People Finder: A Disaster Recovery Application for Delay-Tolerant Networking

Ayman Elkadi
ayman.elkadi@aalto.fi

Aalto University
Department of Communications and Networking
Helsinki, Finland

Special Assignment

Instructor: Teemu kärkkäinen
Supervisor: Professor Jörg Ott

April 7, 2015

Abstract

We present the People Finder System, a delay-tolerant database management system designed to help people find their loved ones in case of natural or humanitarian disasters. The system is build on top of the Liberouter framework [1], a framework deploying the concepts of delay-tolerant networking, allowing messages sent to and from nodes in the system to traverse several wireless devices, reaching their destination without relying on an end-to-end connection between the nodes.

Contents

List of Figures

Chapter 1

Introduction

After any natural or humanitarian disaster people living within the disaster zone escape their homes to seek shelter. Whether these shelters are built by humanitarian movements like the International Red Cross and Red Crescent Movement or by neighborhood volunteers, they are considered meeting points to which food, medicines and other basic supply can reach people in those disaster zones. These meeting points also allow humanitarian organizations to form a database of people who are found alive or dead and who are still missing. Managing such a database from multiple locations and allowing people outside and inside the disaster zone to create, query and update records in the database is a hard, and time consuming process, primarily due to low man power and low resources [18].

Due to the significant importance of database management in disaster zones in saving lives and giving peace to family and friends we sought out to build a Database Management System (DMS) that will help humanitarian organizations to keep track of people affected by the disaster in a more efficient , robust and autonomous way and would allow people anywhere in the world not only to find information about other people affected by the disaster but also to contribute information that they might have obtained from elsewhere that will help in finding people who are still missing.

The trend for data storage nowadays is heading towards 'cloud storage', where databases and applications are hosted remotely on cloud servers. This trend allows users to benefit from more storage capacities and variety of database applications. To use these services users should have constant high-speed broadband connections.

It's hard to imagine today places in the world where there is no Internet

access available for people. Among those few places are disaster zones. When a place is hit strongly by a natural disaster, the forces of the disaster lead to the destruction of available communication infrastructure that was set up in that area. This means that when designing a DMS in a disaster area we will have to move against the trend of 'cloud storage' as we cannot rely on Internet access. Also our design will have to be different than traditional DMSs prior to 'cloud computing' which also rely on Internet access.

Building a network infrastructure from scratch is expensive and time consuming. So our design should use cheap portable equipment to mitigate those issues. In order to do so we will use some delay-tolerant routers [1] and rely on other portable devices to send data back and forth between those routers to gather data from victims in different parts of the disaster area.

The focus of this report is on disaster scenarios which include humanitarian organizations presence, where they would be able to deploy the People Finder system using their resources. In other disaster scenarios that do not include humanitarian organizations presence, designing such a system would be different due to the lack of a well defined logistic structure that humanitarian organizations follow in tackling those disasters.

The rest of the report is structured as follows. The following chapter, chapter 2, is the background. In chapter 3 we drive our system design. Then in chapter 4 we discuss our implementation of the system. And in chapter 5 we evaluate the performance of the system through data gathered from testing the system on Aalto University campus. Finally, we highlight some concluding remarks.

Chapter 2

Background

2.1 Delay-Tolerant Networking (DTN)

Delay-Tolerant Networking is a computer network approach that was initiated by Vint Cerf at NASA for Interplanetary Internet (IPN) [5]. NASA wanted to tackle the delay(minutes to hours) in the connections between communicating devices in space that arises due to interplanetary distances.

TCP , the most widely used transport layer in the Internet today make certain assumptions about a connection between any two parties on the network, such as, the connection is end-to-end, negligible packet loss rate, short Round trip time, the connection is not required any more after a period of inactivity (default 2 minutes), and an infrastructure should exist between the parties to allow domain names to be resolved to addresses (ex. DNS servers) before the actual data could be sent [29]. These assumptions don't always hold in networks with high delay like space networks.

Back on Earth there are some networks that have the same challenges that space networks have concerning delay and intermittent connections, examples of such networks include military, marine, disaster response, and mining.

Due to the need for new type of delay-tolerant networking architecture and protocols, the Internet Research Task Force (IRTF) has launched The Delay-Tolerant Networking Research Group (DTNRG) [6] for addressing the architecture and protocol design of DTN and standardizing them.

In the DTN architecture, we do not assume an end-to-end connection anymore. Instead, routing protocols take a "store and forward" approach, where a message is forwarded over a series of hops until it reaches its desti-

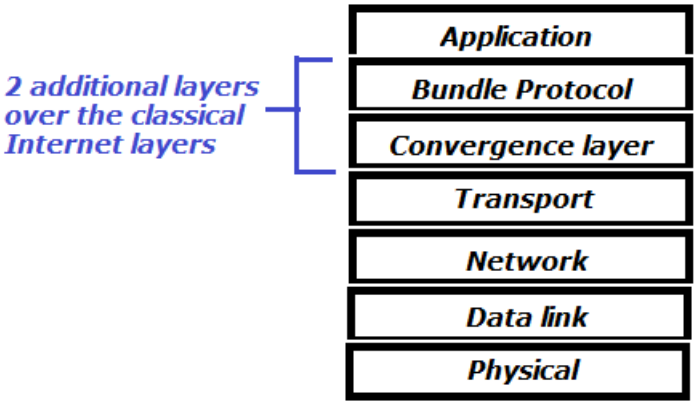


Figure 2.1: Network Layers of DTN showing the Bundle and Convergence layers

nation. Intermediate nodes along the path buffer the message until the next hop becomes available, then forwards the message to the next hop, eventually reaching it's destination. This store-and-forward network structure is implement by a new 'bundle layer' that sits on top of the transport layer, as shown in figure 2.1.

The bundle layer is implemented using a Bundle protocol(BP) [7]. Using convergence layer adapters [26], the bundle protocol is able to encapsulate bundles (data units of bundle protocol) into different transport protocols, allowing for interconnection of different types of networks [27]. If a connection is required on an IP-based network BP uses a TCP/IP convergence layer adapter.

Packets in the new DTN architecture are called bundles. Those bundles consist of a bundle header inserted by the bundle layer, source application's user data, and source application's control information addressed to the destination application describing how data should be handled [25].

2.2 SCAMPI Service Platform

One implementation of DTN architecture and protocols having baseline conformance with the Bundle protocol and full conformance with TCP convergence layer [8] is the SCAMPI service platform [9] developed in Aalto

University to allow opportunistic communication between Smartphones.

Later SCAMPI became part of a larger framework called Liberouter [1] which combines the SCAMPI service platform with a Raspberry-pi [10] hardware platform having a Linux distribution installed, to form a cheap , do-it-yourself 'opportunistic router' that can be used to route messages between devices in delay-tolerant networks.

The SCAMPI service platform is implemented as a Java application that can run on desktop computers and as an Android application that can run on Android devices. The application works as a background process that tracks available communication interfaces (e.g., local IPv4 addresses) and opens links through those interfaces.

We will refer to an 'opportunistic router' by using the name 'Liberouter' and refer to 'SCAMPI service platform' by using 'SCAMPI platform' for the rest of the report.

The Liberouter appears as an ordinary open Wi-Fi access point to mobile phone users within coverage of the Liberouter. After connecting to it , the users are able to download the SCAMPI platform on their devices along with other applications already on the router through the router's captive web portal.

