InstaCircle - A Location-Bound Ad-hoc Network for Android Devices

Project Report

Juerg Ritter (rittj1@bfh.ch)

Bern University of Applied Sciences
Engineering and Information Technology
Biel, Switzerland

January 16, 2013

Abstract

The E-Voting group of the Bern University of Applied Sciences is planning to evaluate distributed E-Voting systems for practical use. The term "distributed" in that context means that there is no central server infrastructure involved in the voting process. The main application of such E-Voting systems are polls with a low number of participants, for example the board of directors in a company. Since the whole implementation of such a system would exceed the time budget of this project, it has been broken down to separate parts. In virtually every E-Voting protocol, messages have to be exchanged between multiple parties. To build up such a messaging platform was the main goal of this project. During this project, an application which allows to connect mobile devices such as smartphones or tablet computers has been built. It allows participants to easyily create an ad-hoc network where messages can be exchanged among all participants. In order to make the configuration of such a conversation a simple as possible, the initiator can share his configuration using QR codes and NFC tags. This allows participants to join a conversation without having to reconfigure their devices manually. The communication is based on network broadcasts within a network segment and therefore also location-bound. The application has been developed for the very popular Android platform. In following projects, the E-Voting part itself can be built on top of this communication infrastructure.

1 Introduction

One of the research fields of the Bern University of Applied Sciences is the area of E-Voting. E-Voting has become a big field of research in the past couple years. Still, there is no generic approach which meets all the criteria such as privacy, transparency, robustness, etc. which we want in E-Voting. The E-Voting research group of the Bern University of Applied Sciences [3] tries to improve this situation with the following approaches:

- Develop new approaches and provide them to the community for review
- Take existing approaches and evaluate them in terms of practicability. These approaches are usually available as scientific papers

The evaluation of these approaches is usually done by implementing them into a prototype level application to show that the approach actually works.The E-Voting group would like to start a new project, whose goal is to implement an E-Voting system for mobile devices such as smartphones or tablet computers. The scenario would be, that each voter uses his/her own device for casting the vote. The communication between the devices happens in a decentralized way (ad-hoc), meaning there is no central server infrastructure. This assures a certain privacy.There are theoretical approaches to realize decentralized elections. For example, in 2012, Hao, Khader, Ryan and Smyth [4] published an approach of a decentralized, fair and robust E-Voting system.

The time budget of this project is a 15 ECTS point project during the MSE studies at the Bern University of Applied Sciences. This time budget doesn't allow to implement a fully-fledged decentralized distributed E-Voting system. Therefore, we split this project into several pieces.In a first stage, the communication infrastructure has to be implemented. A system is required where users can build up an ad-hoc network between multiple devices and exchange messages on this platform. In a later stage, E-Voting protocols can be built on top of this infrastructure, for example the HKRS12 [4] system or any other system of this category.

Having said this, the target of this fist-stage project is stated as follows: The goal of this project is to implement a communication platform where messages can be exchanged in an ad-hoc network which is established between mobile devices. No additional infrastructure (wireless hotspots, servers, etc) should be required. In particular, this communication platform has to provide the following features:

- Initialize an ad-hoc network from a master device
- Allow a group of physically present people to join their devices into the network
- Send point-to-point messages to other devices
- Send broadcast messages to all devices

For the moment, we restrict the implementation to the Google Android platform.

2 Background and Related Work

This section gives a short overview of the basic technologies which should be considered during this project.

2.1 Wireless Communication Technologies

At the time being, there are mainly two standards which allow wireless device to device communication, namely Bluetooth and Wi-Fi (IEEE 802.11g). Another standard called NFC is currently appearing on the surface, but it doesn't play a big role yet nowadays. This section roughly explains the technologies which are used in mobile devices in order to communicate with each other.

2.1.1 Bluetooth

Being introduced in 1994 by Ericsson, Bluetooth [7] evolved to a standard which is used in a wide range of devices for point-to-point communication over short distances. Bluetooth is designed as an ad-hoc protocol, meaning both devices act as master and slave at the same time. Although this seems like a perfectly suitable protocol, it has a major drawback: Bluetooth is designed for two-device communication, and therefore it is not offering a broadcast function which we need in our project.

2.1.2 Wi-Fi (IEEE 802.11g)

Wi-Fi [12] is a protocol that allows to establish an IP network wirelessly. The communication protocol which is used is TCP/IP and hence allows to use all the features of the TCP/IP stack. In ordinary Wi-Fi setups, the devices usually connect to a so called wireless access point which handles all connections. This wireless access point usually covers a certain physical area and let devices in that range connect to the network.

