. (2014, December 30). Russians Are Organizing Against Putin Using FireChat Messaging App. *Bloomberg*. Retrieved from https://www.bloomberg.com/news/2014-12-30/russians-are-organizing-against-putin-using-firechat-messaging-app.html

Moltchanov Dimitri, M. D. (n.d.). Routing protocols for ad hoc networks. Retrieved July 28, 2018, from http://www.cs.tut.fi/courses/TLT-2756/lect05.pdf

Nordic Semiconductor. (2016, December 22). Nordic Semiconductor – Bluetooth 5 Long Range Test with nRF52840 [Video file]. Retrieved May 22, 2018, from https://www.youtube.com/watch?v=w4PbxVwg83M

Nordic Semiconductor. (2018). *nRF52840 Product Specification*. Retrieved from http://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.0.pdf

Nordrum Amy, N. A., Clark Kristen, C. K., & IEEE Spectrum Staff. (2017, January 27). Everything You Need to Know About 5G. Retrieved June 27, 2018, from https://spectrum.ieee.org/video/telecom/wireless/everything-you-need-to-know-about-5g

Open Garden. (2016, June). Metro Manila Development Authority - Shake Drill, Philippines. Retrieved June 26, 2018, from https://www.opengarden.com/case-studies.html?lang=en

Open Garden. (2018). FireChat. Retrieved June 16, 2018, from https://www.opengarden.com/firechat.html

Open Garden. (2018). MeshKit SDK. Retrieved June 25, 2018, from https://www.opengarden.com/meshkit.html

Poole Ian, P. I. (n.d.). Bluetooth Host - L2CAP, SDP & GAP. Retrieved May 29, 2018, from https://www.radio-electronics.com/info/wireless/bluetooth/l2cap-sdp-gap-host.php

Qorvo. (2017, November 7). Getting to 5G: Comparing 4G and 5G System Requirements. Retrieved June 27, 2018, from https://www.qorvo.com/design-hub/blog/getting-to-5g-comparing-4g-and-5g-system-requirements

Ramblings Eric, R. E. (2017, December 7). GoTenna Mesh Hands-On review [Blog post]. Retrieved June 12, 2018, from http://www.jenericramblings.com/2017/12/07/gotenna-mesh-hands-on-review/

Rhodes Margaret, R. M. (2014, July 17). A Pocket-Sized Antenna That Lets You Text Even in a Disaster Like Sandy [Blog post]. Retrieved June 12, 2018, from https://www.wired.com/2014/07/a-brilliant-antennae-that-lets-you-text-even-during-a-catastrophe-like-sandy/

Rogerson James, R. J. (n.d.). What frequency bands will be used for 5G in the UK? [Blog post]. Retrieved June 27, 2018, from https://5g.co.uk/guides/5g-frequencies-in-the-uk-what-you-need-to-know/

Shoemake Matthew, S. M. (2001). *Wi-Fi (IEEE 802.11b) and Bluetooth Coexistence Issues and Solutions for the 2.4 GHz ISM Band*. Retrieved from http://www.ti.com/pdfs/vf/bband/coexistence.pdf

Sims Gary, S. G. (2017, May 26). Why you won't be using Bluetooth 5 on your Galaxy S8 just yet – Gary explains [Blog post]. Retrieved June 26, 2018, from https://www.androidauthority.com/bluetooth-5-samsung-galaxy-s8-774560/

Sommerhoff Peter, S. P. (2018, January 30). Kotlin vs. Java: 9 Benefits of Kotlin for Your Business [Blog post]. Retrieved July 29, 2018, from https://business.udemy.com/blog/kotlin-vs-java-9-benefits-of-kotlin-for-your-business/

StackOverflow. (2018, July 4). Multi peer connection using Google Nearby Connection [Forum post]. Retrieved July 27, 2018, from https://stackoverflow.com/questions/51177985/multi-peer-connection-using-google-nearby-connection

Swain Daniel, S. D. (2017). Nearby Chat - Beta [Mobile App]. Retrieved June 16, 2018, from https://play.google.com/store/apps/details?id=com.danielcswain.nearbychat

Tehrani Mohsen Nader, T. M. N., Uysa Murat, U. M., & Yanikomeroglu Halim, Y. H. (2014, May). Device-to-Device Communication in 5G Cellular Networks: Challenges, Solutions, and Future Directions. *IEEE Communications Magazine*, -(5/14), 86-92. Retrieved from http://sce.carleton.ca/faculty/yanikomeroglu/Pub/ComMag-May2014-mntmuhy.pdf

The Serval Project. (2015, January 7). Serval Mesh. Retrieved July 28, 2018, from http://developer.servalproject.org/dokuwiki/doku.php?id=content:servalmesh:main_page

The Serval Project. (2017). The Serval Project. Retrieved June 14, 2018, from http://www.servalproject.org/

Wi-Fi Alliance. (n.d.). How far does a Wi-Fi Direct connection travel? Retrieved June 25, 2018, from https://www.wi-fi.org/knowledge-center/faq/how-far-does-a-wi-fi-direct-connection-travel

Woolley Martin, W. M. (2017, February 13). Exploring Bluetooth 5 - Going the Distance [Blog post]. Retrieved May 29, 2018, from https://blog.bluetooth.com/exploring-bluetooth-5-going-the-distance

Xavier Thomas, X. T. (2017, August 11). goTenna Mesh Off-Grid Communication Device Review [Blog post]. Retrieved June 13, 2018, from https://morethanjustsurviving.com/gotenna-mesh-off-grid-communication-device-review/

Zello. (2018). Best free push-to-talk (PTT) app, walkie-talkie PoC, and Wi-Fi voice app - Zello. Retrieved June 16, 2018, from https://zello.com/home/

Zhang Sarah, Z. S. (2014, February 25). The Mobile App Driving Venezuela's Anti-Government Protests [Blog post]. Retrieved June 16, 2018, from https://gizmodo.com/the-mobile-app-driving-venezuelas-anti-government-prot-1530045950

APPENDICES

APPENDICES I - SPECIFICATION SHEET

Analysis, design, and implementation of an infrastructure-less situation awareness application

Specification sheet

Business Information Technology

Bachelor Thesis

Written by: Flavien BONVIN

Professor in charge: Yann BOCCHI

Work submitted the 30.07.2018

Table of Contents

Illustration Index

LIST OF ABBREVIATIONS

5G	5th generation wireless systems

CONTEXT

This work has been done in the context of the bachelor in Business Information Technology at the his-SO Valais. It was proposed by the professor Yann Bocchi and Gianluca Rizzo. The student received the task to develop a tool that will explore the potential of opportunistic communication offered by the Android system.

