Information and Communication Technologies

EPIWORK

Developing the Framework for an Epidemic Forecast Infrastructure

http://www.epiwork.eu

Project no. 231807

D 5.6 Design and implementation of software for contacts patterns detections

Period covered: M1-M20
Start date of project: February 1st, 2009
Due date of deliverable: M32
Distribution: Public

Date of preparation: 4/10/2011
Duration: 48 Months
Actual submission date: M33
Status: Final

Project Coordinator: Alessandro Vespignani
Project Coordinator Organisation Name: ISI Foundation
Lead contractor for this deliverable: CREATE-NET

Work package participants

The following partners have taken active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

- CREATE-NET
- Fondazione Istituto per l'Interscambio Scientifico

Change log

Version	Date	Amended by	Changes
Table of Content	30/05/2011	Iacopo Carreras	
Initial version	1/09/2011	Iacopo Carreras	
Integrated version	30/09/2011	Piret Saar	Added figures and description of mobile application
Revised version	7/11/2011	Iacopo Carreras	Integration of material and revision
Revised version	11/11/2011	Daniela Paolotti	Added description of contact patterns through dedicated devices
Revised version	15/11/2011	Daniele Miorandi	Revision

Table of Contents

1. Introduction

The EPIWORK project proposes a multidisciplinary research effort aimed at developing the appropriate framework of tools and knowledge needed for the design of a pan-European epidemic forecast infrastructure. The research considers most of the needed development of modeling, computational and ICT tools such as i) the foundation and development of the mathematical and computational methods needed to achieve prediction and predictability of disease spreading in complex social systems; ii) the development of large scale, data-driven computational models endowed with a high level of realism and aimed at epidemic scenario forecast; iii) the design and implementation of original data collection schemes motivated by identified modeling needs, such as the collection of real-time disease incidence, through innovative web and ICT applications. iv) the set up of a computational platform for epidemic research and data sharing that will generate important synergies between research communities and countries.

One of the crucial factors for accurate data-driven modeling is the access to regularly reported data that is representative of the European population. This is of uttermost importance for developing models and monitoring tools which are based on realistic data, and can help to better understand how diseases and pandemics could potentially spread in an European setting. Volunteers from the population will be reporting their health status on a weekly basis through an Internet-based Monitoring System (IMS) from where the data will be stored. Anonymized subsets of the data will be forwarded and stored in a central database and used by researchers for epidemic forecast modeling. In return, IMS will offer to the population on-line education about the diseases, information about epidemics situations in various countries, graphical representations of the progression of the diseases as well as some elements of entertainment such as games.
According to the EPIWORK Project's description of work, Work Package 5 is aimed at the developing, set-up and deployment of innovative web monitoring and data gathering tools that provide a continuous stream of data to the project. One direction that has been pursued in order to involve larger fractions of the population in the data gathering process is to extend existing reporting tools to a mobile environment, thus considering mobile platforms such as, e.g., smartphones, PDAs, netbooks, as candidate platforms for running a mobile extension of the IMS. Mobile phones are nowadays becoming the most widely diffused platform, with the 100% penetration barrier already passed in various European countries.

A first direction pursued during the first two years of activities has been to extend the existing Web-based monitoring platform to a mobile environment. As documented in D5.4, we have developed a mobile application that can be used by mobile users to report on their health status through their mobile device. The mobile application is fully integrated with the web platform, and allows additional mobile features such as crowd sensing.

However, with the advent of modern smartphones, mobile devices are increasingly becoming a proxy to users' habits and daily patterns. By exploiting the many sensors available on such devices, it becomes possible to regularly monitor users' daily activities and interactions. As an example, through the use of GPS it is possible to track users' location over time, while with accelerometers it is possible to infer users' activities.
In this deliverable, we aim at exploring the use of modern smartphones for monitoring the contacts among people, where a contact is defined as the physical proximity between two persons for a limited period of time. Such information is of paramount importance for the modeling of pandemics, as it represents a primary cause for the spreading of an epidemic. Since smartphones represent users' most "intimate" device, they represent the natural

candidate for measuring contacts. Two are the main technological challenges to be addressed when designing a system for contacts:

1. *Proximity detection*: this refers to the ability of the system to detects when two mobile users are in close physical proximity. This requires the system to be able to discern the distance among two users with a granularity of few meters.
2. *Interaction detection*: this refers to the ability of the system to recognize when two users are having some social interaction. This differs from the proximity detection as it tries to infer if two users are close by chance, or because they are interacting with each other.

In this deliverable we will address both challenges and explore how the sensors available on modern smartphones can be exploited in order to measure proximity and interaction. Further, we will discuss alternative technologies for measuring contacts, which do not rely on the use of smartphones.
The reminder of the deliverable is organized as follows. In Sec. 1 we will formulate the problem of contacts pattern detection, and present the most significant work in this domain. In Sec. 2, we will present our study on the detection of contacts by means of smartphones, and the software developed for running experiments. Sec. 3, will dig into the approach that provided the best results. Finally, Sec. 4 will conclude this deliverable with some discussion on the various approaches.

2. Contacts Pattern Detection

Human behaviour analysis is a research area that has a wide-reaching, multi-disciplinary impact. It has the potential to fundamentally advance the frontier in disciplines including psychology, psychiatry and anthropology. One important focus of human behaviour analysis is measurement of person-to-person interaction to detect social patterns. Information about social and interaction patterns can be applied to a number of different application areas, including i) healthcare, where the level of socialization may have a direct, positive impact on the self-reported mood of people as shown from experiments carried out in [21], ii) social network analysis, where interactions are seen as a proxy towards the social graph of a person, iii) productivity, where social interactions are shown to be correlated to the productivity [22] iv) epidemiology, where contacts among people represent the main cause behind the spreading of an epidemic.
Most of the approaches to monitoring person-to-person interactions required users to wear a dedicated sensing device. For example research work carried out in [1] used a wearable Social Badge to analyze proximity of users and their interactions. The badge was visibly worn in the shape of a pendant and used infrared line of sight to detect proximity. In a similar manner, authors in [2] used an active badge to measure social interactions in closed environments such as hospitals or conferences.
Alternative approaches based on the use of smart phones have been proposed, but they typically identify proximity as a consequence of users' co-location [9, 10], or rely on external infrastructure for detecting interactions [10,11].
In contrast to the previous studies, the solution explored in this deliverable leverages on the sensors embedded in mobile phones to detect user proximity.

2.1. Related Work

Proximity detection refers to the ability of a system to recognize the co-presence of two or more individuals and it is closely related to a number of research efforts which apply the combination of sensing and advanced modelling paradigms to support the monitoring and understanding of human dynamics. Depending on the specific device utilized (commodity vs. dedicated) or sensing techniques (for example, RF, ultrasound, image/video processing), proximity can be detected at various levels of time/space granularity, and for various application scenarios.

The pioneering work in this domain is represented by the Sociometer [1], a computational framework for detecting and modelling face-to-face interactions. The sociometer, consisted of a wearable badge to be carried by users during their daily activities and, in its initial implementation was able to detect i) physical proximity through the use of IR technology and ii) conversations from raw audio segments. This information was afterwards used to detect interactions among users and to study social dynamics within groups.

