UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

LOST: LEADING OTHERS THROUGH SECURE TRAILS

André Gonçalves da Silva

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2014

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

LOST: LEADING OTHERS THROUGH SECURE TRAILS

André Gonçalves da Silva

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

Trabalho orientado pelo Prof. Doutor Luís Manuel Pinto da Rocha Afonso Carriço e coorientado por Diogo Homem Marques

2014

Agradecimentos

O presente trabalho e toda a minha formação não teriam sido possíveis sem a presença de pessoas e entidades que considero importantes. Quero começar por agradecer aos meus pais por me possibilitarem o acesso à faculdade e me darem suporte durante todo o meu percurso académico. Agradeço também a todos os professores da FCUL que contribuíram para a minha formação técnica, e por vezes social, através de projetos aliciantes e aulas apresentadas com rigor. A todos os colegas com quem trabalhei na faculdade, mostrando o grande espírito de entreajuda que existe entre os alunos do DI. Quero agradecer especialmente ao Daniel Silva, Nuno Pinto, Adelino Silva, Bruno Neves, Vera Conceição e Sara Aragão pelos bons resultados que obtivemos ao trabalhar juntos e também por boas conversas e momentos divertidos. Muitos outros colegas também tiveram uma participação importante na minha formação académica, principalmente na concretização de trabalhos de grupo. Todos, de uma forma ou outra, contribuíram para o que sou hoje e para a minha formação técnica. Apesar de não os incluir numa listagem exaustiva, também lhes deixo aqui os meus agradecimentos.

Sobre este trabalho em concreto, começo por agradecer ao professor Luís Carriço pela oportunidade que me possibilitou trabalhar neste projeto e pelas reuniões com partilhas de ideias e novas funcionalidades interessantes para o projeto. Agradeço ao Diogo Marques que me permitiu avançar em etapas mais difíceis do projeto, partilhando o seu conhecimento, experiência e recomendações para fazer um trabalho melhor. Quero também agradecer às pessoas que participaram nos estudos, especialmente ao André Rodrigues, que contribuiu no desenvolvimento de alguns módulos essenciais ao projeto. Sem estas pessoas, seria impossível melhorar o projeto e ter a opinião de possíveis utilizadores do mesmo. Por fim, quero agradecer às pessoas com quem trabalhei dentro do LaSIGE, que me auxiliaram com dicas para o meu trabalho e me ajudaram a integrar no posto de trabalho. Estendo ainda estes agradecimentos ao LaSIGE como grupo e à ADMIN por me fornecerem as condições e ferramentas necessárias para o desenvolvimento deste trabalho.

Resumo

A ocorrência de catástrofes no Haiti, Japão, e mais recentemente nas Filipinas, trouxe a lume situações de isolamento, dificuldade de procura e salvamento, mesmo dispondo das mais modernas tecnologias. Essas tecnologias orientam-se sobretudo para uso pelas equipas de salvamento, ficando as vítimas com um papel passivo durante a situação. Se é verdade que essas tecnologias melhoram o desempenho das tarefas de salvamento, a atuação das vítimas e/ou de equipas de voluntários locais no terreno pode ainda introduzir mais sucesso nessas tarefas, desde que não interfira com as primeiras. Os voluntários costumam ser pessoas muito motivadas para ajudar amigos ou conhecidos que tenham sido afetados pelo desastre, e as vítimas podem ter informação sobre a situação para partilhar. Por isso, existe a possibilidade de colocar voluntários e vítimas a trabalhar em conjunto para auxiliar os salvadores nas operações de salvamento. Desta forma, as vítimas poderiam partilhar informação potencialmente valiosa sobre o local e as condições onde se encontram atualmente, enquanto os voluntários poderiam, de acordo com o seu conhecimento sobre a localidade, encontrar facilmente essas vítimas através dessas pequenas pistas e indicações.

Assim, o projeto Leading Others through Secure Trails (LOST) tem como objetivo o desenvolvimento de um conjunto de ferramentas que auxiliem as vítimas e voluntários em situações pós catástrofe. Estas ferramentas são sobretudo direcionadas para dispositivos modernos, tais como computadores portáteis, *smartphones* ou *tablets*, devido à sua popularidade e cada vez maior presença no quotidiano das pessoas. Mesmo sem acesso a redes infraestruturadas típicas, como redes móveis ou *hotspots* de ligação sem fios púbicos, as vítimas devem ser capazes de pedir ajuda usando mensagens de texto, que podem adicionalmente conter informações sobre o contexto atual, ou no mínimo, ter acesso a um mecanismo que lhes permita ter a sua presença assinalada num mapa para que salvadores, oficiais ou não, possam encontrá-las com mais facilidade. Essas aplicações devem ainda considerar situações de interação fortemente limitadas: os dispositivos que as vítimas transportam consigo têm um nível de bateria limitado e, por vezes, recursos computacionais de baixo nível; a ligação a redes de comunicação públicas pode ser intermitente ou até inexistente; as vítimas podem estar feridas e, por isso, indisponíveis para interagir com os seus dispositivos, etc.

Atualmente, o projeto LOST conta com três ferramentas base para suportar os requisitos mencionados: uma ferramenta de comunicação independente de redes

estruturadas, cuja comunicação depende exclusivamente dos dispositivos que as vítimas transportam consigo; um mapa dinâmico baseado em tecnologias *web* onde é possível ter uma visão geral da situação, ver as vítimas, categorizá-las e obter elementos que possam ajudar salvadores e voluntários a inferir sobre o seu estado, havendo uma forte correlação da situação do desastre com o posicionamento numa mapa real; um mapa melhorado de modo a ser possível de utilizar em operações no terreno, através de dispositivos modernos, nomeadamente *tablets*. Esta última ferramenta é fruto de uma evolução do mapa dinâmico, de modo a ser mais usável em equipamentos Android. Este trabalho tem por objetivo explicar a concretização e o desenvolvimento das duas primeiras ferramentas, LOST-OppNet e LOST-Map.

A ferramenta que promove a comunicação de vítimas para voluntários tem o nome LOST-OppNet. O seu nome deriva de *opportunistic networks* (redes oportunistas), que são extensamente utilizadas nesta ferramenta. Este tipo de redes é adequado para cenários onde os canais de comunicação existentes nesse momento são intermitentes ou estão inoperacionais. A ferramenta permite que os dispositivos das vítimas sejam instruídos para atingir dois objetivos principais: recolher informação diversa sobre a vítima de forma automatizada e independente desta, sempre que possível, uma vez que a vítima pode não estar disponível para interagir com o dispositivo; criar um canal de comunicação razoavelmente estável para que os dados recolhidos das vítimas possam navegar longe o suficiente e assim chegarem a salvadores e voluntários. Concretamente, é esperado que as mensagens geradas pelos dispositivos sejam disseminadas pela rede oportunista, até chegarem a um ponto em que exista ligação à Internet. Consequentemente, estes dados serão enviados para um serviço *online*, onde podem ser visualizados pelos voluntários e armazenados de forma permanente.

Por outro lado, o LOST-Map é uma ferramenta de suporte a voluntários que tem por objetivo obter os dados previamente recolhidos pelos dispositivos das vítimas e transformá-los em informação útil para o seu salvamento. Assim, os voluntários têm acesso a um mapa dinâmico onde, para além de poderem visualizar um mapa atualizado com a geografia da região, também podem ver a localização das vítimas sobre o terreno, assim como quaisquer outras informações que tenham chegado com sucesso ao sistema. Adicionalmente, os voluntários têm acesso a um conjunto de opções integradas no mapa que lhes permitem personalizar a vista sobre a situação, com a finalidade de poder destacar certos grupos de vítimas que possam ser considerados prioritários. Por exemplo, é possível observar o caminho feito por uma vítima ao longo do desastre e assim tentar encontrar outras vítimas, ou apenas observar a sua evolução. É também possível analisar vários elementos recolhidos de forma automática pelos equipamentos das vítimas, e assim tentar inferir sobre o seu estado físico, isto é, se a vítima se

consegue ou não mexer, se tem capacidade para reagir ao dispositivo, etc. Consoante os critérios escolhidos pelos voluntários, é possível definir uma escala de prioridade, análoga a um semáforo, para destacar vítimas que pareçam estar em situação de maior risco.

Em conjunto, estas ferramentas visam proporcionar um sistema de apoio mútuo entre vítimas e voluntários, dando a ambas as partes a oportunidade de participar nas operações de salvamento. Utilizando o LOST-OppNet, as vítimas têm a oportunidade de comunicar textualmente acontecimentos relevantes no terreno de acordo com a sua visão e perceção. No caso da vítima não se encontrar em condições de transmitir informação usando a ferramenta, é ainda assim possível recolher dados importantes que possam levar ao salvamento da mesma, nomeadamente informações sobre a sua localização geográfica. Por sua vez, ao usar o LOST-Map, os voluntários podem ajudar em operações de salvamento de acordo com as instruções de salvadores profissionais, enquanto lhes podem fornecer informações valiosas para o sucesso da operação, recolhidas diretamente da cena de desastre. Com o auxílio da informação presente no mapa, podem não só perceber onde se encontram as vítimas, mas também ter acesso a um conjunto de pistas adicionais que podem ajudar a descobrir melhor o seu paradeiro. Para além de uma visão geral sobre a situação, os voluntários podem ainda personalizar a sua vista, de modo a visualizar a cena de desastre de diferentes perspetivas.

Este trabalho pretende dar a conhecer os passos tomados para a concretização e desenvolvimento das ferramentas supracitadas, assim como mostrar as suas funcionalidades mais relevantes. Concretamente, são explicadas as decisões relativas ao desenvolvimento de *software*, técnicas utilizadas, plataformas de suporte (quando aplicável) escolhidas ao longo do desenvolvimento de cada uma das ferramentas. De seguida, são mostrados alguns casos de uso que mostram as ferramentas em ação, com o propósito de ilustrar o papel de cada uma em determinados contextos. Por fim, foram realizadas avaliações para cada uma das ferramentas, de modo a aferir se estas cumprem os requisitos a que foram destinadas e funcionam de acordo com as expectativas. Os resultados dos estudos indicam que os utilizadores, mesmo sem conhecimento específico de operações de salvamento, são capazes de utilizá-las.

Palavras-chave: Gestão de Desastres, Resposta a Emergências, Cidadãos Salvadores, Redes Não-Estruturadas, Comunicação Ponto-a-Ponto

Abstract

Disasters such as those that happened in Haiti, Japan and more recently in Philippines, often results in instances of isolation, difficulty in rescuing victims, even with the use of currently available technology. Those technologies were designed primarily to aid rescuers, leaving victims with a passive role in their rescue. While it is true that such applications can enhance the performance of rescuing works, with the help motivated volunteers and even victims there are more chances to execute a successful rescue. Victims often have local data that may be useful in their rescue, such as geographical information or health condition status. Then volunteers could find those victims by following these clues.

The Leading Others through Secure Trails (LOST) project is composed by three tools designed to help victims and rescuers in post-catastrophe scenarios: LOST-OppNet, LOST-Map and RescueOppus, being the first two the subject of this work. LOST-OppNet is a tool design to be included in victims' devices, such as smartphones or tablets, and make use of networking capabilities to create a dedicated opportunistic network. This allows victims to establish a communication channel, allowing them to send volunteers text messages, along with other indicators, for instance, their geographical location. On the other hand, LOST-Map is a tool designed for volunteers, allowing them to see the location of the victims over a real-world map. This map also allows the volunteers to personalize their view of the disaster scene, with a set of filters operating on the information received from the victims.

This document describes the engineering process of such tools, their functionalities and the rationale behind the main design decisions. Then, some studies are presented to validate both tools and show typical use cases that victims and volunteers may need in disaster scenarios. Results indicate that both tools are usable, even by people unfamiliar with rescue operations.

Keywords: Disaster Management, Emergency Response, Citizen Rescuers, Unstructured Networks, Peer-to-Peer Communication

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural disasters can have devastating effects. For instance, the recent typhoon Haiyan in Philippines destroyed from 70 to 80% of the local infrastructures[1]. Not only people became in isolated state, away from their relatives, but also without the capability of communicating with them using long-distance communication methods can be affected. This barrier can happen if the infrastructure supporting them is destroyed, leaving cell phones or Internet unavailable. Therefore, people may have difficulties in communicating with others to ask for help or locate their relatives.

This work proposes a project called Leading Others through Secure Trails (LOST). The project comprises three main components. The first component is a communication tool to be used by victims under disaster scenarios. The tool helps the victims in affected areas to make others aware of their presence and to send messages to possible rescuers. It also uses some resources of the devices to passively generate data which can later be used by others to infer about the victims' status. The second component is a visualization tool, allowing volunteers to visualize data generated by victims and helping rescuers to have a general overview of victims in the disaster scene. The data is updated in real-time to allow better rescue planning by showing the most recent victim-generated information and give an overview of tracks. The third component is an enhanced version of the second. It consists on a map to be deployed on Android devices, giving users a more native experience. This work will essentially focus on the development and evaluation of the first two components.

1.1 Motivation

During disasters, people are the key component to a successful rescue. It is possible to have all the best rescuing equipment and still fail to successfully rescue victims. People's knowledge and action are essential to ensure that the rescuing operations are

[1] DailyMail: Rescuers battle to reach site levelled by Typhoon Haiyan while city of 35,000 is 80% underwater – http://www.dailymail.co.uk/news/article-2499851/

carried successfully. There are multiple stakeholders during such scenario. On one hand, rescuers are people specialized in rescuing works and have access to a set of tools to help them. They are highly organized and previously plan their operations to ensure a successful rescue. On the other hand, volunteers are willing to help. They can be victims' relatives, neighbours or even strangers who are in the field and offer to help. Volunteers are often untrained and don't have many resources. However, they are motivated to rescue people and may be capable of unveiling clues to rescuers by finding victims on the field. They can also participate in the rescuing works under supervision of rescuers. Generally, rescuers will not allow untrained people to enter in dangerous zones, meaning that the volunteers' scope may be limited. Also, victims are interested part on being rescued, to be safe and return to their lives. At a first glance, victims are passive elements during a rescue, just waiting to be saved. However, this is not necessarily true. Victims are inside of the disaster scene. They can describe the surroundings, find others and collaborate with them. The problem they find is the lack of a communication channel with rescuers and volunteers to propagate such useful information.

Technology can assist people in communicating with others at high distances. Telephones, cell phones and Internet are just a few examples of long-distance communication tools. Usually, people carry with them a device that allows them to communicate with others, namely cell phones. In fact, according to a mobiThinking study in 2014[2], there are about 780 million subscriptions of mobile cellular services in Europe. The same study also gives an estimate that the percentage of cellular subscriptions per 100 people is 124.7%. This indicates that having a cell phone is very common in Europe and people are likely to carry them daily. However, in very destructive disasters, cell towers may become inoperative.

Smartphones are in heavy expansion. Another study by Gartner[3] points that the worldwide smartphone sales surpassed feature cell phone sales in 2013, accounting for 53.6% of every cell phone sold. While feature cell phones typically offer the essential functions expected in a cell phone, such as making calls or sending messages, smartphones are capable of providing additional functions to the end user. For instance, they are capable to connect to other networks, such as Wi-Fi and Bluetooth, both to public structured networks and point-to-point networks created at the moment. This increases the chance of finding a suitable communication channel, giving victims greater chances of being capable to communicate with people rescuing them. Given that

[2] Global mobile stats 2014 – http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/

[3] Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013 – http://www.gartner.com/newsroom/id/2665715

usually people carry a cell phone with them, and the smartphone usage is increasing, there is an opportunity to develop a tool that is already included in the victim's device, ready to be used when necessary. This is important because it is easily accessible by the victim when needed. Current smartphones also include a large number of sensors, such as accelerometers, proximity sensors, etc. They offer the possibility to gather raw data and explore it to extract potentially useful information.

Such information could be used by rescuers and volunteers to plan a better rescuing operation. Usually, rescue teams use maps to plan their strategy. Although functional, paper maps have several limitations and may contain incorrect information due misunderstanding or human error (Gunawan et al. 2009). This may compromise the whole rescue operation, leading to a waste of resources. Technology can again help to enhance this form of communication. There are services offering digital maps. For instance, Google Maps[4] supports the visualization of real-world maps using standard web technologies. It allows searching for a country or even a street name, calculating routes between multiple user-selected points along many other useful functions. This component is even flexible enough to address other use cases besides real-world maps. For instance, Quartermaester[5] is an online map tool based on Google Maps that offers Game of Thrones fans a complete map representation of the series' fantasy world and their characters travels. For volunteers and rescuers, such map could be useful by having a single view over the disaster scene, always updated and accurate according to data gathered from the one important source: victims who are on the field. Due the versatility of Google Maps, it is possible to create a visualization tool based on geographical locations, built using the standard web technologies. This is useful to provide a standard real-world map with an overlay containing victim locations and associated data gathered by the victims' tool. This also ensures that someone willing to see the location of victims can access the map using a standard web browser, making it accessible by a laptop computer or even a tablet.

The combination of these challenges and opportunities opened multiple proposals in the literature to include Information Technology on rescuing works. This work in particular presents a prototype solution to solve the problems mentioned and give volunteers a little more power to help in rescuing operations.

1.2 Objectives

The main objective of LOST project is to create a set of tools to help rescuers and victims to communicate. These tools are directed for people without previous

[4] Google Maps – http://maps.google.com

[5] Interactive Game of Thrones Map with Spoiler Control – http://quartermaester.info/

knowledge on rescuing, allowing them to increase the chances of a successful rescue operation. Specifically, the tools should be able to help victims, by allowing them to advertise their presence, and to help rescuers, by pinpointing victims on a map and allow understanding if they are safe. They aim to provide support to people during disaster scenarios, by complementing the actual mechanisms to save people, and giving volunteers and rescuers more information to increase the probability of a successful rescue.

1.3 Contributions

This work contributes with a system that allows even inexperienced volunteers to locate victims and infer about their condition based on data gathered automatically. Specifically, the system contains a tool for victims, allowing them to ask for help and send small clues to advertise their presence, and a tool for volunteers, allowing them to follow the clues, identify and locate victims on a local map.

Additionally, two user studies were conducted to evaluate LOST tools and their results published in this work, along with the methodology to concretize them. The first study resulted in a publication to an international conference:

- André Silva, Diogo Marques, Carlos Duarte, Maria Ana Baptista and Luís Carriço, 2014. LOST-Map: a victim sourced rescue map of disaster areas. Accepted for publishing in CRIWG'14.

1.4 Document structure

This document is structured following a bottom-up approach, starting by explaining LOST-OppNet, the network component used by victims' application and then LOST-Map, the volunteers' visualization tool and data aggregator.

- Chapter 2 presents a brief analysis of the state-of-art regarding disaster management using Information Technology, with special attention to mobile and easy-to-deploy tools;
- Chapter 3 briefly explains the LOST project, showing the component integration and making an overview about each component;
- Chapter 4 explains the details of LOST-OppNet, in a bottom-up approach. An introductory context is given to allow the understanding of how the core tool works and the reasons to adopt the idea. Then the software architecture is explained to allow the understanding of concrete implementation details and the reasons that led to the development of a reusable component. To proof that the prototype works, a small proof-of-concept Android application to support victims under a disaster scenario is also presented;

- Chapter 5 explains the development and actual status of LOST-Map, its relation to LOST-OppNet and how the data is passed between these components. These components are presented individually and their relationship explained. A set of useful functionalities is also explained, to unveil the full potential of LOST-Map regarding victims detection on a real-world map.

The following chapters present the studies done to understand the viability of the tools developed and their usage with potential users:

- Chapter 6 presents the LOST-Map validation study from a human-computer interaction perspective, presenting the procedure methods and the results of the study;
- Chapter 7 contains a validation study for LOST-OppNet, using the proof-of-concept Android application in a simulated disaster scenario. The study shows the procedure and results regarding tool usage.

The Chapter 8 concludes the document, summarizing the work that was done and possible future work perspectives for the LOST project and its tools.

Chapter 2

Related work

Victim rescuing using Information Technology (IT) is a focused research topic in the literature. There are already some proposals of systems to help victims in disaster situations and provide support to rescue teams. The following sections are an overview of the current state-of-art research about systems with similar purpose and contributions that were fundamental for building the LOST project.

2.1 Tools for rescuers

Rescuing works have a lot of potential to take advantage of IT. From automating common tasks to providing long-distance communication, IT integration in rescuing operations may benefit rescuers.

Wu et al. (2011) proposed a centralized communication system to provide collaboration between rescuers. The idea is to provide a shared map containing photos taken by rescuers. According to the study, a photo can reduce communication interferences such as misinterpretation or wrong description of events. Photos make possible for rescuers to have a shared view of the incident. A rescuer willing to communicate with others could take a photo of the scenario and then append a textual description of the incident. Eventual follow-ups to allow better understanding of the scenario can be done in a chat window. Command centre can then design a plan based on information gathered by field rescuers. The plan would be visible on the shared map, allowing all rescuers to follow it. All data is stored in a centralized webserver, from which rescuers on the field should connect to receive new information and participate on the public chat window. This contribution provides a simple approach to solve communication problems faced by rescuers in the field, by reducing poorer forms of communication that alone would cause misinterpretations and wrong information to spread. The LOST system tries to simplify the gather of information, making it as automatic as possible.