Theoretical research on ad hoc communications has been conducted in the past but because of software and hardware limitations, it was difficult to implement them. New technologies like Google Nearby and Apple iBeacon offers an abstraction layer and can provide a solution to some of the problems past studies coped.

This thesis has multiple objectives, in the first instance, the student will explain and give an overview of the solutions offered by the different actors of this sector. Some technologies that won't be used during the implementation will be analyzed as well (e.g. Apple iBeacon, 5th generation wireless systems (5G)).

After this research stage, the student will implement an application capable of exchanging messages in ad hoc mode. The original idea was to implement a vehicular service such as traffic warning notification. However, the use case is flexible and can deviate from the initial concept.

At last, the student will detail the actual limitations of the technologies he took during the implementation. On top of that he will provide a recommendation guide that could help future development of opportunistic communications. Those guidelines will provide advice on which technology to choose depending on the needs of the project.

1. Work Description

The world of cars is moving towards automation and electricity, new services will be needed to allow cars to spare fuel, shorten travel time and provide a safer driving experience. To make this change possible vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communications are mandatory.

Use case of those type of exchanges are multiple, the first that come to mind are related to road safety. By creating a vehicular network, cars would be able to exchange information about incoming hazards (bad road conditions, obstacles, accidents, …), create a more secure way to change line, warn other vehicles in an intersection, …

Because cars would communicate one to another, traffic could be more efficient, smart intersections (with automatic traffic lights) that adapts to the incoming traffic would become a reality. Traffic routing in a city allows the reduction of the amount of traffic jam during spike hours.

To avoid unnecessary network exchange, potential delay increase and poor service quality in rural places all communications will be made ad hoc (from one car to another, no network infrastructure needed). This involves the creation of a meshed network that allows automobile to join and left it freely.

The goal of this work isn't to create the protocol of communication between but rather to see if Android can handle this type of exchange in a secure and reliable way by implementing a real world prototype. The expected result is an instrument using opportunistic communication, this tool will help understand the implication of such type of data transmission both on a system and architectural level.

2. Goals

In this bachelor thesis the objectives are the following:

- Design of a vehicular service, such as traffic warning notification (the application may vary depending on the input from the student).
- Design and implementation of an android app capable of exchanging messages in ad hoc mode using various network interfaces, and implementing the given vehicular service.
- Evaluation of the proposed solution, critical review of the main implementation choices made, and documentation of the main lessons learned.
- Give advice regarding the way to implement an application using Google Nearby (or any other technologies) depending on the use case.

3. Work Packages

3.1 State of Art

Before starting the implementation of the use case or any programming, it is critical to know if any solution exists on the market. So far theoretical studies have been made, but they have seen relatively little implementation. The student will take a special care to see what are the obstacles that holds the development of those communications.

The goal of this phase for the student is to be able to know what are the potential problems he will encounter during the development of the application.

3.2 Technologies Study

Because the student has to create an Android application, he has to know what are the different API provided by Google.

The Google I/O occurs from the 8th to the 10th of may. It is usual for Google to announce new technologies during this event, the student will follow what is unveiled during those days to know if something new can be used in this thesis.

The technology choice should comply with a set of requirements, speed (bandwidth, latency, …) and security will be a priority.

At the end of the study, the student will be able to know what technologies can or can't be used. All technologies that do not respond to the security and speed criteria can be discarded and won't be tested.

3.3 Test and Choice of Technologies

To be able to see what technology suits the project the most, the student will implement an application that allows him to familiarize himself with the different technologies provided by Android and Google. The student will implement a latency test to know how fast communications can be in a controlled environment. On top of the test of latency, a reliability test will be performed to see if a connection can be maintained between two cars in a highway. The latest test the student will implement is a connection time test, to see how long it takes for a connection to be established, the student will also try to see if this time can be reduced with some parameters change.

Once the tests finished, the student will be able to choose a technology and have data to motivate his choice. A comparison chart will be established by the student to allow a quick overview of the differences.

3.4 Implementation of the Use Case

Based on the result of the test the student will implement the use case, the subject of the use case can vary depending on the ideas of the student and professor. The goal of this implementation is to create a showcase app that can be used as a demonstration of Google Nearby.

3.5 Report Writing

During the whole project, the student will create, update the report he has to handle back. This document contains all the choice he made, the test results, the problem he had.

Some points (related to the thesis but not used) need to be addressed by the student:

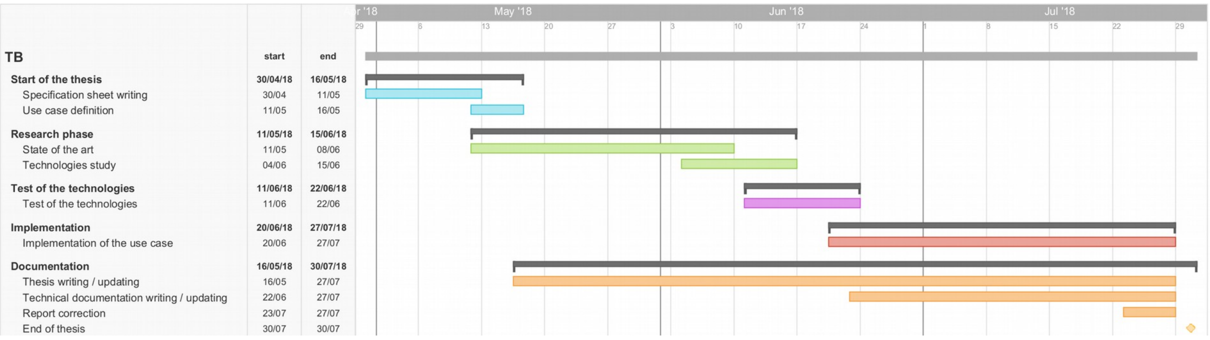
- The state of the 5G network and the possibilities this new technology offers. The legal position this in Switzerland will be discussed too.
- The technologies proposed by Apple for its ecosystem.