The use of dedicated devices to monitor physical proximity has been replicated in various projects. Outside the Epiwork project, the Sociopatterns (http://www.sociopatterns.org/) project, led by ISI Foundation, has developed and deployed a sensing platform employing wearable electronic badges to sense face-to-face proximity between persons, with a fine spatial and temporal resolution. It has so far been utilized in many deployments and data are publicly available for a number thereof. The Sociopatterns project relies on a framework for monitoring social interactions that pares scalability and resolution using an inexpensive and unobtrusive active RFID device. The devices are capable of sensing face-to-face interactions of individuals as well as spatial proximity over different length scales down to one meter or less. The data collection and processing allows the tuning of the scale over which the interaction mapping works. In fact, the approach is highly scalable: the data collection protocol has been deployed in social gatherings involving from 25 to 575 individuals, from schools to hospital wards and conferences. Analysis of the results shows a remarkable self-similarity in the statistical signature characterizing personal interactions, despite the different social contexts and scales of the deployments.

The use of dedicated devices provides a fine granularity in the data being collected, but it is limited to supervised environments and for limited periods of time (limited by, e.g., the battery life-time of the dedicated device). Furthermore, since such devices are not part of individuals' everyday life, they can be perceived as obtrusive or simply may not be always carried by the users. Finally, they typically rely on an external infrastructure to collect the data for a later analysis. The use of dedicated devices is therefore appropriate for scenarios confined in space and time, such as conferences, enterprise environments, and hospitals.

To overcome this limitation, smart phones have been proposed in the literature as an alternative device to monitor proximity among individuals. They follow users during their daily patterns and, with the increasing computing, communication and sensing power capable of performing complex tasks. Through the use of smart phones, proximity can be either sensed directly or inferred in terms of users "co-location". In the former case, the sensing capabilities of the mobile device are exploited to detect nearby users. As an example, by performing a Bluetooth scan, Bluetooth-enabled device may be captured [9, 10] and this information used to infer users' proximity and to reconstruct human dynamics over time. This approach, while being fully distributed and relying solely on Bluetooth for detecting proximity, provides only a coarse spatial granularity, since the Bluetooth communications range is in the order of ten meters. In [13], ultrasound has been explored as a mean to achieve a fine-grained relative localization. Through the emission of ultrasonic signals, the time-of-flight is used to calculate a range and angle of arrival estimates. This information can then be used to calculate the relative position of devices thus detecting their proximity. The approach provides a very accurate estimate of proximity, but requires ultrasonic emitters/receivers which are not available in standard smart phones and in addition, they are very sensitive to the device positioning. To the best of our knowledge, there is only one work that, similarly to our work, exploits transmitting mechanisms embedded in smart phones to achieve the proximity detection with a granularity of a few meters. The project called Virtual Compass [21] detects nearby mobile devices with an average accuracy of 1.9m. However, the authors used the combination of Wi-Fi and Bluetooth technologies in order to achieve high resolution distance estimation, relying on more neighbouring devices to calculate inter-distances in a two dimensional plane using the Vivaldi method.

In the case of co-location, proximity is inferred whenever users share the same spatial location. This can measured through the use of Bluetooth beacons [6] or Wi-Fi scans [7]. The NearMe platform [7] exploits the similarity of Wi-Fi fingerprints performed by different users in order to infer proximity. While the platform requires a minimal set up and can run on

portable devices, it is based on a centralized infrastructure for "matching" the fingerprints, and is applicable only to places covered by a Wi-Fi network.
Approaches based on image and video processing have also been proposed [3,12]. Relying on rich datasets, they are computationally complex and also require line-of-sight access to the monitored spaces, hence these approaches hardly cope with large scales. This limits such methods to very specific application scenarios (such as security/surveillance/intrusion detection). Similarly, speech processing techniques can be used to monitor conversation dynamics and to model social interactions [11]. This solution provides an ubiquitous approach for revealing proximity but, at the same time, can hardly be applied to a large scale and uncontrolled setting due to the initial training phase needed for creating a model of each user.

3. Contacts Pattern Detection using Mobile Phones

In this deliverable, we have developed the necessary software to explore the use of mobile devices for monitoring the contacts among people. The software (see Figure 1) consists of:

- a mobile application that can collect data from a variety of sensors present on modern smartphones. This includes WiFi, accelerometer, compass, Bluetooth, GPS, location from network, audio, light.
- a server component: the server component is able to gather and store the data on a remote server, to be used for a later analysis. This has proven to be very helpful for building a dataset that was then used to design and implement the necessary algorithms.

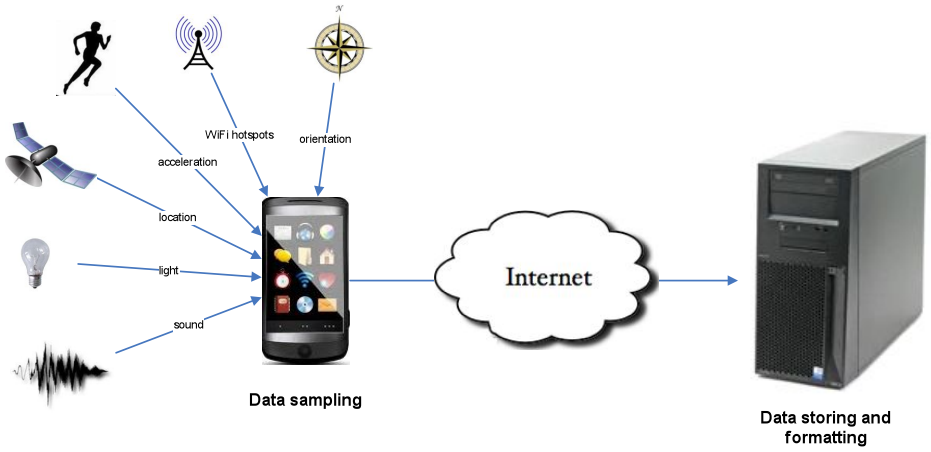


Figure 1: The system architecture for the contacts pattern detection architecture.

The mobile application was used to gather any data that could be captured with a sensor available on the phone. Through the application it is possible to properly configure the sensing period, as well as specific sensor settings. Such data is then transmitted to the server and stored. During the various data gathering campaigns that we run, the application was run by various users in parallel, carrying the phone during their daily activities and occasionally having interactions (e.g., meetings, coffee breaks, lunch breaks, etc.).
Appropriate data mining algorithms were then developed for identifying specific patterns, relevant for the detection of contacts among persons. Such algorithms were developed, starting from the dataset stored on the server.

3.1. Mobile Application

PatternCollector is the mobile extension of our platform, and consists of an Android application that offers two main functionalities: data collection and labelling of the data. The **data collection** is performed through periodical sampling of data from the various sensors available on the phone. Some examples of such data are the orientation of the phone, the direction and strength of the magnetic field or the acceleration of the phone. The collected data is temporarily saved in the phone's database and later forwarded to the web server for formatting and storage. To ease the analysis of the collected data, the *PatternCollector* offers a possibility to **label** portions of the collected data during the sampling. For example, if a user is running a day long experiment and is interested in highlighting some specific events during the sampling, he/she can initiate the data labelling at the beginning of the event and complete the label when the event is over. The start and end times for this label will be saved and associated to the collected data. In addition to automatically saved timestamps, the labelling interface offers a possibility to quickly fill in more information about the event. As the *PatternCollector* is used to collect the contacts patterns data, the label can contain information about the orientation of the facing between people, distance between people, users who were present during the event etc (see Data Labeller UI in Figure 2).

In the following sections we will present the software architecture of the *PatternCollector* (see Figure 2). We will describe the four main parts of the architecture: User Interface, Database, System Modules, Third Party Software.