In order for applications to use the SCAMPI platform running on the same device, a TCP-based API was developed. The API can be used by native applications to publish and subscribe to messages in the Liberouter network. Application developers using the API will work with SCAMPIMessages. SCAMPIMessages are 'self contained' application layer objects that are mapped to bundles of the Bundle Protocol at lower layers. Nodes who subscribe to services will receive copies of published SCAMPI Messages from those services. [1]

The SCAMPI platform has already been used in a practical use case connecting distant mining equipment with an operator console through intermediate nodes such as trucks and people's mobile phones [11]. In this report we will use the platform in another practical use case, namely disaster response.

Using DTN concepts in disaster response networks has been explored in other projects as well. In [12] the authors introduce a new disaster information system to provide a unified user interface showing the state of a disaster varying in time. In [13] DTN concepts where used to allow a suite of applications intended to build a network of citizen disaster responders to work without a continuous Internet connection. And finally, in [2] the au-

thors Propose a mobility model that includes the impact of a disaster on the movement of people in the area, transportation networks, and disaster-relief vehicles.

2.3 Google Person Finder

Google has developed an open-source web application called 'Google Person Finder' [14] to help people find their missing loved ones in the aftermath of humanitarian or natural disasters. The application was launched after an earthquake hit Haiti in 2010. Google opens a new repository for each disaster it chooses to add to the Google Person Finder web application. Anyone that has access to the Internet can visit Google Person Finder's website and add information about missing people in the corresponding disaster repository.

Having access to the Internet maybe easy for people outside the disaster area, but for those inside the disaster zone, who are most affected by the disaster; they have real difficulty in accessing the Internet due to unavailability of communication infrastructure in the aftermath of a disaster. Accordingly, we would use DTN concepts in extending a similar application to allow people in a disaster area to use such an application without requiring an Internet connection to be present.

2.4 PFIF

People Finder Interchange Format (PFIF) [15] aims to facilitate the distribution of digital information about missing people between different people, organizations and databases through an open data standard. In addition to facilitating distribution, it facilitates information convergence as well, through convergence of information about the same missing person coming from different sources. Also it supports traceability of data. Where information about a certain person can be traced back to the people (or organizations) that originally provided it, allowing readers to validate the trustworthiness of the data [15].

PFIF consists of person records and note records. Person records contain meta-data information (9 fields) that allow users to be assured of the reliability of the record, and other information (16 fields) that allow the person to be identified. As for note records, they include meta-data information (8

fields) as well, and status information (6 fields) that allows users to updated information about a person's status. One person record can have one or more attached note records.[15]

For a list of fields that make up person record and note record information described in PFIF 1.4 Specification [15] please refer to APPENDIX, and for a more elaborative description about what each field represents please refer to the specification [15].

Original repositories are repositories in which records where initially created. While records can be sent to other repositories, the authority on the records remains in their original repository, and any updating or deletion of records have to be done by the records' original repository. In PFIF there is no single central authority, each repository can choose to accept PFIF records exported to it by other repositories or not based on its own assumptions like the level of trust it has for the sending repository. Each repository can be comprised of 'original records' which are records to whom the repository is the original repository and 'clone records' which are records originally created in other repositories. A record's original repository is identified by a domain name which is a name that person_record_id and note_record_id fields start with. Inside a repository each PFIF-aware application can implement its own database schema for storing PFIF data. One suggestion given in the specification is to store PFIF records in two tables, a person table for the person record information and a note table for the note record information. As for exporting records each application should export the data in the form of 'PFIF XML document format' like the one given in the specification so that other PFIF applications could easily parse the received data and store them in their relevant fields in their databases.[15]

Figure 2.2 shows how PFIF is used to send and receive records between two organizations having repositories for collecting information about missing victims in a disaster scenario, and how users of such organizations interact with the repositories.

For a PFIF XML document to be valid it should include a single PFIF element consisting of one or several person or note elements, where each person or note element would include 'child elements' which are the fields that make up a person or a note record. The mandatory fields of a person element are: person_record_id, source_date, and full_name. And for a note element they are: note_record_id, author_name, and source_date fields. The other fields are all optional. The reason for not having person_record_id as a mandatory field in the note element is that a note element in a PFIF XML

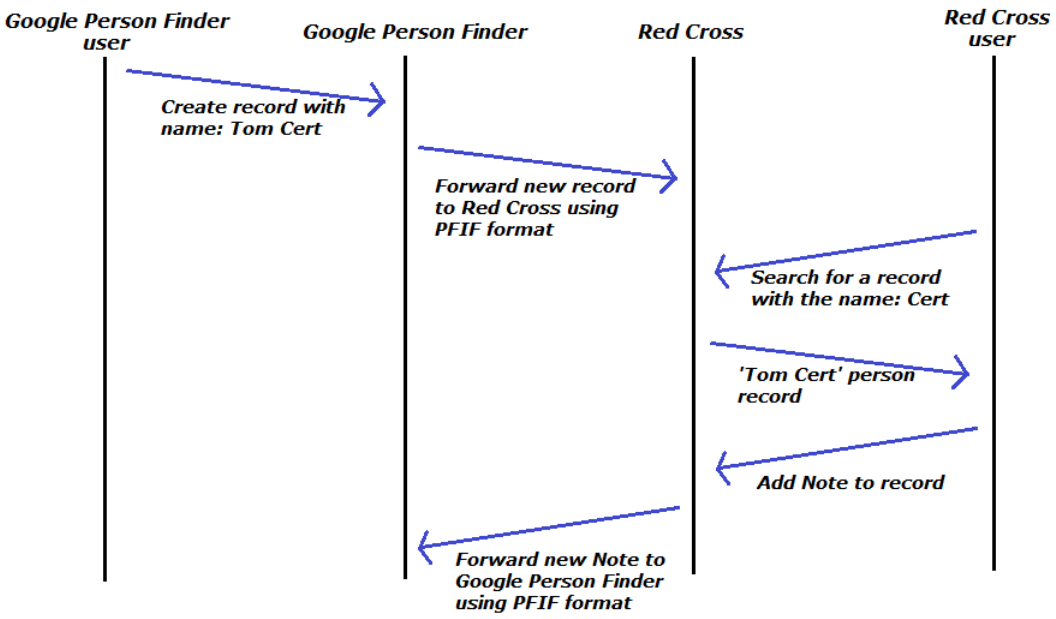


Figure 2.2: PFIF used to send/receive records between two organizations

document can be included inside a person element. But if a note element exists outside a person element, it must contain a person_record_id field.[15]

The following lines show a PFIF element with a single person record and a single note record in a PFIF XML document with only mandatory fields. Notice as the note element is outside the person element so it has to include the person_record_id field.

start = element pfif:pfif { person* & note* }

person = element pfif:person{
element pfif:person_record_id { record_id } &
element pfif:source_date { time } &
element pfif:full_name { text } &
note*
}

note = element pfif:note {

```
element pfif:note_record_id { record_id } &
element pfif:person_record_id { record_id } ? &
element pfif:author_name { text } &
element pfif:source_date { time } &
}
```

Also the format in which each PFIF application would use to display person and note records to its users is left to the application's developer to choose.

Chapter 3

Design

In this chapter we first present how disaster recovery operations are generally structured, then we look at the design considerations that we need to meet when designing a DMS in such a structure. We then move on to present an overview of the architecture of the People Finder Database Management System and its elements in a basic scenario; finally we discuss how this architecture addresses the design considerations mentioned earlier.

3.1 Disaster Recovery Operation Structure

In [2] the disaster recovery operation following a disaster is modeled by three components: Location of participating centers, moving agents, and Communication between centers. This model is derived from documents prepared by the Federal Emergency Management Agency (FEMA) [20], part of the department of Homeland security in the United States; outlining guidelines [21] and functions [22] of nation-wide disaster recovery operations.