Using a hotspot to establish a wireless connection is not exactly an ad-hoc setup. Therefore, a substandard called Wi-Fi Direct [13] has been introduced. This standard allows to use the Wi-Fi technology without having a hotspot. Two or more devices can establish an ad-hoc network with this technology. Tablets and smartphones are not designed to act as a powerful Wi-Fi access point. The Wi-Fi direct standard doesn't limit the number of connected devices, but it is surely not designed to serve more than 5 connected devices. Behind the scenes, Wi-Fi direct acts as a low scale Wi-Fi hotspot to which other devices can connect. The connecting devices don't necessarily have to be Wi-Fi direct capable. The Android platform provides an API which allows establish Wi-Fi direct connections and exchange data over it, has a major drawback at this point though: During the handshake of two devices, a confirmation pop-up shows up asking to confirm the connection. Unfortunately this is not very practical for the usability of our context.

As a third option, Android offers a hotspot which can be activated on an Android device. This can be done without having the user confirming each connection. Other devices can discover and connect to this hotspot just as any other wireless network. At this stage, the

Android API only offers only an unstable API for controlling this hotspot, but it turns out to work quite well, at least on Samsung devices.

2.1.3 Near Field Communication (NFC)

Near Field Communication [8] is an evolving technology which is designed for short-range communication (typically 4cm or less). It has a wide range of applications, for instance it can be used for sharing content, electronic payments, access control, etc. It is strongly related to the RFID technology, which is also a standard for communication over short distances. NFC is not suitable for exchanging a large amount of data, it can be seen as an alternative for a QR code for example.

In this project, NFC could be used for the handshaking process. For example, instead of entering a predefined PIN code, the participants would just touch their mobile devices in order to enroll for the upcoming election or poll.

2.2 Mobile Platforms

In the past few years, three platforms have proven to be the most successful of its kind. The following sections give a short description of each one.

2.2.1 Google Android

The Android [1] platform of the internet company Google has evolved into the most common platform for mobile devices. It has been built on top of a Linux Kernel. The Android operating system is running on a wide range of devices from different manufacturers. This is probably the main reason why this platform has evolved to the most widespread system used on mobile devices. Google offers a Java Software Development Kit for developing software on the Android Platform. This SDK is available free of charge. For software development, an ordinary computer running either Windows, Linux, or Mac OS can be used.

2.2.2 Apple iOS

The second big player in the area of mobile devices is Apple with their iPhone, iPod and iPad devices. All these devices are running the iOS [2] operating system. The programming language which is used for developing iOS applications is Objective-C. The integrated development environment (IDE) which can be used for developing iOS applications is called XCode, which comes for free with the newest version of the MacOS operating system (10.7).

2.2.3 Microsoft Windows Phone 7

Microsoft recently launched its Windows Phone 7 [14] platform, which is supposed to evolve as a competitor to Apple and Google in the smartphone sector. Microsoft proposes its rather proprietary programming languages Visual Basic and C# for developing applications for Windows Phone 7.

2.3 Sharing credentials

The first application which should be built upon this platform is an ad-hoc E-Voting application. For performing an election, a known set of potential voters must be defined. In the ad-hoc E-Voting scenario, assume that all the potential participants are physically in the same location. Wireless communication is usually not strictly bound to physical locations. Therefore, an additional mechanism is needed to restrict the access to the network. An access credential of some sort has to be distributed using a separate communication channel which is bound to the physical location where all the participants are. The following 3 approaches are considered as a solution for this problem:

2.3.1 PIN Code

The simplest way to control the access to the network is by defining a PIN code and communicate it to the allowed participants. This communication should be done either acoustically or visually, for example displaying it on a screen using an overhead projector. All the participants have to enter this PIN code on their devices in order to get access to the network. This solution doesn't require any additional features of a phone and is therefore applicable for a wide range of mobile devices.

2.3.2 QR codes or bar codes

A bar code [6] is an optical, machine-readable representation of data. It can be found on many products, and is used to identify them quickly using an electronic device. QR codes [9] are the newer version of bar codes. While the data in bar codes is encoded using bars in a line, QR codes are built up using filled or blank pixels on a two-dimensional area. One QR code can encode more data than the relatively old bar code.

Almost every modern mobile device contains a camera nowadays. Using a software which is running on the mobile device, a photographed bar- or QR code can be found on the picture and the data can be extracted. Using this technique, a credential for accessing the network could be encoded and provided to the participants in the physical realm, for example by projecting it to a screen using a overhead projector or printing it out and pass along the paper.

2.3.3 NFC

Near Field Communication (NFC) can be used to communicate wirelessly over short distances. The counterpart can either be another mobile device or a so called NFC tag, which is a passive chip which usually has the size of coin.

For instance, a NFC tag containing the conversation credentials could be passed to all participants. As long as the tag doesn't leave the physical realm, no one from outside the realm can join the network.

Since NFC is a relatively new technique, not all mobile devices are NFC capable yet. Chances are quite high that there are participants with devices that are not NFC capable.