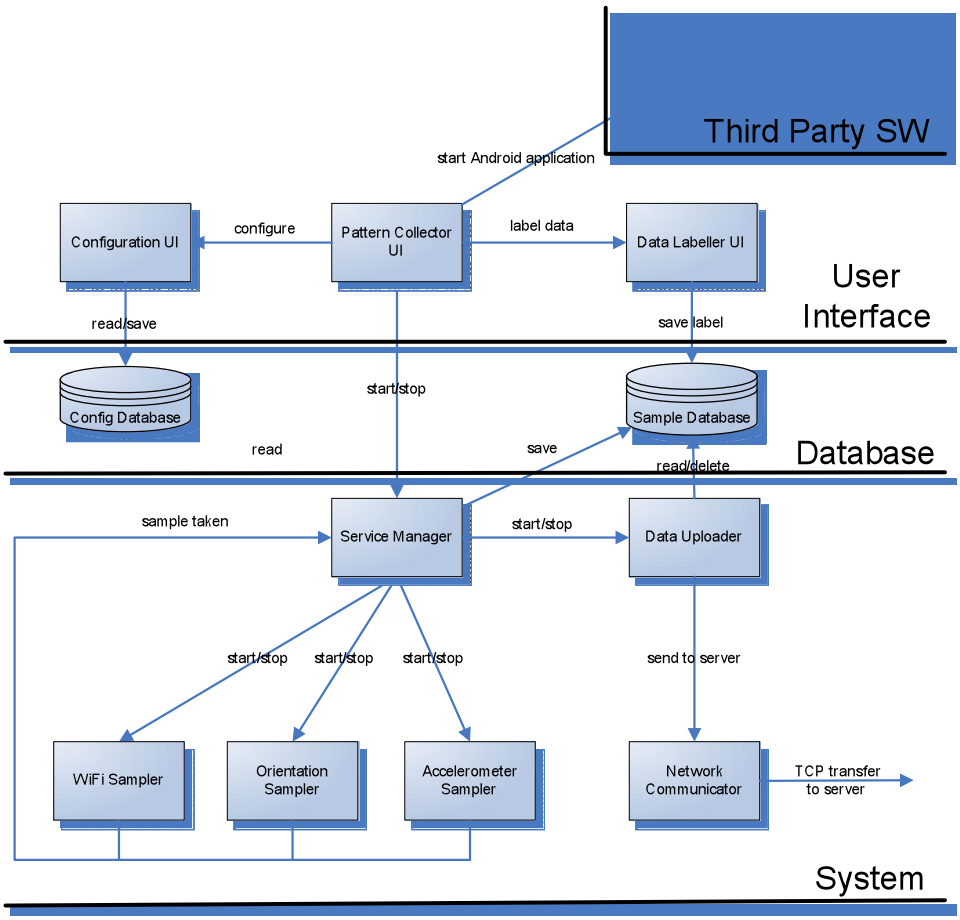


Figure 2: Pattern Collector software architecture

3.1.1. User Interface

The user interface consists of three components where each corresponds to one screen that can be displayed on the phone. The entry point into the application is the Pattern Collector UI from where the Configuration UI or the Data Labeller UI can be activated. Figure 3 shows screenshots from these three user interfaces.

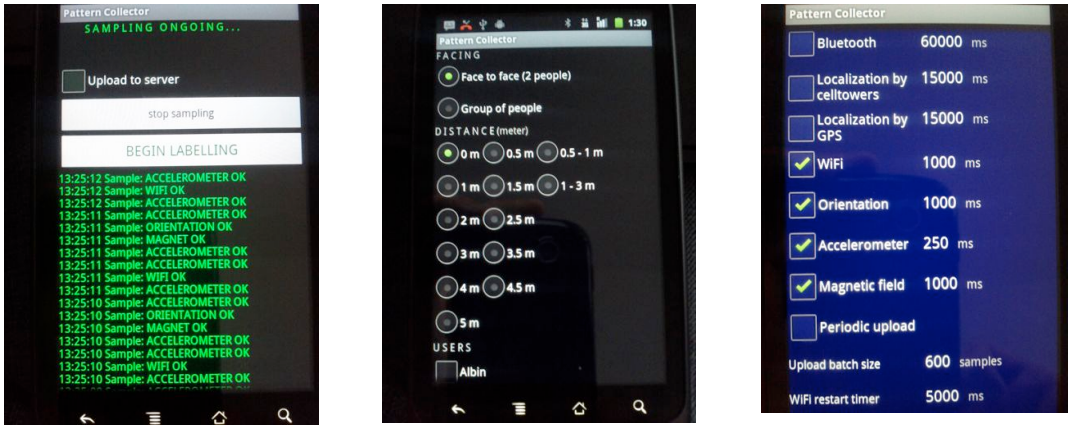


Figure 3: The Pattern Collector UI during sampling, the Data Labeller UI and the Configuration UI.

The **Pattern Collector UI** is activated when the application is started on the phone. If it is the first time that the application is run, this component creates all the persistent components that will continue existing until the application is uninstalled. An example of such components are the databases for data and background services that must continue sampling even when the application is not on the foreground of the phone's screen.

The *PatternCollector* UI is the main screen of the application where sampling and data labeling can be started and stopped.

From the *PatternCollector* UI, the user can activate the **Configuration UI** through the menu at the bottom of the screen. The configuration UI allows the user to configure all the parameters needed for sampling and data upload. For example, activation/deactivation of certain samplers, sampling period values, the URL of the server where the data should be uploaded. All configuration data is saved in the phone's persistent database and will be available the next time the application is started.

The **Labeller UI** can also be activated through the Pattern Collector UI menu. This UI allows the user to describe the labeled data. The data can be described in terms the distance between participants, orientation (face to face or angle), free text etc

3.1.2. Database

The *PatternCollector* uses Android's built-in persistent SQLite database for storing the sampled data and the application configuration. The **Sample Database** contains all the samples and their labels (if any) until the data has been successfully uploaded to the server. The **Configuration Database** contains all application configuration data and will not be cleared until the user explicitly chooses to clear the application data on the phone. As many of the application components depend on the configuration in this database, the Configuration Database offers notifications about the configuration changes to the components that are interested.

3.1.3. System

The system part of the *PatternCollector* application consists of the sampling and sample upload management to the server.

The **Service Manager** is the central component that manages the configuration and the states of all the samplers and the data uploader component. It listens to the configuration changes in the Configuration Database and makes sure that its subordinate components are appropriately configured and in a correct state. The Service Manager is also responsible for receiving all the samples from the samplers and saving it in the Sample Database.

The **Samplers** are responsible for reading the data from a sensor The sampled data is packed into a common sample format and sent to the Service Manager. The Samplers are the central part of the system that perform the task of data collection.

The **Data Uploader** handles the data upload to the server and data deletion from the Sample Database processes. The data is deleted from the database only if has been successfully received on the server. All communication with the server is done through the Network **Communicator** component that is responsible for the communication protocol between the mobile phone and the server.

3.1.4. Third Party Software

The Wireless Tether is a third party software and used for configuring Access Point related settings on rooted Android phones. In the Android context, rooted means that the user replaces the Android software of the phone with an unofficial version that gives the user the root access to the phone, unlocking all the functionalities of the phone that are hidden to the normal user.
In the Wireless Tether we used the functionality for activating/deactivating the Access Point mode, to choose the WiFi transmission channel, frequency and the transmission power. We have modified the original code in order to be able to change the WiFi transmission power on all our rooted phones, as the original version managed only some phone models.

The modified version of the Wireless Tether can be activated from the Pattern Collector menu.

3.2. Server Application

The server side of the system was used to gather the data sensed from mobile phones, and store it into a database. Such data was then used to develop models for detecting contacts, starting from the sensed data. The server implemented a REST architecture, where each call specified the data to be stored, with the corresponding values. The main design requirements for the server has been its ability to handle large amounts of data in real-time. This both when (i) receiving and saving data, and when (ii) processing it.
The server consisted of a J2EE web application, build using standard technologies:

- Apache/Tomcat 7.* as servlet engine
- Hibernate + Mysql: used for the persistence of Objects

Figure 4 shows the graphical interface used for navigating and browsing the data stored in the platform.