The location of participating centers depends on the type of each center. Different types of centers are modeled, but we mention only two of them which are used later in our disaster recovery scenario.

The two center types are:

1. Main coordination centers: These centers are responsible for coordinating the recovery operation, supplying food and equipment to the rest of the Evacuation centers. These are generally located in an area which has good

transportation routes, to be easily accessible; and good service infrastructure.

2. Evacuation centers: These centers are places where people affected by the disaster can keep cover and find basic necessities like food and aid services. Evacuation centers are usually created in each affected neighborhood, this way people in the disaster area can reach those centers without much difficulty from their homes. These centers are referred to as 'Shelters' in our scenario later in this section.

In addition to the centers, the recovery operation includes moving agents. Moving agents consist of one or more transportation types that are used to carry supplies, aid workers and other services between centers.

Moreover, Centers should have the means to communicate together and share digital information that can help in the recovery process.

3.2 Design considerations

Now that we have looked at the structure of the disaster recovery process, we have to look at the design considerations and challenges of designing a DMS that would function in such a process.

Here are some design considerations that have to be taken into account when designing such DMSs.

3.2.1 Establishing Communication Infrastructure

Infrastructure Availability

In a disaster area, it is likely that communication infrastructure such as cellular towers and Internet fiber cables are affected by the disaster, so we cannot base our design on any infrastructure that was available in the area prior to the disaster.

Scalability

Due to long distances between shelters, and low resources that humanitarian organizations have in a disaster area, we cannot install routing nodes stretching from one shelter to the other whether it is wired or wireless. We

need a delay-tolerant system that will allow just few numbers of 'store-carry-forward' routing nodes equaling to the number of shelters in a disaster area to be installed, and rely on transportation services belonging to humanitarian organizations or other rescue groups to carry additional 'store-carry-forward' router nodes for spreading information from one shelter to the other. This allows the design to be scalable. [2]

3.2.2 Accessibility

The system should be easy to read from and write to by everyone, so it should allow accessibility through common devices like Smartphones, and for reaching people outside of the disaster zone, one or more gateways to the Internet should be installed in some central location where information from all shelters are gathered in one place having access to the Internet. Those locations should be accessible by transportation services carrying the additional 'mobile routers' as well. As main humanitarian organization centers supply food, medicine, etc..., to the shelters they are usually seated in places where they have access to communication infrastructure of some sort, so the gateways can be installed there [3] [4].

3.2.3 Interfacing With Other Databases

Finally, the database system should allow easy interfacing with other global digital databases for two reasons, one, to allow more spreading of information about people in a disaster and second, to allow the system to be used side by side other digital database systems that humanitarian organizations use today and would be reluctant to replace right away.

After mentioning the design considerations, next we present an overview of the architecture of the People Finder Database Management System and its elements through a basic scenario, and then we discuss how this architecture addresses the design issues mentioned above.

3.3 Basic Scenario

We will use a basic scenario to help build our design of the People Finder system. Although our scenario is limited to few elements and that in a real-life scenario the number of elements would be more, elements would interact

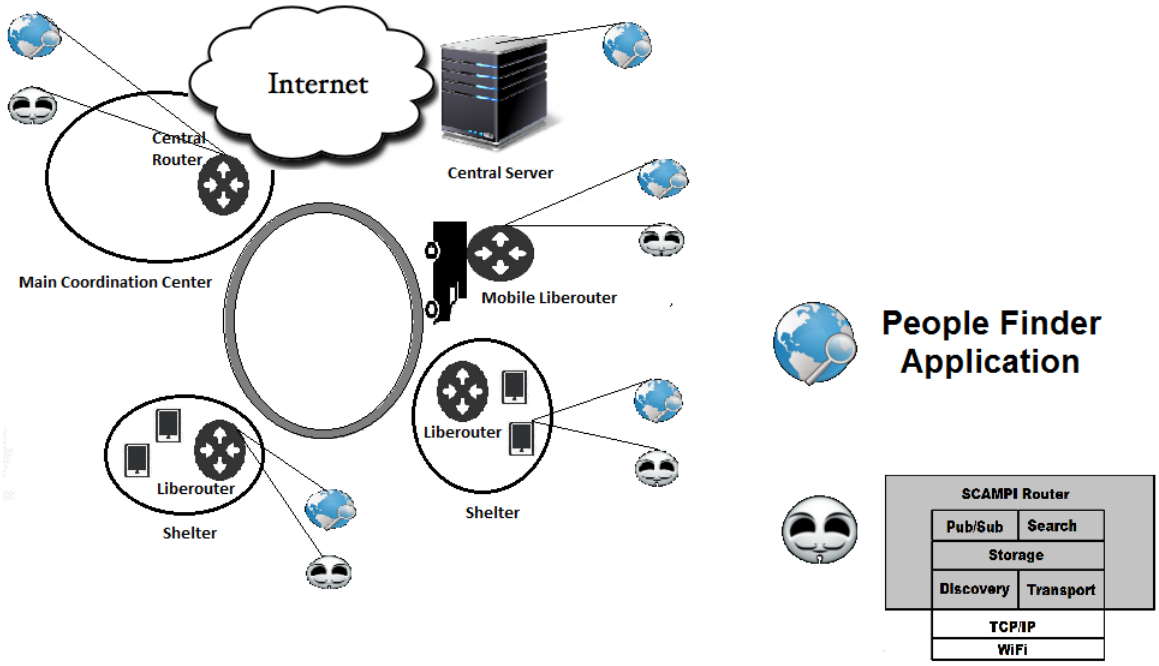


Figure 3.1: People Finder System Architecture

with one another in the same way in both scenarios, so implementing such a design in a real-life scenario would be straightforward. Our scenario is based on two shelters (Evacuation Centers), one main coordination center, and one moving agent, which is a truck in this scenario. A road exists linking the shelters with the main coordination center which the moving agent takes to supply food, equipment and share information between the main coordination center and both shelters.

Figure 3.1 shows an overview of the People Finder system architecture being used in a disaster zone by a humanitarian Organization.

The architecture is composed of:

1. One or more fixed Liberouter(s) - located in each shelter running the SCAMPI platform and the People Finder application.

2. A mobile Liberouter - that is carried by the type of transportation that the humanitarian organization already uses to deliver food and water

from main coordination centers to the different shelters in the disaster area. This router is used to send peoples' records back and forth between different shelters, so that each shelter would have the complete database of missing people. This router can be installed on board of the transportation facility or carried by one of the members on board. Due to a feature in the Liberouter framework that allows Liberouters to leverage nearby Smartphones as additional routers[1], the router can be replaced with a Smartphone having the SCAMPI platform and the People Finder application installed in case a Liberouter wasn't available.

3. One or more Central Router(s) - located in the main coordination centers. Central routers update their databases from the mobile Liberouter and then send the updated database to a Central Server through the Internet. These routers update their databases from the central server as well and send it to the mobile Liberouter so that all shelters would update their databases accordingly.

4. A Central Server - that is accessible to the public through the humanitarian organization's website so that any person could create or query information about a missing person. The server should be located in a safe place away from the disaster area, having a fast broadband connection . There could be more than one central server in different locations to have no single point of failure in the system.