Figure 2.1: NFC tag

2.4 Network Topologies

A classical and simple Wi-Fi network topology is a star topology, meaning there is a central device which handles all the traffic which is flowing through the network. Usually, this central device is a Wi-Fi hotspot. Since we have the requirement that no infrastructure other than the mobile devices of the participants should be used, we have to find another solution. A simple solution is to replace the hotspot with a mobile device of a participant. This device initializes a Wi-Fi hotspot which is running on the device itself, allowing other devices to connect to the network. The big disadvantage of this approach is, that mobile devices are not built to handle a large number of connections from other devices. Hence, this setup is only applicable for a small number of participants (about 5 participants). In order to solve that problem, a number of approaches are available:

2.4.1 Installation of a Wi-Fi hotspot

The limitation of the small number of participants using a software hotspot on a mobile device could be solved by installing an actual hotspot to which all devices are connecting. The management of the participants would still be done on a participant's device since this is not a task of a classic wireless hotspot.

2.4.2 Using an existing hotspot

Many conference facilities are providing a WLAN infrastructure. This infrastructure could be used to act as a communication hub. The principle is the same as above, only that we are using an existing infrastructure rather than bring our own.

2.4.3 Building an ad-hoc mesh network

The most sophisticated approach would be to establish a self-organizing mesh network which links all the devices together without having a central instance which manages all the connections. Using this configuration, all devices can act as server and client at the same time.

Since the connection load is shared in this configuration, the limitation of the small device number is removed. Establishing a robust mesh network is not a simple task tough. Since some devices are acting as a communication gateway for other devices, a failure of these gateway nodes has to be handled somehow, which is not an easy task. Furthermore, the Android API which is going to be used in this project lacks of some functionality which is crucial for this kind of communication platform (for example autoconnect without any user interaction).

The setup of such a mesh network would exceed the time budget of this project. The implementation for such a network infrastructure would certainly be an interesting topic for an upcoming project.

2.5 TCP vs. UDP

On top of IP (Internet Protocol) which is responsible for the addressing and routing in a network, there are two well known concept of communicating within a network:

- **TCP:** TCP (Transmission Control Protocol) [10] is probably the most widely used transmission protocol in the modern network world. Simply speaking, TCP adds a reliability layer on top of IP which assures the delivery of each single packet which is transmitted to a recipient. The recipient has to acknowledge the arrival of every single packet. If the sender doesn't get this acknowledgement, it tries to resend the packet.

- **UDP:** UDP (User Datagram Protocol) [11] is a protocol which allows unreliable communication within a network, meaning the recipient doesn't have to acknowledge every received packet. This reduces the footprint of the protocol and makes it a very efficient way of transmitting data, of course without any guarantee that the content will arrive. It is used primarily in the area of multimedia streaming where the loss of a packet (or a datagram in the case of UDP) is not a big deal.

2.6 Broadcast vs. Unicast

A state-of-the-art computer network offers several ways of communicating with peers in the network, among them broadcast (one to many) and unicast (one to one).

When we evaluate the needs for this project, the choice of a transmission protocol would clearly go to TCP as we clearly don't want to loose any messages. There is one problem though with TCP, which is that we can't broadcast messages. Since the protocol is reliable, the sender expects an acknowledgement from every recipient. In a broadcast scenario this recipient is not clearly defined and therefore the protocol cannot run properly. Hence, TCP can be used for unicast communication only. It would be possible to set up a network chat application with unicast when we go for a central messaging server which distributes all messages among all participants. In this project we would like to avoid to depend on a central infrastructure, which makes this topology a poor choice. Another approach with TCP and unicast communication would be the sender sending each message explicitly to all participants. This approach is inefficient though because for each message we need to open a connection to each participant in the network.

As an alternative to unicast, we could have a look at broadcast. Broadcast seems to be exactly what we are looking for, messages go to all participants in a network segment which is exactly what we want in a chat-like application. We have already discussed that TCP is not suited for broadcasting, so we need to switch to UDP, which in turn is not reliable.

As a compromise, a hybrid setup could be built where the recipient of a message asks the sender using an unicast channel to resend the message when it has been detected as missing.

3 Results

The result of this project is an application called **InstaCircle** which is running on Android based mobile devices such as smartphones or tablet computers. This application allows to connect multiple mobile devices to a network which allows the devices to exchange messages. Messages can be exchanged in two fashions:

- **Broadcast:** A message which is sent in broadcast mode will be sent to all participants in the network.
- **Unicast:** A message which is sent in unicast mode is designated to a recipient, meaning only this particular participant will receive this message.

The following sections give some more details about the architecture and techniques which have been used in this project.

3.1 Architecture

The Android platform offers a wide range of pre-built components which allow to build applications which integrate themselves neatly into an Android device. Most important ones are the following:

- **Activities:** An Android activity basically represents a "screen" on a device. It displays components such as buttons, textboxes, labels and other controls which allow the user to interact with the application. In InstaCircle this technique has been used to implement the user interface.
- **Services:** Services can be used for running code in the background, meaning the user is using another application. InstaCircle uses this technique for implementing handling the interface to the network and process incoming messages even if the InstaCircle user interface is in the background.
- **Local Broadcasts:** Local broadcasts can be used in order to let components like activities, services, etc. communicate with each other. Example: As soon as a message arrives and got processed by the service, it notifies the user interface with a broadcast

that a new message has been arrived. The user interface can update the content accordingly if necessary.