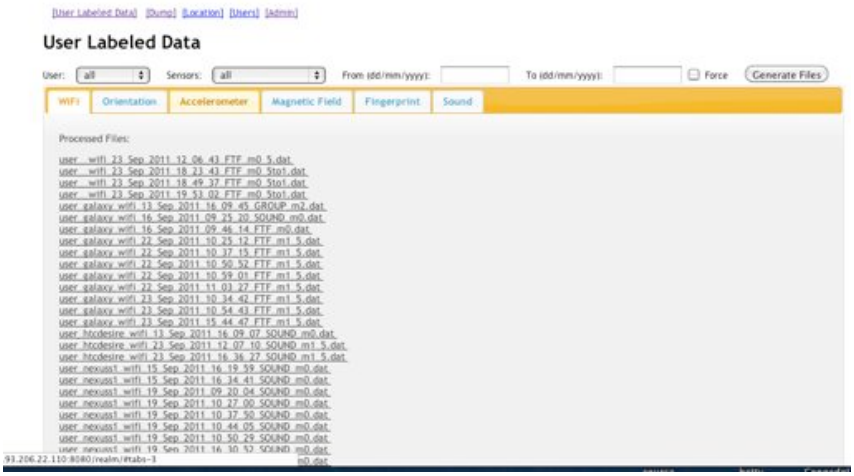


Figure 4: Server side web interface

3.3. Exploring the use of different technologies for detecting proximity

Starting from the data collected through the *PatternCollector* mobile application and stored in the server, we have analyzed various techniques for detecting the proximity of users. In the following we will describe each one of them.

3.3.1. WiFi fingerprinting

WiFi fingerprinting is a method that takes a snapshot of the information of all visible WiFi access points and compares it to another fingerprint. The information that can be saved in one fingerprint is the visible network IDs and their signal reception strength. When two mobile phones are close by and scan for available WiFi access points, they should be able to detect the same access points and experience similar patterns in the strength of the received signal from these access points. By expecting to detect strong similarity in WiFi fingerprints from phones that are close by, compared to those that are further away, we conducted experiments for estimating the distance between two phones.

In our experiments, we used two phones and gathered WiFi fingerprints each second for five minutes, and we repeated this for several distances between the phones. The fingerprints were saved on the phones and sent to the server for analysis after the experiments. The experiments were repeated several times combining different phone models and changing the experiment location.

Unfortunately, the method did not prove to be useful for proximity detecton, as we discovered several serious weaknesses of this approach. In particular, the approach was not able to detect proximity with the accuracy needed for inferring social interactions (~ 3 meters).

- *Received WiFi signal power varies greatly*. WiFi fingerprints that are measured by a phone with an interval of only a few milliseconds can display great differences in signal strength values for the same WiFi access points. Thus, comparing signal strengths from one measurement is not meaningful. To avoid this problem, we tried to use the mean values from several fingerprints, which seemed promising, but made us discover the next weakness:
- *Received signal power is strongly effected by the shape of the room*. The transmitted signals reflected from the floor and walls have much larger effect on the received signal strength than the increasing distance (we tested up to 5 meters) between the phones. This problem makes the WiFi fingerprinting method unsuitable for measuring shorter distances.

- *Received signal strength is perceived differently by different phones*. If two different phone models measure the WiFi access point signal strength in the same position, it is very likely that the values will differ. We noticed that each model had its own sensitivity, resulting in a constant offset difference in measurements between the phones. This problem can be corrected by calibrating the phones before the experiments, but this would require an extensive training and calibration phase, which does not comply with our requirements.

3.3.2. Acoustic Proximity Detection

In search for unintrusive distance estimation methods we decided to experiment with high-frequency sounds that are not audible for the human ear. Sound, having a relatively low velocity, seemed to be a suitable means for measuring distance with modern mobile phones, which support high sampling rates. The idea was to find a way to measure the time that it takes for the sound to travel from one phone to another. Knowing this and the velocity of the sound, the distance could be easily estimated.

Approach 1: Sound recording with real-time data processing.

In this approach, the phones were sending out high-frequency signals and listening to replies from other phones at the same time. When a reply was received, the time from sending the signal and receiving the reply could be calculated, providing an easy means for estimating the distance.

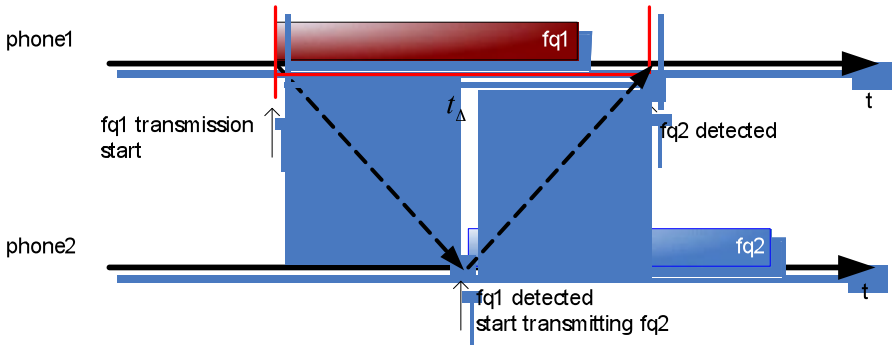


Figure 5: Real-time sound processing protocol.

In our experiments two phones were configured to generate signals with frequencies $\mathbf{fq_1}$ respectively $\mathbf{fq_2}$ (see Figure 5). The phone that wished to initiate the distance estimation (*phone1*) started the estimation procedure by emitting a sound with its frequency. When the other phone (*phone2*) detected this signal, it immediately responded by emitting its own signal. *phone1* measured the time ($\mathbf{t_\Delta}$) between emitting its signal and detecting the answer. Thus, $\mathbf{t_\Delta}$ is the time it takes for the sound to travel from *phone1* to *phone2* and then back again. The time it takes to travel from *phone1* to *phone2* is then $t=\mathbf{t_\Delta/2}$. Knowing the velocity of sound ($\mathbf{v_{sound}}$) the distance (d) between the two phones is:

$$d = t\, v_{sound}$$

This approach produced results of very poor quality. We experienced that phones had problems starting up both the recording and sound emission, and also performing frequency detection analysis at the same time. Among the problems that we encountered were the signal emission delays of up to several seconds and received sound packets being discarded due to the data processing taking too long time.

We learned that for this task, today's mobile phone platforms cannot yet support real-time systems that demand high accuracy. It is not yet realistic to expect the services of the phone to react immediately or to be guaranteed continuous execution time. In our case, a 1 ms error would mean 3 meters error in distance estimation, and can already be considered unacceptable.

Approach 2: Light weight sound recording with post processing of data.

To avoid the inaccuracies encountered in Approach 1, we redesigned our distance estimation protocol and minimized all real-time procedures that demand high accuracy. The most significant improvement compared to Approach 1 was that instead of analyzing the sound in real-time, the phones saved the sound data and processed it after the signal emitting/recording process was completed. In this way, the data analysis did not interfere with the execution time of the sound player/recorder, greatly improving the quality of the emitted/recorded sound. The drawback of the post-processing of data was that we needed to add a session layer to our protocol for letting the phones coordinate the start of their measurement sessions. The session layer was also needed for exchanging some measured data between the phones that was required for calculating the distance estimates.