COORDINATORS (Wagner et al. 2004) is another system that supports rescuers' communication, specifically firefighters. It comprises a geo-localization module and a reasoning module. The first allows detection and deployment planning of human resources on the emergency scenario, in order to manage the firefighters. Each point on the map is a team, with an associated name, such as "Team 1". These teams can be coordinated by a mobile command centre using the same application. The incident commander can create a new task on the reasoning engine to be sent to the firefighters, according to the initial knowledge of the situation. After that, firefighters could classify the fire as high risk and then require another task to properly deal with the problems faced. The idea of the reasoning engine is to offer solutions for the current situation as information is being gathered from the scene, allowing the management of actual human and material resources present in the scenario. The communication can be done via ad-hoc networks, minimizing the dependence of a structured network to support the communication. COORDINATORS solves the issue of long-distance communication and coordination of rescue teams, namely firefighters. LOST is a system which splits its efforts for both victims and volunteers. It does not have dedicated features for rescuers, for instance, a coordination component. However, the core concept or distance communication using ad-hoc networks is heavily used in the victims' component.

On a similar approach, a system called WIISARD (Chipara et al. 2012). It consists on a set of tools to be used by rescue teams. Teams that on the disaster field start by identifying victims using a smartphone application, filling some basic information, such as name or the location where the victim was found. Triage teams receive this information and use it in their procedures. Internally, the network used by WIISARD is highly dynamic, without a centralized point where the points connect to. Instead, the nodes create a temporary mesh to connect with others. The information gathered by the team in the field can be obtained when the element reaches the location of the triage team (command centre or cold zone), or can be relayed among the rescue elements via wireless, until it reaches the triage centre. Triage team has equipment with greater computational power, such as laptop computers, in order to process the high amount of information received quickly. LOST has similar approach to WIISARD, however LOST is more oriented to unknown environments. With LOST, it is possible to search for victims by following the clues that they leave. WIISARD seems to be more information-gathering oriented, by collecting information quickly as victims arrive to the triage centre. While WIISARD can be deployed in a small area with the help of a mesh network, LOST can run fully decentralized, using a store-carry-forward model. That means nodes can be disconnected at some point to connect to others that are more far away, and cannot connect to the rescuers network directly.

Another proposal of an emergency medical response system is presented in Hashmi et al. (2005). This system in particular relies on sensors that are deployed with the victims when rescuers on field are rescuing them. These sensors are motes equipped with a pulse oximetry sensor, a GPS receiver, micro-processor, and a module for storing data and transmitting it. This partly automates the victims' monitoring by sending data directly to the local command centre. Rescuers are equipped with a Personal Digital Assistant (PDA) to have feedback about victims on the field by connecting to the local command centre and exchange the most recent situational awareness data. The connection to the local command centre is established through a webservice. Connection from local command centre to the Internet is established via satellite or cellular links, thus enabling information to be sent to remote command centres. This system allows rescuers to leverage their monitoring tasks and situation update efforts by relying them to technology. With the PDA, rescuers can evaluate victims' status with small effort and obtain the most recent status regarding the overall situation. Motes are cheap and low-resource devices that have enough flexibility to gather and disseminate information, being ideal for scenarios where a large number of equipment may be necessary. LOST has the advantage of having access to the information to the victims. That is, victims' devices running LOST are already prepared to announce information to volunteers. While victims who are isolated still need to exchange their information with volunteers when they are in range, their devices still announce their presence. This means volunteers do not necessarily need to have visual contact with the victims to find them.

2.2 Victim collaboration

Victim collaboration is sometimes decisive in their own rescue. Their proactivity allows forming small groups and promoting safety. In fact, it was observed by Vieweg et al. (2010) that during Oklahoma Grassfires of April 2009 and Red River Floods during March and April of same year, affected people used social networks to communicate. Example types of communication were about the environment status, geolocation of victims and exit routes found. An interesting fact about geolocation is the high percentage of users using it: 78% of victims of Oklahoma Grassfires and 86% of victims on Red Rived Floods used some sort of geolocation data in their text messages, such as city name or the street address. This shows that victims often want to provide their own location, or even a location for critical events happening near them. Location sharing may have multiple purposes, such as keeping victims away from a dangerous zone, defining safe exit paths or advertising their presence to possible rescuers. Another interesting fact is the possibility of applying a categorization scheme to the conversations taken during a disaster. While smaller categories are often dependent of

disaster types, it is still possible to successfully apply generic categories such as "Warning" or "Road Conditions" to message and then filter messages according a certain criteria. Qu et al. (2011) conducted a similar study during the Yushu Earthquake in 2010. Messages from an online social network were analysed and successfully assigned to a limited set of generic categories. People often advertised their need for help or advertised their availability to help others on a certain role, such as doctors or translators. These messages were also used to build a local lost-and-found directory of people. With LOST, victims have the ability of communicate with volunteers, using text messages. While the messages are not categorized within the system, some of them are replaced by information gathered automatically. For instance, geographical location is automatically attached to victims' messages when possible. This helps victims to advertise their presence, without using descriptive text, which could be problematic to interpret, as seen before.

Another system where victims can have a proactive role on rescuing is TravelThrough, presented by Gunawan et al. (2012). It is a smartphone application aimed to provide communication and collaboration between victims and rescuers. With TravelThrough, victims can help rescuers in the creation of a safe path to exit an affected area. While on the field, victims can report roads blocked or obstructed, by filling a report which can contain text or pictures taken with the smartphone. The application allows victims to see a map shared with the rescue team. The rescue team can gather the information given by victims and, once confirmed, make it public and design a path on the map, allowing victims to exit the affected area safely. This system demonstrates a close cooperation between victims and rescuers to create an incident map. However, some disasters could be very destructive, leaving long-distance communication methods unavailable. For large disaster areas, this system may require a stable connection from victims to rescuers. LOST does not rely in such communication scheme. For instance, LOST uses unstructured networks to gather the information and spread it as it becomes available. This allows volunteers to split in small groups and search for victims in different areas within the disaster scene. Rather than creating a global view of the incident, volunteers can direct their efforts to a smaller area, and allow some proactivity during the rescue.

Typical smartphone features may also be explored in order to promote means of rescuing victims. Al-akkad et al. (2014) developed Help Beacons, a project that explores the use of Wi-Fi Service Set Identifier (SSID) to exchange messages between victims and first responders. Essentially, there are two Android applications used: a Victim App, used by victims to advertise their need for help, by writing a custom message or choosing an existing one; a Responder App that allows first responders to see nearby victim-generated SSIDs and connect to them, in order to exchange

information. The SSIDs are structured into a special format to allow the Responder App from distinguish between regular access points from the ones generated by victims. This contribution shows that nowadays devices and already existing technology can be widely explored to extend beyond their typical purpose. SSIDs are part of the Wi-Fi standard and thus available in all devices supporting Wi-Fi. This allows victims to exchange information between them, promoting their safety as a group with the help of devices that they probably are already carrying. The strong point of this system is the low need of dedicated resources. However, creating a new message (that is changing the SSID) implies losing the previous one. LOST promotes retaining as much information as possible. With past and current information, it may be possible to establish a route and try to guess the next victims' steps, if no further information is given from them. Moreover, these messages are propagated in LOST from one victim to other. This means that a victim does not necessarily need to be in range of the volunteer to be discovered, assuming that the victim was previously in contact with others, and the respective information was propagated later.

2.3 Unstructured networks

Unstructured networks are also frequently cited in the literature. They represent an alternative communication model in situations where deploying a structured network would not be feasible. Typical scenarios are disaster areas, where the physical structures may not have the necessary conditions to host a structured network. Another concern is time: giving the fact that people may need help, deploying a structured network could take a long time to conclude. Since unstructured networks are usually easy to deploy and are often self-organized, they offer the possibility to create a dedicated communication channel with less effort.

There are several proposals of unstructured networks aiming for communication with others. Belblidia et al. (2011) presents PACS as a network application that uses an opportunistic network scheme. The objective is to disseminate a large amount of data, such as High Definition videos or simply large files, over a network. Due the unstructured nature of opportunistic networks and the possible low processing and storage capacity of nodes, distributing a single large file over the network and expecting that it reaches the destination may not be feasible in every scenario. Assuming that nodes have low bandwidth capabilities, this may lead to an unsuccessful file transfer. Also, this prevents the creation of a reliable and stable communication channel. The concept of PACS is to split the file in different pieces, so they can be distributed over the network. Results presented by PACS research suggest that the order of content dissemination is also important. For instance, the order of propagation may have different results on content availability. Propagation of sequential pieces of data may

lead to an incomplete file over the network. These results suggest that opportunistic networks are highly adaptable to the environments, but have some degree of unreliability. It is essential to ensure that important information is retained by nodes in the network and spread to other in order to make it highly available. Sammarco et al. (2012) presented a prototype named PePit that implements PACS as an Android mobile application to allow the dissemination of large files using a peer-to-peer network. PePit allows sending parts of a picture to an ad-hoc network, and receivers can exchange those parts in order to get the full picture. The application also contains a graphical interface, where a user can see the progress, how many parts the device received so far, and a preview of the image, when all parts are present on the device. This study helps to understand that while opportunistic networks are useful to propagate data, there are some constraints to have in account. For instance, pieces of information can never be propagated in the network. In the case of LOST, messages are constantly propagated until they achieve a secure place where they are stored and available to volunteers. This means that all efforts are centred in the idea of every node in the network having the complete set of the information disseminated. While this requires a large amount of storage to save high volume of information, it allows a single victim to be able to propagate a large part of the victims' network information: a volunteer could find several victims, by only needing to reach to one.

Opportunistic networks are also cited in studies related to emergency communication scenarios. A device-to-device communication system is presented by Nishiyama et al. (2014), aiming to provide a communication channel during disaster scenarios. The network operation scheme is opportunistic, toggling between the use of a Mobile Ad-hoc Network (MANET) approach and a Delay-Tolerant Network (DTN) approach. Changing between these two approaches improves the nodes' availability and dissemination under different scenarios. For instance, when an application running on a smartphone detects movement through the accelerometer sensor, a DTN is used instead of a MANET because while the node is moving, there are less probability of successfully connect to a wireless network. Distance between nodes is increased, decreasing the wireless signal strength, and thus the connection viability. Due this lack of connectivity, the node should store the current data and try to send it to other nodes nearby which it can connect to. This technique is called store-carry-forward. If the node is likely to be stationary, and have a reasonable amount of power, such as battery, then its operation enters in MANET mode, actively broadcasting its presence and sending data to the nearest nodes. This combination of techniques is useful to increase the reliability of communication channel. Instead of relying in a single connection method, toggling between multiple methods helps the technology to adapt itself to the environment, achieving a higher degree of connectivity and data dissemination.

Another opportunistic network allowing victims of a disaster to create a connection with others was proposed by Ramesh et al. (2012). The idea is to disseminate victims' information which may lead to their rescue. These connections rely on Bluetooth in order to propagate the messages. Victims can send a personalized message, along with a phone number, an identification string and geolocation data. The messages are propagated over the network, and when reaching a node having cellular connection to a mobile network provider, they are sent to destinations, such as an emergency command centre or even relatives who are outside the disaster.

Opportunistic networks are fundamental to establish a communication channel where physical barriers and distance between victims and volunteers may be present. While being independent of a well-known structure, opportunistic networks are also capable of adapting themselves to external conditions, such as movement of number of neighbours, in order to achieve the most efficient communication possible, attending to the resources available in the node. This kind of networks is the core concept of the LOST communication from victims to rescuers. It does not require a dedicated infrastructure, making it possible to deploy in many disaster scenarios.

2.4 Interoperability

Using a common technology inside a project is an approach to promote higher compatibility within different internal systems or modules. However, this scenario is not always feasible. For instance, systems with a high level of heterogeneity may not be directly compatible with each other. Nowadays, these systems are very common. From consumer electronics to enterprise-level solutions, heterogeneous systems are a reality. In spite of efforts to create standards that promote compatibility between heterogeneous devices, sometimes an additional platform to promote communication is needed. These platforms are called middlewares. One specialized type of middleware to web applications is the webservice. Webservices allows clients using Hypertext Transfer Protocol (HTTP) and its derivatives to exchange information and call methods on remote platforms in order to get results for more complex operations.

The use of webservices may be deployed in emergency scenarios in order to transfer the need of computational power from the devices on the disaster to dedicated devices located somewhere with adequate conditions. For instance, the emergency medical response system based on sensors presented in section 2.1 by Hashmi et al. (2005) uses a webservice to allow communication between the rescuers' PDAs and the local command centre. By using a webservice, the system can ensure compatibility with heterogeneous devices and promotes data openness, for integration with external

systems. This example demonstrates that webservices are able to couple with heterogeneity and allow future systems to be integrated with the current one easily.

LOST system takes advantage of webservices to create a connection between tools with different technologies. This allows the tools to exchange data in a common format and simplify the retrieval and storage of data.

2.5 Summary

This chapter offers an overview of the state-of-art about victim rescuing using information technology. Some effort was already done and some lessons may be learnt from these studies. One of these lessons is that infrastructure may fail. In greater disasters, the structure supporting communications may become unavailable. A scenario like this would require an alternative communication channel to be established on the disaster scene. Opportunistic networks are suitable for this kind of scenarios, due their great tolerance to changes in the network elements. For instance, if a node is communicating with other, if by some reason that communication fails, the sender node would simply look for others in order to transmit the message and disseminate it. For this reason, opportunistic networks are widely used in LOST.

On the other hand, there are more stakeholders in a rescuing operation besides official rescuers. Neighbors, relatives of victims or people on the surroundings make possible volunteers to help rescuers finding victims. They are motivated and often know the local area. However, they don't know where victims are. In turn, victims are inside the disaster and may accurately describe the situation. However, there are not able to communicate with rescuers or volunteers. The main goal of LOST is to solve this problem, by establishing a communication channel that allows victims to advertise their presence and allows volunteers to detect those victims using a user-friendly tool.

Chapter 3

The LOST project

This chapter aims to give an overview about the Leading Others through Secure Trails (LOST) project. The project goal is to build a set of tools allowing victims to advertise their location and ask for help, while also aiding volunteers in rescuing these victims, in an after-catastrophe scenario. The tools should be capable of providing useful and appropriate mechanisms allowing victims to advertise their presence and exit the scene safely, and allow volunteers or even official rescuers to detect victims who may still be present on the disaster field.

These tools should, however, consider additional challenges. In previous disaster scenes, such as typhoon Haiyan in Philippines (2013) or the earthquake and tsunami in Japan (2011), the effects were highly destructive. This may lead to infrastructure disruption thus affecting standard communication systems, such as cellular networks or public Wi-Fi. In such scenario, an emergency communication system should be expected to adapt itself to the actual conditions and intelligently change its communication strategy in order to promote the best communication channel possible given the current conditions. Furthermore, these tools should be capable of being used in off-the-shelf devices, such as laptop computers, smartphones or tablets. This also imposes other limitations. Such devices frequently have a low quantity of resources available, meaning that any tool running on these devices should be aware of the battery impact and limited computational power.

The idea is to use the already available technology and introduce it into rescuing operations, thus increasing their success. It should not, in any case, replace rescuing knowledge or interfere with it. On an ideal scenario, volunteers and rescuers should take the best decisions according to their knowledge, while LOST tools give them additional information to support those decisions.

An initial set of tools was developed to implement some of these requirements. Actually, LOST project comprises three tools designed for victims and volunteers. While being standalone applications, they were developed to work together and

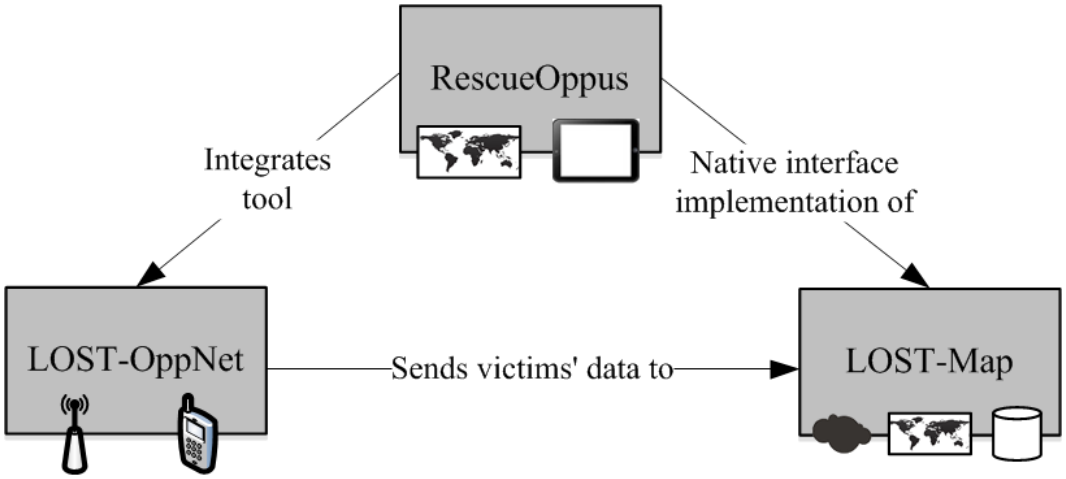


Figure 3.1 – LOST project with the tools developed and their conceptual relationship inside the project.

communicate with others when necessary. Figure 3.1 summarizes the actual LOST project tools and their relationship. Each tool is also briefly described below.

LOST-OppNet. Victims should be equipped with a tool that allows them to advertise their presence without requiring user intervention. This ensures that victims with a smartphone are still capable of telling others the incident location regardless their condition. LOST-OppNet promotes an alternative network environment based on opportunistic networks to provide a communication channel along an intermittent or inexistent infrastructure. Chapter 4 thoroughly presents the LOST-OppNet tool.

LOST-Map. Maps help volunteers to understand the zone and the disaster impact. However, static maps do not allow further investigation about people present in the scene. LOST-Map is a dynamic map that allows viewing a real-world map with an additional overlay containing information about victims. LOST-Map also comprises a persistent storage mechanism to safely store victims' data and allow its exploration. Chapter 5 explains how LOST-Map works and shows features that make it useful to rescuing operations.

RescueOppus. An interface implementation of LOST-Map was created to be included in a native Android application, exploring the full potential of these devices, namely tablets. It contains part of the features offered by LOST-Map interface and other enhancements. It was externally developed by other members of the LOST project and included in the diagram for completeness.

LOST-OppNet and LOST-Map are the focus of this work. They are described in detail and their integration explained. Additionally, studies were conducted to assess their effectiveness in simulated disaster scenarios.

Chapter 4

LOST-OppNet: Knowing the victims

This chapter presents in detail the LOST-OppNet, a reusable software component to allow the detection of victims by gathering data that may be useful for inferring about their location and condition. This component creates a temporary wireless network connection to share data with other victims who also may be running an application based on LOST-OppNet. An example of an application designed for victims using this component will also be presented as a proof-of-concept in this chapter.

4.1 Background

Wireless networks are highways to several types of communication systems. Examples of these systems are radio, Wi-Fi, cellular connections, among others. Modern systems allow not only typical voice communication, but also the exchange of digital information at high speeds. These networks are usually physically structured according to previously planned deployment of access points in order to maximize the signal availability to the desired area. While being reliable for everyday use, structured networks are often unsuitable to use during disasters. For instance, natural disasters may destroy towers that physically support the wireless network access points. This leads to a failure in the structured network, thus becoming unreliable or even unusable.

On the other hand, unstructured networks don't rely on a default deployment and are more flexible for environment changes. For example, an unstructured network may be deployed with the objective of being adaptable to a disaster scenario. Instead of having wireless access points in well-known locations, a set of devices capable of providing an ad-hoc network for a small range could be distributed among the affected area, thus offering a reasonable information exchange mechanism.

Opportunistic networks are a specialized type of unstructured networks. Besides the unstructured nature, nodes on these types of network usually adapt themselves to the environment conditions, and choose their connections opportunistically, according to certain criteria. For instance, a node could intelligently connect to its neighbours when

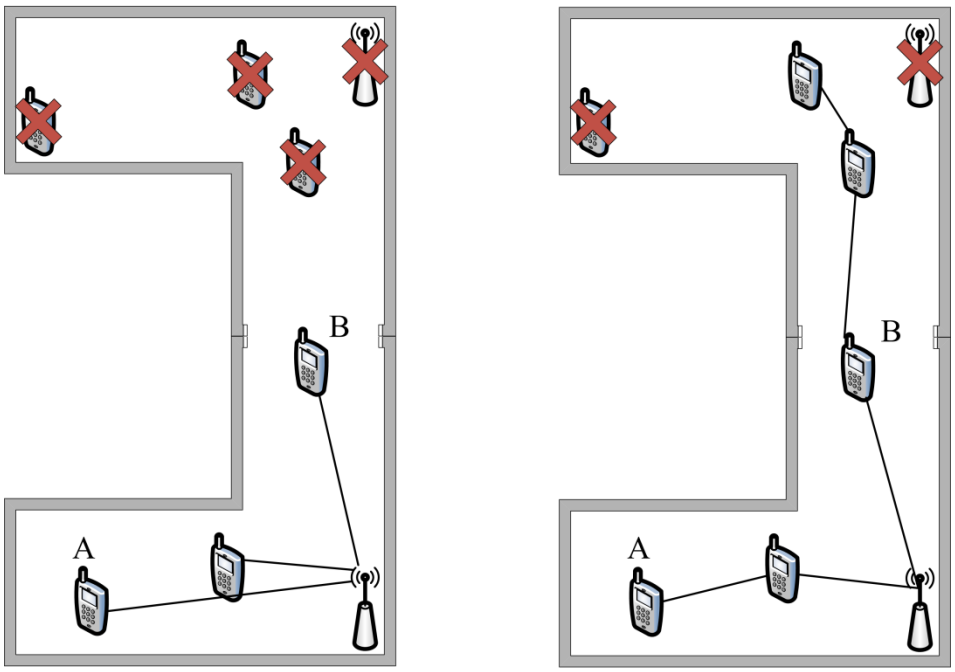


Figure 4.1 – Two sample scenarios showing the possible behaviour of a structured network (at left), and an unstructured network (at right).

the access point connection is unavailable. Figure 4.1 shows a scenario where this actually happens. On the left, a structured Wi-Fi network was deployed in a building floor, with two access points, one at north and other at south, delivering optimal wireless range. During a disaster, the north access point gets broken due physical damage on the wall supporting it. Nearby nodes are disconnected from the network because they are out of range. The south access point is still working and providing network connection to nodes in range. On right, the exactly same scenario for the structured Wi-Fi network happens, with the broken north access point and the still functional south access point. However, nodes opportunistically choose their routes. For instance, node A prefers to connect to its neighbour instead of the access point because of the higher wireless signal strength offered the adjacent node. Node B extends the network to the north side of the building, allowing nodes to connect to it. The role of Node B is to act as a bridge between the south access point and the nodes on the north.