- **SQLite Database:** The Android platform tightly integrates the popular lightweight database SQLite which makes it easy to store relational data. InstaCircle uses this feature for storing the conversations on the device.

3.1.1 Software components

Figure 3.1 gives an overview of the software components which have been implemented in InstaCircle and how they interact with each other.

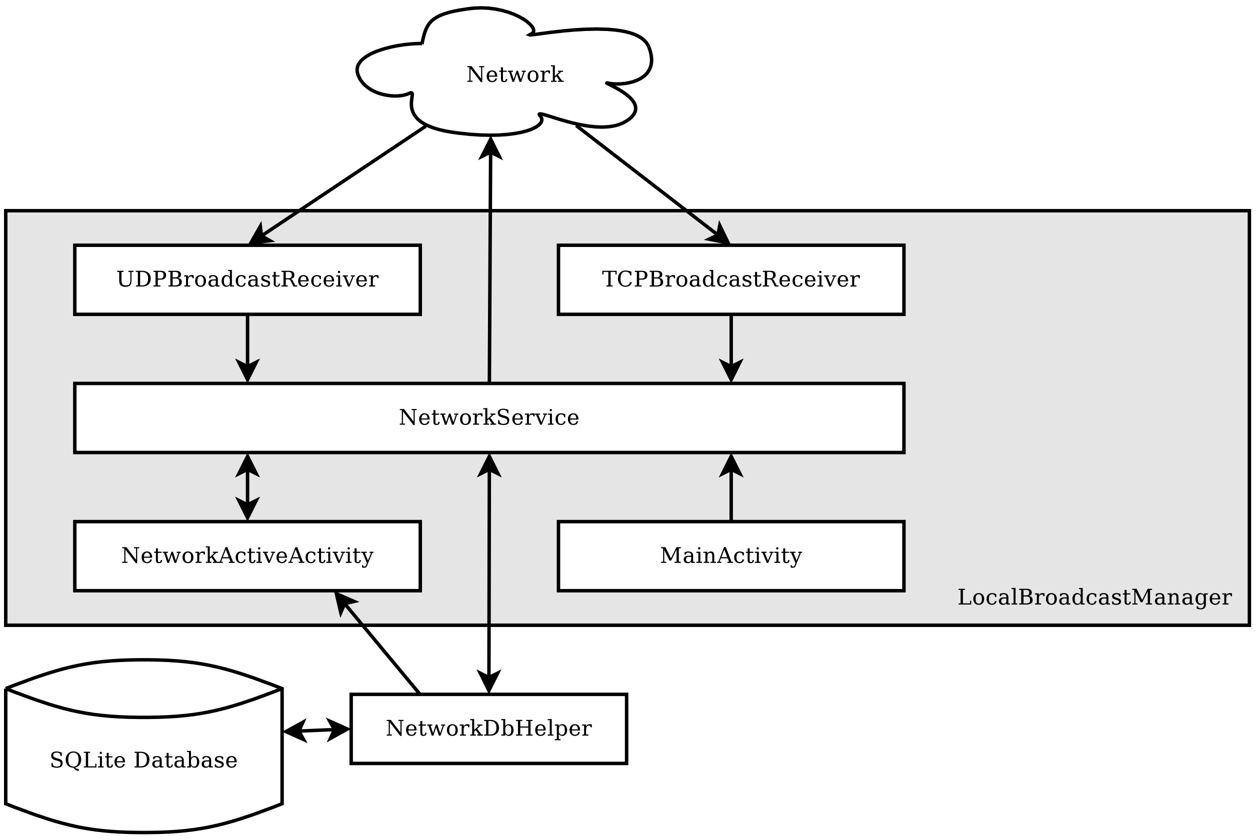


Figure 3.1: Architecture

- **MainActivity:** The MainActivity implements the user interface which is displayed after the application launch. It is responsible for configuring the Android device in order to participate in a conversation.

- **NetworkActiveActivity:** The NetworkActiveActivity is displayed during a conversation and keeps the user informed about incoming messages, the participants and general network information. It also offers the possibility to share the network configuration using a QR code or a NFC tag.

- **NetworkService:** The NetworkService implements an Android service. It is responsible to process incoming and outgoing messages. The service runs in the background and is therefore able to handle messages even if the user is using other applications on the device. It uses the local broadcasting infrastructure in order to communicate with the Activities.
- **UDPBroadcastReceiver:** The UDPBroadcastReceiver is an implementation of a Java thread. It allocates a UDP socket and waits for broadcast messages to arrive and passes them back to the NetworkService where they are being processed.
- **TCPBroadcastReceiver:** The TCPBroadcastReceiver is an implementation of a Java thread. It allocates a TCP server socket and waits for unicast messages to arrive and passes them back to the NetworkService where they are being processed.
- **NetworkDbHelper:** This component acts as a gateway to the SQLite database and hosts methods to query and manipulate data records in the SQLite database of InstaCircle.

One big advantage of this architecture is that the user interface which presents the messages (in this case the NetworkActiveActivity) can be exchanged. This comes in handy if this communication platform should be used for something else than a simple chat application.