When the *phone1* in this approach wishes to estimate the distance to *phone2*, it first initiates a session (using TCP) to *phone2* (see Figure 6). After the session is initiated, both phones immediately start to record and to send their signals. The player/recorder initiation and starting processes are always quite CPU intensive on the mobile phones and therefore the quality of the sound produced/recorded is also quite poor during that period. In general, the phone recovers from this intensive workload of player/recorder startups in less than 1 second and thereafter continues to perform with a satisfactory real-time quality. To avoid the bad sound quality in the beginning of the session, we added $\mathbf{t_{silence}}$ seconds of silence in the beginning of the emitted signals. This way, the sound emitter can start emitting sound from the beginning of the session and enter into a steady state by the time the actual frequency is going to be emitted.

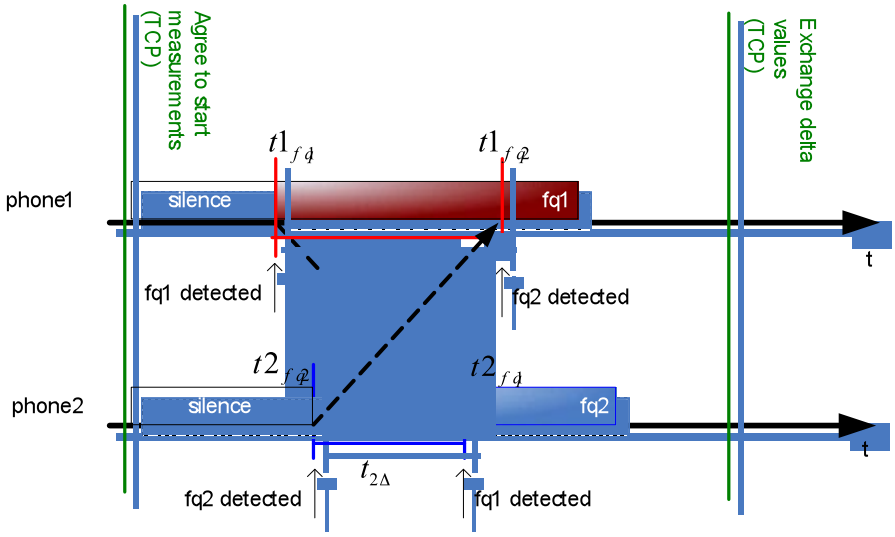


Figure 6: Post processing protocol of the sound.

As the phones are recording and emitting their signals at the same time, both signal frequencies will be captured by the recorders on each phone (if the phones are close enough). After the recording and signal transmission are finished, the recorded sound spectrum is analyzed. We cannot use absolute times for detecting the frequencies in our calculation, as the clocks of the phones may not be perfectly synchronized. Instead, we aim to measure on each

phone the time difference ($\mathbf{t}_{\Delta}$) between detecting the signals $\mathbf{fq}_1$ and $\mathbf{fq}_2$ and use these relative measurements for our calculations.

$$t_{1\Delta} = t1_{fq2} - t1_{fq1}$$
$$t_{2\Delta} = t2_{fq1} - t2_{fq2}$$

At the end of each session the phones exchange these time difference values and calculate the distance between them.

The time for the sound to travel the distance between *phone1* and *phone2* can be expressed as:

$$t = t2_{fq1} - t1_{fq1} = t1_{fq2} - t2_{fq2}$$
$$\Leftrightarrow$$
$$2 \cdot t = t2_{fq1} - t1_{fq1} + t1_{fq2} - t2_{fq2} = t_{1\Delta} + t_{2\Delta}$$
$$\Leftrightarrow$$
$$t = \frac{t_{1\Delta} + t_{2\Delta}}{2}$$

And finally, the distance (d) between the phones will be:

$$d = \frac{t_{1\Delta} + t_{2\Delta}}{2} \cdot v_{sound}$$

Using this approach, the distance estimation can be very precise. In Figure 7 we see that the error of the estimated average value lies almost constantly around 0.4 m and does not seem to increase with the distance. In time units, this error corresponds to an error of 0.001 seconds. We believe that the reason for this error lies in the implementation of our frequency detection algorithm that analyzes data in large chunks and might miss the frequency if it begins somewhere in the middle of a chunk. We believe that by refining the frequency detection algorithm we could decrease the error.

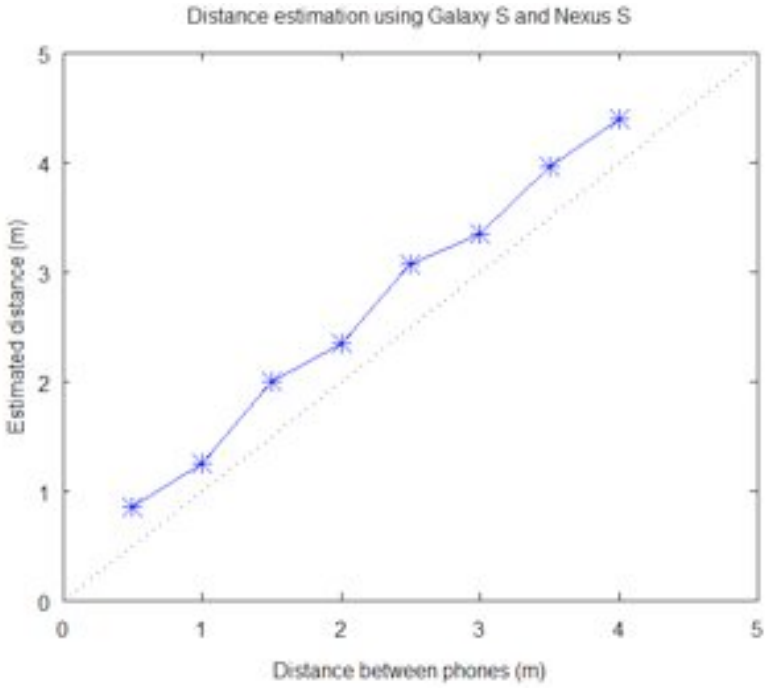


Figure 7: Accuracy of the acoustic proximity detection for various distances.

As the error has a constant nature, another way of decreasing the error is removing it through the calibration of the phones in the beginning of an experiment. By performing an estimation to a known distance and remembering the error, all the future estimated values can be adjusted.

Although this method is very precise in the distance estimation, we also discovered a serious drawback of using high frequencies on mobile phones in general for distance estimation in everyday life. The problem is that high frequency sound waves do not spread out very much and if a phone's speaker does not face the direction of the other phone's microphone, the signal will miss the microphone. The greater the distance, the harder it is to find the right position so that the phones can hear each other. Another problem with high frequency signals is that they do not penetrate materials well. If a phone is in the pocket, only very close distances can be measured. Using low frequency sounds would solve the weaknesses of this approach, but they would be impractical to use since they are audible for the human ear.

3.3.3. Proximity Detection via RSSI

As the last approach, we used the ability of modern devices to act both as Wi-Fi transmitters and Wi-Fi receivers. We exploited this feature to estimate the distance between a transmitting and a receiving peer by analyzing RSSI (Received Signal Strength Indicator). Since this has been the most promising approach, we have dedicated the following section to describe it in details.

3.4. Proximity Detection via RSSI

In this section we present our proximity detection system intended for sensing social interactions using technologies available in modern smart phones. The goal is to achieve a fine-grained distance estimation related to social interactions (in the order of a few), while not relying either on external infrastructure or on any additional hardware but smart phones. Our approach to proximity detection between two smart phones (one acting as a Wi-Fi receiver, the other one as a Wi-Fi transmitter) is based on the Wi-Fi RSSI analysis and data mining techniques in order to estimate the distance between them.