The use of an opportunistic network may bring some of the already described advantages. There are research efforts made towards opportunistic networks and scenarios where they can complement or even replace structured networks. WiFi-Opp (Trifunovic et al. 2011) is a setup relying on opportunistic networks. It consists on a network model that actively changes the strategy to establish a communication channel with other nodes. For instance, WiFi-Opp relies on hotspot functionality present in

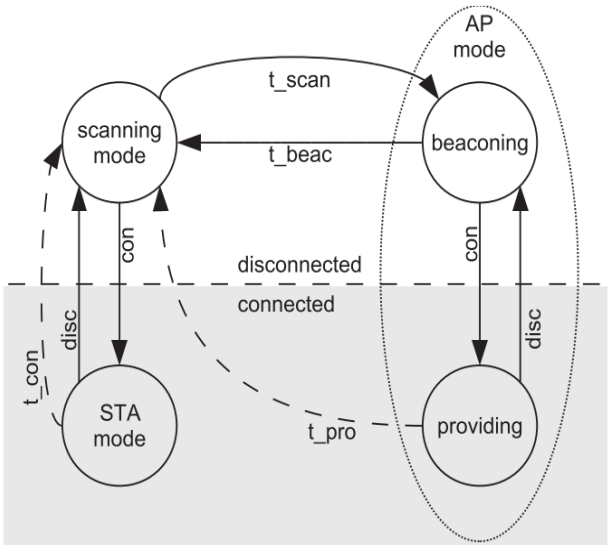


Figure 4.2 – Transition state diagram for WiFi-Opp (Credits: Trifunovic et al. 2011).

nowadays smartphones to allow the creation of a communication channel between devices. The proposal suggests the implementation of four states to achieve the communication: Scanning, Beaconing, Station and Providing.

Each of these states has a specific role. Scanning mode is responsible for finding suitable hotspots for connecting to. Nodes in this mode should scan the surroundings for available hotspots in order to connect and be able to exchange information with others. Beaconing mode is the state that allows nodes in Scanning mode to connect, that is, these nodes are acting as hotspots. It allows multiple clients to connect, thus allowing devices to communicate. When a node in Scanning mode connects successfully to a node in Beaconing mode, the first node changes to Station mode while the second enters in Providing mode. Nodes in Station mode are connected to a network and have access to its available resources. Providing mode indicates that a node is offering network communication to other nodes connected to it. Transitions are usually done via timers with a small distortion in order to avoid similar transition times. Figure 4.2 summarizes all the states and respective transitions present in WiFi-Opp.

The work also presents a study confronting a static WiFi-Opp method and set of flexible methods. In static method, nodes in Scanning mode search for nodes in Beaconing mode as usual. However, when a Beaconing node enters in Providing mode it only changes to Scanning mode if it has no clients connected. Clients remain connected once in Station mode. Flexible methods are similar, but they allow nodes in Station or Providing modes to disconnect from each other. This allows nodes in Station mode to disconnect from the actual hotspot and search for another, while also allowing nodes in Providing mode to disconnect its clients and enter in Scanning mode to search for other hotspots. Flexible methods may allow more expansion for network. While in

static method nodes may form small clusters with the same nearby nodes, flexible methods may allow nodes to connect to other hotspots, making possible the exchange of information with nodes besides their neighbourhood.

Studies regarding the optimal amount of time to stay in a given mode were also conducted. These studies confront the WiFi-Opp approach with the Ad-hoc wireless mode in terms of content dissemination. The first conclusion is that network performance is almost independent of the time amount a node remains in Scanning mode. This allows nodes to actively scan for hotspots without compromising the network significantly. On the other hand, according to the second conclusion, increasing the time a node stays in Beaconing mode has negative impact to the network performance in terms of dissemination. The reason why it happens has to do with the number of hotspots available near each other. Recalling that hotspots cannot communicate directly with other hotspots, this would lead to an area without a chance of creating communication channels if beaconing time is too long.

Battery consumption was also subject of study. Because this communication strategy may be deployed in emergency scenarios, victims carrying smartphones may desire that the equipment have a reasonable amount of energy in order to make a phone call or send a text message if cellular network is available again. This is also true for the opportunistic network. If a critical node runs out of battery, it may dictate the whole network dissemination efficiency. Three HTC Nexus One[6] devices running Android 2.3.4 were used to understand power consumption. Conclusions were once again dependent on amount of time each node spends on a given mode. Scanning time was concluded to be the less battery consuming state. Due this and the previous conclusions regarding network performance being independent of Scanning time, it is possible to perceive that a long Scanning time can have positive effect on the opportunistic network. Nodes can scan for hotspots during long periods of time and then having more chances of connecting to a working hotspot. On the other hand, increasing Beaconing time leads to more battery consumption. As increasing Beaconing time does not increase performance gains on network, it can be concluded that Beaconing time should be kept to a minimum to avoid additional battery waste while promoting communication for nearby nodes.

WiFi-Opp is an important research effort. It shows that opportunistic networks are feasible in nowadays devices and more importantly, their functions can be integrated for real-world needs. LOST-OppNet takes advantage of the opportunistic networks

[6] HTC Nexus One Specifications – http:// phonearena.com/phones/HTC-Nexus-One_id4512

behaviour by using a customized WiFi-Opp implementation in order to provide message exchange with devices in an opportunistic network created with victims' smartphones.

4.2 Implementation

LOST-OppNet is a customized implementation of WiFi-Opp presented in the previous section. Besides the opportunistic network component and state transition proposed by WiFi-Opp, LOST-OppNet extends that work to include features that may help to detect victims and allow inferring about their condition. This includes a set of sensors that allow data gathering and a suitable message exchange format.

This section presents in detail the development start point of LOST-OppNet, extensions made to the original WiFi-Opp implementation, the actual software architecture and a description of features that make LOST-OppNet suitable for rescue operations.

4.2.1 Start point and expansion

The start point for LOST-OppNet was based on an already existing prototype of WiFi-Opp. Oppus[7] implemented the four machine states proposed by WiFi-Opp and allowed basic communication with nodes. When two nodes were successfully connected, they received a basic hardcoded string, such as "I'm alive". Oppus also have a good integration with Android devices, allowing to setup the timers for transitions between states, such as Scanning, Beaconing, Station or Providing. A simple graphical user interface was also developed in Oppus to allow the comprehension of the actual network status, machine state transitions and messages exchanged.

While being a suitable prototype to present a proof-of-concept regarding the WiFi-Opp implementation, it had some barriers that prevented its immediate use. The first barrier was related to the message format. The hardcoded string was appropriate to show that the prototype works. However, it was non-customizable message, serving for the solely purpose of allowing devices to say "I'm here" without any useful additional information. Another problem in the implementation of the machine state mechanism was detected. The implementation was problematic due a small mistake on the software design. Machine state transitions were called recursively. Because the machine state was continuously looping through states *ad aeternum*, this led to excessive Java recursive calls and premature execution termination. This behaviour was only seen when the prototype was executing during a couple of hours using 30 seconds to each state. On the other hand, Oppus was thoroughly documented and implemented using an

[7] Oppus on Github – https://github.com/diogomarques/oppus/tree/ee9be9b20f

interface-based programming approach. These characteristics were especially useful when new transitions states were developed and integrated. Other components were also extended easily using the same approach.

In LOST-OppNet, the mentioned problems were corrected and new features integrated. The actual implementation was developed using a similar architecture. The message format was changed to a semantically richer one, allowing carrying more useful data between nodes. Messages are now disseminated using a store-carry-forward mechanism, in order to store the messages they receive, and disseminate them when establishing new connections with other nodes later. State transition was also corrected and tested to ensure correct operation for at least a couple of hours, while theoretically being capable of running until device runs out of battery. Moreover, a range of new features was integrated in LOST-OppNet: a couple of new state transitions were adapted to allow integration with the an external webservice; a set of sensors was developed and integrated in the actual implementation to allow getting the geolocation of the device and allow inferring about the surrounding environment; a new message management model was developed to allow more controlled message exchange, preventing duplicate dissemination and storing data received persistently; a couple of logging tools to allow the gathering of statistics about state transitions and messages exchange for future review. LOST-OppNet was also ported to an Android service, in order to allow background execution of the app without requiring user intervention.

To develop LOST-OppNet, the Android Developer Tools (ADT) software suite was used, with conjunction of four Android smartphones, being three of them Samsung Galaxy Mini and the other a Samsung Galaxy Ace, all running Android 2.2.

4.2.2 State Machine

The State Machine suggested in WiFi-Opp was augmented to include communication with a webservice, allowing messages from victims to be persistently stored out of the opportunistic network. This expansion was needed in order to push data from victims directly to an aggregator, making it available to possible rescuers. The actual implementation of LOST-OppNet has five conceptual machine states. Four of these states are from the actual implementation of WiFi-Opp, already included in Oppus and only were changed to include minor improvements and integration with new features. The newer state was created in conjunction with the makers of RescueOppus. The purpose of this state is to check if there is an Internet connection available, and if it is the case, send all the information gathered so far by the device to the webservice.

Figure 4.3 shows a simplified diagram with the transitions between states and the required criteria to allow the change. By default, every node starts in the Scanning state.

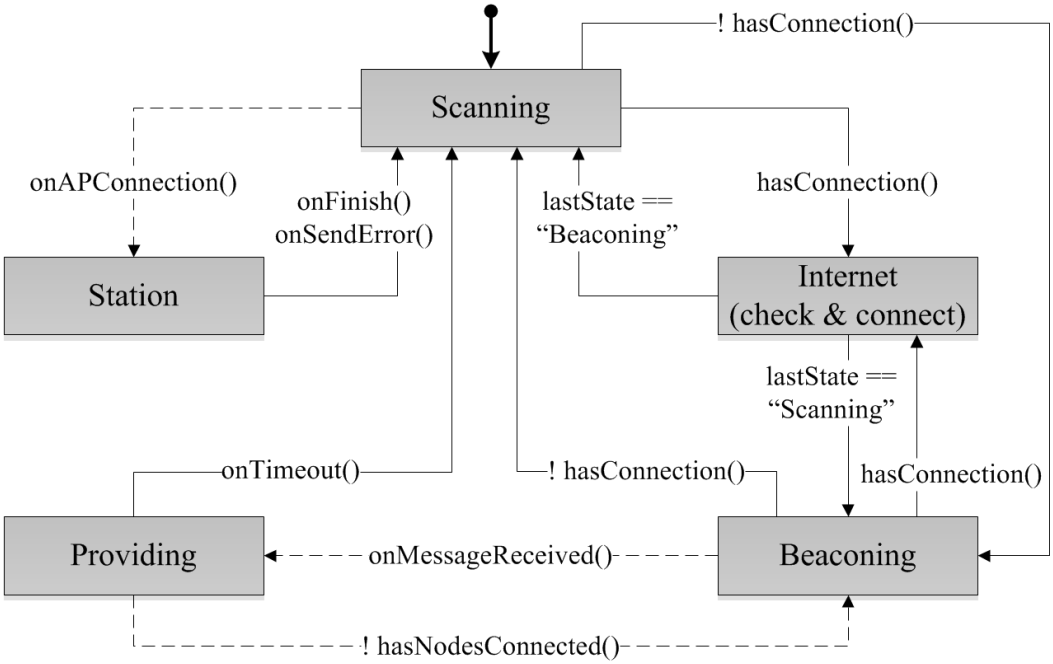


Figure 4.3 – LOST-OppNet state transition diagram. Full lines represent default transitions due timeout while dashed lines represent transitions due external events.

Each transition is one-way only and made when all conditions apply. Some transitions are always made at a certain timeout attribute of each state. Full lines represent the default transition when timeout is reached, while dashed lines represent possible premature transitions due changes in a state. Premature transitions are common in states that wait for response of an external source. For example, when a node in Scanning state finds an access point to connect, it does not need to wait for its timeout, and can change to Station state on successful connection to other node.

4.2.3 Software architecture

LOST-OppNet is an Android application based on the Oppus architecture. Due its modular architecture, it is possible to split the software into several small components. Oppus includes a centralized component called Environment. This component is responsible for initializing the remaining components and accesses their functions. Each of these components may also have subcomponents, responsible for a small and well-known task. Figure 4.4 summarizes the actual LOST-OppNet architecture. Each of the components is thoroughly described below.

State Machine. The State Machine allows transitions between the different states proposed by WiFi-Opp and the additional states introduced by LOST-OppNet. Each of the states has a specific role. The transition from one state to another is only done when the state meets certain criteria or when the timer reaches the time limit defined for the current state. There is also a special state called Stop that stops the execution of the state

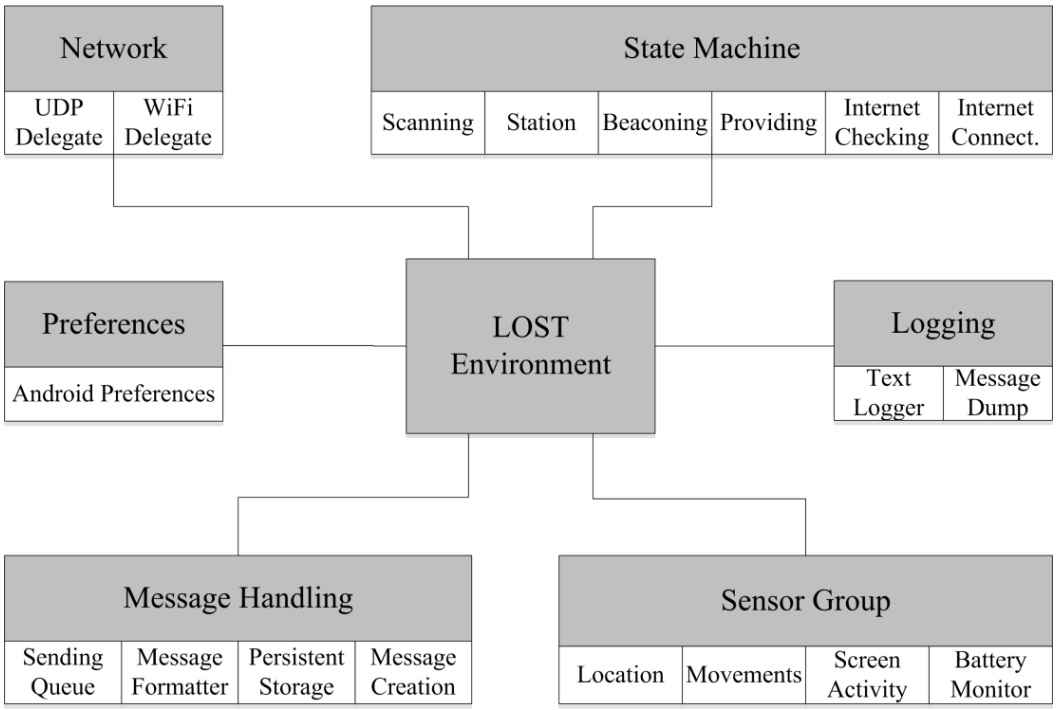


Figure 4.4 – LOST-OppNet modular architecture. Each component contains a set of smaller subcomponents highly coupled to their ancestor.

machine (not present in the Figure 4.3). This state is only reached when explicitly requested. For instance, it could be a button on a graphical interface labelled "Stop machine", allowing LOST-OppNet to stop its execution gracefully, and freeing any used resources, such as any used device sensors.

Logging. Text logs may be useful for debugging or future analysis of a specific run. There are two subcomponents of Logging component: Text Logger is a log written on a human-readable text file inside the Android device, containing information about state transitions, messages received and other useful information that a developer working with LOST-OppNet may find useful to log; Message Dump also stores in a text file all messages received, encoded in a Comma Separated Values format. Message Dump was especially useful for manual data insertion in other tools, such as LOST-Map. In the actual implementation, both of these subcomponents are disabled, although the Logging component is still present and available to developers.

Sensor Group. Android smartphones usually have a variety of sensors available to be used by application developers. LOST-OppNet uses four types of sensors: Location sensor to retrieve the device's current geolocation; Movements to detect when the user is walking or operating with the smartphone; Screen activity to understand if the user is interacting with the smartphone after an idle period; Battery Monitor to watch the battery usage and current charge level. Section 4.2.4 explains in detail the sensor usage and the data gathered.

Message Handling. Messages play a crucial role in the current implementation of LOST-OppNet. With messages, it is possible to exchange information between nodes. For instance, a customized text message sent with a LOST-OppNet powered Android application would carry the message along data gathered automatically from sensors. Multiple subcomponents allow messages to be handled easily: Sending Queue controls the message flux to the network, filtering out duplicate incoming messages and avoiding resending them; Message Formatter allows converting the message to another human readable representation of the data, such as Comma Separated Values or to JavaScript Object Notation (JSON) format to send messages to an external webservice; Persistent Storage components stores the received messages on the device for future analysis or even for recovery from a previous application abnormal termination. Message Creation is not a subcomponent *per se*. Messages are created directly in the Environment with a special Java method that gathers information of all sensors and put them on the message. However, since it plays a crucial role on the Message Handling, it is included on the diagram for completeness.

Preferences. Android has a mechanism that allows storing user preferences on a protected location, making them impossible to edit manually for the average user. Preferences uses this mechanism by implementing a specialized version of Preferences, called Android Preferences. LOST-OppNet uses this mechanism to store user preferences containing application settings, such as timer limits for each state or the external webservice address. Following this approach gives two immediate advantages: Android preferences are usually easy to manage for the final user, through a graphical screen generated from a XML file, without the need to develop additional code to store preference values; the Android preferences can be updated in execution time, allowing changes to be reflected on application behaviour in real-time. For instance, if the webservice address pointed by the application is offline at start up, one could change to a working webservice address at execution time, allowing LOST-OppNet to discover the new address without restarting the application. All preferences are accessible via a configuration screen included with LOST-OppNet.

Network. The Network component did not changed significantly from the Oppus implementation. It comprises two subcomponents, being UDP Delegate used for communication between devices and WiFi Delegate used to establish connections with other nodes and toggling between infrastructure and ad-hoc modes. UDP Delegate suffered minor modifications to accommodate with new features. For instance, UDP Delegate now handles the sending of message streams instead of the hardcoded message present in Oppus. Section 4.2.5 details the format of message streams and how they are sent over the network.

LOST Environment. Environment is the core of LOST-OppNet. It uses all the previous mentioned components in order to control the machine state execution and successfully handle received messages. Environment also allows external apps to be aware of the LOST-OppNet execution state, by maintaining a public key-value table accessible by other Android applications. This table contains various entries related to some of its components. For instance, it is possible to know if the service is currently active and the current machine state. Messages received are also accessible in a similar method. These methods are described in detail in section 4.3, where an example of an external application accessing LOST-OppNet and interacting with LOST Environment is presented.

By using a modular architecture, LOST-OppNet ensures a correct role distribution among several small components. This is especially important, as it facilitates the development of new components and integration with LOST-OppNet, allowing expanding its functionalities.

4.2.4 Automatic data gathering

One of the LOST-OppNet components is the Sensor Group. It manages and allows access to device sensors in order to extract data from them. Android Compatibility Definition Document (Android CDD) [8] is a document where hardware vendors are informed about how they should design parts of their hardware in order to meet the minimum requirements to run Android on their devices. This contains information about compatibility and recommendations about using some features in Android devices. Hardware vendors can check it to be aware of the expected capabilities that devices should or may have in order to run Android.

The Android CDD contains a section with sensor support and recommendation. Table 4.1 summarizes the sensor support recommendations for devices running Android. Some of these sensors are preferred to others. For instance, the recommended sensors are the Accelerometer, Magnetometer, GPS and Gyroscope. Other sensors such as Barometer or Proximity sensors are allowed, but may be absent in mass-market devices. To achieve a compromise of having support to a large range of devices and to allow extracting data that may be useful to infer about device surroundings, LOST-OppNet uses data from the following device sensors:

- GPS, to obtain the device's geolocation;
- Accelerometer, to understand if the device is moving;
- Touchscreen, to know if the user interacted with the device.

[8] Android Compatibility Definition Document – http://source.android.com/compatibility/android-cdd.pdf

Sensor	Recommendation summary
Accelerometer	Device implementations SHOULD include a 3-axis accelerometer.
Magnetometer	Device implementations SHOULD include a 3-axis magnetometer (i.e. compass).
GPS	Device implementations SHOULD include a GPS receiver
Gyroscope	Device implementations SHOULD include a gyroscope (i.e. angular change sensor).
Barometer	Device implementations MAY include a barometer (i.e. ambient air pressure sensor).
Thermometer	Device implementations MAY but SHOULD NOT include a thermometer.
Photometer	Device implementations MAY include a photometer (i.e. ambient light sensor).
Proximity Sensor	Device implementations MAY include a proximity sensor.

Table 4.1 – Android Compatibility Definition Document recommendations regarding sensor support in hardware devices.