5. Smartphones - having SCAMPI platform and People Finder applications installed. These are used by the humanitarian members in each shelter to input data into Liberouters. So the victims in each shelter are divided over several humanitarian members within that shelter, where each member would input information about each victim into the People Finder application running on his Smartphone and each time he finishes entering the information, he pushes a publish button that updates the database of the People Finder application on the phone as well as on the fixed Liberouter , providing the phone is within coverage of the router. Moreover, victims who already have their own Smartphones can install the People Finder application and the SCAMPI platform application from the web portal of the Liberouter and start updating the database of the fixed Liberouter present in the shelter themselves

3.4 Meeting Design Considerations

3.4.1 Establishing Communication Infrastructure

As we do not want to rely on any communication infrastructure that was available in the disaster zone prior to the disaster , we have to support our own infrastructure. Our infrastructure has to be set up as quickly and cheaply as possible aligned with humanitarian organization's aid process , requiring no(or limited) additional resources and no change in the disaster management structure of the humanitarian organization.

Choosing Network Devices

The fastest and cheapest way to deploy a network infrastructure on a new site is to use available devices on ground, in this case its peoples' Smartphones. As most humanitarian members have their own Smartphones, as humanitarian organizations nowadays use Smartphone applications to facilitate communication among its members [16] ,we can use their devices' built-in Wi-Fi adapters to build a mesh network formed of Smartphones communicating in an Ad-hoc fashion.

Many projects have used Smartphones to build mesh networks, such as [17] , however, they have all ran into these common problems when using Wi-Fi Ad-hoc mode for direct communication between Smartphones:

1. Different vendors support different Wi-Fi adapters, resulting in interoperability issues between different mobile brands.
2. Continuous peer discovery drains phone's battery, so it shouldn't be used for long.
3. Mobile phone applications leveraging Wi-Fi Ad-hoc mode require users to root (on Android phones) or jail-brake (on IOS phones) their phones which is undesirable by many users. [17]

The issues of Wi-Fi Ad-hoc mode in mobile phones call for 'opportunistic routers' that gather information from Smartphones in each shelter and share this information with other opportunistic routers in other shelters through a 'mobile opportunistic router'. These 'opportunistic routers' should store the data and forward it to 'mobile opportunistic routers' when they are in coverage.

Moreover, These opportunistic routers should be cheap , light, mobile and allow storing of thousands of peoples' records.

The Liberouters fit that description, so we are using them in our design.

Relying on a mobile router to forward messages between the different shelters and the Main coordination centers avoids the need to build complex network Infrastructure, and allows quick deployment of such a network.

3.4.2 Accessibility

Since main coordination centers have Internet connection so with this architecture we are able to connect people from the disaster zone (who are staying in shelters) with the outside world , where each side can get or contribute valuable information from/to the other side about a missing person.

3.4.3 Interfacing With Other Databases

To allow more involvement between people in the disaster zone and the outside world, the database system allows easy interfacing with other global digital databases such as Google Person Finder [14]. This is done by using People Finder Interchange Format (PFIF) open data standard for information about missing people that was discussed earlier in the previous chapter.

3.4.4 Routing between elements of People Finder System

As the identity and number of Smartphone devices communicating with each liberouter in the system is not known prior to a disaster and would probably change throughout the life span of the disaster recovery operation, and as we need all those devices to receive all of the messages generated by or delivered to each liberouter device, accordingly we are using 'epidemic routing' [28] for routing messages wirelessly between the different wireless devices in the system.

In epidemic routing two communicating devices would start by exchanging their 'Summary Vectors'. A 'Summary Vector' is a message containing a summary of all message Identifiers corresponding to all messages a device currently stores [28]. This way each device could identify which messages it is missing and request them. Epidemic routing ensures that all devices in the network would receive only a single copy of each message (no duplications),

and that they would eventually receive all the messages that other devices have sent.

Even though spreading of unnecessary messages is limited due to exchange of Summary Vectors, Summary Vectors could still be quite long in populated disaster zones, limiting the scalability of the routing algorithm. To help in mitigating the scalability issue, we can set a limit over the number of hops a message can take. In our scenario we had two shelters, one main coordination center and a single moving agent moving between both center types. So we can calculate the maximum number of hops a message would need to take inside the disaster zone to reach its destination, which could be set in our scenario to be 4 hops. First hop between a center (shelter or main coordination center) and the moving agent, then between the moving agent and another center (shelter or main coordination center). Third hop is between the center which had already received the message from the moving agent, and Smartphone devices (running people finder application) in range. And finally the forth hop is added as a safe guard in case the mobile router on his way to one of the centers would pass in the range of a Smartphone device which had just received a message from a Liberouter, this way when the moving agent receives this message from the Smartphone device, it would have already made two hops. The hop limit can be tailored to suit any scenario.

3.4.5 Domains

All entities of the system(in a single deployment instance) are part of a single authoritative domain, which we call SCAMPI domain. All records originated by any device running the People Finder application have unique person_record_ids and note_record_ids starting with ' scampi-people-finder/', only those are accepted by the different entities of the system, except for the Central Server which accepts records from databases outside the SCAMPI domain. Externally we can interface with other authoritative domains outside our SCAMPI domain through importing their PFIF records or exporting ours. The only entity of our system that is allowed to interact with devices located in external domains is the Central Server.

3.4.6 Security

To identify users inside the SCAMPI domain and authorize them, so that all system entities can accept their records, we use an asymmetric cryptography scheme within the domain. This is done by a Certificate Authority (CA) (we use the central server in our scenario), which is used to issue digital certificates to authorized users within the domain. All authorized users before starting to use the people finder application contact the CA to issue them a digital certificate and also to get digital certificates of all authorized users within the Scampi domain. How an authorized user would prove his identity to the CA so that the CA can issue him a digital certificate is left for the organization deploying the People Finder system to decide on how it would implement this. Digital certificates certify that an authorized user has ownership of a specific public key(contained in the certificate). An authorized user generates a private and a public key before starting to use the people finder application. The private key is kept secret, and only known to the authorized user. The public key is available for anyone who wants to communicate with the authorized user and is obtained through the CA. Each authorized user signs its own messages with the digital certificate after encrypting the data (or a hash of the data) with it's own private key. Receivers can verify the authenticity of a message by using the signer's public key, accepting the user as the rightful owner of the record if they can verify the message signature. This facilitates the process of updating/deleting of records as each record can be traced back to its original user. Records are allowed to be updated or deleted only by the users that created them in the first place.

The system allows as well creation, storage and sending of records that are not signed. The reason behind that is the fact that we support victims inside the disaster zone to engage in creating records by their own Smart phones. Since they might have no access to an Internet connection in the time they download the application and start using it, we cannot provide them with digital certificates. As their messages cannot be authenticated and traced back to them, they cannot update or delete records. A receiver device receiving an unsigned message will store it normally in its database but mark it as 'unauthenticated', so that the message is not considered in case of update or deletion. In this case another form of verification could exist, namely face-to-face verification, where an authorized user who knows an unauthorized user could verify records sent by the unauthorized user after receiving it on his device. This will be forwarded to the rest of the system

nodes, who will consider the record to be an authenticated one.

3.4.7 Database Management

We have our own implementation of storing person and note records inside People Finder application's database, which follows the suggestions given by the PFIF standard on the format to use for storing people and note records [15] with some additional fields that help each application in keeping track of which records lying in its database, have already been sent to other entities and which records still need to be sent. As for exporting and importing records internally, between all entities of the system; we use the 'PFIF XML document' format introduced earlier in the previous chapter.