3.6 Technical Report

To allow further exploration of Google Nearby a full technical documentation is needed, this paper will facilitate an eventual future student to understand what the student did for this bachelor thesis, to have access to the different accounts that were created for this project, …

4. Planning

The following planning has been established by the student. He will document his work and any eventual change of the scheduling during his e execution of his thesis. All modifications (decrease or increase of time on a task) will be explained. references



TB	start	end
Start of the thesis	**30/04/18**	**16/05/18**
Specification sheet writing	30/04	11/05
Use case definition	11/05	16/05
Research phase	**11/05/18**	**15/06/18**
State of the art	11/05	08/06
Technologies study	04/06	15/06
Test of the technologies	**11/06/18**	**22/06/18**
Test of the technologies	11/06	22/06
Implementation	**20/06/18**	**27/07/18**
Implementation of the use case	20/06	27/07
Documentation	**16/05/18**	**30/07/18**
Thesis writing / updating	16/05	27/07
Technical documentation writing / updating	22/06	27/07
Report correction	23/07	27/07
End of thesis	30/07	30/07

60. Illustration: Gantt Diagram

4.1 Tasks Detail

Name	Start	End	Hours
Specification sheet	30 April	11 May	20h
Use case definition	11 May	16 May	10h
State of the art	11 May	8 June	60h
Technologies study	4 June	15 June	35h
Test of the technologies	11 June	22 June	35h
Implementation of the use case	20 June	27 July	120h
Thesis writing/updating	16 May	27 July	40h
Technical documentation writing/updating	22 June	27 July	20h
Report correction	23 July	27 July	20h
		TOTAL	**360h**

4.2 Holidays

The student won't be able to work on his thesis for 2 weeks. He has to make a military service from the 22nd of May to the 2nd of June.

Mr. Yann Bocchi has a conference in Bilbao from the 4th to the 7th of June. He will be on holiday from the 9th to the 23rd of July. Project delivery.

All papers related to the project (documentation, code, …) will be delivered by the student on the 30th of July at 12 o'clock.

4.3 Oral Defense

The oral defense of the project will take place the ______________________

Analysis, design, and implementation of an infrastructure-less situation awareness application

Use Case Definition

Business Information Technology

Bachelor Thesis

Written by: Flavien BONVIN

Professor in charge: Yann BOCCHI

Work submitted the 30.07.2018

Table of Contents

INTRODUCTION

In this document, we will detail the potential use cases we came up with during our research phase. The goal of this file is to give several choices of application implementation that might be chosen.

We analyzed two different use cases during our research phase. However, only one was selected, and we decided to keep all the research made and place them in this document, chapter 5 —Use Case analysis—Peer reviewing (p. 5)

1. Car Communication Experimentation

This idea is the closest to the one explained in the original document of the thesis. The objective is to create an experimental tool to see if Android and Google Nearby can provide a safe and reliable way for vehicles to communicate.

The aim is to test the feasibility of this type of communication and to be able to say whether the technology used allows the vehicle to exchange information or not. Some aspects are fundamental (connection time, latency, reliability) and will be extensively tested. A real-world test is planned.

A mesh network would be dynamically created with the car's surroundings. One challenge of the application is the transmission of data, not all the cars have to receive the messages; it would be useless for a car 2 kilometers ahead to revive information about slowdowns behind it. The data have to go up the traffic.

1.1 Personal Note

One downside of this use case is the research orientation it has. It might be hard to have something easy to demonstrate in a salon.

Another problem that might come is the distance of connection. In the technical documentation range of up to 100 meters (Google Developers, 2017) is given, and this might make the demonstration harder in a confined space.

2. Peer Reviewing

It is always hard to know what other people in a classroom, a conference thought of a presentation. An application that provides a reliable way to vote for a given set of criteria could help the orator to know what is right and what needs to be improved in his speech and idea.

The concept relies on the start strategy, the person who creates the room would be the center of the star. Other people could join a room and see what the criteria are. He could then vote and send the results to the administrator (the person who created the room).

2.1 Personal Note

This application is inspired by a project we had to do this year, we had to implement something similar but on a website and it requires a lot of infrastructure (a server, a computer) With a solution like this one, users can in less than 3 minutes create a simple room where people can join and vote. Because we would use the Connection API of Google Nearby, we do not need any internet access (Google Developers, 2017), which is a bonus.

App like this is way easier to demonstrate in a salon than the other use cases.

3. Disaster assistance

When there is a disaster (earthquake, tsunami, avalanche or even war), wireless communication is hard or impossible to maintain. That is why an application that will provide offline wireless message exchange could be useful.

The application would allow anyone in a given area to send a message with the position of the sender attached to it (as precise as possible). This message would be available to anyone who has the app. A list of messages that are in a given range (up to 100 meters [Google Developers, 2017]) will be displayed, helping the rescue team to go where help is needed.

Another feature of the app could be a channel reserved for the rescue team where they could ask for help, supply or even transport. This would be separated to avoid confusion and help the organization of the people.

3.1 Personal Note

This use case is by far the most technically challenging, in our opinion. It involves many technologies that are not included in the Google Nearby API like floating content. This is also the application with the most “useful” use case.

4. Messaging App for a Music Festival

Sometimes it is hard to communicate or find friends at a music festival, some people do not have internet access with their mobile phone contract, and there is no Wi-Fi network available. An application using Google Nearby Connection API can solve this.

The possible features in this application are nearly endless, we can think of private groups of friends, groups related to a group playing at the festival or by music genres. This can become quite complex, but it is relatively easy to use it as a demo in a salon.

This application involves floating content, this is not supported by default and might need much work. This means we will have to experiment with floating content to see how it works and how easy it is to use it. Depending on the results, we can change the scope of the application.

4.1 Personal Note

This application can be quite lovely to implement because it responds to a real-life problem. On top of that, the HES SO has a booth at the Paléo Festival; it could be an excellent way to promote the application. One problem is that the Paléo Festival takes place from the 17th to the 22nd of July, the app might not be ready in time (regarding the planning I established).

This is also a nice research use cases because it will try to implement floating content.

5. Use Case analysis—Peer reviewing

It is always hard to know what other people in a classroom, a conference thought of a presentation. An application that provides a reliable way to vote for a given set of criteria could help the orator to know what is right and what needs to be improved in his speech and idea.

The application uses a star topology, the person who creates the room would be the center of the star. Other people could join a room and see what the criteria are. Visitants could then give feedback and send the results to the administrator (the person who created the room).

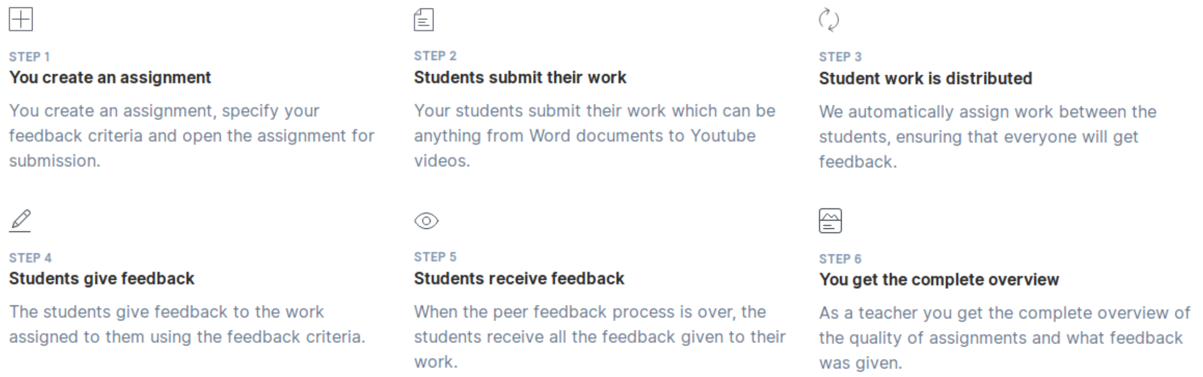
5.1 Similar Solutions—Peergrade

peergrade

61. Illustration: Peergrade logo (Nordicmakers.vc, n.d.)

Peergrade is the fruit of Davide Kofoed Wind; he was teaching Master course in the Technical University of Denmark and view his lesson attendance grow from 20 to 130 students. He did not have the time to evaluate and give sufficient feedback to every student. He had to find a way that will allow students to submit homework or paper and get decent feedback in return.