3.4.1. Methodology

Similar to indoor positioning systems that use fingerprinting technique, we have studied a method for distance estimation that is based on analyzing RSSI values, observed on an unknown distance from the phone which transmits Wi-Fi signal (for instance, this feature is typically called “Portable Hot Spot” in Android phones and “Personal Hot Spot” in iPhone). The RSSI values are processed by our system which estimates the unknown distance (Online phase – Figure 8) by applying the model built using a database that matches RSSI values with actual distances (Offline Phase – Figure 8).

For acquiring the training set, two smart phones were used (one in transmitting, the other one in receiving mode) to carry out Wi-Fi signal measurements on different distances following a grid of 0.5m thus collecting patterns of RSSI in the database to be used as training set. Using a shorter grid spacing, which corresponds to acquiring more points in the training set, improves the system’s performance but only up to a certain threshold when the accuracy starts to saturate or even to degrade [15]. In our experiments we found that a grid spacing of 0.5m provided optimal performance, considering the accuracy and the distances relevant for social interactions [18]. For the same reason, we limited training to distances of up to 8 m.

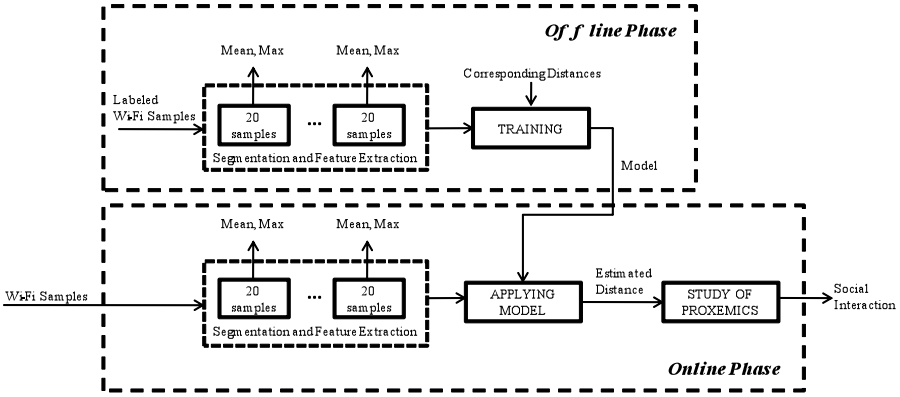
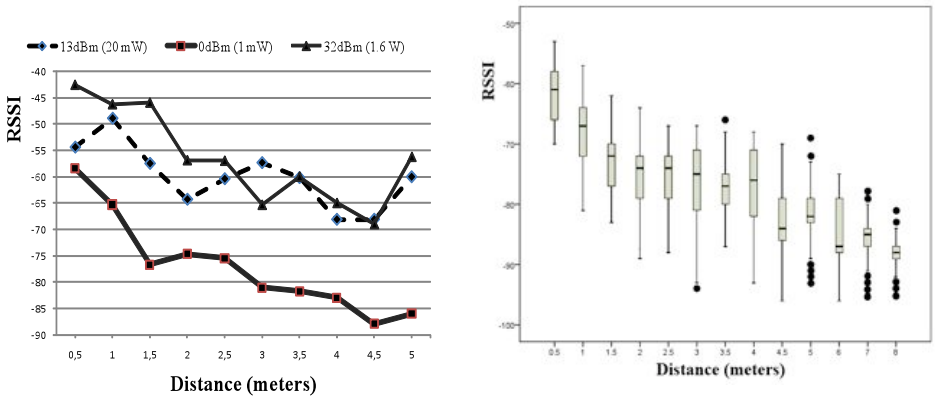


Figure 8: Block-diagram of our approach

3.4.2. Preliminary RSSI pattern analysis for short distance estimation

In order to evaluate the feasibility of distance estimation based on Wi-Fi RSSI, we have analyzed the RSSI dependence on distance for three different transmitting power levels (Figure 9a): 32dBm (1.6 W) – maximal available power level, 0dBm (1mW) - minimal power level, and 13dBm (20mW). The measurements were carried out in the same environment using an HTC Desire smart phone, while recording 300 samples with the sampling rate of 1Hz for each of the distance values. The transmitting power of 0dBm provided the smoothest and the most monotone characteristics (as can be seen from the mean RSSI values in Figure 9a), thus proving to represent the best fit for short distance estimation. Afterwards, we further analyzed RSSI patterns, setting the transmission power to 0dBm, in seven different environments (indoor areas ranging from 30m² to 90m²) recording 300 samples every 0.5m up to the distance of 8m. The cumulative RSSI patterns are presented in Figure 9b where small circles represent outliers, thick horizontal lines correspond to median values, bottom and top of each box corresponds to the first and the third quartiles of distribution and the whiskers extend up to 1.5 times the interquartile range (IQR). It can be seen that the RSSI shows relatively monotone characteristics across different environments. However, the large whiskers suggest that WiFi signal is not fully stable and can present considerable fluctuations, a known phenomenon typically due to environmental factors [17]. Therefore, the distance estimation approach based on a simple RSSI threshold analysis (assigning ranges of RSSI values to corresponding distances) did not suffice. This led us to considering machine learning techniques for refining the distance estimation procedure.



a) Three different power levels, one environment b) Power level of 0dBm, seven different environments

Figure 9: **RSSI dependence on the distance**

3.4.3. Estimating distance through classification and regression techniques

Data segmentation and features extraction

As the first step of data processing we segmented RSSI values by grouping every 20 consecutive samples (which, in our experiments, corresponded to 20 seconds) and calculated signal characteristics for each group separately. In this way, we aimed to mitigate the short-time Wi-Fi signal instability issue [14]. Reducing the number of grouped samples led to degradation of the overall system's accuracy, while working with larger sample sets yielded no notable improvement in the distance recognition. Since RSSI distribution varies according to its mean [14], we selected the mean value as a candidate parameter to represent the RSSI pattern. It turned out that among other tested signal characteristics (such as standard deviation, minimum and median), the combination of the mean and maximal value was proven empirically to provide the highest accuracy in distance estimation Adding more parameters did not result in significant improvements of accuracy while, on the other hand, it increased computational requirements. Hence, every block of 20 consecutive RSSI samples (recorded over approximately 20 seconds) was represented in the training set with its mean and maximal value and was assigned to the corresponding distance.

Techniques Selection

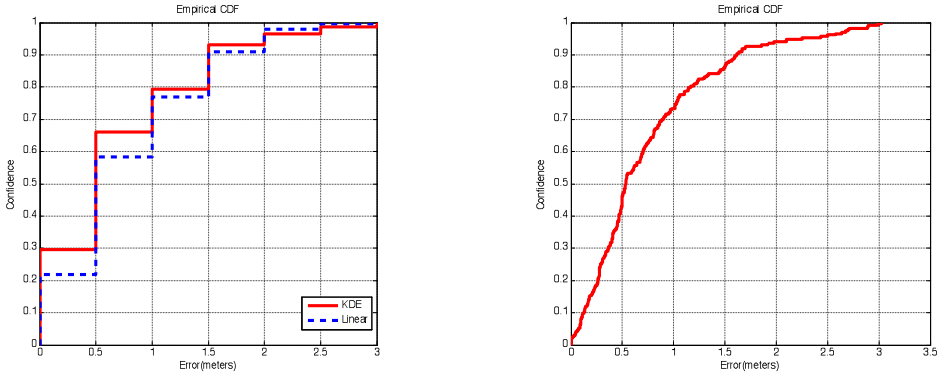
In order to estimate the distance between two phones, we applied well-known machine learning algorithms to perform pattern matching between an unknown distance and the training set. Regarding the distances relevant for social interactions, suitable methods may be considered as mapping from observed RSSI samples either to one of the distances pre-defined by the training process (discrete target) or to one of the distances from the infinite set that contains all possible distances (continuous target). Therefore, distance recognition can be addressed using both classification and regression techniques.
Although the lognormal distribution of Wi-Fi RSSI was often assumed in the literature, it only represents a part of real RSSI distributions [14]. Therefore, we opted for using a Naïve Bayes classifier with Kernel Density Estimation (KDE), which represents a powerful yet flexible nonparametric classification technique. However, several classification techniques that we tested demonstrated similar performance in distance estimation; as an illustration, we will report the results for Linear classification.
On the regression side, Gaussian Process (GP) is a well-suited regression method for localization for the following reasons: a) it does not require a discrete representation of an environment, b) being non-parametric approach it provides adequate approximation for a wide range of non-linear functions, c) GP parameters can be estimated from training data applying well known algorithms [16].