Detecting and Removing Duplicates

So as not to overwhelm all the databases in the system with different Person records about the same person we need to find duplicate records and leave only one record for each person. The People Finder application on each entity of the system would calculate the similarities between two person records by comparing same fields of both records (ex. Phone Number) and based on that gives a decision on whether to delete one of the records or not. For example when it comes to similar records, the record having the higher person_record_id could have higher priority, so the record with the lower person_record_id will be deleted. If two records were found to be similar, the application will merge the 2 records into one , including non-similar data from both records (Ex. 2 different E-mail addresses). The new record would still store both person_record_ids even if it will use the higher one, this will allow updating and deleting of duplicate records that have lower ids by the people who created them. This feature would be implemented on all five entities of the system to limit the number of records sent between all entities. To decrease the amount of duplicate records from the first place, the People Finder application on the Smartphone could show the user a list of possible similar person records already in the database of the device when he/she is entering a new person record on the device so that the user can look if a record with the same name already exists before creating a new one.

Chapter 4

Implementation

In this chapter, we discuss our implementation of the different elements of the People Finder system.

4.1 Android Application

We are using the People Finder android application that was developed here [1] . The application allows users to search for person records and notes attached to each person record. If users have information about a missing person, they can submit a new person record. In case the missing person was already in the database, they can submit a new note record for that person record.

Each time a person record or a note record is inserted into the application, this record is updated in the internal database of the application, in the same time it is sent through the SCAMPI API to the SCAMPI platform running beside the People Finder application on the same device so that other devices within coverage who have subscribed to messages from the People Finder application will receive updates of the newly inserted records, updating their databases accordingly.

The application can be installed from the Liberouter's web portal right away in the disaster zone.

In the following, we will describe the application's main classes.
Application's main classes:

1. AppLibService.java

It is the service responsible for handling messages received by the SCAMPI platform, which are messages sent by other devices running SCAMPI platform. It is also responsible for building SCAMPI messages from unpublished records that the user enters into the application's database, then sending them to the SCAMPI platform. After that the records could be sent to other devices running the SCAMPI platform. It uses a SCAMPI API protocol part of the SCAMPI API to connect to a local router on the device. [1]

2. DatabaseController.java

It is the class responsible for creating the database, creating tables in the database, inserting and updating records in the tables, and accessing records already in the database. It is the only class that can interact with the database. All other classes that want to insert, update or access records should do it through this class. The interaction with the database is done in an asynchronous fashion.

As suggested in the PFIF specification [15] the database created has two tables one for person records and the other for note records. All the person record and note record fields, mandatory and optional, mentioned in the PFIF specification (version 1.4) are included in the database. In addition to those fields two more fields are added to the Person records, one is used to save the location of photos on the local file system, and one is used to signal whether the record was published(seen by SCAMPI platform) or not. Also two more fields are added to note records for the same reasons.

The database engine used here and used by all entities of the system is SQLite database. SQLite is a popular choice for embedded database implementations due to its high speed (2-3 times faster than MYSQL [19]), small memory footprint, functionality and above that it is open source. SQLite is not a good choice when it comes to server database implementations, that is why the use of SQLite in our Central server is just for testing purposes, for real life implementations we will have to use a more suitable database engine.

3. Other classes

Other classes in the application deal with the user interface which allow users to search a person or a note record in the database and add records as

well, but we will not discuss those as we are focused on how the application stores records and how it interacts with the SCAMPI platform.

4.2 Liberouters

The Liberouters (fixed and mobile) are implemented on Raspberry-pi boards connected to a Wi-Fi USB card, and booted from an SD Card having a Linux distribution along with the SCAMPI platform. The Liberouter could take its power from a normal power supply (in case of fixed Liberouters) or could run on batteries (in case of mobile Liberouters). The fixed Liberouters can also run on batteries depending on the condition of the electric supply in the shelters.

Apart from the Liberouter image that contains the SCAMPI platform running on the Raspberry-pi platform , People Finder Java application was developed to run beside the SCAMPI platform on the router. This application is similar to the People Finder android application but developed in Java programing language to be able to be run on Linux distributions running on the Liberouter. This application has no user interface to input records in its database and relies on input from other devices running People Finder applications.

The main reason for having an application beside the SCAMPI platform and not relying only on SCAMPI for storing and forwarding records is that we implement some functions which require record manipulation, such as duplicate detection. These functions are not included by default in the SCAMPI platform.

The application has two java classes, other than the 'main', namely AppLibService.java and DatabaseController.java. These two classes perform the same functionality as their counterparts in the Android application but without the user related functions.

The default settings of the SCAMPI platform are used in all Liberouters. Discovery mode is set to IP Multicast, The bundle service is set to TCP convergence layer, routing is set to Epidemic routing, and topology is set to full mesh.

4.3 Central Router

For the Central Router located in the Main coordination center we have added an extension to the People Finder application used in the Liberouters to allow the router to periodically open a TCP connection with the server, check for updates in its database, and if new records are found it sends them right away to the Server, then it waits for updated records from the Server to be received before closing the connection.

Data fields of Person and Note records in TCP packets are organized and formated similarly to the PFIF standard format, with each field separated by a keyboard 'Tab', to help in parsing the data in the receiver side. An additional field containing a 'carriage return' character is added to the end of each record signaling the end of the record, allowing optional fields to be added later on if needed.

So in addition to the two Java classes AppLibService.java and DatabaseController.java which correspond to the People Finder Android application classes, we have a new class namely CentralRouter.java that communicates with the database directly without the help of DatabaseController, to check for records in the database that have not yet been sent to the server, and if it finds one or more, it sends them to the server. It also receives new records from the server and add them into the database directly. So here the DatabaseController is not the only class to interact with the database.

For marking records to identify the record that has been sent to the server, we add an additional field to both the Person and note records. Whenever a record is sent to the server, the field is changed to reflect that.

4.4 Central Server

An application running on the server will receive updated records from Central Routers and update the Server's database accordingly, and send new records found in the database to the router after accepting TCP connections from the routers.

In the server side we have no SCAMPI platform, so there is no AppLibService. And instead of a single class for database interaction (DatabaseController.java) that was used previously here the database interaction and server/client connection is done in a single class named CentralServer.java.

The class implements a multiple socket server so that it can support

multiple connections at the same time from many Central Routers and from devices situated in domains outside the SCAMPI domain.

Finding and deleting duplicate records feature was not implemented in any of the applications.

Chapter 5

Evaluation

After implementing the different elements of the People Finder system, we ran a test at Aalto University's Otaniemi campus located in Espoo, Finland, to evaluate the performance of our People Finder system.

5.1 Test Scenario

Our test scenario was based on the People Finder System Architecture shown in figure 3.1 in chapter 3.The test scenario is shown in figure 5.1. It is composed of a fixed Liberouter located at 1, a Central Router located at 2 and a mobile Liberouter kept inside a car going back and forth between locations 1 and 2. The route the car takes between locations 1 and 2 is shown by a blue line in figure 5.1.

Locations 1 and 2 are distant enough to prevent both the fixed Liberouter and the Central Router from directly communicating with one another.

In addition to those three elements, one of the servers on campus ran a Central Server application which communicated with the Central Router over the campus Wi-Fi network.