This was when he started developing Peergrade, a simple tool that allows students to give and get the feedback they deserve. In six simple steps, it is possible to create a platform where students can interact and exchange ideas (Peergrade, 2018).



STEP 1
You create an assignment
You create an assignment, specify your feedback criteria and open the assignment for submission.

STEP 2
Students submit their work
Your students submit their work which can be anything from Word documents to Youtube videos.

STEP 3
Student work is distributed
We automatically assign work between the students, ensuring that everyone will get feedback.

STEP 4
Students give feedback
The students give feedback to the work assigned to them using the feedback criteria.

STEP 5
Students receive feedback
When the peer feedback process is over, the students receive all the feedback given to their work.

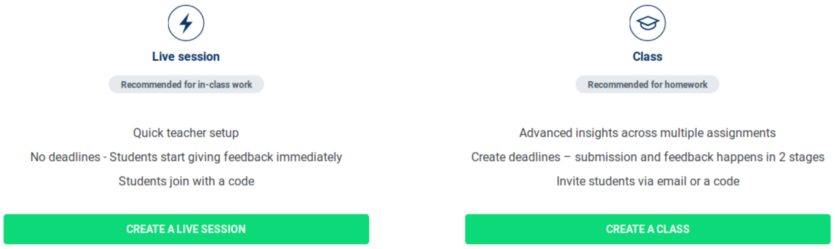
STEP 6
You get the complete overview
As a teacher you get the complete overview of the quality of assignments and what feedback was given.

62. Illustration: Steps needed to evaluate peers Peergrade (Peergrade, 2018)

5.1.1 How It Works

Peergrade allows teachers to create two types of rooms, “Live Session” this is recommended for in-class work and “Class” that are more features packed and can be used during a whole semester. In both cases a Peergrade account is needed, this is not possible to connect and evaluate other people’s work without an account.

The main difference between “Live Session” and “Class” is the deadline. In “Live Session” students start giving feedback as soon as files are submitted. Unlike “Live Session”, “Class” are time planned, a submission and evaluation deadline are set, and the student cannot evaluate before the beginning of the evaluation phase. Also, the two blocks cannot overlap.



Live session

Recommended for in-class work

Quick teacher setup

No deadlines - Students start giving feedback immediately

Students join with a code

Class

Recommended for homework

Advanced insights across multiple assignments

Create deadlines – submission and feedback happens in 2 stages

Invite students via email or a code

63. Illustration: Type of assignments (Peergrade, 2018)

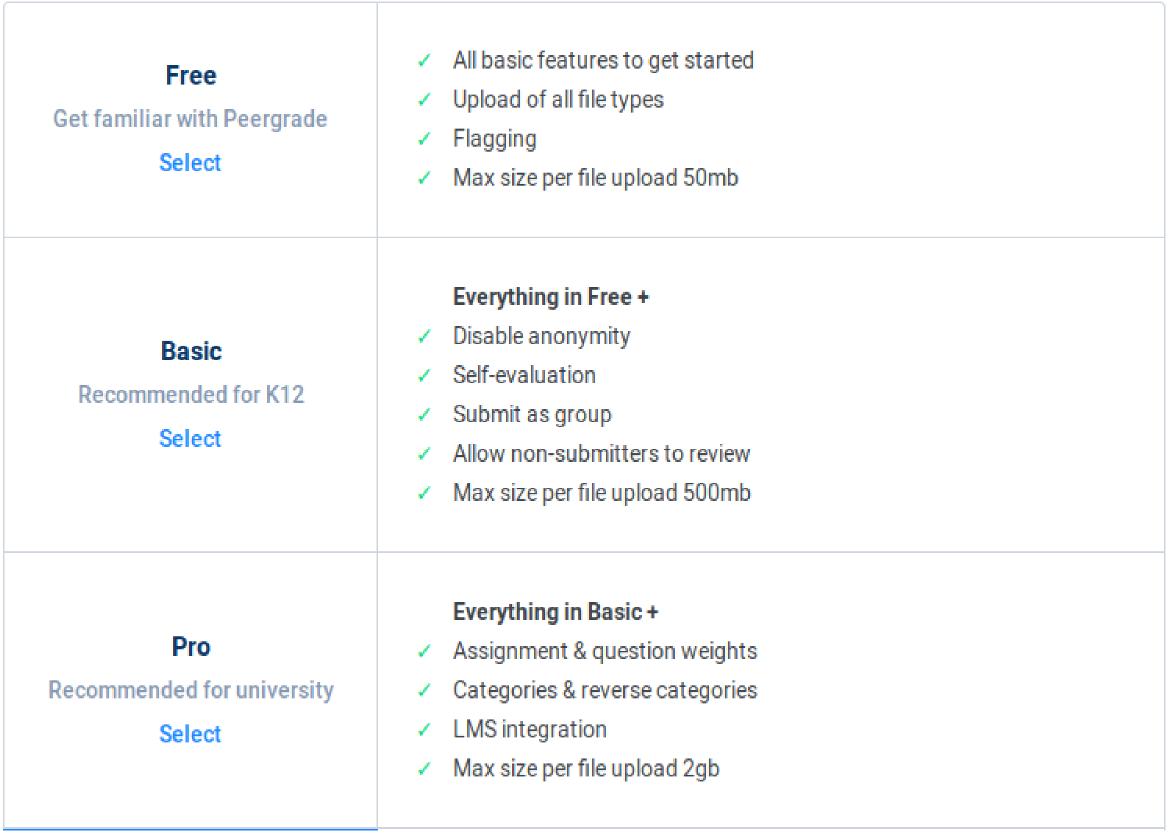
A typical evaluation, regardless of the type of room the teacher created, takes place in the following way:

1. The teacher creates the room, if it chooses a “Live Session”, a code will be generated and is needed for the student to join the room. If the room is “Class” an invite email is sent to every participant, the professor added.
2. Criteria can be set in a menu called “Feedback Rubric”, multiple choices of answers are available. “Text” allows students to write some words about a given question. “Scale” give the possibilities to create multiple-choice questions. Finally “Yes/No” is a binary question, it can either be true or false, completed or not.
3. Once the criteria are defined, the teacher will, if he created a “Class” define a deadline for submissions and another for the evaluation.
4. Now that the room is entirely created and customized to the need of the professor, students will be able to access it and submit their work. They can deposit a file, give an external URL, send a Google Drive file or have a text area where they can write.
5. Once the evaluation phase is reached, students can evaluate other student work. They have to fill the criteria defined by the teacher in the “Feedback rubric”.