With these sensors, it is possible to explore data that may be transformed into useful information for volunteers. GPS can provide geographical coordinates allowing detection of the device location with high precision. This gives the advantage of knowing where a victim is, assuming that the victim is in possession of the respective device. It is also possible to understand if the victim is moving from a place to another, register the trail done or even measure the total distance travelled. On the other hand, accelerometers can be useful to detect movement on a smaller scale than GPS. An accelerometer is capable to report the orientation of the device regarding the Earth's magnetic field and the acceleration of the device, usually due user movement. Information regarding victim movements can thus be obtained from data generated by this sensor. This may allow inferring if the victim is moving at all.

These sensors can work without user intervention. This gives two advantages: first, victims do not need to worry giving an explicit order to the device start gathering

data, since the gathering process is done transparently in the background. This leverages the dependence of LOST-OppNet on the victim, who may be unavailable to interact with the device; second, the accuracy of data is independent of the victim knowledge. For instance, a victim can be uncertain or not aware at all of the current location. Instead of sharing a possibly incorrect location and thus spreading erroneous data to rescuers, LOST-OppNet relies in sensors data, namely GPS, to get an estimate of the victim location automatically.

Data gathered from sensors may not be always accurate or useful. These situations may require some sort of data confidence indicator levels. For instance, when GPS loses the connection with GPS satellites, it may not report any geolocation until it connects again. During this period of time, LOST-OppNet reports the last known location with a flag indicating that the confidence on location is low. An example of high confidence for the location would be when GPS successfully reconnect to GPS satellites and then be able to provide updated coordinates. A low confidence flag allows volunteers analysing the data to understand that particular geolocation data may be treated as a best effort for providing a valid geolocation and may not exactly represent the place where the victim is.

4.2.5 Network messages

Data would be valueless if there was no channel to share it. Being possibly distant from rescuers, victims would like to tell others about their condition. Message exchange between devices is thus an important part of LOST-OppNet.

In Oppus, the message exchange was done when the node is in Station state. While being connected to a node in Providing state, it prepares an UDP datagram containing a Java string, followed by a termination character, namely the End-Of-Transmission character (ASCII: 0x04). The datagram is re-sent periodically until the Station state timeout is reached, in order to increase the probability of the message reaching destination. While being sufficient to share a simple text message, this approach has problems under certain requirements. Text messages are often unstructured. Although they are flexible enough to represent other formats, an agreement must be done in order to ensure that the same format is being used in both parts of communication channel. In spite of existing formats, such as JSON, these may be more useful when using a communication between different platforms.

LOST-OppNet uses the same communication scheme present in Oppus. A connection channel is established when a node is in Station state. Then, an UDP datagram is created to send the message. The difference resides in the message format. The actual implementation uses a Java class called Message to represent a message.

This class is suitable to be exchanged in the network environments. It was implemented based on the following requirements:

- Every message has an origin identification, a time of creation and a fixed set of sensor values;
- Some messages may contain an optional text message;
- Two messages should be considered equal if they are created by the same node, at the same time.

Message class contains a set of private fields that allow saving the values as specified by the requirements. Furthermore, it is possible to compare two messages directly in order to verify if they are the same message. This requirement was implemented by overriding the native's Java Object equals and hashcode methods. For instance, the nodes should avoid duplicates when saving messages, in order to prevent duplicates dissemination. Figure 4.5 contains the actual code that is responsible for filtering out duplicates messages. The technique consists in using a Java Set containing hashcodes from all messages already stored in the LOST-OppNet. When a message is received, it is added to the sending queue ("mQueue" property) and persistently stored on the device ("storeMessage" method). However, only new messages are stored and forwarded. A set containing hashcode of every message received ("duplicates" property) is used to verify if a received message was already stored. If there is a match, the message is just ignored, since it is a duplicate. Otherwise, the hashcode of the message is calculated and then stored in the duplicates set, so in case of receiving that message again, it is correctly pointed as a duplicate and not processed.

The Message class also implements the Java Serializable class. This allows the message to be sent over a network, using sockets, and without using additional code to process the message format, given that both sender and receiver are devices running Android. However, unlike Oppus, LOST-OppNet frequently needs to send more than one message at a time. A simple method would be sending all messages sequentially, each in a different datagram. This approach presents some problems. It was observed that creating multiple messages using a dedicated UDP datagram for each message

```
public void pushMessageToQueue(Message m) {
	// duplicate check
	Integer msgHashcode = Integer.valueOf(m.hashCode());
	if( !duplicates.contains(msgHashcode) ){
		mQueue.add(m);
		duplicates.add(msgHashcode);
		storeMessage(m);
	}

}
```

Figure 4.5 – Example of the actual implementation of message duplicate filter.

would put an additional delay in communication. Recalling that messages are re-sent periodically, messages from the first period often overlapped those of the second period, causing problems in the communication channel. To solve this problem, a class called MessageGroup was implemented. It consists on a typical Java List containing instances of Message class and the total of messages transmitted. Because this class also implements Java Serializable, it is used to be sent over the network instead of using multiple instances of Message class. Using this approach, a single UDP datagram is created instead one per message[9], reducing the needed time to prepare the datagram and send the request.

Messages can also be generated automatically or on-demand. Automatically generated messages are created when the machine state changes to the Station state, that is, the node is connected to other. These messages are created to ensure that the current conditions are periodically tracked, including the current geolocation. They require no user intervention as all content can be extracted automatically. On the other hand, on-demand messages are explicated created by victim request. These messages contain a custom text message along automatic gathered data. This means that on-demand messages are essentially automatic messages with a custom text, explicitly generated when a victim requests it.

4.2.6 Internet state

In LOST-OppNet, a new state was developed to provide an additional communication channel. The Internet state allows connection to the Internet, when available. This connection is established to send information gathered by the device and the network to the LOST-Map webservice.

The Internet state has two functions. The first function is to test if an Internet connection is available on the device. If such connection is unavailable, a transition to either Beaconing or Scanning state is made, depending on the state before Internet was Scanning or Beaconing, respectively. This is made in order to ensure the typical behaviour present in WiFi-Opp, that is, the transition to Scanning is made if the Beaconing timeout is reached and vice-versa. On the other hand, if a connection to Internet is available, the data gathered is prepared to be sent to the LOST-Map aggregator.

[9] Due Maximum Transmission Unit (MTU) size, a Message Group may be split into multiple datagrams. However, given that a Message Group can carry multiple messages and still fit into a single UDP datagram, there is still an advantage in using Message Group class as opposed to transmitting a single Message at a time.

The integration with LOST-Map is achieved with a remote webservice that allows the insertion of data. While on Internet state, the device uses the MessageFormatter subcomponent to create an alternative representation understood by both parts. Since the LOST-Map webservice runs on a different technology, the agreed format is JSON. MessageFormatter allows the conversion of a Message to other representations, including JSON. The Messages that are present in the sending queue are converted to JSON objects. Then all of these objects are included into a JSON array, in order to send multiple messages within a single request. When the device successfully establishes a HTTP connection to the webservice, the JSON array is sent and the device waits for a reply. The webservice must reply with an HTTP status code telling if the request was successfully processed. Once the webservice inserts the messages, these are deleted from the sending queue. This is done to reduce network duplicates for data that is already secure on the centralized webservice and to reduce the amount of memory needed by devices to store the Messages that may already be available in the network. These Messages are, however, persistently stored in the device for later manual retrieving.

Internet state is optional. On the LOST-OppNet configuration screen, it is possible to enable or disable the Internet state in real-time, and change the webservice HTTP address. This was done to make LOST-OppNet more flexible. For instance, in a situation where Internet is not accessible at the start, the Internet state can be disabled in order to bypass it. This allows devices to save battery and spend more time on states that may lead to a successful connection with other devices present in the scene, allowing the exchange of local information. On the other hand, if an Internet connection is provided later, this state can be activated without restarting LOST-OppNet, allowing devices to send victim data to the centralized LOST-Map webservice.

4.2.7 Communication with other applications

Integration with other Android applications is also possible. In fact, LOST-OppNet is implemented as an independent Android service. This means that there is no graphical interface dedicated to interact with LOST-OppNet, but instead, a new or existing application can be integrated with the service. However, LOST-OppNet is independent of a graphical interface in order to run. The service could be initiated from a remote authority while still there is Internet connectivity, only needing to implement the communication with such authority.

Communication between LOST-OppNet and an external application is done with an Android Content Provider. Content Providers are similar to white boards, where an application can write data and others access that data easily. Android relies in Content Providers for its own operation. For instance, custom words added by the Android

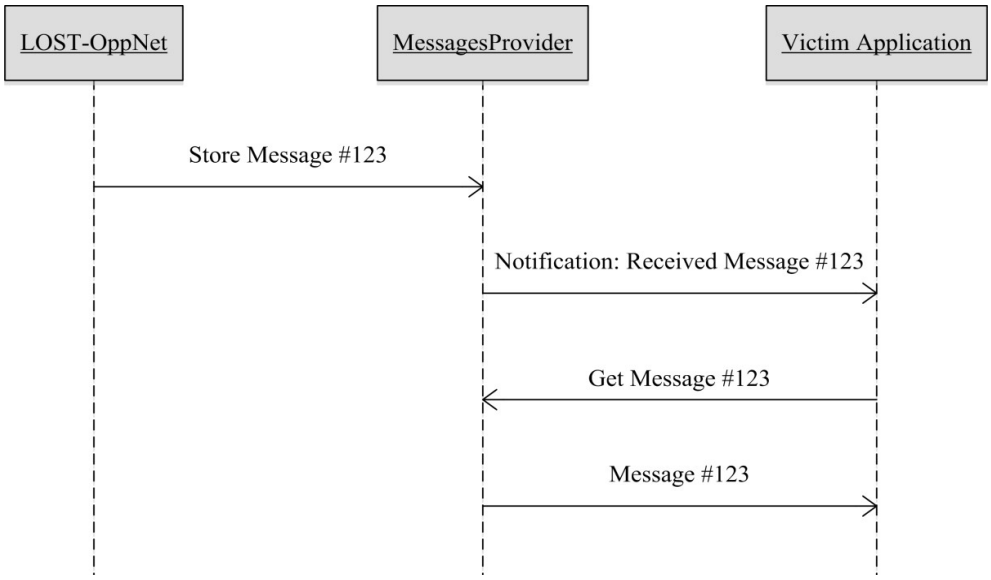


Figure 4.6 – An external application receiving a notification of a new message arriving the system, and fetching it from the MessagesProvider.

device user's dictionary are stored in the UserDictionary Content Provider[10]. Technically, a Content Provider allows other Android applications to access shared data, stored in a single or multiple relational database tables, using SQLite. The typical Create, Read, Update, Delete (CRUD) operations are available on Content Providers, to allow full control of data stored. Each Content Provider is accessible with a Uniform Resource Identifier (URI). This URI can be split in authority and path. The authority part is exclusive to each provider. It consists on the fully qualified name for the Content Provider class. The path is customizable, and is typically the resource name the application wants to access. LOST-OppNet exposes a Content Provider called MessagesProvider for public use. The resources available to other applications include messages stored (received and sent) and contains information about the LOST-OppNet service status. An external application can access this data to interact with the service not only to read data, but also to create new messages with a custom text message. For instance, an external application can create a custom string and append it on the MessagesProvider, to be sent later.

Another interesting mechanism offered by Content Providers in general is the capability of notifying interested parts on data changes in resources, through an Android Content Observer. For instance, the MessagesProvider promotes and relies in the use of this mechanism. When a message arrives to a device running the LOST-OppNet service, the MessagesProvider generates a notification telling that a new message arrived, and includes an identifier allowing observers to directly get the corresponding message.

[10] UserDictionary – http://developer.android.com/reference/android/provider/UserDictionary.html

Figure 4.6 shows an example of an application interested in receiving messages from LOST-OppNet as they arrive. The notifications are also used internally to insert custom messages from other applications in the sending queue. When a new custom text message is submitted, LOST-OppNet is notified of its contents and creates a new message including the custom text. The message is then pushed to the sending queue, and sent when there is a connection. Although these notifications work as explained on devices running at least Android 4.1 (API level 16), a workaround is needed to simulate the same functionality in devices with lower Android versions. This constraint is due the fact of Android not passing the complete notification URI (i.e. including the new message identification at the end of the URI) on these versions. Instead, the application only knows that some data was updated on the resource it was listening for updates. Basic workarounds consist on fetching all messages and getting only the last one, or comparing the messages with the already received ones. Although these methods allow achieving the goal of getting the new message, they require additional workload on the system.

In summary, the MessagesProvider component allows external applications to access data received or created by the LOST-OppNet. By relying in the Android Content Provider, it is possible to allow controlled access from external applications with ease. The Android Content Observer allows applications to be notified in real-time when LOST-OppNet changes it states or receives a new message. This is an important concept, since it allows more expansion for the LOST-OppNet component. Any interested programmer could easily deploy an application to work with LOST-OppNet and expand the functionalities offered by the service, by presenting the data dynamically to the final user. The next section presents an application designed to interact with LOST-OppNet following this approach, exploring the full set of functionalities offered by the service and the MessagesProvider.

4.3 VictimApp: LOST-OppNet in action

The LOST-OppNet also comprises an external application to allow interaction with the service capabilities. This application is a frontend prototype as would be expected to see if LOST-OppNet was deployed on real-life scenarios. The application allows a user to see the LOST-OppNet status and to send a custom message.

The application is called Victim Application or VictimApp for short. The idea of VictimApp is to provide victims a basic and easy-to-use application to see LOST-OppNet working. Recalling LOST-OppNet is an Android Service, it has no graphical interface besides a simple notification on the Android notification centre. This

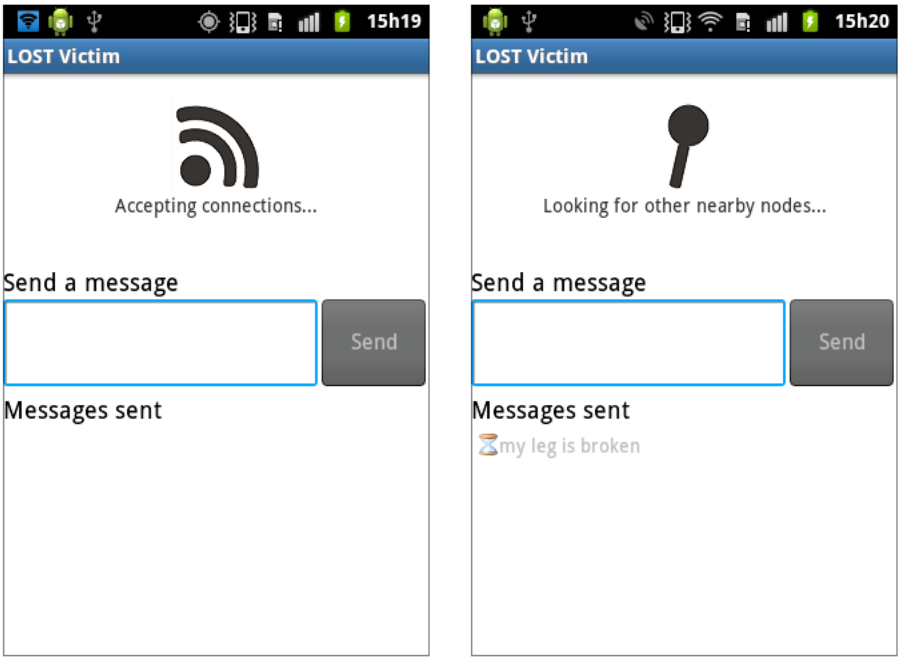


Figure 4.7 – VictimApp user interface. On left, current status of LOST-OppNet is shown. On right, an on-demand message created by the victim is waiting to be sent.

notification alone provides insufficient feedback to the victim. VictimApp provides a simple graphical interface allowing that tries to answer these questions:

- Is LOST-OppNet running?
- What LOST-OppNet is doing?
- What can I do with LOST-OppNet?

The actual interface of VictimApp can be seen on Figure 4.7. It shows the same and only screen in two different states. The screen at left shows the application in Beaconing state, that is, announcing its presence and waiting for connections. The state is identified by a representative icon and a friendly description. The description was opted instead of the actual state name due their technical nature. Friendly descriptions try to describe what is happening with simple language. There is also a text input field allowing victims to send a custom text message. The send button is disabled until the victim types at least two characters in the text input, to avoid accidental empty messages. After typing the desired message and sending it, the text message is added to the messages sent list, as shown on the screen at right. It is also possible to notice that the state changed. The old icon is replaced with the new state's representative icon and the respective friendly description, being Scanning state in the case.

Connection between the application and LOST-OppNet is done with the Content Provider described on section 4.2.7. The VictimApp contains two Content Observers. Each one is responsible for different resources. One of the Content Observers is

responsible for obtaining the most updated LOST-OppNet status. The status includes the current machine state and whether the service is active or inactive. This allows VictimApp to update the graphical interface, namely the status icon and the respective description. The other Content Observer is responsible to listening for messages received and sent. While not showing all the messages sent by LOST-OppNet, namely those automatically generated, receiving updates for all messages sent allows the interface to notify the victim when the on-demand message created by the victim was sent to the network or even directly to the webservice.

Custom text messages have three possible statuses: waiting, sent to network and sent to webservice. The "waiting" status is the default for a new message. This indicates the message was successfully created and is in the sending queue. Its graphical representation is an hourglass in front of the message. The "sent to network" status indicates that the message was successfully sent to other devices. It appears as a green checkmark before the message. However, there is no guarantee that the message was actually delivered. This is due the nature of the UDP protocol. As seen before, there is an effort to make the message reach the destination, by sending it periodically while a connection is open. However, there is no way to confirm that the message was effectively received by other nodes at this time. The last possible message status is "sent to webservice", that indicates the message was successfully sent and received by the LOST-Map webservice. It appears as a cloud icon before the message contents. Unlike the sent to network status, this status ensures that the message was successfully delivered to the webservice. This is possible due the use of HTTP protocol, and the response code that allows knowing that all messages sent were correctly processed.

In summary, VictimApp is a prototype that takes advantage of LOST-OppNet capabilities allowing victims to interact with the service and use its functions. It is also a proof-of-concept to show that an external message can communicate and interact with LOST-OppNet, taking advantage of it reusability. The VictimApp was also integrated into a user study to understand how easy to use the application is, and to understand if it allows people to understand the role of LOST-OppNet status in their rescue. Chapter 7 details this user study, and presents the conclusions reached.

4.4 Summary

This chapter presented LOST-OppNet as a modular and reusable software component. By implementing an opportunistic network based on the WiFi-Opp (Trifunovic et al., 2011), LOST-OppNet can connect to other devices running the service and provide a communication channel to allow automatic data exchange, leveraging the hotspot functionality available in off-the-shelf devices. This data can be

useful to save victims, without depending on them to be gathered. If an Internet connection is available, LOST-OppNet can also send messages directly to a remote webservice in order to make them immediately available to possible rescuers.

VictimApp was also presented as a proof-of-concept prototype to help victims on the field. The application allows victims to understand the current status of LOST-OppNet and to send a custom text message asking for help or giving information in free text form. These messages can be exchanged with other devices in surroundings in order to facilitate the message dissemination.

Important data is generated and gathered using LOST-OppNet. However, data itself does not provide immediate information about victims. Tools are needed to transform it into possibly useful information. The next chapter presents LOST-Map as a set of components to manage and process all data gathered and make it accessible to volunteers or even official rescuers. This way they can analyse the information and make a better planning to rescue the victims.

Chapter 5

LOST-Map: Detecting victims on a dynamic map

When natural disasters occur, people living in proximity are often the first candidates to help rescuing other people. Knowing the region is an advantage that some of the official rescue team members may not have. Even with partial destruction, this knowledge may still be useful. However, destructive effects may lead to disruption of line of sight between victims and rescuers. This may cause several problems: physical barriers can interfere with vision but also with verbal communication, leaving victims in an isolated state.

Additional tools may help finding victims in such scenarios. Maps are tools commonly used for an initial deployment of rescue teams at the start of rescuing works. These maps are usually made of paper, thus static and with low to none forms of interaction. Besides that, paper maps lack the ability to have context. For instance, the map itself only shows a static view of the geographical representation of an area. While it is still possible to draw over the map, research suggests that this may lead to confusion after some update iterations along other problems (Gunawan et al. 2009). A dynamic map in a digital format may help to overcome these limitations.

LOST-Map is a dynamic map using currently existing technology. It was essentially designed for visual exploration, helping volunteers to discover victims needing help on the field. Important decisions such as which victims should be rescued are entirely left to the users. It also allows volunteers to infer about victims' status, giving the possibility to prioritize victims to rescue and leverage the management of rescuing resources they may have available.

This chapter starts by explaining the current technology status that allowed the development of the existing features in LOST-Map. Then concrete implementation details and functionalities are presented and discussed in order to allow the understanding of the tool. The graphical component is also presented from a user's perspective to allow better comprehension of the dynamic map in a rescuing scenario, and to present the options LOST-Maps offers to volunteers.

5.1 Background

Current technology provides exciting features in every branch of Information Technology. Web applications are an example of technology evolution. Recently, the most used web browsers were updated to support new HTML5 technology. A large part of these technologies can disrupt the traditional role of a webpage. For example, the new HTML5 Canvas[11] allows web developers to draw graphics on web pages on-the-fly without the need of previously generating them. On the other hand, Asynchronous JavaScript and XML (AJAX) is a group of technologies also heavily used in modern web applications. For instance, AJAX allows developers to load content dynamically on the background without user intervention. This means that the user does not need to manually refresh the webpage to check for updated content.

Examples of successful web applications include online maps. These applications provide the same functionalities as provided by paper maps with the addition of real-time and advanced exploring functionalities. For instance, with online maps a user could easily change the map scale to see a particular region. Moreover, online maps are frequently updated. Google Maps is an example of an online maps service. It is free to use. With Google Maps, users can explore the world map from bird's eye view to street view, search for directions and get real-time traffic conditions, among other features. This service also provides an Application Programming Interface (API) for web developers needing to integrate the Google Maps functionality on their websites. An API is a well-defined software contract between the service provider and the developer. The API is free to use and allow web developers to use the full-featured online maps according to the business rules for a given project. As seen before, Google Maps API is flexible enough to use even for simulated world maps.