As for generating records, two Android mobile phones running an instance of the People Finder application and SCAMPI where used. One was located beside the fixed Liberouter(at location 1) and the other was located beside the mobile Liberouter (inside the car). To automate the process of generating records we added a feature to the People Finder application running on both mobile phones that allowed records to be generated periodically every minute. In our test, a total of 84 People Finder records where generated by both

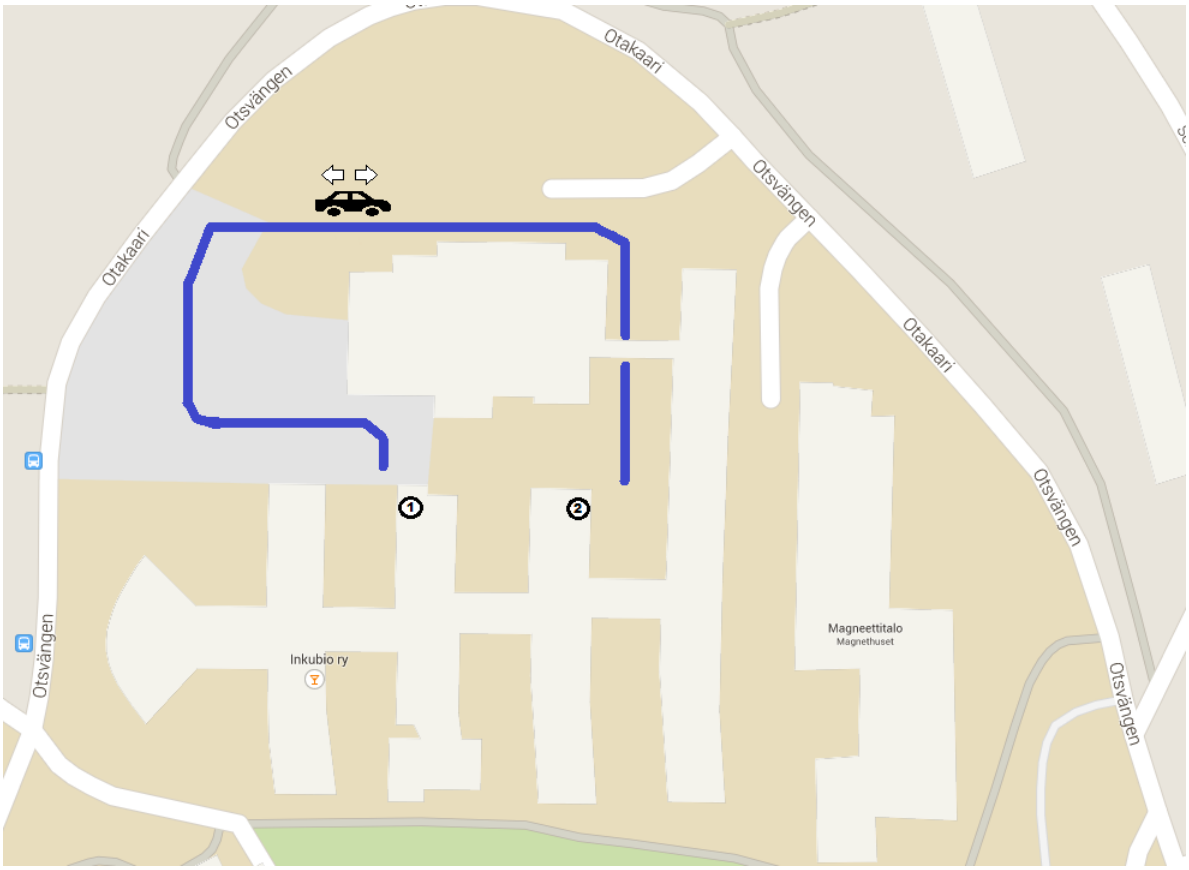


Figure 5.1: Test Scenario

mobile phones.

To be able to track each record throughout the test, each node recorded the time when a record was received by that node.

5.2 Test Results

Accordingly, for each node we have plotted the delay faced by each People Finder record from the time the record was generated until the time it was received by the respective node. The plot is shown in figure 5.2. The X-axis is the record's relative generation time, which is measured from the generation time of the first record generated in the test signaling the start time of our test. While the Y-axis is the delay.

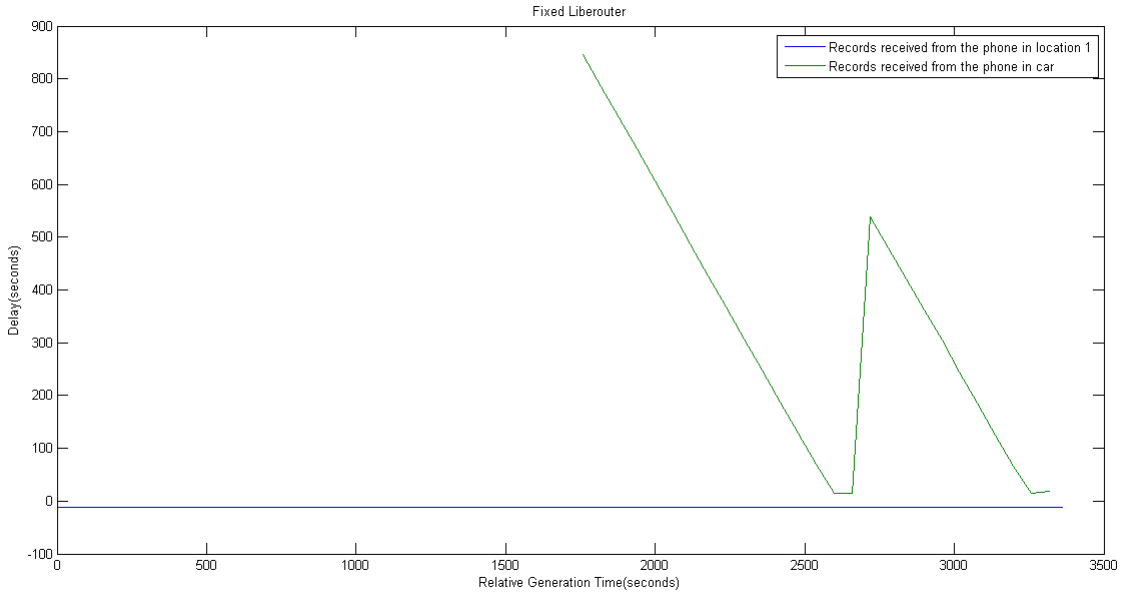
For the fixed Liberouter (at location 1) plot, due to the close proximity of one of the mobile phones to the fixed Liberouter there is almost no delay between the time a record was generated and the time it was received by the Liberouter. This can be seen from the blue curve in the fixed Liberouter plot.

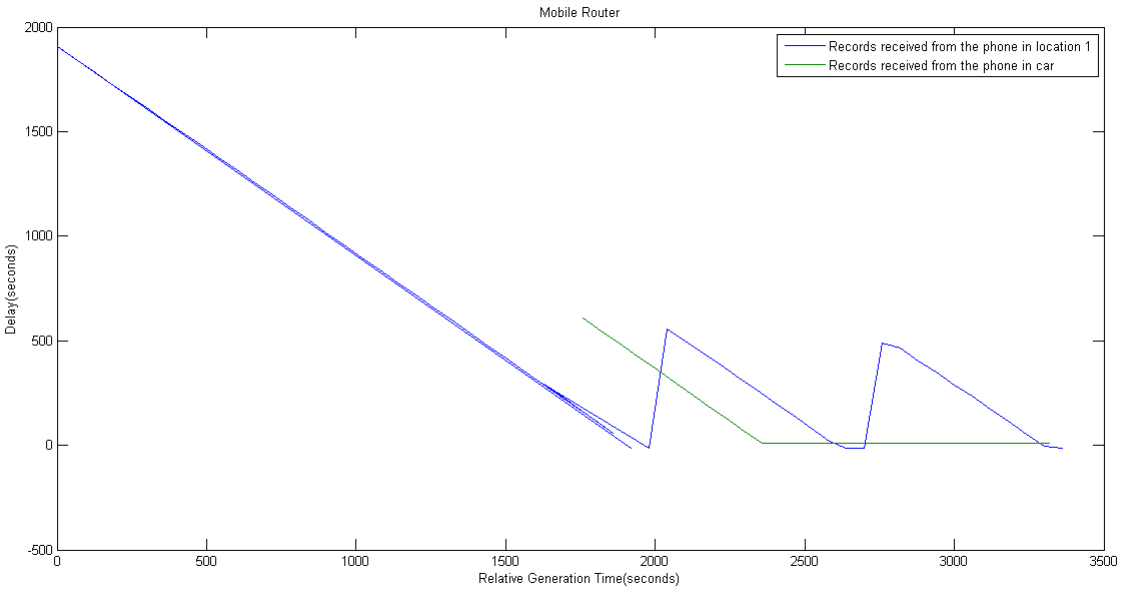
Note that the delay shown in the blue curve is slightly less than zero, which is due to small time differences (less than 13 seconds) between both devices.