6. At this point the student can see the feedback established by his peers, before a summary of all the criteria, it has to give a "Feedback critic". This section allows the student to give a critic the feedback they received; the evaluator will receive the critic. If someone is biased and give bad feedback, the teacher can be notified.
7. Finally, the student can see a summary of all the feedback he received. On top of that, he can see each feedback. Those results are anonymous to avoid conflicts between users.

5.1.2 Pricing

There are three tiers of pricing for Peergrade, one free account that let users get familiar with the website and see if it suits their need. The free account is valid for one month only. Other accounts are Basic and Pro. They decided to base the cost of the subscription based on the number of students that will join the class. For a basic account, 30 students will cost $55, and the same amount of student will cost $137 for a pro-subscription.



Free Get familiar with Peergrade Select	✓ All basic features to get started ✓ Upload of all file types ✓ Flagging ✓ Max size per file upload 50mb
Basic Recommended for K12 Select	**Everything in Free +** ✓ Disable anonymity ✓ Self-evaluation ✓ Submit as group ✓ Allow non-submitters to review ✓ Max size per file upload 500mb
Pro Recommended for university Select	**Everything in Basic +** ✓ Assignment & question weights ✓ Categories & reverse categories ✓ LMS integration ✓ Max size per file upload 2gb

64. Illustration: Pricing of Peergrade (Peergrade, 2018)

5.1.3 Mobile Friendly

During our research, we tested the website on both mobile and web browser, and the experience is the same on both devices. The website is very responsive, and even if some tasks are more accessible to achieve on a larger screen, like uploading a document to an assignment, it still works on the mobile.

Peergrade did not develop an Android or iOS application. We do not find it problematic, because the majority of the work will be done on a computer. However, everything can be done from the mobile version of the website if the student or teacher do not have access to a computer.

5.1.4 Advantages of Peergrade

There is much area where Peergrade shines. We enjoyed the website that is nice looking and fast.

- The design of the website is simple yet perfect. All the information are where users can expect them; there is no hidden settings and features. We were able to create and customize a “Class” without any help.
- This is quick and easy to set up; the website is well thought. No help was needed to configure the “Class”. Even the “Feedback Rubric” was easy to understand, we were able to change the existing question, add new ones.
- Everything works automatically; this is particularly visible with “Live Session” for the students. As soon as a step is completed, the next one load automatically. Evaluating work is a breeze, the web page is split in half, one is for the paper to evaluate, and the other is for the criteria.

5.1.5 Flaws of Peergrade

Peergrade is an excellent website but works perfectly with school work only. Here are the points that were limiting for the use case we developed.

- An account is required to submit and evaluate other people’s works; this made the first evaluation longer.
- There are no guest accounts; visitors cannot evaluate other people’s work if they do not submit a work. This is understandable because it is aimed at students and classrooms, but an external person who would vote for a school project will not be able to give its feedback.
- The product is excellent but works correctly only with school work. Features are limited as soon as it is used outside school. Because a work needs to be uploaded, it is hard to recommend it for evaluating an orator in a conference, for example.

5.1.6 Difference With Our Use Case

There are several differences between our use case and Peergrade. Firstly, our application will not evaluate work or papers but presentations, this means that no document will be uploaded to the platform and that some people will vote without presenting something, as it is the case in conferences.

We do not want to make the account creation mandatory; this has two advantages, ease of access and respect for privacy. Ease of access because anybody can launch the application and vote for a nearby presentation. Privacy because a user might want to keep his personal information private.

Everything is centralized with Peergrade; we decided to make our application decentralized by implementing ad hoc networking. There is multiple motivation to make our application this way, poor quality or unavailable networks, no need for us to implement a server back end, allow only people near the presentation to vote.

Lastly, we will not implement a website but an Android application, this is not important, but it needs to be specified.

5.2 Similar solution—SpeakUp

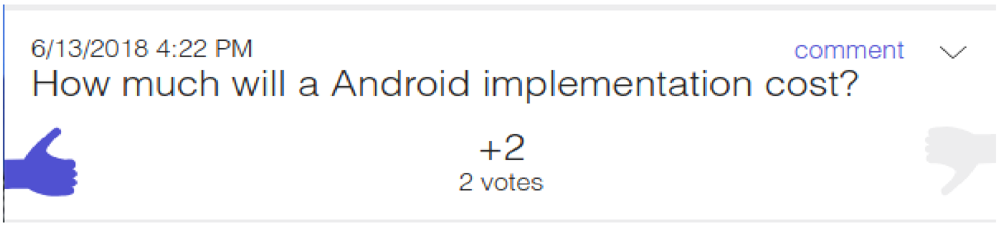
65. Illustration: SpeakUp logo (SpeakUp, 2018)

Developed by the Ecole Polytechnique fédérale de Lausanne (EPFL), SpeakUp is an application that let the audience share and rate each other's questions. SpeakUp wants to address the issues there is with the questions at the end of a presentation. There is either nobody that can remember or wants to share or a lot of questions and not enough time to answer them all letting people disappointed.

5.2.1 How It Works

SpeakUp wants to improve interaction in conferences, brainstorming sessions and even classroom through anonymous comments. Users can create and vote for multiple-choice questions, ask and vote for a question.

The service is available in any web browser and on both Google Android and Apple iOS making the service highly available.



6/13/2018 4:22 PM

How much will a Android implementation cost?

+2
2 votes

66. Illustration: Example of questions on SpeakUp. From the author

We can see in the above illustration a typical question on SpeakUp. The concept of the application is pretty straightforward, here are the steps it will take to use SpeakUp correctly:

1. Creation of the room by someone in the organization, nothing to set except the name and two options. “Nickname” that will ask people for a nickname when they connect and “Here & Now” that will create a room for 24 hours and make it visible by nearby people. We will detail the “Here & Now” option later on this chapter.
2. Participants can join the room, either by entering a number generated by the website at the creation or by browsing nearby rooms.
3. As soon as they join, users can ask questions and vote for existing one. If there is a multiple choice question, they can also vote. If a question is not clear, it is possible to comment on it to add information.
4. At the end of the presentation, the orator can merely see what questions have the higher score and answer them.