3.1.2 Data schema

As already mentioned, InstaCircle uses a relational data structure in order to store information which accumulates during conversations. Figure 3.2 shows the the simple relational data model which is used in this application.

There are three entities in the model: A conversation, which itself can have multiple participants. Participants in turn can be the sender of multiple messages in the conversation.

3.2 Messaging model

A goal of this project is to keep the necessary infrastructure as minimal as possible, so as already discussed in section 2.6 we need to think about implementing a protocol using broadcast and the unreliable UDP transmission protocol. The solution applied in this project looks as follows: Each message which contains conversation relevant content is assigned a sequence number by the sender and is then sent as a broadcast UDP datagram into the network. All participants can pick up this message, process it and store it into the database. During the processing, the recipient compares the sequence number of the received message with the sequence number it expects from this participant. If this comparison succeeds, the recipient knows that it got all the messages from this particular sender during this conversation. In case the comparison fails, there are missing messages from that sender. In that case, the recipient asks the sender to resend the missing messages using a unicast TCP channel.

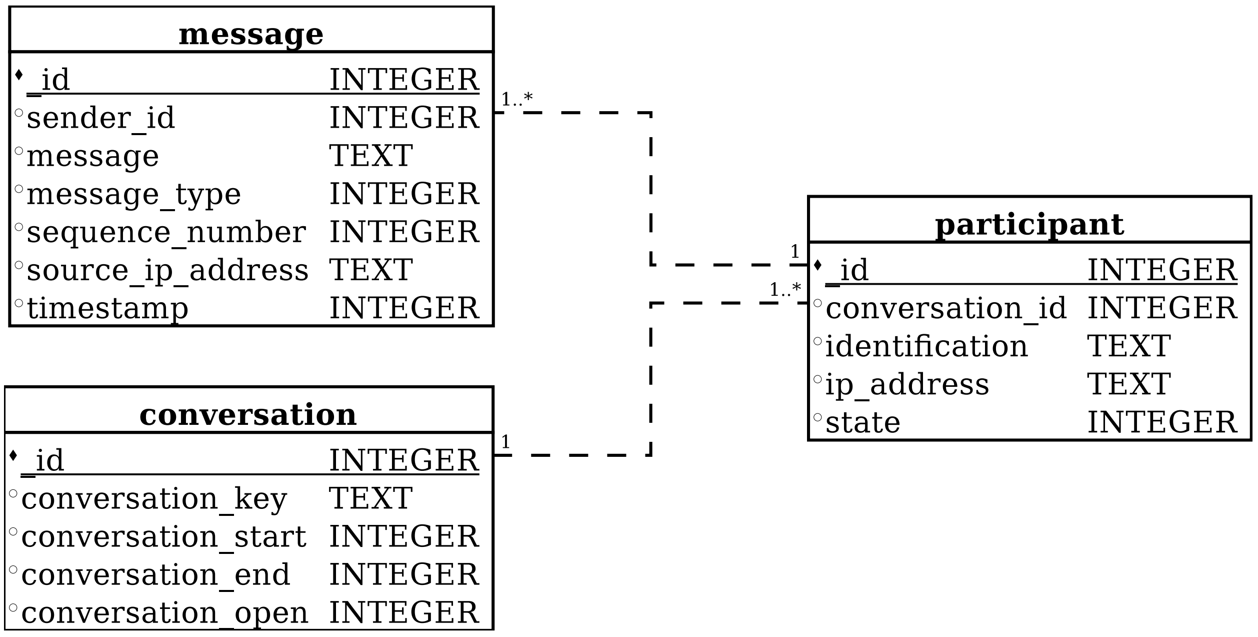


Figure 3.2: Data Model

3.2.1 Message types

InstaCircle uses 7 different message types in order to communicate. 3 of them hold communication relevant information, the other 4 types are used to govern the message flow during the conversation. Table 3.1 gives an overview of the message types used in InstaCircle.

3.2.2 Weaknesses

At the first sight this protocol seems to be straight forward, has a couple of weaknesses though:

Currency. Missing messages are only detected as soon as the following message arrives. This problem could be solved by broadcasting a beacon containing the sequence number in a certain interval. Using this technique, the maximum delay each participant could get is the interval of the beacon. The problem remains tough if a participant leaves unexpectedly between sending a message and the next beacon.

Manipulation of sequence numbers. The order in the conversation could be disturbed if a participant is manipulating sequence numbers, causing the other participants to ask for missing messages. This could lead to a denial of service attack. This problem is generally considered hard, in TCP which is using a similar sequencing technique this flaw also exists.