Another specific type of API is a webservice. Webservices are routines or functions provided by web applications in order to allow external applications to interact with a given product in a controlled way. From spell-checkers to plane ticket reserves, webservices can provide a range of useful services to web developers. From a perspective of a service provider, launching a webservice could allow easy integration of existing code with newer systems. Another common use of webservices is the construction of a bridge between systems using different technologies. Their role in these situations is to provide a compatibility layer between the systems to allow the same form of communication. It is then important to establish a common software contract in order to ensure data consistency. There are two major types of webservices. The first type is Representational State Transfer (RESTful). RESTful services use a

[11] Canvas element specification – http://www.w3.org/TR/html5/scripting-1.html#the-canvas-element

simple approach by defining a Uniform Resource Identifier (URI) to distinguish between the methods offered by the service. It also has deep integration with the HTTP application protocol. For instance, if developers would like to use a webservice to store books on an online database, they just need the URI and the data of the book in an agreed format, such as JSON or XML. Then, developers would have to issue a HTTP PUT request to the URI corresponding to the book resource with the data for that particular book. For retrieving the information, a simple HTTP GET request to the same URI is the standard way to get information about the previously inserted book. The other type of webservice is the Simple Object Access Protocol (SOAP). This type is not as coupled as RESTful is to the HTTP protocol. In fact, it is possible to use SOAP over other protocols, such as Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP). The communication type is also different. While in a RESTful service a developer would include the data inside a HTTP request, with SOAP the developer would need to use a XML envelope to include the request and associated data.

Originally, the initial planning for LOST-Map was to develop a tool to allow visualization of nodes exchanging messages in the network. While technically significant and feasible using current technology, the tool would be too difficult to be used by volunteers without previous knowledge in computer networks. The first problem would be a lack of projection on the real world. To be useful, the data should be projected over a real-world map, in order to correlate the line of sight between two devices with the actual paths available on the ground. Other problem would be the technical terms used. Volunteers would have no interest in knowing the rate of messages received by nodes. Instead, they would like to know if victims tried to send any messages and the respective contents.

LOST-Map takes advantage of the technologies mentioned before to provide volunteers a repository of data regarding victims, with the possibility of seeing that data correctly aggregated on a dynamic real-world map. On the frontend, LOST-Map uses Google Maps API to provide an online map to volunteers. Due the flexibility of this API, it is also possible to include information related with the victims over the map. On the background, an aggregator contains a database that stores and provides access to victims' data. The access to data is done via a RESTful webservice also included in the aggregator, allowing controlled read and write of information by external applications, such as LOST-OppNet.

The LOST-Map was implemented with several technologies. The frontend was developed in HTML5, CSS and JavaScript. The aggregator component was developed with the common Linux, Apache, MySQL and PHP (LAMP) stack. These technologies

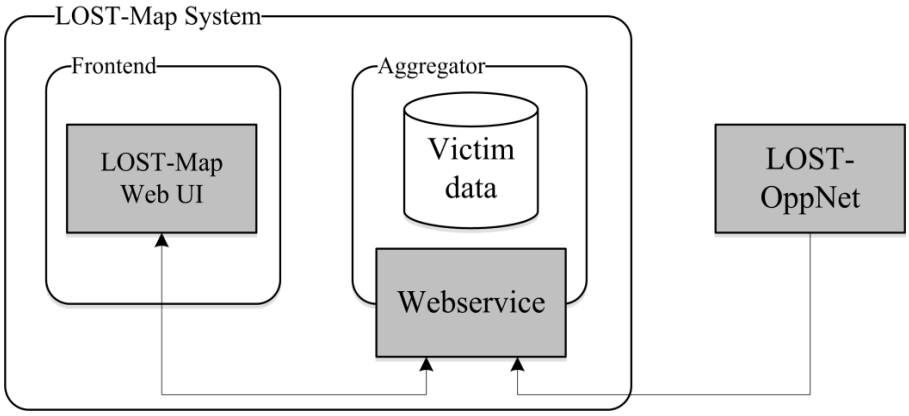


Figure 5.1 – LOST-Map system architecture. The frontend and aggregator components are loosely coupled to reduce the dependence on each other.

were chosen due the previous knowledge on them, and also due their popularity and easy deployment in case of needing an unplanned service up and running.

5.2 Implementation

LOST-Map can be split in two large components. It consists on a web application that provides a graphical user interface to volunteers, and a webservice to allow access to victims' data by the interface and other interested external applications. Each of these two features was implemented as separate components. The first one is the frontend, which contains the graphical user interface to volunteers. It consists on an online map with a special overlay containing information about victims. The other component is the aggregator. It comprises a repository for all data gathered and sent to the LOST-Map and a webservice to provide controlled and abstract access to the data stored on the aggregator. These components were designed to be loosely coupled, so they are physically independent, that is, the victims' data could be stored elsewhere, and the frontend still be able to access the data. Figure 5.1 contains a diagram with the representation of actual LOST-Map architecture.

5.2.1 Frontend

The frontend is the component that offers volunteers an online map with victims' information. The map is based on the Google Maps API and offers the basic functionality available, such as zooming and general map navigation. On top of that, other functionalities were developed. An overlay was implemented in order to show victims' geographical locations on the map. That overlay offers volunteers useful information regarding the victims' trail. A trail made by a victim is a set of points gathered periodically by a LOST-OppNet device. A point may include automatically detected movements or screen activations, to understand if the victim is interacting with

the device. It also includes messages sent by the victim at that point, if any. By default, the map shows only the last known point of each victim at the corresponding geographical location. Then, it is possible to see the trail for a single victim, along the messages that the victim may have sent. This means that it is possible to choose a victim at a time and then see the complete trail that the victim did during the disaster scene.

The map also allows showing these points dynamically, that is, to only show points matching certain criteria. LOST-Map includes features to filter data based on time, information contained by each point, and by victim rescue status, that is, if the victim is safe or not. Other tools are also available to help volunteers on their mission. For instance, it is possible to search for a victim using their name or part of it. By writing the victim's name on a search box, the system compares the search term with the victims' names in real-time and then generates a list of all matching victims. A functionality called Critical Area was also developed to allow volunteers drawing a zone on the map for their personal reference.

Another interesting feature of LOST-Map frontend is the real-time update capability. Volunteers can have the map always open and receive data from new victims automatically. This allows the interface to update the existing information according to most updated data about the victims. Volunteers can then follow the victims that they are interested in rescuing more accurately. For instance, if a trail for a victim is drawn on the map, when a new point for that victim is added to the map, the trail is updated to show the new location. These real-time updates were implemented using the pooling method. This means that the webservice is periodically asked for new points.

5.2.2 Aggregator

The other main component of LOST-Map is the aggregator. Its responsibility is to ensure that victims' data is correctly stored, remains consistent and that read and write requests done to this data are satisfied. It is possible to split the aggregator in two large subcomponents: data storage and webservice.

The data storage is the place where all victims' data is stored. It consists on a MySQL database with a relational schema to retain several records, each of them with a well-known number of properties. The actual relational schema is summarized on Table 5.1, with an indication of the concrete fields used, their restrictions and their purpose for the system. Some efforts were done to improve data consistency while writing data. When inserting a new record, the nodeid and timestamp fields form a unique pair that uniquely identifies that record, that is, a single and unique message from a victim at a given time. This was done to prevent duplicate messages from the same victim to be inserted in the system. Recalling that nodes on LOST-OppNet may generate and

Table "Points"		
Name	**Field parameters**	**Field purpose**
nodeid	Char(100), Primary Key	Victim unique identifier
timestamp	Long, Primary Key	Time of message creation
latitude	Long	Latitude of victim
longitude	Long	Longitude of victim
llconfidence	Integer	Confidence level for latitude/longitude pair
battery	Integer	Current battery level of device
movements	Integer	Number of movements detected by device so far
screen	Integer	Number of screen activations detected by device so far
distance	Long	Unused. Reserved for future use
safe	Integer	Code to indicate if the victim was marked as safe or not
added	Long	Local date when the record was added to the table

Table 5.1 – Schema of the table responsible for storing the victims' data.

disseminate some duplicates, the nodeid and timestamp pair prevents a duplicate to be inserted into the victims' data thus granting a reasonable level of data consistency.

The other subcomponent of the aggregator, the webservice, provides read and write access to the victims' data. The webservice acts as a controller to simplify the access to the data. It offers several methods for reading data, and a single method to insert new records. Writing requests are typically issued by devices running LOST-OppNet. As previously described on section 4.2.6 LOST-OppNet creates a JSON array containing one or several messages from the victims. That array is sent directly to the webservice through the write method using the HTTP protocol. The points are then inserted into the victims' data set, excluding any duplicates of data already existing. On the other hand, there are multiple read methods to access victims' data. This was done to make the webservice simpler to be used by developers. By using multiple methods, it is possible to filter victim's data in several arrangements and obtain a customized view over the data according to specific needs. For instance, the webservice has methods to filter data by victim, by timestamp and even by a given region. The Annex A contains detailed information regarding the methods available on the webservice and examples on how to use them.

5.3 Dynamic visualization tool

LOST-Map interface was implemented as a web application. The interface implements the features described so far, making them accessible to potential users. Furthermore, a study was conducted in order to validate the ease of use of the tool. Details regarding this study are presented in Chapter 6.

The application can be used for two purposes: viewing historical data and viewing real-time data about a disaster scene. Viewing historical data implies the possession of a file containing the data to be seen. That is a Comma Separated Values (CSV) file having all messages exchanged during the previous disaster scene. The file is imported through a dedicated webpage. After the successful import, the map is cleared and shows the data that was uploaded. All filtering functions remain available in order to allow restriction by time and victims' data. The other mode allows seeing real-time data from a current disaster scene. For a live disaster scene, this mode should be used. It requires no configuration at all from the volunteers. They can just open the LOST-Map interface and wait for data to be displayed on the screen. When devices running LOST-OppNet find an Internet connection available, they immediately send all messages they collected so far to the webservice. The result is that data being displayed on the map, when the most recent data is requested. The volunteer can then use all the functionalities available on the interface to refine the search for victims.

The focus of this work will be regarding the real-time mode, since it allows exploring the full potential of the tool. Next sections contain interface details and functionalities available to volunteers using LOST-Map.

5.3.1 Interface layout

The interface elements were disposed like a typical online map application, as seen on Figure 5.2. This ensures that users start with a familiar structure and quickly understand how to operate with the interface and locate the resources they need. The figure shows a possible start screen when a volunteer opens the map during a disaster scene. It is possible to immediately identify three large areas on the interface:

Time-frame control (top). The first area at the top is the slider responsible for controlling the time-frame visible on the map. With this control, a volunteer can restrict the period of time to show data on the map. The slider contains two handles, allowing restricting the time between two known dates. For instance, a volunteer willing to analyse the moment before the disaster should moving the left handle to the desired start time and the right handle to the moment immediately before the disaster. Then the volunteer can only see the victims that appeared during that period of time. When the

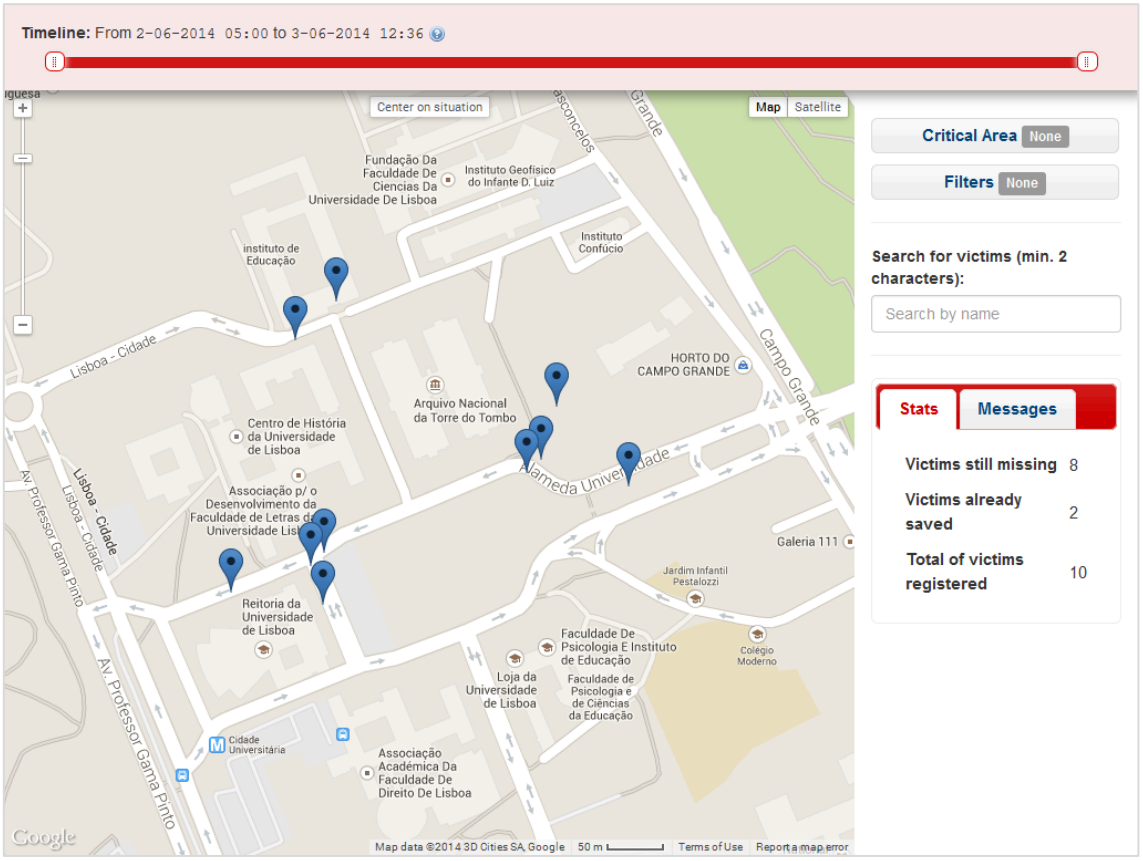


Figure 5.2 – Typical screen for LOST-Map interface.

slider is fully extended, as in Figure 5.2, it shows all victims, regardless time constraints.

Map (left). The second and most prominent area is the map. It was developed using the Google Maps API, allowing the display of a real-world map with ease. In the current scene, it is possible to see some markers drawn on the map. Each marker represents a different victim. The initial set of markers indicates the last point where each victim was located within the selected time-frame. The map allows the standard Google Maps views, such as Map view, with an actual vectorial representation of the real-world map, and Satellite view, to allow bird's eye viewing of the scene, using static satellite images. These modes can be changed without affecting the markers already existing on the map. Zooming is also available, allowing volunteers to geographically filter some zones. A button at top centre of the map was placed to allow volunteers returning to the original view. This is useful when a volunteer needs to temporarily check the neighbourhoods and then wants to return to the disaster scene when feeling lost. By using this functionality, the volunteer can return to the disaster scene immediately at any time in a single action.

Tools bar (right). The sidebar at right allows access to a set of tools designed to help volunteers on finding victims. The buttons at top allow access to the Critical Area and filters. The first button allows the setup of a Critical Area. It allows a volunteer to draw circle on the map for private future reference. The circle can be resized at any time and also removed. The second button provides access to the filters that allow categorizing victims, using colours to distinguish between them. These filters will be discussed shortly. On the sidebar, the volunteer can also search for victims with a known name. For instance, a volunteer can search for the name of a known victim or part of it. The search is done at real-time. All matches are presented in a list, containing the victim name and the last-seen time. Then, the volunteer can choose the most suitable victim and check the associated information. Figure 5.3 shows these functionalities in action to give an idea of their behaviour in a typical scenario.

5.3.2 Features implementation

LOST-Map contains filtering, categorization as well as other features to help volunteers analysing the information presented on the map dynamically, according to their needs. These features are described verbosely in this section.

The time based filtering is controlled by the slider positioned above the map. It allows restricting the time interval to a certain period of time. This filter may be useful in cases when there are several points during a large irrelevant period of time for the situation. For instance, if the data gathering started at Sunday morning, and the disaster occurred between the 8 PM and 11 PM of that day, victim information during morning may be irrelevant to the rescue scene. With the time based filter, a volunteer could restrict the time-frame to 7 PM of Sunday and 2 AM of Monday to see the victims immediately before, during and after the disaster.

Another useful filter is based on data gathered by the victims. Recalling that sensor information gathered on LOST-OppNet may include user movements or screen activations, among others. This filter allows volunteers to make a basic categorization system allowing them to have more focus on victims showing stronger signals about a given sensor value. This filter does not actually remove points from the map, but shows them in a different colour, using a semaphore analogy. Figure 5.4 and Figure 5.5 show an example of this colour filter functionality. For instance, a volunteer that would like to have more focus on victims having lower movements could define the colour scale within the predefined values. The points on the map would be represented as red for victims with lower movements, yellow for intermediate values and green for victims with a high number of movements. Using the colour scheme ensures that no victim is left out of the map while still providing a method of categorizing them. This scheme allows volunteers to focus more on victims that may need immediate rescuing or split

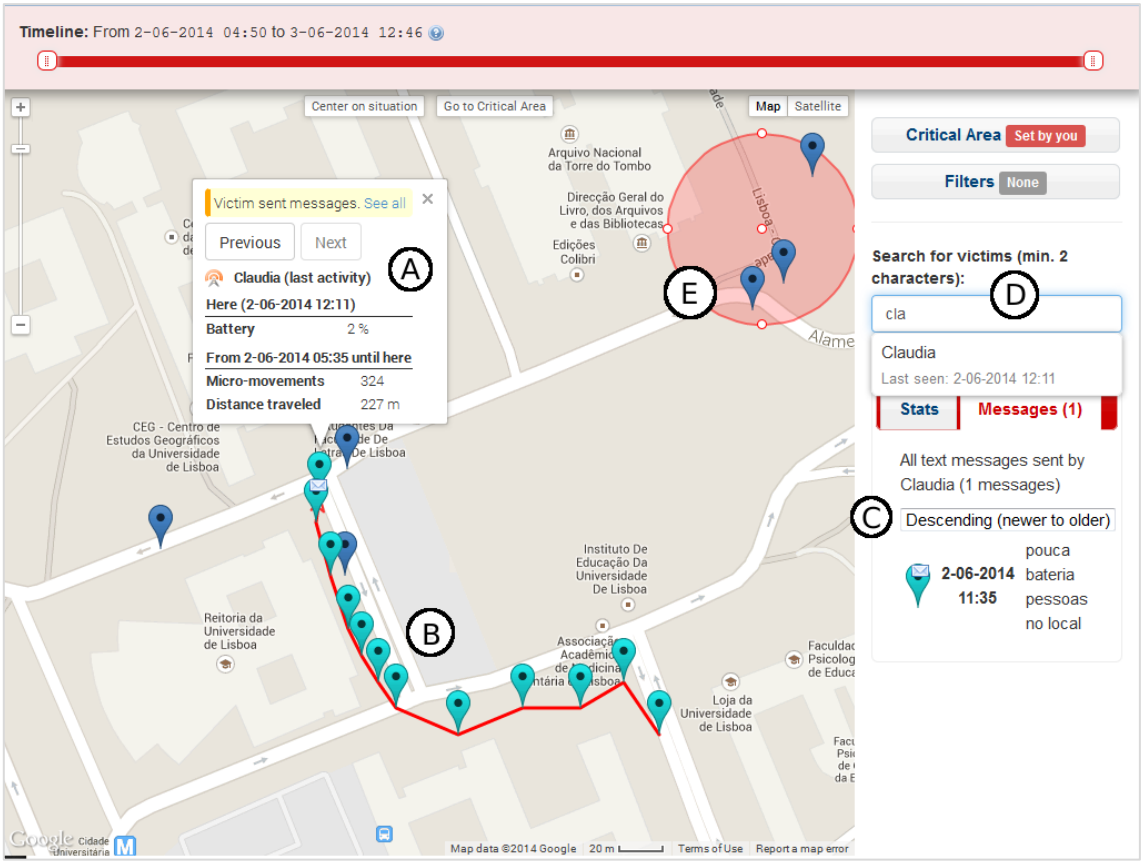


Figure 5.3 – LOST-Map interface with the (A) balloon, (B) trail, (C) message list, (D) search and (E) Critical Area functionalities in action.

efforts among victim groups. The last filter available is the victim safety status. It is a binary option, where it is possible to filter out some victims, that is, hiding victims that are already safe and out of the scene, in order to reduce the visual clutter on the map. This filter can also be seen of the filter settings screen (Figure 5.4).

Each marker on the map also contains information about that particular point. When a volunteer clicks on a certain marker, the corresponding trail appears on the map along a pop-up with sensor data for that particular point (Figure 5.3 (A) and (B)). This allows the volunteer to view the distance travelled by the victim. The pop-up also allows navigation between points, to the next or previous point. With these buttons, the volunteer can navigate geographically and temporarily, thus being able to analyse the route of that particular victim. Each pop-up contains information about the data gathered at that point, such as the time of the day, battery level and number of movements. With this data, a volunteer could infer about the victim status at that position and try to understand the evolution of the victim's condition by observing the path points and their characteristics.