As shown above, only four steps are needed to use SpeakUp fully. The process is quick and easy to master. We have two points to clarify concerning the operation of the application:

- “Here & Now” option will create a room that is visible by anybody, no code is needed to join the room. After 24 hours the room is automatically deleted.
- Only room administrators can create multiple choice questions; they can choose to show the answer directly or not.

5.2.2 Another Service Offered by SpeakUp

If there is no internet connection where SpeakUp is used, the team developed a portable device called SpeakUpBox that creates a local Wi-Fi network. The device does not need the internet either the service is hosted on the device making possible to create SpeakUp room anywhere.

67. Illustration: SpeakUpBox (SpeakUp, 2018)

5.2.3 Pricing of SpeakUp

Apart from the SpeakUpBox, the service is free. We had to contact the company to know the price of the device; nothing is indicated on the website. However, they never responded to our request, we do not know the price of the SpekaUpBox.

5.2.4 Advantages of SpeakUp

Speak up address an interesting issue, and we found no competitors, the main advantages are the following:

- The service is fast and easy to understand. There are not many settings, and both administrators and users understand how the application works instantly.
- Commenting on the question is vital to add missing information. This is well implemented.
- The website and the application stick to its functionalities; the design is uncluttered and well thought.

5.2.5 Flaws of SpeakUp

There is not a lot to criticize about SpeakUp the biggest problem we found is the SpeakUpBox; they allow anybody in the world in any condition to use the service which is excellent. However, ad hoc solutions exist and can also run a decentralized version of SpeakUp without the need of the box.

We understand that they did it because ad hoc solutions such as Google Nearby or Apple MultipeerConnectivity works on smartphones only, their box make the service available for laptops, computers, and smartphones but it is quite bulky.

5.2.6 Difference with our use case

SpeakUp is an excellent solution, it works great and is fast. The focus is on interaction and question during a presentation. Our service will not cover those use case but wants to give feedback to an orator on a set of defined criteria. Because the criteria as manually established, they can target anything, from the quality of the presentation to the evaluation of the product presented.

We will not implement a website, only an application. This choice is made to have an application that can work without infrastructure without the need for external hardware as it is the case with the SpeakUpBox.

REFERENCES

Google Developers. (2018). Nearby Message API Overview. Retrieved June 23, 2018, from https://developers.google.com/nearby/messages/overview

Google Developers. (2018). Nearby Connection API Strategies. Retrieved June 23, 2018, from https://developers.google.com/nearby/connections/strategies

Analysis, design, and implementation of an infrastructure-less situation awareness application

Product Backlog

Business Information Technology

Bachelor Thesis

Written by: Flavien BONVIN

Professor in charge: Yann BOCCHI

Work submitted the 30.07.2018

1. Product Backlog

N°	Day Added	Role	Functionality	Goal	Acceptance criteria	Priority	Story Point	Done Done	MoSCoW
1	29.06	Dev	Prepare the work environment	Share code across computer	GitHub created	10000	1	01.07	Must
2	29.06	Dev	Create the message structure	Know how messages will be sent	Know how messages will be sent and read	9500	2	07.07	Must
3	29.06	Victim	Send a message to nearby devices	Ask for help	Have a view where message can be sent Choose a category of message Have a button to send the message	9000	8	07.07	Must
30	03.07	Both	Have an automatically generated name	To identify the message I created	The generated name is visible in the application	8950	1	05.07	Must
31	03.07	Both	Change the automatically generate name	Be able to give my name to help	There is a button allowing the name change The name is kept in the preferences of the application	8940	2	07.07	Must
4	29.06	Victim	Delete a message if it is not relevant	Avoid information overflow	Have a button where the user can delete a message Only the creator of a message can delete it	8900	5	13.07	Must
5	29.06	Helper	Send information about resources	Inform nearby people where resources are	Have a view where message can be sent Choose a category of message Have a button to send the message	8700	2	07.07	Must
6	29.06	Helper	See messages nearby	Give help to people in need	See a list of nearby messages Have the distance to the message Easily see the message category	8500	5	09.07	Must
8	29.06	Helper	Confirm that an information is still relevant	Maintain up to date information	Receive a notification when someone ask for update Confirm if an information is still correct or not	7500	3	19.07	Must

					Have a list of choice of possible choice				
9	29.06	Helper	Confirm that information are not up to date	Avoid wrong information on the application	Have a button to say that the information is not up to date The message is still displayed on other phones Other users can confirm that this is not up to date Other users see that the message might be outdated	7000	2	19.07	Must
10	29.06	Helper	Filter messages by category	Manage the type of information I see	Have a filter button on top of the list of message See the different categories Message of the correct category displayed Have a way to reset the filter	6800	3		Should
11	29.06	Helper	Define the radius of the zone	Know message for a zone of a given time	Have a setting or a textview where distance can be set Have a button to confirm the change Have a toast that confirm the change	6500	5	18.07	Must
12	29.06	Helper	Have a map of the nearby messages	Easily see where are the nearby messages	Have a tab with a map See the map if available	6300	8	16.07	Must
19	29.06	Helper	Dynamic expiration time	Delete the messages that are old	Once a message is confirmed wrong 3 to 5 times it is deleted The message is deleted automatically but is placed in bin	6200	5	19.07	Could
13	29.06	Helper	Different color and icon for every category	Make easier to see the different messages	Each category type have a distinct color This can also be a different type of icon instead	6000	2	07.07	Must
14	29.06	Helper	Order message by date	Know the latest messages	Have a a button to filter message by date Filter ascending or descending	5500	3	19.07	Should
15	29.06	Helper	Order the message by distance	Know where are the closest messages	Have a button to filter message by distance	5000	3	19.07	Should

					Filter ascending and descending				
16	29.06	Helper	Give feedback about information	Simple confirm if a message is true or not	Have two buttons in the detail of a message One confirm that the information is correct One is to say that the information is not up to date Information are linked to the message and other can see	4900	2	19.07	Should
17	29.06	Helper	Download the map offline for future use	Have the map ready for an eventual disaster	Have a message asking if the user wants to download The map of an area is downloaded	4800	8		Should
18	29.06	Helper	Share the map to other users	If the map is not downloaded on another Phone, it will be shared by other users	Have an automatic message that ask user if he wants To download a map The map will be get from nearby peers	4700	8		Could
20	29.06	Helper	See deleted messages on a bin	See what are the messages that has been deleted	Have a way to access the bin Have a list of all deleted messages by other users	4500	3	25.07	Could
21	29.06	Helper	Retrieve message from the bin	Get message back if they are correct	When in bin a button allow to retrieve the message A confirmation message ask if it is sure to retrieve it Once confirmed it is taken out the bin	4400	3	25.07	Could
22	02.07	Helper	Send the APK of the application	Other people will access the application	There is a sharing button in the setting Other users will be prompted to accept the incoming file Installation of the app start as soon as the file is received	4300	8		Could
23	02.07	Both	Have a tutorial for the first launch of the app	Understand how the application works	At the first app launch, tutorial launch	4200	5	17.07	Must
24	02.07	Both	Launch the tutorial again	See the tutorial again, if something is unclear	In the setting there is a button to allow that When the button is pressed, the tutorial play again	4100	2	17.07	Should