As for the green curve, whenever the car approaches location 1, the records already received by the mobile Liberouter would be sent to the fixed Liberouter. Records having the lowest delays are those generated when (or just moments before) the car came within range of the fixed Liberouter.

For the mobile Liberouter plot, the first record received from the fixed Liberouter took a delay of about 1900 seconds which came at a time of first contact between both Liberouters. This can be seen at Relative Generation Time of zero from the blue curve. As for the green curve, we started generating records from the mobile phone (kept in the car) before turning on the phone's Wi-Fi adapter. Once we opened the adapter, all the records which had been previously generated by the phone were sent to the mobile Liberouter. Afterwards, all preceding records had almost zero delay due to the close proximity between the phone and the mobile Liberouter.

For the Central Router plot, the first record received (shown from the blue curve) took a delay higher than that of the mobile Liberouter, the difference between those times is the time taken by the car to reach location 2 from the time the mobile router reached location 1, neglecting transmission delay.


Fixed Liberouter
Records received from the phone in location 1
Records received from the phone in car
Delay(seconds)
Relative Generation Time(seconds)
900
800
700
600
500
400
300
200
100
0
-100
0
500
1000
1500
2000
2500
3000
3500


Mobile Router
Records received from the phone in location 1
Records received from the phone in car
Delay(seconds)
Relative Generation Time(seconds)
2000
1500
1000
500
0
-500
0
500
1000
1500
2000
2500
3000
3500

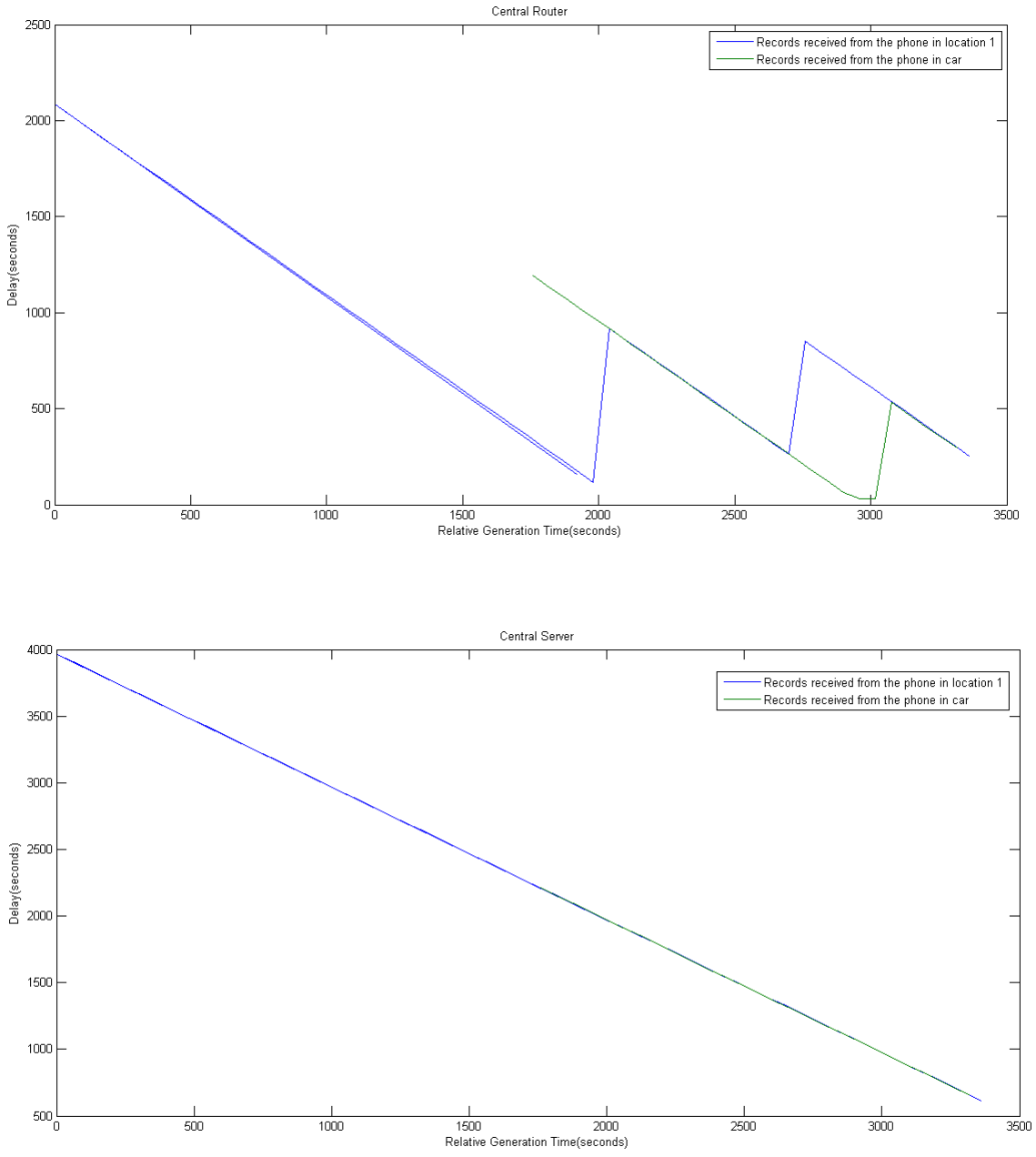


Figure 5.2: Test results showing the delay in receiving People Finder records

For the Central Server, although the plot should have been similar to the Central Router plot as the transmission delay is within milliseconds range, we can see that it differs. The reason behind this is due to the fact that the Central Router had lost connection to the campus Internet at the start of the test, and was not able to reconnect until a late period in the test. After reconnecting, all the records which have been received by the Central Router were sent to the Central Server.

5.2.1 Metrics

The first metric that we can use to evaluate our implementation of the system is the bundle delivery ratio. Bundle delivery ratio shows the percentage of bundles received at the destination from the ones generated at the source. Between all entities, except between the Central Router and Central Server, we had 100% bundle delivery ratio. This means that all the records that were generated at one node were received on the other node when both nodes were within wireless range. Even between the Central Router and Central Server when the connection was re-established, all the records from the Central Router were received by the Server.

Another metric is the delay faced by a record to reach its destination. As we saw from our results, the delay due to sending records or receiving records is negligible compared to the delay the vehicle takes to go from one node to the other. So the delay will depend on the distances between shelters and main coordination centers, and the frequency and speed of the transportation that will be used to collect and provide People Finder records.