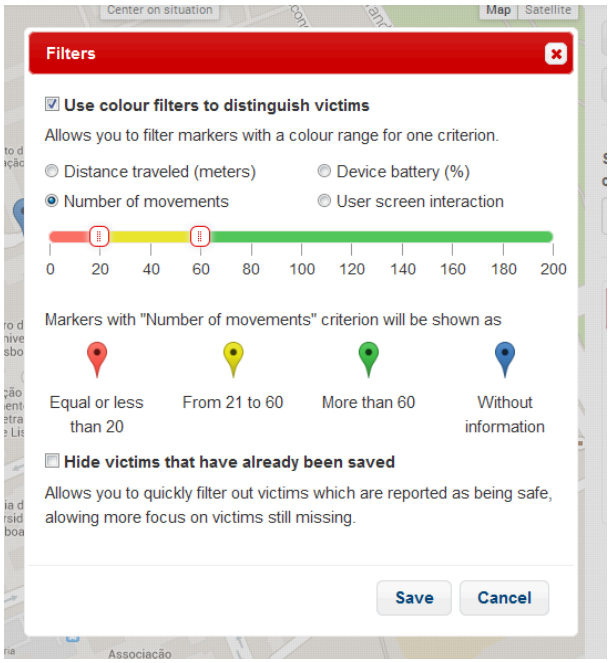


Figure 5.4 – Filter settings screen. It is possible to choose a measure and the colour range according to a semaphore analogy (from left to right: red, yellow and green).

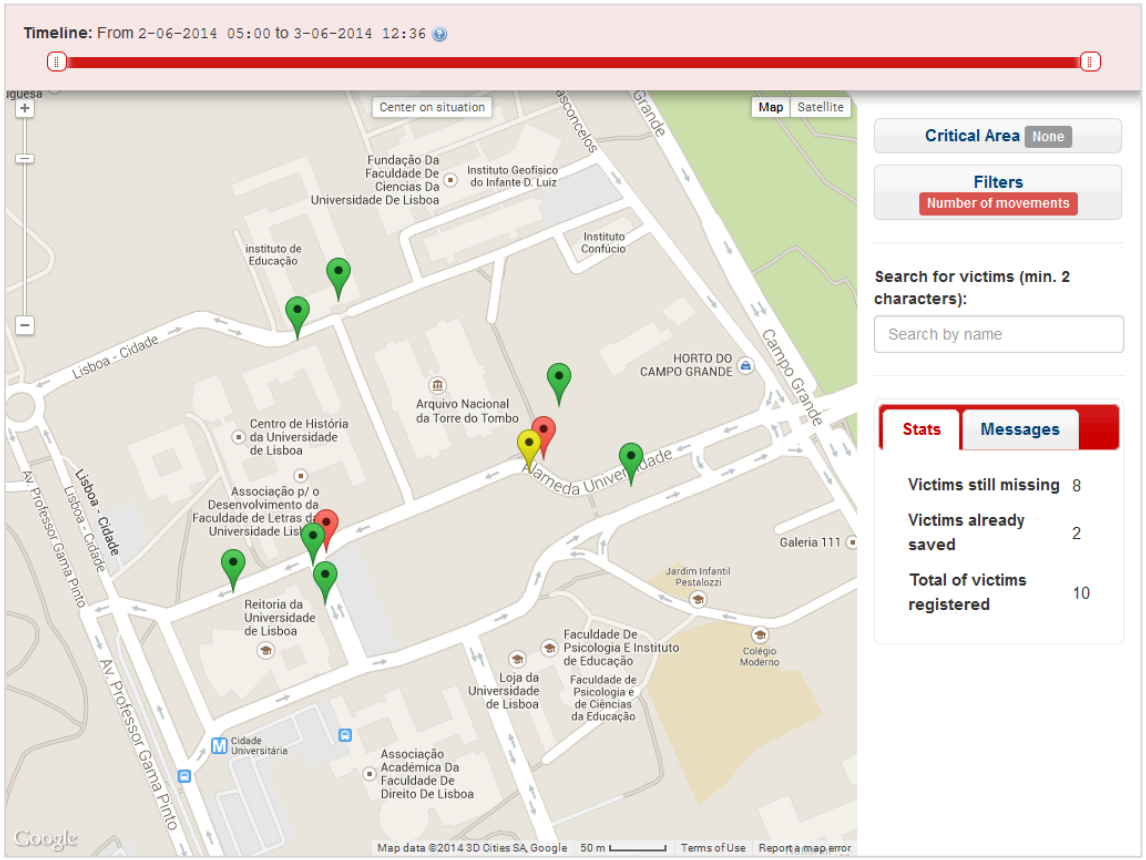


Figure 5.4 – LOST-Map interface after applying a filter. The markers are coloured according to the chosen scale on the filter settings screen.

The pop-up also contains a notification of text messages sent by victims, if any. A point carrying a message additionally contains the contents of the message written by the victim in a highlighted form. The marker icon has a mail envelope, telling that particular point has a text message. In case the point is out of the sight of the volunteer, or for other reason it was not noticed, a small notification is included in the pop-up for every point of the trail. It offers the option to show all messages on the toolbar (Figure 5.3 (C)). The toolbar has the indication of the total amount of messages sent by the selected victim, a list of every message sent by the victim and allows centring the map on a particular point containing a message. This shortcut allows a volunteer to immediately check the messages of the victim without needing to analyse the complete path and look manually for text messages.

Other additional features are also included in the LOST-Map interface. It is possible to search for a victim using their name or part of it (Figure 5.3 (D)). By writing the victim's name on the search box, the system compares the search term with the victims' names in real-time. A list of matching victims is shown and the volunteer could pick up the one that seems more appropriate. The result would be the map centring on that victim, in the last-seen position. Given that volunteers often try to look for relative or friends in first place, this functionality allows them to quickly search for known victims. Other functionality present is the Critical Area (Figure 5.3 (E)). This function allows a volunteer to draw a circle on a certain zone of the map. The original idea was to promote interaction between volunteers and giving them the opportunity to share a location where a special event may occur. This event could be a safe zone or even a dangerous one. However, the functionality was not fully developed. Currently, it is implemented as a tool allowing volunteers to define a region for any purpose, such as future reference. That region can be resized or recreated any time and any changes are visible only locally. Figure 5.3 shows these functionalities being used simultaneously.

5.4 Summary

This chapter introduced the LOST-Map tool and its components. LOST-Map contains an aggregator component responsible for managing the victims' data and a frontend component responsible for promoting a way of using the data available to transform it into useful information to detect victims in a disaster scenario.

The integration with LOST-OppNet should be clear at this point. LOST-OppNet sends its data to the aggregator component, specifically to the webservice. This webservice stores the data on the data storage to be later accessed by volunteers. This interaction is done using standard web technologies. An overview and examples on how to use the LOST-Map interface were also presented. LOST-Map frontend contains a

graphical interface with features to allow discovering victims more efficiently with the combination of filter and additional functionalities.

The next chapter presents an evaluation of the LOST-Map tool to understand if the tool can be used by untrained volunteers and if they understand how to use the tool without having previous knowledge about its functionalities.

Chapter 6

User study: evaluating LOST-Map

To assess if LOST-Map can be a useful and usable tool for untrained volunteers, a user study was conducted. It comprised a number of tasks that simulated the typical use cases that LOST addresses, such as searching for victims or analysing the situation after a disaster. This study was done for an early version of LOST-Map. While the study may contain issues that were already addressed in the current graphical interface, the conclusions were important to enhance the tool and to understand how potential users interact with LOST-Map.

6.1 Apparatus

A typical desktop computer was used to access the LOST-Map interface. Specifically, the users accessed the map through the Firefox web browser, chosen due its compatibility with Google Maps and its immediate availability on the equipment. For data gathering, paper questionnaires were given to participants and a cell phone with voice recording capability was used to record the final interview.

6.2 Participants

A convenience sample of ten participants was recruited. Ages ranged from 22 to 51 (mean=27.3 , SD=8.6). All participants had some experience with using Google Maps and good knowledge of the scenario areas. Participants were offered no compensation.

6.3 Procedure

Participants were given a paper script containing an introductory description of LOST-Map and its purpose, along with three tasks to do. At the end, a semi-formal interview was done with each participant in order to gather additional feedback from them regarding the tool and its usefulness.

A task consisted in a fictional disaster scenario present on the LOST-Map interface. The goal of each task was to use the resources that LOST-Map offered to locate and

identify victims of the disaster according to certain requirements expressed by the task. These scenarios were generated in advance using a computer program and inserted right before each task start, in order to ensure a clean scenario in each run. Annex B contains the detailed script for all tasks, along screenshots exemplifying the expected results on the screen.

The moderator was responsible for measuring completion times for each task. After each task, participants were asked some questions to understand their comprehension regarding the interface and functionalities. After that, they were asked to answer a between-task survey with a single question. After all tasks were completed, participants were administered a final questionnaire, to assess their overall perception of LOST-Map, and a final semi-structured interview.

6.4 Measures

For each task, the following set of measure was collected:

- Total amount of time to conclude each task;
- Whether the participant concluded each task without giving up;
- Number of questions that participants asked the moderator;
- Ease of use as measured by the Single Ease Question (SEQ) (Sauro et. al. 2009), from 1 ("very difficult") to 7 ("very easy").

SEQ is a standardized usability measure, whereby users are asked to complete the statement "Overall, this task was:" using a Likert scale. Overall perceptions were measured with the AttrakDiff (Schaik et. al. 2012) questionnaire of user experience, in the ten-item version. AttrakDiff is a set of semantic differentials that inform on subjective perceptions of pragmatic quality, hedonic quality and attractiveness.

6.5 Results

All participants concluded successfully all tasks proposed. The mean time to conclude each task is presented in the Table 6.1. As expected, as tasks were completed in increasing degree of difficulty, participants needed more time to conclude task #3, which was closer to a real world situation, requiring a combination of techniques learnt from the first two tasks. Scores for the SEQ are also presented in Table 6.1. The average SEQ score is in line with the task completion time: the more complex the task is, the less easy users found it.

The total number of questions asked by participants was also measured. By comparing the average number of questions between task #1 and task #2, as well as the observations done by the moderator on the study, conclusions about the decreasing

	Avg. task completion time	Avg. SEQ score	Avg. help requests
Task #1	1m48s (SD=37s)	5.50 (SD=1.08)	0.7 (SD=0.7)
Task #2	1m58s (SD=46s)	5.70 (SD=1.34)	0.6 (SD=0.5)
Task #3	2m43s (SD=99s)	4.50 (SD=1.58)	1.0 (SD=1.2)

Table 6.1 – Average completion time, average SEQ score and average number of help requests for each task.

number of questions can be made. For instance, the task #1 was relatively easy, by having a small number of victims all of them being visible, without requiring the use of filters. However, the participants were presented the interface for the very first time and asked about less clear interface elements. It was also observed that some participants preferred to confirm tasks with the moderator before actually doing it on the interface. The moderator tended to allow the participant to discover how to continue autonomously. General conclusions are that participants asked more of the moderators in the most complex task. It was also observed that the number of help requests increased as the task was getting more difficult.

An AttrakDiff questionnaire was used to evaluate the full experience provided by LOST-Map. Each semantic differential was given a score ranging from one to seven, the latter being the most positive score. The first four differentials are indicators of pragmatic quality and second four are indicators of hedonic quality. All results are between five and seven in the seven points scale. Responses to AttrakDiff were on the positive side of the differentials, but it seems participants perceived LOST-Map interface to have greater pragmatic quality than hedonic quality. The primary focus during the development of LOST-Map interface was the introduction of useful functionalities to filter and understand data gathered from LOST-OppNet. While this does not necessarily mean that the hedonic aspect of user experience was neglected, it was a factor with a slight lower priority. On the other hand, participants seemed to understand how to work with the proposed tool and found that it is easy to operate with.

The final interview also allowed gathering additional feedback regarding functionality. In general, participants missed text search functionality, not available in this early prototype. They claimed that searching for a specific victim with textual information, such as name or phone number would be easier then looking victim for victim, while trying to find a specific victim. Since the tool was primarily designed for volunteers, who may know the victims, this suggestion seemed to be relevant to the current work.

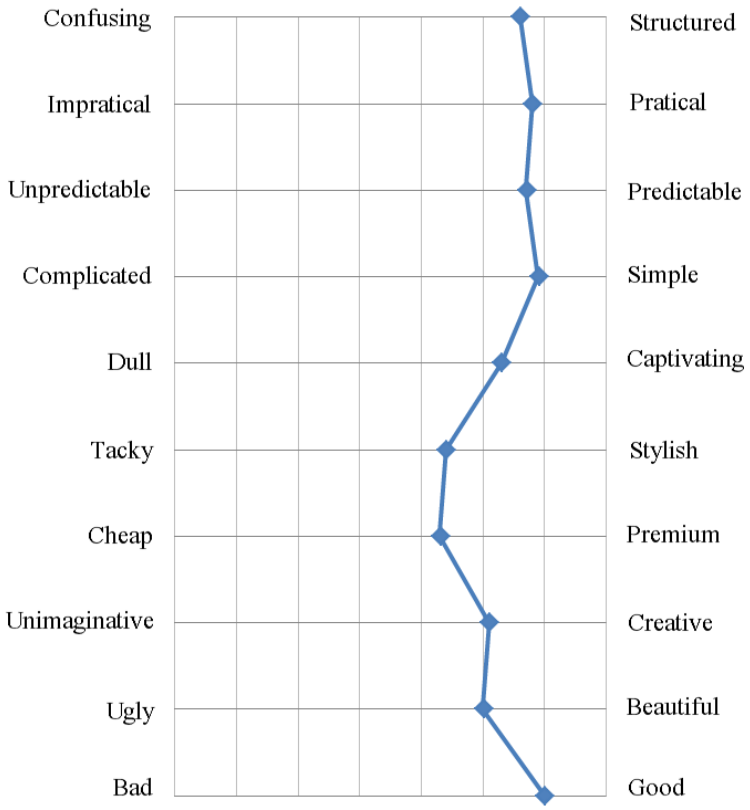


Figure 6.1 – AttrackDiff results for LOST-Map as perceived by participants.

Another functionality criticized was the colour filter. The colour filter allowed participants to give more focus to victims with certain criteria. This filter was used in task #3 instead of manual search, due the elevated number of victims present on the map. Problems were mainly generated by the slider used to change the colour range. The assessed version of LOST-Map offered the possibility of omitting a colour from the scale. Recall that the slider range had three colours to define ranges, namely red, yellow and green. For instance, if a volunteer would only want red and green, volunteer could drag one handler into another in order to have only two colour scale. Participants found this behaviour unusual and unexpected, while some participants found the feature an unnecessary complement. Concerns about the expected effect after changing the scale were issued. Participants did not feel confident about the feedback provided by the scale alone. In general, a pre-visualization was desired to understand the changes made by the colour range filter. In the assessed version of LOST-Map, this kind of feedback was inexistent.

Messages were a topic of concern by the participants. The previous LOST-Map version contained a simple list of messages when choosing a victim from the map. However, that list was only a text list of messages sent by the victim along the corresponding sending date and time. Participants missed an option to quickly jump to

the location where the message was sent. The order of the messages was also unexpected for some participants. The list was formerly ordered by a descending date and time, that is, the most recent message would appear at the top. A small fraction of participants were expecting older messages to be at top, order chronologically. Furthermore, there was no indication about the order neither an option to personalize this aspect.

6.6 Conclusions

Overall, the results validated LOST-Map interface as an effective and easy-to-use tool for use by untrained volunteers. Participants seemed to adapt easily to the majority of the features offered by the tool. The study allowed finding some areas of improvement regarding the tool's usability. Some of them were addressed on the current version of the LOST-Map interface. A search functionality was implemented, allowing direct access to a victim, by knowing the victim's name. The colour filter was also subject of several improvements. For instance, the slider was redesigned to remove the feature of omitting a colour from the scale. Moreover, a new preview below the slider now allows volunteers to understand how the markers on the map will be recoloured.

The results and observations indicate that the LOST-Map can be efficiently used by untrained volunteers. This aspect is important as volunteers may often need to quickly deploy rescuing tools in disaster scenarios. As previously demonstrated, LOST-Map may be quickly deployed on scenarios where LOST-OppNet powered devices are already operating.

Chapter 7

User study: evaluating LOST-OppNet

A user study was conducted in order to assess if LOST-OppNet can connect to other devices and evaluate the overall perception about the VictimApp usability. Specifically, RescueOppus and VictimApp introduced in Chapter 3 and Chapter 4 respectively were used in conjunction with LOST-OppNet in order to simulate a disaster scenario and allow message exchange on the field. Recalling that VictimApp uses the LOST-OppNet service, by evaluating the performance of VictimApp, it is also possible to see if LOST-OppNet is working as desired.

7.1 Apparatus

In this study, RescueOppus and VictimApp applications were used. RescueOppus is a native Android application with a purpose similar to LOST-Map. It contains a map showing detected victims and allows real-time updates of the current situation. This tool is suitable for volunteers. VictimApp is a tool suitable for victims, with the possibility of advertising the victim's presence and send a text message to nearby volunteers. Both tools use LOST-OppNet to communicate with each other. The RescueOppus was deployed on a Samsung Galaxy Tab2 10.1 device, while the VictimApp was deployed on a Samsung Galaxy Ace smartphone. Both devices were previously disconnected from the Internet. The RescueOppus application was designed to alert the user both visually and audibly for unexpected events. For instance, when an unexpected event occurred, the device issued an audible alert along with a small notification at the bottom of the map.

Paper questionnaires were administered between tasks. Two moderators were responsible for ensuring the completeness of each task, helping the participants if strictly necessary.

7.2 Participants

Ten participants were recruited to take part in this study, three of them being female. Their ages ranged from 23 to 29 (mean=25 , SD=2.3). All participants had previous knowledge on using Internet and online maps, and good knowledge on the area used by the scenarios. Participants were offered no compensation.

7.3 Procedure

The study consisted on a rescue gamification, with two tasks, each one with its own scenario. Each task was intended to simulate a disaster, with the inclusion of two participants: one played the role of a volunteer and other played the victim role. The goal of the volunteer was to use the RescueOppus application to find fifteen victims who were still alive and make decisions according to the information that the application provided in real-time. On the other hand, the role of the victim was to use the VictimApp in order to give the volunteers clues about the victim's presence. Other victims were fictitious, computer-generated, and were physically represented by post-its placed in advance at fixed locations. Each of those victims had unique characteristics, such as name and personalized indicators regarding the victim's condition.

Each scenario consisted on a small region that was allegedly affected by a natural disaster. It had multiple groups of computer-generated victims. Some of the victims inside a group could be alive while others were unconscious. To rescue one or multiple alive victims from a group, the volunteer had to stay two minutes in the place where they found them. Then, a popup on the RescueOppus application would indicate that the rescue was complete and the volunteer could continue its mission. During this time, the volunteer had to time analyze the map and decide the next group to move on. Both scenarios were identical at the start. Two groups of victims at equal distance were presented in RescueOppus. Both groups had eleven victims. However, one group had more victims alive than the other. The volunteer's decision was to choose a group to start. After choosing the initial group, the volunteer had to save the victims in that group. From that point on, more groups appeared, with different situations according to the scenario.

On the scenario #1, after the first group, volunteers were confronted with another challenge. They had to choose between two groups, both at the same distance and containing an equal number of victims alive. The difference between groups was the most prominent indicator. For instance, victims on one group had stronger indicators regarding screen activity while others had stronger indicators regarding micro-movements. The goal of this challenge was to understand if volunteers had preference for particular indicator and if indicators were important to them.

The scenario #2 had a slightly different approach, with more unexpected events. After saving the initial group of victims common to both scenarios, volunteers were confronted with two groups, one being closer to the volunteer than the other. However, a careful analysis of the victims on the closer group would reveal a text message telling the most direct way between the groups was blocked. So, it was the responsibility of the volunteer to choose an alternative way to reach those victims. Also, the victim responsible for sending the alert had left a trail on the map, suggesting an alternative way. If the volunteer missed those clues, the moderator would be forced to give new instructions on following the alternative way. The goal was to understand how easily volunteers would interpret clues from victims. After finding the alternative way, another unexpected challenge took place. During the path to the final group of victims, other group appeared. This new group was close to the path, however away enough to require the volunteer to change its path. The idea was to test how volunteers would react to unexpected events occurring on the map. There were two actions possible: save the new group of victims or ignore it. In both situations, the volunteers had to save the final group in order to reach the fifteen victims goal and terminate the task.

On the other hand, the victim role on both scenarios was to interact with VictimApp and send some messages. These messages contained the text message along information gathered automatically by LOST-OppNet, including the victim's geographical location. The victim was inserted on one of the groups and under supervision by a moderator, who was available to ask for help if strictly needed.

During the study, each group of two participants was invited to be part of two tasks. In the first task, one of the participants was the volunteer while the other played the victim role. On the second task, they inverted their roles to give everyone the opportunity to use both applications. A questionnaire was given to each participant at the end of the task, according to their role. The questionnaires goal is to assess users' experience with the applications and the communication. Annex C contains the model questionnaires presented to users.

7.4 Measures

A set of measures were gathered in order to allow evaluation of the applications used. For VictimApp, the measures were:

- User perception that a message leaves the device;
- User thinks that the VictimApp is useful during the rescue;
- Ease of use as measured by the Single Ease Question (SEQ) (Sauro et. al. 2009), from 1 ("very difficult") to 7 ("very easy").

Measures for VictimApp allow understanding if messages were successfully exchanged, and if users were aware of messages being sent from their devices to the network. The second measure is extracted from the victim questionnaire, using the question "Did you think that VictimApp was relevant to your rescue?" with Yes or No being the possible answers. The last two measures allow gathering the users' opinion regarding the VictimApp and its usefulness during a disaster scenario. The questionnaires also contained additional questions to check if users could retain some VictimApp interface elements immediately after using the application, in order to check if the interface is simple enough to allow victims retaining all the information provided. Annex C contains the complete set of questions presented to users.