25	02.07	Both	There is the number of people that Received the message	Reassure people in need for help	There is in the list of message the number of readers This number automatically update itself	4000	3	13.07	Must
26	02.07	Both	Not see outdated messages	Avoid wrong information on the application	The outdated messages are automatically deleted The exact protocol established is in documentation	3900	2	19.07	Should
27	02.07	Both	See the creation date of a message	Know if it is recent of not	On the message list there is the time of creation This can be the date or the xx times ago This is also shown on the map	3800	2	09.07	Must
28	02.07	Both	Update message the user created	Make modification to an old message	There is a button to edit messages The message is updated for everybody The way this is handled is detailed in the documentation	3700	8		Should
29	02.07	Both	Have a help on how to use the app	Make sure that people can have offline help	A button in either the settings or main menu will launch help The help might have text or image, the better	3600	3		Should
32	3.07	Both	Have a chat	Exchange direct message to other person	A new tab is available on the application Incoming messages are displayed	3500	8		Would
7	29.06	Helper	Ask for confirmation about a message	Confirm that an information is still relevant	Have a button to ask for the confirmation of a message See in the message list that need confirmation	0	2	19.07	Must

Total SP	**127**
SP done	**79**
SP rejected	**2**

APPENDICES IV – LOGBOOK

Analysis, design, and implementation of an infrastructure-less situation awareness application

Logbook

Business Information Technology

Bachelor Thesis

Written by: Flavien BONVIN

Professor in charge: Yann BOCCHI

Work submitted the 30.07.2018

1. Logbook

Week	Day	Task	Detail	Hours done	Week total	Note about the week
Week 1	07.05	Technologies study	First researches about the existing technologies	3		
		Specification sheet	Beginning of the writing of the specification sheet	3		
		Use case definition	Wrote in the specification sheet the use case I thought of	1		
	08.05	Specification sheet	Writing of the specification sheet	2		
		Technologies study	Further research on the technologies	2		
	09.05	Specification sheet	End of the writing of the first draft of the specification sheet	4		
		Technologies study	Further research on the technologies	2		
		Meeting	1st meeting with Yann BOCCHI and Gianluca RIZZO	1	**18**	
Week 2	14.05	Specification sheet	Correction of the specification sheet, remarks made during the 2nd meeting	2		
		State of the art	Beginning of the research about the state of the art, focus on the Android APIs	3		
		Technologies study	At the same time of the state of the art, studying of the technologies	1		
	15.05	Specification sheet	Correction of the style and errors of the specification sheet	1		
		State of the art	Research about the state of the art, focus on the Android APIs	3		
	16.05	State of the art	Research about the Google API, focus on WiFi and Bluetooth	6		
		Meeting	2nd meeting with Yann BOCCHI and Gianluca RIZZO	1		
	17.05	Specification sheet	Final correction of the document after last meeting	3		
	18.05	Specification sheet	Final proofreading of the document	2	**22**	
Week 3	27.05	Use case definition	Writing of the file	3		Army
		State of the art	Starting to write the document	1	**5**	

Week 4	28.05	State of the art	Documentation writing, focus on Google's Bluetooth API	2		Army
	29.05	State of the art	Documentation writing, focus on Google's Bluetooth API	2		
	30.05	State of the art	Documentation writing, focus on Bluetooth 5G	2	**6**	
Week 5	04.06	State of the art	Writing the beginning of the file, chapter one	6		
	05.06	State of the art	Step back of the project and took time to organize the work, end of part 1	6		
	06.06	State of the art	End of the first part of the state of the art (chapter 1.1 to 1.4)	5		
		Meeting	3rd meeting with Yann BOCCHI and Gianluca RIZZO	1		
	07.06	State of the art	Correction and rereading of what have been done Working on other part of the state of the art	4		
	08.06	State of the art	Writing of the routing protocols part	6	**28**	
Week 6	11.06	State of the art	Analysis of the existing – Peer reviewing use case	5		
	12.06	State of the art	Analysis of the existing – Peer reviewing use case	3		
	13.06	State of the art	Analysis of the existing – Disaster assistance use case	5		
		Meeting	4th meeting with Yann BOCCHI and Gianluca RIZZO	1		
	15.06	State of the art	Correction of document after feedback from Yann BOCCHI Analysis of the existing – Disaster assistance use case	3		
	16.06	State of the art	Correction and proofreading of the use case analysis	5	**22**	
Week 7	19.06	State of the art	Technology study (update of the Bluetooth part)	3		Exam session
	20.06	State of the art	Proofreading of the bluetooth part Writing of the WiFi Direct part	3		
	23.06	State of the art	Writing of the existing technologies, end of Google and of MeshKit	6	**12**	

Week 8	25.06	Meeting	5th meeting with Yann BOCCHI and Gianluca RIZZO	1	
		Administration	Converted the transcripts to digital format	2	
		State of the art	Correction of some remarks made by Mr BOCCHI and RIZZO	5	
	26.07	State of the art	Existing technologies – Apple, End of MeshKit, Start of Several Project	7	
	27.06	State of the art	Existing technologies – End of Several Project, Start of 5G	6	
	28.06	State of the art	Existing technologies – End of 5G	5	
	29.06	State of the art	Use case analysis – Improvement of the explanation of the use case	4	
		Implementation of the use case	Creation of the backlog	3	
	30.06	Implementation of the use case	Designing of the mockups	6	**39**

Week 9	02.07	Meeting	6th meeting with Yann BOCCHI and Gianluca RIZZO	1		
		Implementation of the use case	Update of the backlog	3		
		Administration	Creation of the intermediate planning	1		
		State of the art	Correction of some feedback Mr.BOCCHI and RIZZO gave me	4		
	03.07	Implementation of the use case	Beginning of the implementation of the application, Nearby tests	7		Implementation is slower than expected due to nearby problems.
	04.07	Implementation of the use case	Further testing of Google Nearby, problems with the connection process	8		
	05.07	Implementation of the use case	Fixing some Google Nearby problems, further testing	8		Lack of documentation about it Is the cause
	06.07	Implementation of the use case	Design of the application, fixing some google nearby problems, overall improvement	9		After question asked on Stackoverflow Everything quicker and works fine
		Administration	Updating of the logbook, add intermediate planning and task hour count	1		
	07.07	Use case definition	End of the document, final correction and title change	1		
		State of the art	Correction of abbreviation, self references and title change	1		
		Implementation of the use case	Change use name, send messages, see nearby messages	6	**50**	