Chapter 6

Conclusions

Areas affected by natural or humanitarian disasters, often suffer from lack of suitable communication infrastructure, making it hard to collect and distribute information about affected victims. In this report, our goal was to design and to implement a database management system that is scalable, accessible and would be easy to deploy by a humanitarian organization to collect and distribute digital information about victims after a disaster. Although we have shown from our test results that the People Finder system is feasible, further tests would be needed before the system can be used in a disaster area, involving more test nodes, more records generated, and longer test durations.

Bibliography

[1] Karkkainen, T. and Ott, J. n.p. Liberouter: Towards Autonomous Neighborhood Networking.

[2] Uddin, M. Y. S., Nicol, D. M., Abdelzaher, T. F. and Kravets, R. H. 2009. A post-disaster mobility model for delay tolerant networking. pp. 2785–2796.

[3] Ifrc.org. n.d. Shelter and settlements - IFRC. [online] Available at: https://www.ifrc.org/en/what-we-do/disaster-management/responding/services-for-the-disaster-affected/shelter-and-settlement/ [Accessed: 21 Mar 2014].

[4] Region4a-mrc.org. n.d. AMERICAN RED CROSS GUIDE FOR SHELTER MANAGERS. [online] Available at: http://www.region4a-mrc.org/documents/2009march/AM [Accessed: 21 Mar 2014].

[5] Ipnsig.org. 2014. InterPlanetary Networking Special Interest Group (IPNSIG). [online] Available at: http://ipnsig.org/ [Accessed: 21 Mar 2014].

[6] Dtnrg.org. 2013. Home - Delay Tolerant Networking Research Group. [online] Available at: http://www.dtnrg.org/wiki/Home [Accessed: 21 Mar 2014].

[7] K. Scott, and S. Burleigh, Bundle Protocol Specification, IETF RFC 5050, experimental, November 2007.

[8] M. Demmer, J. Ott, S. Perreault, Delay Tolerant Networking TCP Convergence Layer Protocol. [online] Available at: http://tools.ietf.org/html/draft-irtf-dtnrg-tcp-clayer-07 [Accessed: 21 Mar 2014].

[9] Ict-scampi.eu. n.d. ICT-SCAMPI. [online] Available at: http://ict-scampi.eu [Accessed: 21 Mar 2014].

[10] Raspberrypi.org. n.d. [online] Available at: http://www.raspberrypi.org/ [Accessed: 21 Mar 2014].

[11] Ginzboorg, P., Karkkainen, T., Ruotsalainen, A., Andersson, M. and Ott, J. 2010. DTN communication in a mine.

[12] Sasaki, Y. and Shibata, Y. 2010. Distributed disaster information system in DTN based mobile communication environment. pp. 274-277.

[13] Cmu.edu. n.d. Disaster Management Over Disruption-Tolerant Networks-Silicon Valley Campus - Carnegie Mellon University. [online] Available at: https://www.cmu.edu/silicon-valley/industry-connect/practicums/projects/2011/geocam-dtn.html [Accessed: 21 Mar 2014].

[14] Google.org. n.d. Google Person Finder. [online] Available at: http://google.org/personfinder/g [Accessed: 21 Mar 2014].

[15] Zesty.ca. 2014. People Finder Interchange Format 1.4. [online] Available at: http://zesty.ca/pfif/1.4/ [Accessed: 21 Mar 2014].

[16] Redcross.org. 2014. Volunteering — American Red Cross — Volunteer for Charity. [online] Available at: http://www.redcross.org/support/volunteer [Accessed: 21 Mar 2014].

[17] Gardner P. 2011. The Serval Project: Practical Wireless Ad-Hoc Mobile Telecommunications

[18] IFRC. 2013. World Disaster Report. [report] IFRC.

[19] Talks.php.net, (2014). Introduction to SQLite. [online] Available at: http://talks.php.net/show/sqlite/3 [Accessed 1 May. 2014]

[20] Fema.gov, (n.d.).FEMA.gov — Federal Emergency Management Agency. [online] Available at: http://www.fema.gov/ [Accessed 8 Jun. 2014]

[21] NRF-Docs 2001. NRF documents. Available via¡http://www.fema.gov/pdf/emergency/nrf/nrf core.pdf¿[Accessed Jun. 8, 2014]

[22] ICS 2001. National incident management system. Available via ¡http://www.training.fema.gov¿ [Accessed Jun. 8, 2014].

[23] Wood, L., Peoples, C., Parr, G., Scotney, B. and Moore, A. (2007). TCP's protocol radius: the distance where timers prevent communication. pp.163–167.

[24] Vaishali.S.Rajet al, DELAY Disruption Tolerant Network (DTN), its Network Characteristics and Core Applications. International Journal of Computer Science and Mobile Computing Vol.2 Issue. 9, September- 2013, pg. 256-262

[25] Web.mst.edu, (2006). Welcome to the Mobile Data Management and Applications Website. [online] Available at: http://web.mst.edu/ mobildat/DTN/index.html [Accessed 11 Jun. 2014].

[27] Segui, J. and Jennings, E. (2006). Delay tolerant networking-bundle protocol simultion.

[28] Vahdat, A. and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks", Duke University , Technical Report CS-200006, April 2000.

[29] Farrell, S.; Cahill, V.; Geraghty, D.; Humphreys, I.; McDonald, P., "When TCP Breaks: Delay- and Disruption- Tolerant Networking," Internet Computing, IEEE , vol.10, no.4, pp.72,78, July-Aug. 2006.

Chapter 7

APPENDIX

Person Record's Metadata:

person_record_id: Unique identifier for the record, which consists of an ASCII domain name followed by a slash and a local identifier. The record's original repository is identified by the domain name. As for the local identifier it is up to the original repository to decide on its format.
entry_date: Date in which this copy of the record was stored.
expiry_date: Date in which this record should be deleted.
author_name: full name of the author of the note.
author_email: E-mail address of the author.
author_phone: phone number of the author.
source_name: Name of the original repository of the record.
source_date: Date in which the original record was created in its original repository.
source_url: URL of the record in the original repository of the record.

PersonRecord's Other Information (used for identification of missing people):

full_name: Full name of the person.
given_name: First name of the person.
family_name: Family name of the person.
alternate_names: Other names the person was called by.
description: Description of the person, his face description, height, etc,
sex: Gender of the person

date_of_birth: Date of birth of the person
age: Approximate age of the person
home_street: Home address of the person.
home_neighborhood:Home neighborhood of the person.
home_city: Home city of the person.
home_state: Home state of the person
home_postal_code: Postal address of the person's home address.
home_country: Home country of the person.
photo_url: URL of an image of the person.
profile_urls: URLs of the profile page of the person if he has one.

NoteRecord's Records Metadata:

note_record_id: Unique identifier for the record, which consists of an ASCII domain name followed by a slash and a local identifier. The record's original repository is identified by the domain name. As for the local identifier it is up to the original repository to decide on its format.

person_record_id: The note refers to a person record which has an ID of person_record_id .

linked_person_record_id: This is another person_record_id of a Person record that is linked with the person_record_id of the note.

entry_date: Date in which this copy of the record was stored.

author_name: Full name of the author of the note.

author_email: E-mail address of the author.

author_phone: Phone number of the author.

source_date: Date in which the original record was created in its original repository.

Note Record's Status information:

author_made_contact: If the author of this note has contacted the person then the value of the field is set to true otherwise its false.

status: Status of the person, whether he is believed to be missing, alive or dead.

email_of_found_person: E-mail address of the person.

phone_of_found_person: Phone number of the person.

last_known_location: last known location of the person.

text: Description of the person's current condition.

photo_url: Image URL .