7.5 Results

All tasks were completed successfully by all participants. Table 7.1 presents the details for each component evaluated. In general, participants understood how to operate the application, namely the message-sending mechanism. When questioned about how to send a message without looking at the application screen, every participant was capable of correctly describing the steps to send a message to the network. They also understood that after sending a message it would appear into a list along a symbol indicating whether the message was sent or is still waiting for a connection. However, the status icon associated with each message was subject of confusion. Some of the participants did not notice it until a message was sent to the network, where the icon would change to a different one. A suggestion made regarding the icons was to put a label on the screen indicating the possible icons and their respective meaning. The LOST-OppNet status was also identified by majority of the participants. When inquired about the current status of LOST-OppNet, participants revealed more confidence. The status is represented by an icon as well, however it also contained a descriptive text such as "Connected, sending messages" or "Looking for

VictimApp evaluation results for both scenarios	
Application usefulness/relevancy during a disaster	90%
Victims that noticed messages being exchanged	80%
Single Ease Question	mean=6.3 , SD=0.7

Table 7.1 – Detailed results for the aspects evaluated in VictimApp.

other nearby nodes...” indicating the status purpose along the icon. This could explain the better performance on identifying the application status opposing to identifying the message status.

Regarding the tool usefulness during a disaster scenario, the majority of the participants told that it would be useful in that kind of situation. Participants’ comments confirm the drawbacks found in message feedback. The features that participants notice lacking of where: 1) more feedback regarding message sending confirmation (e.g. either using a popup or device vibration); 2) more information regarding the statuses icons, that is, what is the application doing exactly.

Regarding the VictimApp usability, participants found the application easy to use: the average SEQ score was 6.3 points out of 7. This means that the tool is simple enough to understand, as desired. Under disaster scenarios, victims may not pay attention to their devices, due panic or other external factors. With this in mind, the VictimApp was designed to be easy to understand and operate when needed. Additionally, some information is gathered on the background, as part of LOST-OppNet functionality, and thus needing a short attention span from the victim to operate with the application.

7.6 Conclusions

A study to verify that users understood the purpose of the VictimApp was carried out with positive results. At the same time, LOST-OppNet was subject of testing, and conclusions are that the tool is able to connect to other instances. However, this connection may be subject of failures due some environment constraints. Namely, the GPS connection may interfere with the search for victims, due to unknown or imprecise location. Besides that, the messages are correctly exchanged in the majority of the cases, some of them even when nodes are dozens of meters apart. VictimApp tool is also easy to use, according to the test results, which may reveal useful when users’ attention is affected by stress or other factors during a disaster.

Chapter 8

Conclusions

This section presents a final overview regarding all work developed so far. It also comprises a critical review of the developed LOST project components along with perspectives of future developments for them.

8.1 Overview

The LOST project is a rescuing oriented prototype to allow untrained volunteers to look for victims in the field and allow these victims to communicate back to them. The base premise of LOST is that people on the field should be empowered with tools to allow helping victims in need. LOST-OppNet is a shared component developed to allow network communication using readily-available devices in scenarios where typical connections may be unavailable. With that goal in mind, an Android application designed for victims was developed. VictimApp uses LOST-OppNet to communicate with other devices and requires little to no interaction by the victim. On the other hand, a tool to empower volunteers was also developed. LOST-Map allows volunteers to look for victims in a dynamic and updated map. The interface offers dedicated functionalities, allowing them to quickly find victims by name or even by indicators made available by the victim's device sensors. The tool also acts as a shared repository of data, allowing communication between LOST-OppNet and the map interface, as well as applications that may be developed in the future.

Overall, developing these components of LOST project was a challenging work. First of all, LOST contains more than one technology. This required the development of common platform to cope with technologies differences. For instance, the communication between LOST-OppNet and LOST-Map is currently established using a webservice; the LOST-OppNet network message format was also target of some modifications to be readable by that webservice. On the other hand, LOST is specially designed for untrained people without ability in rescuing others. The main goal is to give more power to people who are willing to help, and not to replace the typical

rescuing teams, specialized and trained for disaster scenarios. LOST is beneficial from the point of view that it can provide additional information about the victims' location along with other data that may be useful to understand if the victim is fine and responds to external events, such as the notifications generated by VictimApp.

Studies also benefited both tools. By receiving users' feedback, it was possible to develop newer functionalities that were more in line with the view of possible rescuers. By putting people in fictional scenarios of disaster with the mission of saving friends and relatives, they were able to think "how can I use this tool to my benefit?" and provide feedback accordingly. This can specially be seen on the LOST-Map study, where users suggested additional features according to their needs. Some of the suggestions were interesting and highly requested among the participants. Some of them were integrated in the current version of LOST-Map presented in this work.

8.2 Limitations

Some limitations were found in the final stages of LOST project development. They represent opportunities to reflect about the current implementation and improve the project.

LOST-OppNet communication can be significantly enhanced. Currently, it only allows one-way communication, from victims to rescuers. The communication could be done both ways, also allowing rescuers to communicate with victims. This is currently a challenge, as such communication often is preferred in real-time, requiring a stable connection channel, which may be inexistent in disaster scenarios. The geographical location gathering method should also be enhanced. In the current implementation, a smartphone GPS sensor is used to guess the victim's location. However, GPS performance can be poor inside buildings and other situations, as concluded in the Chapter 7. This is especially important when victims are indoors. Thus, a better mechanism of gathering the victim's geographical location could be researched in order to allow the detection in such environments, even with a small error margin. Data gathering in general is also currently limited to the available sensors for each device. Moreover, smartphone sensors were developed to smartphone operation purposes, that is, detect the device orientation or if the device is falling. This means that data would not be recommended to be used for medical purposes, and thus, it may be inaccurate to evaluate the victims' condition.

LOST-Map could also be improved. Volunteers are still in doubt when using some of the functionalities, revealing several points to improve the tool. For instance, categorization of victims is still dependent of information gathered by device sensors, which may reveal less useful in more specific scenarios. Other problems with visual

representations are still on the table: how and when victims should be grouped? Should high amounts of victims' automatically gathered information be grouped, being possible to lose small important details? On the other hand, an Internet connection is still needed to exchange data between the LOST-OppNet and the LOST-Map aggregator, since that LOST-OppNet applications would try to contact a predefined webserver running an LOST-Map aggregator. Reducing this dependence would help volunteers to create a local aggregator, enabling information to be exchanged directly from devices to the aggregator, without requiring external connections.

8.3 Future work

LOST project is an ongoing effort. A lot of improvements are possible to be applied in a near future or even added as long-term goals for the whole project.

Currently, there is no way of remotely triggering LOST-OppNet based applications to wake up and start gathering data. Although it can be developed with currently available technology, for instance using a cloud messaging system, such as Google Cloud Messaging, the implementation could represent a privacy concern, since LOST-OppNet may run silently on devices. For example, users could be monitored by using a locally deployed LOST-Map instance and then gather data about the users' device, including their geographical location. Smartphones sensors could also be explored to extract more data from them. Recently, new innovations allowing integration between users' health and smartphones are arising. For instance, wearable technologies such as Sony SmartBand[12] or Samsung Gear Fit[13] allow a deeper interaction with users, by estimating their steps or heart rate. Despite these products being primarily designed for sports, they could be also used in the context of rescuing. This would mean that LOST-OppNet would be able to gather data closely related to the victims' condition, offering better information to be shared with volunteers and rescuers.

Collaboration between volunteers could benefit LOST-Map interface. They could use LOST-Map interface to plan their rescue works and report when finding a victim, in collaboration with official rescuers. This would allow volunteers to work together and summing their efforts to rescue the victims. They could also be able to chat with others and share their views on the scene. There were some steps taken in this direction. The Critical Area feature presented in section 5.3 was intended to be used as a common area, shared by all volunteers. However, the feature was too simplistic and future developments towards it were dropped. In the future, the functionality could intelligently identify zones with large number of victims and label them automatically.

[12] Sony SmartBand – http://www.sonymobile.com/global-en/products/smartwear/smartband-swr10
[13] Samsung Gear Fit™ – http://www.samsung.com/us/mobile/wearable-tech/SM-R3500ZKAXAR

Moreover, it could also be useful for planning purposes, distributing groups of volunteers among specific zones, with reports or tasks to accomplish, in real-time.

On the other hand, LOST-Map aggregator could be redesigned to allow a more decentralized architecture. Currently, LOST-Map aggregator contains a database where points of all victims are stored. This means that an Internet connection may be needed in order to receive information regarding new victims. The LOST-Map aggregator could be partly integrated with every device running LOST-Map interface, storing the scene in their view. This view would be updated as the volunteer was exploring the scene. Then, when two or more volunteers were in range, they could exchange all their views and create a global view of the scene.

In summary, LOST project is capable of providing a basic support to victims and volunteers in simulated disaster scenarios with untrained volunteers. While still not being tested to be deployed in a real-life disaster scenario, LOST tools are being developed aiming to provide support in such conditions. With the consumer electronics evolution, people are more likely to carry a smart device with them, augmenting the chances of deploying a solution like LOST-OppNet in their devices for a foreseen disaster. This would also mean that more data could be extracted from health-related sensors, allowing better estimates regarding victims' condition, giving LOST a bright future regarding a deeper integration with the victim and more collaboration between volunteers.

Bibliography

Al-akkad, A. et al., 2014. Help Beacons: Design and Evaluation of an Ad-Hoc Lightweight S . O . S . System for Smartphones. In M. Jones et al., eds. CHI ’14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM New York, NY, USA, pp. 1485-1494

Belblidia, N. et al., 2011. PACS: Chopping and shuffling large contents for faster opportunistic dissemination. In 2011 8th International Conference on Wireless On-Demand Network Systems and Services, WONS 2011. pp. 9-16

Chipara, O. et al., 2012. WIISARD. In Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys ’12. New York, New York, USA: ACM Press, p. 407

Gunawan, L.L.T. et al., 2012. TravelThrough: a participatory-based guidance system for traveling through disaster areas. Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts - CHI EA ’12, pp. 241–250

Gunawan, L.T. et al., 2009. Collaborative Situational Mapping during Emergency Response. In L. Norros et al., eds. European Conference on Cognitive Ergonomics: Designing beyond the Product - Understanding Activity and User Experience in Ubiquitous Environments. VTT Technical Research Centre of Finland.

Hashmi, N. et al., 2005. A sensor-based, web service-enabled, emergency medical response system. In Proceedings of the 2005 workshop on End-to-end, sense-and-respond systems, applications and services (EESR '05). USENIX Association, Berkeley, CA, USA, pp. 25-30

Nishiyama, H., Ito, M. & Kato, N., 2014. Relay-by-smartphone: realizing multihop device-to-device communications. IEEE Communications Magazine, 52(4), pp. 56–65

Qu, Y. et al., 2011. Microblogging after a major disaster in China. In Proceedings of the ACM 2011 conference on Computer supported cooperative work - CSCW ’11. New York, New York, USA: ACM Press, p. 25

Ramesh, M. V., Jacob, A. & Devidas, A.R., 2012. Enhanced emergency communication using mobile sensing and MANET. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI ’12. New York, New York, USA: ACM Press, p. 318

Sammarco, M. et al., 2012. PePiT. In Proceedings of the third ACM international workshop on Mobile Opportunistic Networks - MobiOpp ’12. New York, New York, USA: ACM Press, p. 79

Sauro, J. & Dumas, J.S., 2009. Comparison of three one-question, post-task usability questionnaires. In Proceedings of the 27th international conference on Human factors in computing systems - CHI 09. New York, New York, USA: ACM Press, p. 1599

Trifunovic, S. et al., 2011. WiFi-Opp: Ad-Hoc-less Opportunistic Networking. In Proceedings of the 6th ACM workshop on Challenged networks (CHANTS '11). ACM, New York, NY, USA, pp. 37–42

Van Schaik, P., Hassenzahl, M. & Ling, J., 2012. User-Experience from an Inference Perspective. ACM Transactions on Computer-Human Interaction, 19(2), pp. 1–25

Vieweg, S. et al., 2010. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In CHI 2010: Crisis Informatics April 10–15, 2010, Atlanta, GA, USA. pp. 1079–1088

Wagner, T. et al., 2004. COORDINATORS coordination managers for first responders. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004

Wu, A., Yan, X. & Zhang, X. (Luke), 2011. Geo-tagged mobile photo sharing in collaborative emergency management. In Proceedings of the 2011 Visual Information Communication - International Symposium on - VINCI ’11. New York, New York, USA: ACM Press, pp. 1–8

Annex A. LOST-Map webservice details

Victim information (data structure)

The victim information consists on a JSON array containing objects. Each object represents a single point from a single victim and has a set of fields, based on key-value pairs, containing useful information regarding the victim condition at a specific location. The next table summarizes the fields you may find while decoding the JSON structure:

Key	Description
nodeid	Victim unique identification.
timestamp	The time in milliseconds since 1-Jan-1970 registered by the victim application. It corresponds to the time when the message was created in the client. It is usually composed by 13 digits.
msg	The text message written by the victim, if any.
latitude	Represents the point latitude, used for geographical positioning.
longitude	Represents the point longitude, used for geographical positioning.
llconf	Confidence of the geographical coordinates. Currently, '0' means that the coordinates were obtained from the last known location, while '10' means that the exact geographical location was retrieved directly from the GPS, and should be treated as accurate. This field is intended for future use.
battery	Current battery level reported by the victim application.
steps	Number of steps detected by the victim application, if sensors are available.
screen	Number of times that the screen was turned on by the victim, if available.
distance	Currently, this field has no meaning. Distance should be calculated by the client consuming the webservice, if needed. It should return NULL of -1.
safe	Tells if the victim marked itself as safe (1) or not (0). Possible values are 1 and 0.
added	The time in milliseconds since 1-Jan-1970 when the message was received by the webservice. This value should be used to get new data periodically, minimizing the number of points to process.

Methods available

The available methods allow control of victims' data and interaction with other features in LOST-Map.

/rest/victims

Retrieves information regarding all registered victims (HTTP GET) or allows the insertion of points to new or existing victims (HTTP POST).

Request

Verb	Parameter	Description
GET	—	Gets all points for every victim registered in webservice. This method doesn't accept parameters.
POST	data (JSON array)	Must contain victim information in JSON format. This method allows the insertion of multiple victim information. You should send an array even if you intend to send only one record.

Response

HTTP code	Internal code	Response

200	—	Returns all victims' points. Example:
		Request:
		`GET /rest/victims`
		Response:
		`[ { "nodeid":"Alberto", "timestamp":"1385888400000", "msg":"", "latitude":"38.7531", "longitude":"-9.15618", "battery":"92", "steps":"0", "screen":"1", "distance":null, "safe":"0", "added":"1385888400000" }, { "nodeid": ... }, ... ]`

		Request:
		`POST /rest/victims/`
		`...`
		`data=[{"nodeid":"Alberto",`
		`"timestamp":"1385888400000", ... }, { ... } ]`
		
		Response:
		`{ "sent":2, "inserted":2 }`

400	801	The sent string was not correctly interpreted. It could be damaged, incomplete or with wrong syntax. Manually check the data sent to this method and ensure that the string is an array with information for each victim in the string format. You should send an array even if you intend to report only one victim.
400	802	No victim information had been received. This means that the information was correctly decoded but there are no victim records. Please ensure that the records are being sent along with your request. Also check if the data is being sent in JSON array format even if you intend to report only one victim.

/rest/victims/mintimestamp

Retrieves information about victims' points with a given minimum timestamp.

Request

Verb	Parameter	Description
GET	long numeric type (ex: 1234567890123)	Represents the minimum timestamp from which points are included in result. The time to compare is the time when the message was registered in the database and not the client application timestamp. The exact match is also included. point.added >= parameter.

Response

HTTP code	Internal code	Response

200	—	Example Request:
		`GET /rest/victims/mintimestamp/1234567890123`
		
		Response:
		`[`
		`{`
		`"nodeid":"Alberto",`
		`"timestamp":"1300000000000",`
		`"msg":"",`
		`"latitude":"38.7531",`

		"longitude":"-9.15618", "battery":"92", "steps":"0", "screen":"1", "distance":null, "safe":"0", "added":"1300000000000" }, { "nodeid": ..., "added": 1234567890123, ... }, ...]

/rest/victims/lastpoints

Retrieves the last points for every victim.

Request

Verb	Parameter	Description
GET	positive integer numeric type (ex: 5)	Retrieves the N last points of every victim. Semantically, the result is a collection of the most updated entries of each victim in the disaster. Without parameters, this method returns the very last record (one entry) per victim.

Response

HTTP code	Internal code	Response

200	—	Example Request:
		`GET /rest/victims/lastpoints`
		
		Response:
		[
		{
		"nodeid":"Alberto",
		"timestamp":"1300000000000",
		"msg":"",
		"latitude":"38.7531",
		"longitude":"-9.15618",
		"battery":"92",
		"steps":"0",
		"screen":"1",
		"distance":null,
		"safe":"0",
		},
		{
		"nodeid": "Bernardina",
		"timestamp": 1234567890123,
		...
		},
		...]

200	—	Example Request:
		`GET /rest/victims/lastpoints/2`
		
		Response:
		`[`
		`{`
		`"nodeid":"Alberto",`
		`"timestamp":"1300000000000",`
		`"msg":"",`
		`"latitude":"38.7531",`
		`"longitude":"-9.15618",`
		`"battery":"92",`
		`"steps":"0",`
		`"screen":"1",`
		`"distance":null,`
		`"safe":"0",`
		`},`
		`{`
		`"nodeid":"Alberto",`
		`"timestamp":"1299999999999",`
		`"msg":"",`
		`"latitude":"38.7531",`
		`"longitude":"-9.15618",`
		`"battery":"92",`
		`"steps":"0",`
		`"screen":"1",`
		`"distance":null,`
		`"safe":"0",`
		`} {`
		`"nodeid": "Bernardina",`
		`"timestamp": 1234567890123,`
		`...`
		`},`
		`...`
		`]`

/rest/victims/llbbox

Retrieves information about victims' points within a given pair of coordinates forming a latitude/longitude bounding box.

Request

Verb	Parameter	Description
GET	Exactly two pair of coordinates split by commas, ex: lat1,lon1,lat2,lon2	Get the victims' point within the given coordinates of bounding box. The bounding box is created a follows: the top left point is composed placed in (lat1,lon1) coordinates, while the bottom right point is placed in (lat2,lon2).

Response

HTTP code	Internal code	Response

200	—	Example Request:
		`GET /rest/victims/llbbox/38.7855,-9.15618,38.7605,-9.145`
		
		Response:
		`[ { "nodeid":"Alberto", "timestamp":"1300000000000", "msg":"", "latitude":"38.7613", "longitude":"-9.15532", "battery":"92", "steps":"0", "screen":"1", "distance":null, "safe":"0", "added":"1300000000000" }, { "nodeid": ..., "timestamp": 1234567890123, ... }, ... ]`

400	802	The bounding box values are incorrectly formatted. You must pass two pair of coordinates split with commas. The . (period) character must be used as decimal separator for each latitude or longitude value.

/rest/victims/id

Retrieves all points for a single victim.

Request

Verb	Parameter	Description
GET	victim id (string)	Gets all point information for a single victim ID. The ID is case-insensitive.

Response

HTTP code	Internal code	Response
200	—	Example Request: `GET /rest/victims/id/alberto`

		Response: [{ "nodeid":"Alberto", "timestamp":"1300000000000", "msg":"", "latitude":"38.7531", "longitude":"-9.15618", "battery":"92", "steps":"0", "screen":"1", "distance":null, "safe":"0", "added":"1300000000000" }, { "nodeid": "Alberto", "timestamp": 1234567890123, ... }, ...]

Annex B. LOST-Map study tasks

Task #1 - Analyze the path of a victim

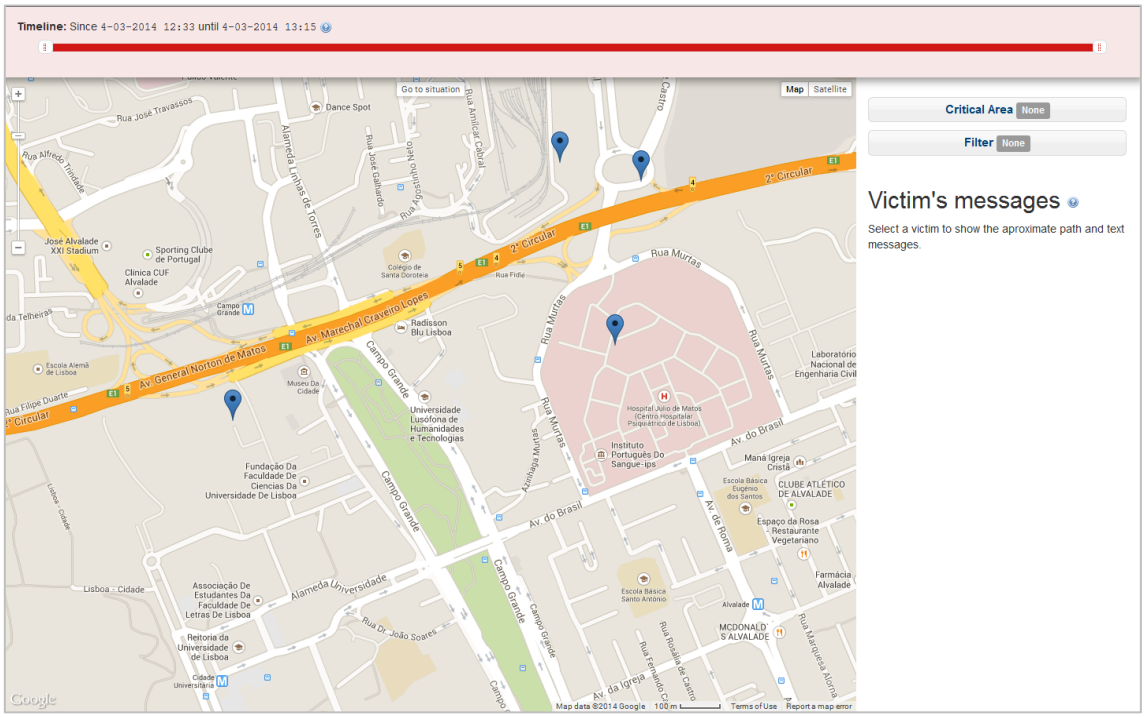
Scenario: Around 12:30 PM, a fire broke out in Cidade Universitária, Lisbon. Despite not causing major damages, the fire created a thick cloud of smoke. Victims felt difficulties in breathing and their vision was affected. You were one of the first to escape the area and to alert the authorities. You don't know if your friend Arnaldo (who was also in the area) escaped. You want to find him without putting your safety at risk, and for that you're going to use the LOST-Map.