Experimental set up and results

Our testbed consisted of six smart phones (with Android operating system) including three different models, namely HTC Desire, Samsung Nexus S and HTC Nexus One. All were modified, as described above, to allow adjustment of the transmitting power. Different phone units were distinguished by the respective MAC address.
Measurements were run in three offices with dimension of 12x8 m, 6x5 m and 6x3 m, a balcony of 12x2.5 m and a meeting room of 10x8 m. For testing the system's accuracy we used a pair of phones – one in transmitting and the other one in receiving mode.
Following a grid of 0.5m, RSSI was measured for 5 minutes on each distance between phones starting from 0.5m to the point in which either signal degraded to its minimal level or it was the furthest accessible point within room dimensions. The maximal distance in the experiments was between 5 and 8 m thus covering all the distances relevant to the study of proxemics [18, 19]. The measurements were stored locally on the phone memory but for facilitating data analysis uploaded on the server during the evaluation period.

The accuracy was estimated in the offline phase by applying a cross-validation method – RSSI pattern captured in one out of five environments (three offices, balcony and a meeting room) was used for building the model (i.e. as a training set) while measurements from the four remaining environments were used for testing. In this manner, the procedure was repeated to cover all the combinations regarding distinct training and test sets across five environments. The RSSI characteristics were calculated over every block of 20 samples (Section 3.3) and queried separately to estimate an unknown distance. The cumulative distribution function of the distance estimation error was plotted to evaluate the system's accuracy.

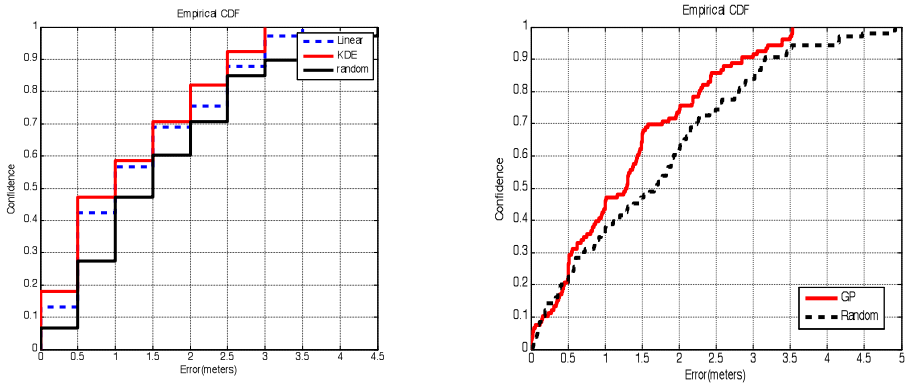
Figure 10 shows the system's accuracy in the case of using the same phone (i.e. same model) acting as a receiver in both training and test phase. The median estimation error (50th percentile) of approximately 0.5m was achieved using all the three applied machine learning techniques. Naive Bayes with KDE showed a slightly better overall performance, providing a distance estimation with an error less than 0.5m in 67%, less than 1m in 80% and less than 1.5m in 92% of cases.



a) Classification b) GP Regression

Figure 10: Distance estimation accuracy (same receiving phone for training and test phases)

When different phone models were used for the training and for the test phase, the system's accuracy significantly degraded. This is due to the fact that RSSI patterns highly depend on the receiver characteristics [15] which are likely to be different across different phone models. Figure 11 presents distance estimation accuracy plotted against the baseline performance represented as a random estimation out of the set containing distances from 0 to 5 meters. The median error was around 1m while 90th percentile error was 2.5m and 3m for Naive Bayes KDE classification and GP regression, respectively. Considering our goal of recognizing distances related to social interactions, the system did not provide satisfactory accuracy in this case. In the next section we describe our approach to tackle this issue.



a) Classification b) GP Regression

Figure 11: Distance estimation accuracy (different phones for training and test phases)

It should be mentioned that the change of the mobile device model acting as transmitter (keeping the same transmitting power) did not result in significantly different RSSI patterns; therefore, we do not report the results for this case.

3.4.4. Fast Calibration based on Propagation Model

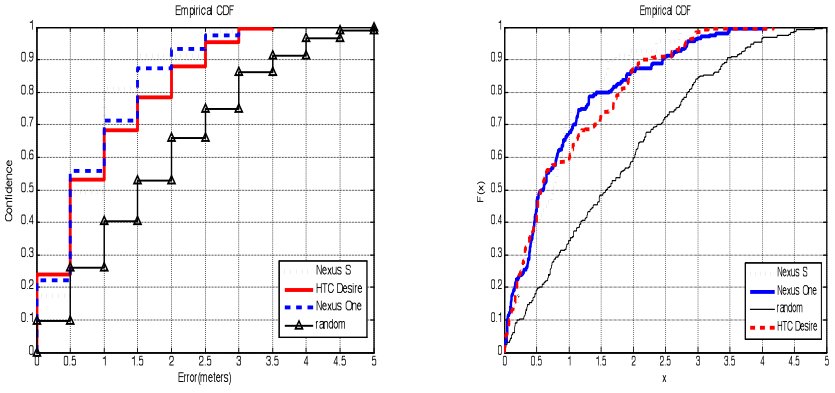
Affording a generic application for inferring social interactions would require having a generic model that would reflect RSSI patterns considering various models of smart phones and environmental conditions. However, the possibility of creating such a model, which would provide a reasonable system's accuracy for a number of phone models, is limited given that the range of RSSI values is a function of effective transmitter power, path loss, receiver antenna gain, and receiver sensitivity [15]. In addition, fluctuations of RSSI may be caused by environmental factors including furniture layout in the room of interest, air temperature, humidity and presence of people [17]. Since some sources of uncertainty are difficult or even impossible to be fully knwon at system design/calibration time, the main goal becomes the identification of the major factors that influence RSSI patterns and the development of models able to account for their effect.

Our analysis shows that the main problem lies in the fact that different phone models present different receiver's characteristics. In addition, RSSI is determined by the transmitter power and path loss [15]. Since the application would keep the transmitting power constant to 1mW, the remaining issues that we aim to address are related to the receiver characteristics and to the path loss. This would require a calibration i.e. acquisition of a RSSI pattern database for any new model of the phone intended to use the application. Repeating RSSI measurements would be a labor-intensive and cumbersome process, since achieving satisfactory accuracy requires a number of distances (in our case following a grid of 0.5 m). To avoid this, our approach is based on calibrating only one point by measuring RSSI for a couple of minutes on a fixed distance of, for instance, 1 m. Once the RSSI is captured, the training set is estimated applying the following propagation model [23]:

$$P(d)[dBm] = P(d_o)[dBm] - 10n\log(\frac{d}{d_0})[dBm] - X[dBm]$$

where n is the path loss exponent, P(d0) is the signal power at the reference distance d0 from the transmitter phone (in our case 1m) and d is the distance in which RSSI is estimated by applying the model. X is a component which reflects the sum of losses induced by each wall between the transmitter and receiver. We have found empirically from the training sets that the best suited value for the coefficient n is 1.5, while for X is zero (as there are no walls or other obstacles between points).