Asynchronous clocks. Since there is no central infrastructure, there is also no reference

Table 3.1: Message Types

Name	**Type number**	**Stored**	**Channel**	**Description**
MSG_CONTENT	1	yes	Broadcast / Unicast	Message type which transfers conversation content.
MSG_JOIN	2	yes	Broadcast	Indicates a participant joining into the conversation.
MSG_LEAVE	3	yes	Broadcast	Indicates a participant leaving the conversation.
MSG_RESENDREQ	4	no	Unicast	Used for asking a participant to resend messages.
MSG_RESENDRES	5	no	Unicast	Used to respond to a resend request. Such a resend response contains an array of the messages for which the requester asked.
MSG_WHOISTHERE	6	no	Broadcast	Can be used to discover who is around in the network, this is mainly used after the join of a participant.
MSG_IAMHERE	7	no	Unicast	Response to the WHOISTHERE request. It will be sent directly to the requester containing the current sequence number.

time which could be used as a reference for the conversation. In order to deal with the fact that messages could be delivered with delays, the timestamp of the message has to be set already by the sender of the message. This is no problem as long as the clocks of the participant's devices are in sync. Otherwise it is possible that the timeline of the conversation is not quite accurate as all devices are using a different reference.Clocks of modern smartphones are usually synchronized using the cellular network or a GPS signal, which somewhat remediates this weakness.

3.3 Security

The goal of this project was to create a protected conversation, meaning it shouldn't be possible for everyone to join in without knowing a credential. Since we are not sure that in a particular wireless LAN are only devices of participants connected, we need to secure the conversation with another technique. Since we share a credential which is used for participation, we can use that credential as a symmetric key. The content of all messages are encrypted with this key using AES. The android is shipped with the Bouncy Castle [5] security provider for Java, so there is no need to install further software to use this feature.

Technically it would be possible to have multiple InstaCircle conversations running in the same network segment. In this case, broadcasts of both conversations would go to all devices in the network. Multiple conversations are kept separate by just ignoring messages if the credential cannot be used for decrypting the message. This ensures that conversations are kept separately as long as the credentials are different.

3.4 Network Topologies

InstaCircle offers the option to activate a Wi-Fi hostspot on the Android device which then acts as a hub where the other participants can connect to. This hotspot needs to be activated on one device only and can be used by the other participants as a communication hub. It is only necessary to activate a hotspot if there is no other suitable wireless network in range though, if there is a wireless network to which all the participants can connect, it is probably the simpler option to connect all the devices to this network and use this hotspot as the communication hub.

3.5 Exchanging the configuration

In order to join an InstaCircle configuration there are basically two parameters which must be known:

- The WLAN SSID
- The credential of the conversation

All devices have to be configured to use these parameters. The classic way of is of course just to connect to the according WLAN and type in the credential, but this is quite painful and error prone. In order to make this process a little simpler, InstaCircle implements two alternative ways of sharing a configuration to other participants.

QR Code. As soon as the network is configured, the user interface provides an option which displays a QR code of the configuration. This configuration contains the WLAN SSID and the credentials, separating the two values by two pipe characters. This QR code could also be sent to a computer and displayed on an overhead projector for example. The main screen of InstaCircle offers an option to capture a QR code using the camera of the phone and instantly connect to the conversation afterwards. The functionality for displaying and capturing the QR codes is implemented in a separate Android application called “Barcode Scanner” which has to be installed separately.

NFC Tag. For NFC enabled devices, InstaCircle also offers the possibility to share the configuration using a NFC Tag. After configuring the network, the user interface offers an option to write the configuration to a NFC tag, which then can be passed through the participants. The InstaCircle application automatically starts and connects immediately

Table 3.2: Devices for testing

Model	Form factor	Android Version	WLAN	Camera	NFC
Samsung Galaxy S1	Smartphone	4.1.1 (unofficial image)	yes	yes	no
Samsung Galaxy S3	Smartphone	4.1.2 (official image)	yes	yes	yes
Samsung Galaxy Tab	Tablet	4.0.4 (official image)	yes	yes	no

after tapping the NFC tag on the back of the device, provided that the application is installed and NFC is enabled on the device.

3.6 Testing

Testing the functionality was quite a challenge because of limited availability of hardware. The Android Software Development Kit (SDK) offers an emulator which can be used to run the developed code directly on the computer, but due to the heavy dependency on the hardware this was not an option. Finally the testing was done on three Galaxy devices of the Samsung corporation. Table 3.2 gives an overview of the testing equipment which was available in order to test the functionality of the software. Moreover there was a writable NFC tag available which allowed to test the NFC functionality. Since there was only one NFC capable device available, the test could only be done by saving the configuration and then rejoining the same device into the conversation which is not a real-life scenario.