Week 10	09.07	Implementation of the use case	Display of the message implemented correctly	4	
		Implementation of the use case	Save of the message sent, received and to send on the phone	4	
		Implementation of the use case	Save messages when no peers available and send them automatically when peer	3	
		Implementation of the use case	General improvement (string extraction, commenting, correcting errors)	3	
	10.07	Implementation of the use case	Message forwarding when new peers are connected	6	
		Implementation of the use case	Improve message handling, testing of message transmission (lot of bugfix)	4	
	11.07	Thesis writing / updating	Writing of the designing of the application (start of challenging part)	5	
		Technical documentation writing	Start of the technical documentation	2	
	13.07	Implementation of the use case	Deletion of message handled, improve UI (add sent tab)	6	
	14.07	Thesis writing / updating	Writing chapter about use case implementation reflection	7	
	15.07	Thesis writing / updating	Writing chapter about general implementation	8	**52**

Week 11	16.07	Implementation of the use case	Display distance for each messages	3	
		Implementation of the use case	Add confirmation screen before sending message	2	
		Implementation of the use case	Add map and show custom marker	4	
	17.07	Implementation of the use case	Add on boarding	4	
		Implementation of the use case	Add settings activity	3	
	18.07	Implementation of the use case	Add floating content	6	
		Implementation of the use case	More on boarding screen	2	
	19.07	Implementation of the use case	Add dynamic expiration time	5	
		Implementation of the use case	Add about page	2	
		Implementation of the use case	Add network status view	3	
	20.07	Report correction	Proofreading of the document	4	
		Thesis writing / updating	Beginning of the details implementation, back end challenges done	4	
		Technical documentation writing	Updating of the technical documentation	2	
	21.07	Thesis writing / updating	Writing of the details implementation, user experience and interface done	7	
	22.07	Report correction	Proofreading of the document	2	
		Thesis writing / updating	Writing of the management part	3	**56**

Week 12	23.07	Thesis writing / updating	Writing forewords, abstract, introduction	5	
		Report correction	Proofreading the state of the art and the specification sheet	3	
	24.07	Meeting	7th meeting with Yann BOCCHI and Gianluca RIZZO	1	
		Thesis writing / updating	Writing of the product backlog chapter	2	
		Report correction	Merging of the correction made by my sister and girlfriend	4	
	25.07	Implementation of the use case	Correction of the feedback made yesterday	6	
	26.07	Thesis writing / updating	Writing of the discussion	4	
		Report correction	Correction of some of the first chapter	2	
	27.07	Thesis writing / updating	End of the discussion, writing of the conclustion	6	
		Report correction	Proofreading of the State of the art	2	
	28.07	Thesis writing / updating	Preparation of the annexes	3	
		Report correction	Proofreading of the thesis	5	
	29.07	Technical documentation writing	Writing of the technical documentation	4	
		Implementation of the use case	Commenting and polishing code	2	**49**
Week 13	30.07	Thesis writing / updating	Final preparation of the report	2	**2**
			Total		**349**

Analysis, design, and implementation of an infrastructure-less situation awareness application

VANETs Technologies

Business Information Technology

Bachelor Thesis

Written by: Flavien BONVIN

Professor in charge: Yann BOCCHI

Work submitted the 30.07.2018

Wireless Standard	F in GHz	Data Transmission Rate	Max Signal Coverage ≈	Signal Interference	Maintenance	Accessibility	Upfront Cost	Security	Mobility supprot in ≈ km/h	Suitable for Safety Application	Suitabe for Non-Safety Application
Cellular System	Operator Dependent	≈ 384 Kbps - 129 Mbps	50 km	Low	Difficult	Contention Based	High	High	100-250	No	Yes
WiMAX 802.16m	2.3/2.5/3.5	≈ 75 - 300 Mbps	50 km	High	Difficult	Schedule based	High	High	60-250	No	Yes
MBWA 802.20	3.5	≈ 4.5 Mbps	15 km	High	Easy	Schedule based	High	High	100-250	No	Yes
Microwave	1-3	≈ 16 Gbps	30 km	High	Difficult	Contention based	High	Low	--	No	No
Wi-Fi 802.11a	5.1/5.8	≈ 54 Mbps	100 m	Low	Easy	Contention based	High	Low	40-120	No	No
Wi-Fi 802.11b	2.4	≈ 11 Mbps	100 m	High	Easy	Contention based	Moderate	Low	40-150	No	No
Wi-Fi 802.11g	2.4	≈ 54 Mbps	140 m	High	Easy	Contention based	Moderate	Low	40-120	No	No
Wi-Fi 802.11n	2.4/5	≈100 Mbps	250 m	High	Easy	Contention based	High	High	40-120	No	No
DSRC 802.11p	5.8/5.9	≈ 27 Mbps	1 km	Low	Easy	Contention based	Moderate	High	40-150	Yes	Yes
CALM M5	5	≈ 6 Mbps	10 km	High	Difficult	Contention based	High	High	40-150	No	Yes
Infrared	300 GHz - 400 THz	≈ 115 Kbps- 4 Mbps	100 m	Low	Easy	Contention based	Low	High	250	No	No
Bluetooth 802.15.1	2.4 GHz	≈ 1-24 Mbps	100 m	High	Easy	Contention based	Low	High	20-30	No	No
ZigBee 802.15.2	868/915 MHz / 2.4 GHz	≈ 250 Kbps	100 m	High	Easy	Contention based	Low	High	10-20	No	No
UWB 802.15.3	3.1-10.6GHz	< 100 Mbps	10 m	Low	Easy	Contention based	Low	High	10-20	No	No

Table comparing different existing technologies for VANETs (Anwer and Guy, 2014, p. 661)

AUTHOR DECLARATION

I declare, by this document, that I have carried out the attached bachelor's work alone, without any assistance other than those duly indicated in the references, and that I only used the sources expressly mentioned. I will not give any copy of this report to any third party without the joint authorization of the head of the business information technology and the professor in charge of the bachelor's work, including the applied research partner with whom I have collaborated, with the exception of individuals who have provided the primary information necessary for the drafting of this work, which I quote below: Yann Bocchi, Gianluca Rizzo.

Sierre, on the 30th of July

Flavien Bonvin