We evaluated this method by measuring RSSI at the distance of 1m in one out of five environments, generating the rest of training set applying the propagation model. We then assessed the accuracy using RSSI measurements from the four remaining environments. The procedure was repeated for all the environments. The overall performance is plotted in Figure 11 as a cumulative distribution function of distance estimation errors for all the phone models considered.



a) Classification b) GP Regression

Figure 12: Distance estimation accuracy using training set generated by applying the propagation model

By investing a minimal effort for performing the calibration (which takes approximately a couple of minutes), we demonstrated that it is possible to achieve similar performance as in the case of acquiring a full training set (Figure 12). Out of the tested models, Nexus One and HTC Desire provided a good accuracy, with a median distance estimation error of 0.5m. On the other hand, Nexus S achieved a lower accuracy, with a 50th percentile error ~0.7m (Naive Bayes KDE) and ~1m (GP), but performed better in terms of 80th percentile error: ~1m (Naive Bayes KDE). HTC Desire and Nexus One estimated distance with an error smaller than ~1.5m in 80% of cases.

Therefore, fast calibration based on the propagation model endows a solution for wide deployment overcoming the differences due to different radio signal receiver sensitivity.

4. Conclusions

In this deliverable, we have explored the utilization of modern mobile devices for the detection of contacts among people, by exploiting the many sensors available on commercially available smart-phones (e.g., WiFi, Bluetooth, sound, etc.). The most promising results were achieved by exploiting the ability of modern smartphone devices to act both as Wi-Fi transmitters and Wi-Fi receivers. We exploited this feature to estimate the distance between a transmitting and a receiving peer by analyzing RSSI (Received Signal Strength Indicator). Results show that it is actually feasible to estimate proximity by using off-the-shelf smartphone, without requiring the deployment of any capital-intensive infrastructure. Our solution could be turned into an app, which – if adopted by a sufficiently large number of users- could provide valuable data for epidemiologists and scientists working on models for contact patterns. At the same time, it is worth recalling that our method required to jailbreak smartphones, in order to access low-level APIs. While this may change with the release of new version of smartphone OSs (at least in the Android case), this is a factor to be taken into account. Also, some common mobile platforms (iOS-X) do not allow to access low-level WiFi primitives, thus inhibiting the use of such platforms for performing proximity estimation according to the proposed method.

References

1. Pentland, A., Choudhury, T.: Sensing and Modeling Human Networks using the Sociometer. In Proc. of ISWC, New York, US, (2003)
2. Cattuto, C, Van den Broeck, W., Barrat, A., Colizza, V., Pinton J-F, Vespignani, A.: Dynamics of Person-to-Person Interactions from Distributed RFID Sensor Networks. PLoS ONE 5(7), (2010)
3. Haritaoglu, I, Harwood, D, David, LS.: Real-time surveillance of people and their activities. In IEEE Transations on Pattern Analysis and Machine Intelligence 22: 809 (2000).
4. Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Pocket switched networks and human mobility in conference environments. In Proc. of SIGCOMM workshop on Delay-tolerant networking, Philadelphia, US (2005)
5. Rachuri, K., Mascolo, C., Musolesi, M., Rentfrow, P.: SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing. In Proc. of MOBICOM (2011)
6. Vu, L., Do, Q., Nahrstedt, K.: Jyotish: A Novel Framework for Constructing Predictive Model of People Movement from JointWifi/Bluetooth Trace. In Proc. of PERCOM, Seattle, US, (2011)
7. Krumm J. and Hinckley K.: The NearMe Wireless Proximity Server. In Proc. of UBICOMP, Nottingham, England, (2004)
8. Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G.: A Relative Positioning System for Co-located Mobile Devices. In Proc. of MobiSys (2005)
9. Eagle, N., Pentland, A., Lazer, D.: Inferring friendship network structure by using mobile phone data. In PNAS, vol. 106, nr. 36 (2009)
10. Carreras, I., De Pellegrini, F., Miorandi, D., Tacconi, D., Chlamtac, I.: Why Neighbourhood Matters: Interests-Driven Opportunistic Data Diffusion Schemes. In Proc. of MobiCom workshop CHANTS, San Francisco, US, (2008)
11. Wyatt, D., Bilmes, J., Choudhury, T., Kitts, J. :Towards the Automated Social Analysis of Situated Speech Data. In Proc. of UbiComp, Seoul, Korea, (2008).
12. Cristani, M., Murino, V. Vinciarelli, A.: Socially Intelligent Surveillance and Monitoring: Analysing Social Dimensions of Physical Space. In Proc. of SISM, San Francisco, US (2010)
13. Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G.: A Relative Positioning System for Co-located Mobile Devices. In Proc. of MobiSys (2005)
14. Kaemarungsi, K.: Distribution of WLAN Received Signal Strength Indication for Indoor Location Determination. 1st International Symposium on Wireless Pervasive Computing, 2006.
15. Bhagwat, P., Raman, B., Sanghi, D. :Turning 802.11 inside out. ACM SIGCOMM Computer Communication Review, vol. 34(1), 2004.
16. Ferris, B., Hahnel, D., Dieter F.: Gaussian Processes for Signal Strength-Based Location Estimation. Robotics: Science and Systems, 2006.
17. Popleteev, A.: Indoor positioning using FM radio Signals. PhD thesis, University of Trento, Italy, April 2011.
18. Noth, W.: Handbook of Semiotics, "Proxemics: The Semiotics of Space", Indiana University Press, 1995.
19. Hall, E. : The Hidden Dimnesion. New York: Double Day Anchor Books, 1966.
20. Trullols-Cruces, O., Morillo-Pozo, J, Barcelo-Ordinas, J., Garcia-Vidalpower, J., : Saving Trade-Offs in Delay/Disruptive Tolerant Networks. In Proc. of WowMom, Lucca, Italy, 2011.
21. Matic, A., Osmani, V., Popleteev, A., Mayora, O.: "Smart Phone Sensing to Examine Effects of Social Interactions and Sedentary Work Style on Mood Changes", Proceedings of the 7th International and Interdisciplinary Conference on Modelling and Using Context (Context '11), Germany, 2011.
22. Fischbach, K., Schoder, D., and Gloor, P.A.: Analysis of Informal Communication Networks – A case study. Business & Information Systems Engineering, Vol. 1(2), pp. 140-149, 2009.
23. Matic, A. , Popleteev, A., Osmani, V, Mayora, O. Tuning to your position: FM-radio based Indoor Localization with Spontaneous Recalibration". In proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2010), Mannhein, Germany, 2010.

5. Appendix

In the following, we report a brief documentation of the software part of the present deliverable.

5.1. Mobile Application Software

The mobile component is a standard Android application and it is contained in the realityMining.zip.

/src: This directory contains the source code of the mobile application

/build: This directory contains the deployable version of the mobile application.

/docs: this directory contains the documentation of the software

5.2. Server Application Software

The server component is a standard J2EE Web application. The server is contained in the realityMining.zip and is organized in the following main directories and files:

/WebContent: this directory contains all the files responsible for the visualization of the data

/src: this directory contains all the main classes of the server. This includes classes for both handling the reception and saving of data, as well as the classes responsible for the processing and data mining

/build: this directory contains the deployable version of the web application

/docs: this directory contains the documentation of the software