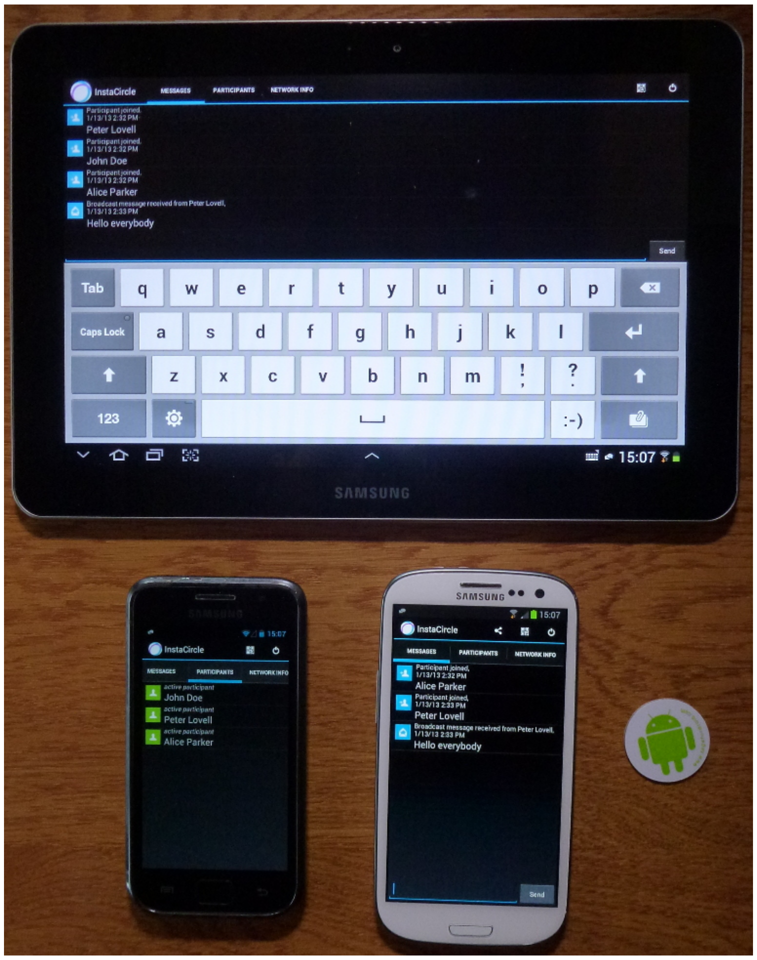
3.7 Documentation

During this project, three pieces of documentation have been created. First of all, this project report describes the technologies and methods which have been applied and gives information about the project structure. To describe how to use the software, a brief user manual can be found at the end of this document in the appendix. Finally, in order to make the code more readable, there are comments in the code which included in the very common Javadoc style which is a standard of documenting the code when programming Java.

4 Problems

As most of the projects, also this one turned out to bear some surprises and problems.

Reliable communication without master. The assignment of building a communication infrastructure without a master doesn't seem to be hard at first. It turns out that there are some obstacles such as time synchronization and the detection of missed messages. These problems can be solved with some mechanisms for the most part, but these solutions are not perfect. It would probably be worth it to look at some sort of centralized messaging platform as an alternative.



Participant joined, 1/13/13 2:32 PM
Peter Lovell
Participant joined, 1/13/13 2:32 PM
John Doe
Participant joined, 1/13/13 2:32 PM
Alice Parker
Broadcast message received from Peter Lovell, 1/13/13 2:33 PM
Hello everybody

active participant
John Doe
active participant
Peter Lovell
active participant
Alice Parker

Participant joined, 1/13/13 2:32 PM
Alice Parker
Participant joined, 1/13/13 2:33 PM
Peter Lovell
Broadcast message received from Peter Lovell, 1/13/13 2:33 PM
Hello everybody

Figure 3.3: The device setup for testing

The Wi-Fi API of Android. Generally speaking, Android has a quite intuitive and well structured API for building applications. The Wi-Fi part of the API seems to make an exception here. It is quite hard to understand the concept. For example there is no universal way for handling the keys for the different encryption methods. The lack of clear error messages made it hard to track down the problems, so a lot of time went into building and troubleshooting the Wi-Fi part of this application.

Homogenous devices for testing. The application could only be tested on Samsung hardware, it would have been interesting to test the software on other hardware as well. Furthermore, the NFC part could only be tested with one device because there was only one NFC capable device available.

5 Future Work

This project has already been realized with prospect to further project which will dive deeper into the world of decentralized E-Voting systems such as the on provided by Hao et al. [4]. At the heart of virtually every E-Voting system is a mechanism of exchanging messages among the participants, and this project provides a platform to do so on Android devices. At this stage, these E-Voting platforms can be built on top of this project.

6 Conclusion

With this project we lay the groundwork for further projects with focus on distributed E-Voting systems. The Android application which has been built during this project allows users to simply connect their mobile devices to an ad-hoc network where messages can be exchanged, either in unicast- or broadcast mode. Exchanging simple messages over this platform is just the first stage of further, more sophisticated applications which can be built on top of this platform. Personally it was also a great experience to dive deeper into the Android platform, from which I only had a very shallow knowledge at the beginning of this project.

Acknowledgments

I would like to thank all the members of the E-Voting group of the Bern University of Applied Sciences who contributed a great deal by giving valuable input in constructive discussions. A special thanks goes to my advisor Prof. Dr. Rolf Haenni who provided me with valuable inputs and ideas. Moreover I would like to thank Prof. Reto Koenig for sharing his great expertise in developing software for Android, which was helpful many times.