Task: Access the map. Locate Arnaldo on the map and observe the path he made. Define a critical area to indicate the origin of the fire zone, according to the place from where people seem to flee.

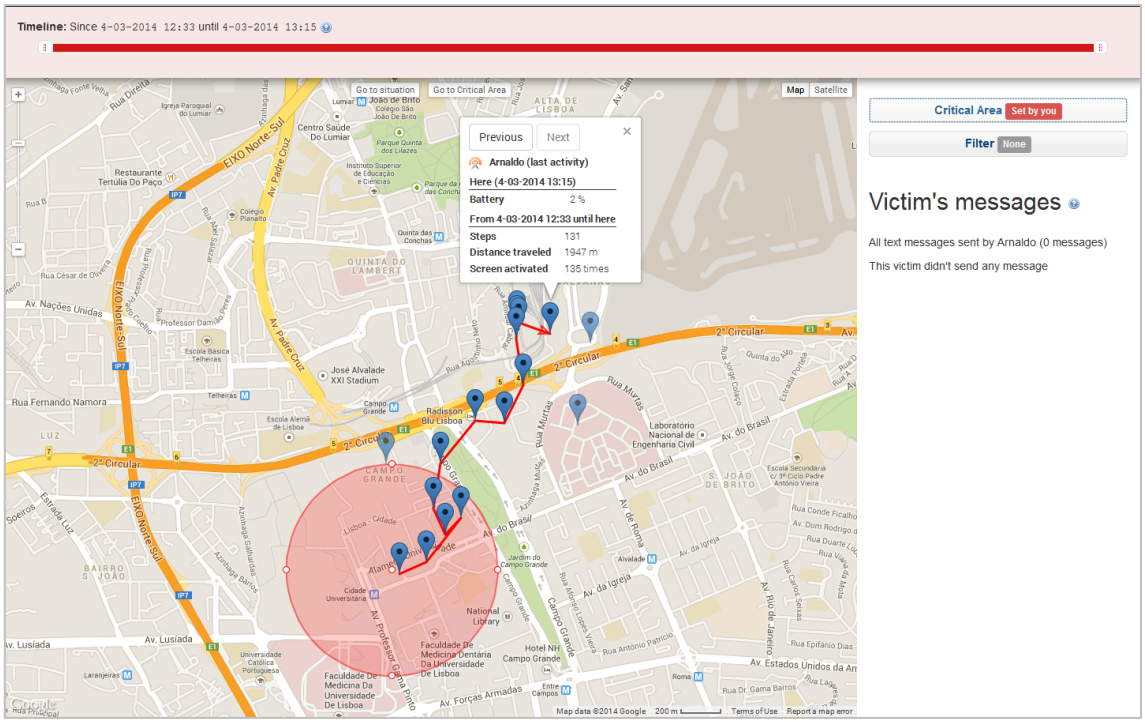
Questions to analyze subject comprehension:

1. Where is the start point of Arnaldo's path?
2. Where is the end point of Arnaldo's path?
3. The path start point of all victims is inside the critical area? (expected answer depends on how subject draws the critical area)

State of the user interface when the task starts:



Typical state at the end of the task:



Task #2 – Limit data temporally

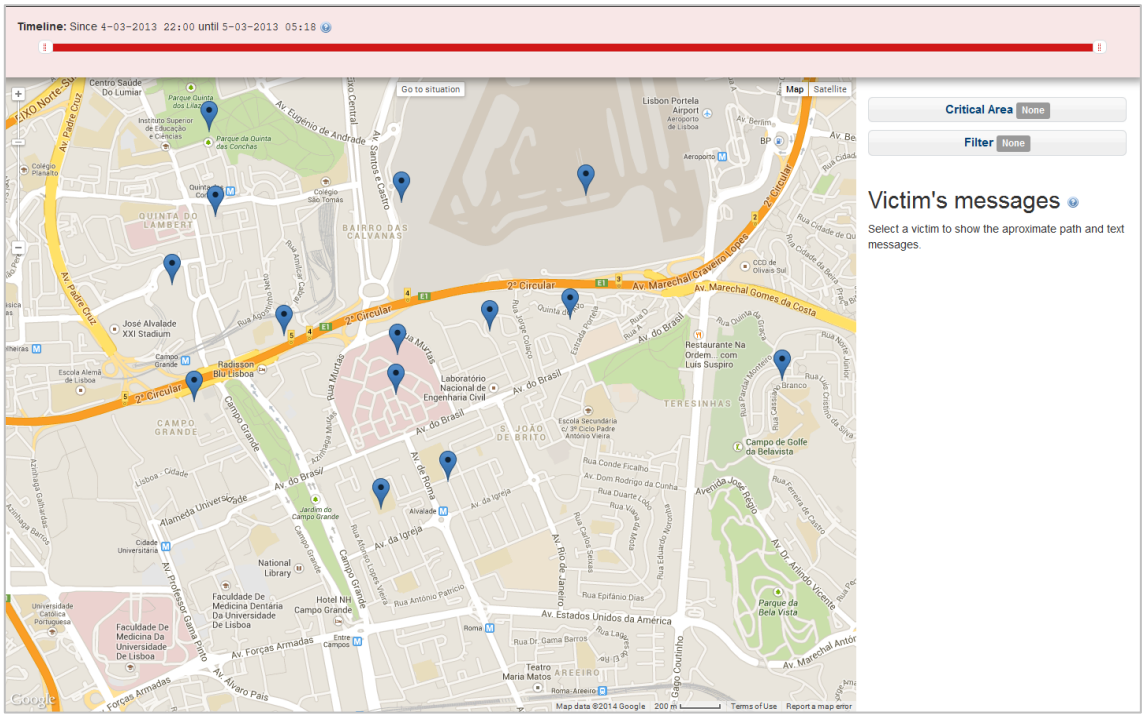
Scenario: Today is March 5, 2013. Yesterday at 10 PM, there was a strong storm in Campo Grande, Lisbon. According to the weather forecasts, the wind reached speeds of 120 Km/h, causing physical damages in the area. This morning you visited your friend Joana, who was on the area. She said to you between 1 AM and 2 AM she sent some messages using LOST-OppNet. However, she was so confused and can't remember what happened. You're going to use LOST-Map to understand what happened.

Task: Access the map. Adjust the timeline to the period indicated by Joana. Locate her position on the map. Check the messages she sent.

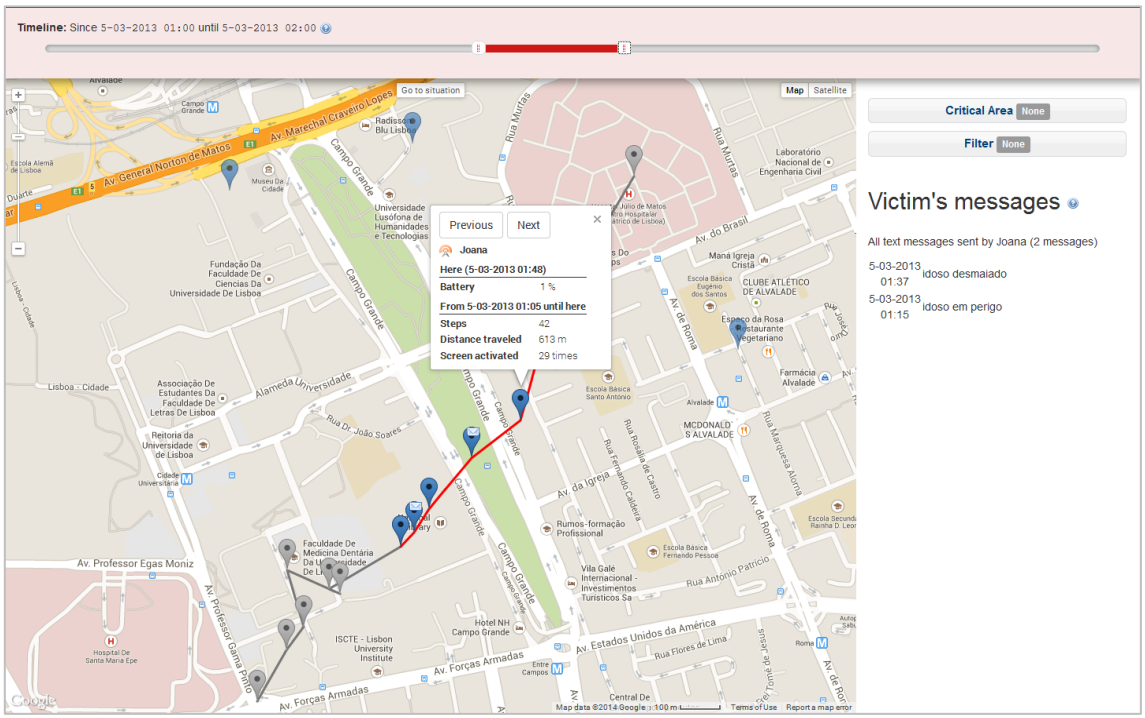
Questions to analyze subject comprehension:

1. What was the temporal space you defined to find the victim? (Expected answer is between 1 AM and 2 AM)
2. How many messages did the victim sent? (Expected answer must be exactly 2 messages)
3. What are the contents of each messages and their time? (Expected answer must be: first message at 1:15 AM and second message at 1:37 AM, telling that an elderly is in danger and needs help

State in the beginning of the task:



Typical state at the end of the task:



Task #3 - Distinguish victims using different criteria

Scenario: Today is December 2, 2013. Yesterday there was a major earthquake in Lisbon, at 9 AM. LOST-OppNet was active on some devices present in the area moments before the disaster. Thus some data of people in the area were collected and

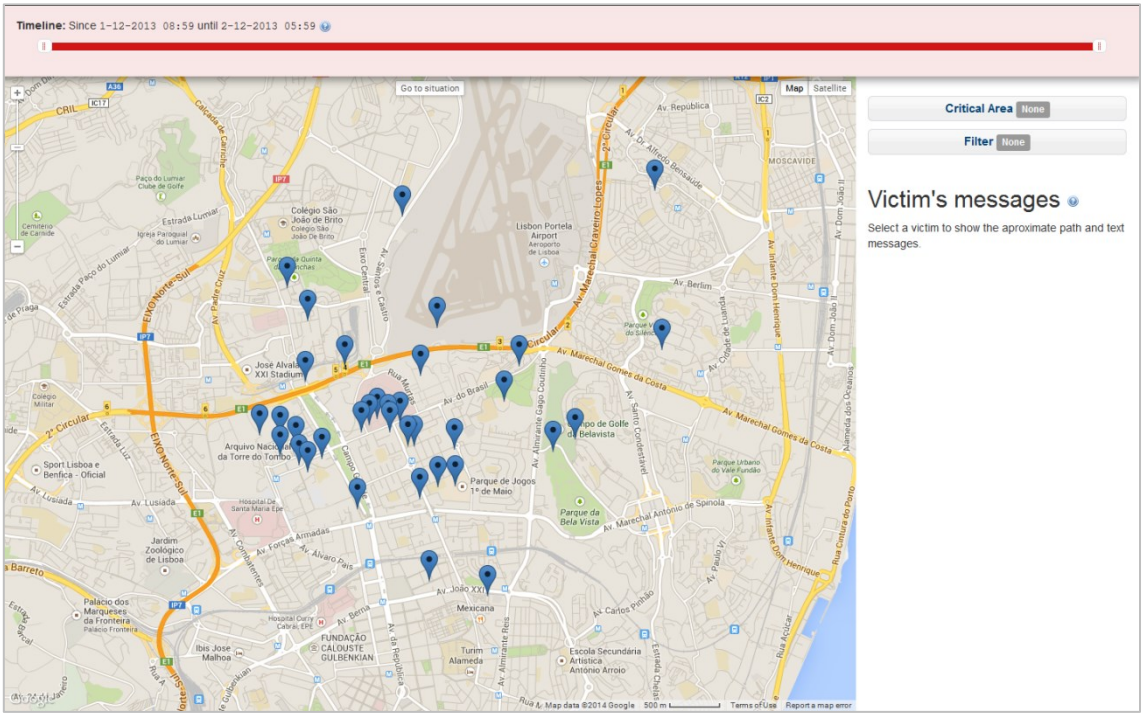
can be viewed on LOST-Map. The recovery work was started yesterday during day time and many of the victims were rescued. It was also reported that some victims managed to escape the area however they were not officially registered as being safe. Today at 6 AM, it is known that there are still some victims that cannot move and are missing. You offer yourself to help other volunteers to find those victims. You decide to use LOST-Map to help you on your mission.

Task: Access the map. Locate victims that there were not already saved (use filters you find suitable to support your task).

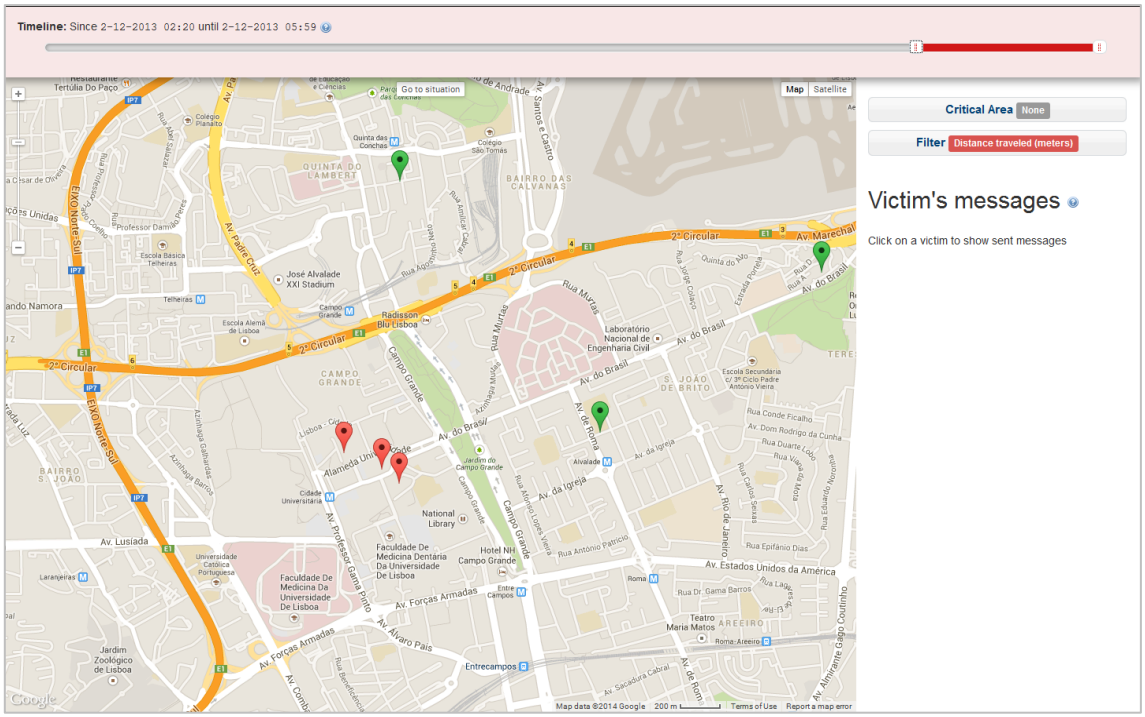
Questions to analyze subject comprehension:

1. How many victims do you find stationary? (Expected answer is 3)
2. Which filters did you use to support your decision? (Expected answer should include "Steps" or "Distance traveled" filters, plus exclusion of people reported as being safe)

State in the beginning of the task:



Typical state at the end of the task:



Annex C. LOST-OppNet study questionnaires

Thank you for participating in this study. The goal of this work is to evaluate a set of components from the LOST project. LOST is a project aiming to provide a functional prototype to allow detection of victims who are affected in natural disasters and provide support to people who volunteer to rescue them. Today we are evaluating the component designed to volunteer rescuers and the component designed to victims.

Assume that a disaster happened at Faculdade de Ciências about 30 minutes ago. It is known that there are victims over the campus and communication channels such as WiFi or cellular network are down. We want to test if LOST-OppNet and Rescue-Oppus can help people to rescue victims and give victims the power to ask for help without relying on the Internet or other communication methods.

Volunteers will be given a tablet running the rescue support application, where they can see victims and some information about them. Victims would be given a smartphone running the victim application, VictimApp for short. With the application they can send messages and their location, allowing a victim to be detected in a map. Some messages are sent automatically in the background, when a connection between a victim device and others (e.g. rescuer device) is available.

To make this study we created a small game. The goal is to rescue 15 people in the least time possible. There are 2 scenarios. One of the participants will be chosen to be the volunteer rescuer and the other will be the victim. When changing scenarios, participants will switch their roles (volunteer will be the victim and vice-versa).

Game rules for the rescuer:

- You can only save victims who are alive. Not all victims are alive, so you must be careful to distinguish them
- To save the victims, you need to be in range of the group you want to save, and then a first-aid symbol will appear on-screen. To save all victims of that group, you must tap the symbol and wait 2 minutes (map will give you further instructions)
- There will be more than 15 victims. This means you can choose some victims over others in order to achieve your 15 victims goal (don’t worry, they will be fine!)
- Each scenario ends when the rescuer saves 15 victims

Games rules for the victim:

- You are expected to use the victim app. You should also send some messages and analyse the application behaviour while sending them.
- On the second scenario, you should walk around the area to indicate that you are alive

Questionnaire Rescuer (scenario 1)

1 – Did you notice groups with people both alive and not appearing to be alive?

O Yes

O No

If you answered "Yes", pick all options that led to your conclusion:

☐ Graph of micro-movements

☐ Graph of screen activations

☐ Victim's last update

☐ Others. Please enumerate: _____

2 – On game start, there were two groups of victims. Which one did you choose first?

O Group near Torre do Tombo

O Group near C5

Please justify your choice (pick all that apply):

☐ The group appeared to be closer to me

☐ The group appeared to have more victims alive

☐ The group appeared to have less victims alive

☐ Others. Please enumerate: _____

3 – Did you notice a new victim appearing on the map, near C5?

O Yes

O No

4 – After saving the victims near C5 the map showed you other two groups to choose from. Which one did you choose?

O Group near C8

O Group near Horta da FCUL/C2

Please justify your choice (pick all that apply):

- ☐ The group appeared to be closer to me
- ☐ The group appeared to have more victims alive
- ☐ Victims had more indication of micro-movements
- ☐ Victims had more indication of screen-activations
- ☐ Others. Please enumerate: _____

Overall, using the Rescue-Oppus was:

Very difficult						Very easy
O	O	O	O	O	O	O

Questionnaire Victim (scenario 1 and 2)

1 – At some moment, you sent a message to the network. Please enumerate the steps you did to send your message and what happened after sending it:

_____

_____

_____

2 – Regarding the message you have sent, do you think that your message reached its destination?

- O Yes
- O No
- O Don't know

If you answered "Yes", please tell how you were informed about message confirmation. If you answered "No" or "Don't know", please tell what do you were expecting to confirm the message was received:

_____

_____

3 – The victim application contains some information regarding connectivity to other devices. Did you notice any activity (at all) on the victim app?

O Yes

O No

If you answered "Yes" please tell the elements/events you remember that allowed you to notice the application was working:

_____

_____

4 – Did you think that VictimApp was relevant to your rescue?

O Yes

O No

Please tell at least one positive/strong point and at least one negative/weak point you found on the VictimApp in the given regarding the situation:

Positive: _____

Negative: _____

Overall, using the VictimApp was:

Very difficult						Very easy
O	O	O	O	O	O	O

Questionnaire Rescuer (scenario 2)

1 – Did you notice groups with people both alive and not appearing to be alive?

O Yes O No

If you answered "Yes", pick all options that led to your conclusion:

☐ Graph of micro-movements

☐ Graph of screen activations

☐ Victim's last update

☐ Others. Please enumerate: _____

2 – On game start, there were two groups of victims. Which one did you choose first?

O Group near Torre do Tombo O Group near C5

Please justify your choice (pick all that apply):

☐ The group appeared to be closer to me

☐ The group appeared to have more victims alive

☐ The group appeared to have less victims alive

☐ Others. Please enumerate: _____

3 – Did you notice a new victim appearing on the map, near C5?

O Yes O No

4 – After saving the victims near C5, some special events happened. Pick all that apply:

☐ There was a victim in the map with a message telling the way was obstructed

☐ The moderator who was with me explicitly alerted that the way was obstructed

☐ There was a victim on the map which trail suggested an alternative way

☐ Others. Please explain: _____

5 – When (if) an additional group appeared on your route to victims near C8, what was your first decision?

O I didn't noticed any additional group

O Immediately save the new group and then the other

O Ignore the new group and save only the other you already planned to rescue

Why did you opted for that decision: _____

_____

Overall, using the Rescue-Oppus was:

Very difficult Very easy

O O O O O O O