A User Manual

A.1 Prerequisites

In order to run InstaCircle, you need a Android device running at least Android 4.0 (Ice Cream Sandwich). It runs on smartphones as well as tablet computers. In order to use the advanced credential sharing features you need the appropriate hardware, i.e. a camera for reading the QR codes and NFC capability to use the NFC sharing functionality. If you want to use the QR code sharing functionality, the application "Barcode Scanner" needs to be installed. This application can be found on the Google Play platform.

A.2 Installation

At this stage, the InstaCircle is not available on the Google Play platform yet, so it has to be installed manually. As a first step, the InstaCircle.apk file needs to be transferred to the Android device either by downloading it over the wireless network or by transferring it by attaching the device using USB to a computer. Second, make sure that you allow the device to install software from unknown Sources (i.e. not the Google Play store). This option can be found here:

Settings → Security → Unknown Sources

It is recommended to switch that option off after the installation. Finally you can start the installation process by just tapping on the InstaCircle.apk file in a file manager of your choice. After the installation the application is available in your application browser and can be started. If you want to use the QR code sharing functionality you also need to install the "Barcode Scanner" application which is available on the Google Play platform.

A.3 Start of InstaCircle

Simply click on the InstaCircle Icon in your application browser in order to start InstaCircle. It will lead you to the start screen as shown in figure A.1. The first thing you'll need to do is to specify an identification name for which will be used to identify you in the conversation. InstaCircle will try to extract the name of the owner of the device and present it as a suggestion in the text box. Feel free to change it if you don't like it. This setting will be saved for more upcoming conversations.

You are now ready to connect to a network. The principle of joining or creating a network is basically the same, with the only difference that if you want to create a new chat, you cannot use the credential sharing mechanisms such as QR code and NFC.

A.4 Creating a new conversation

In order to create a new conversation, you can just tap on the Wireless Network which you would like to use. In the pop-up you can specify the password which will be used for the

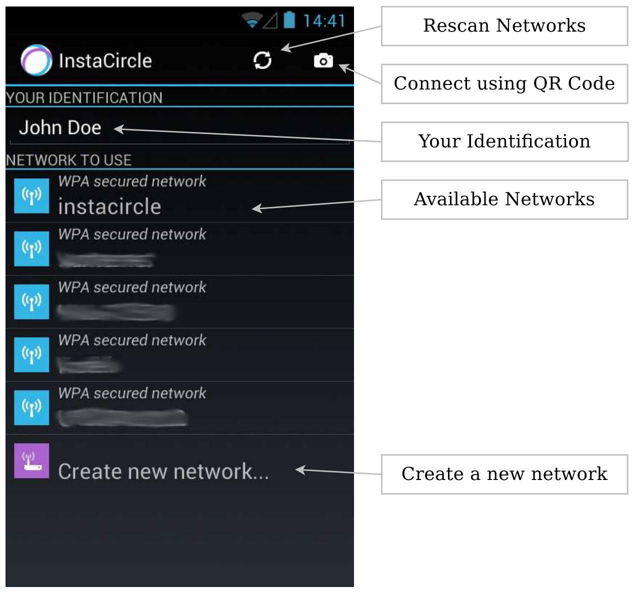


Figure A.1: Start screen

conversation. You'll then be redirected to the conversation screen as shown in figure A.2.

You also have the possibility to create your own network using the integrated hotspot of Android. You can do that by tapping on "Create new network". In the opening pop-up you'll have to specify the a name as well as a password for the new network. After clicking on the button "Create Network", the hotspot will be activated and you will be redirected to the conversation screen as shown in figure A.2.

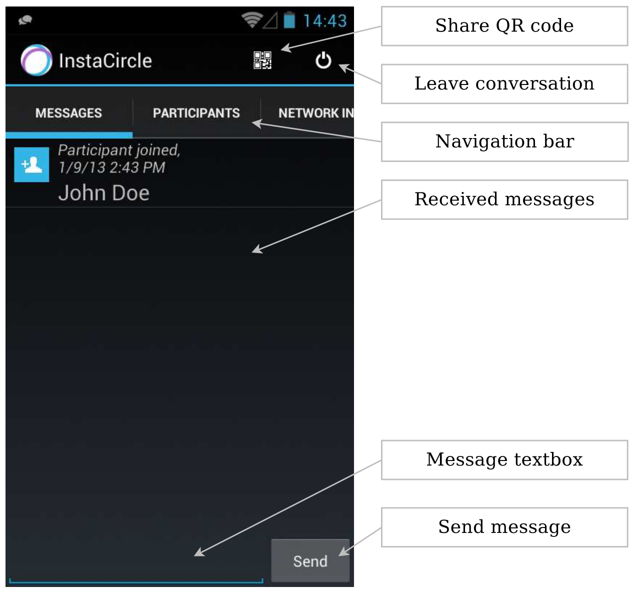


Figure A.2: Conversation Screen

A.5 Sharing the credentials

In order to connect to a conversation, one needs to know the network name and the password for the conversation. These two parameters can be communicated orally or with other media. The InstaCircle application provides two more sophisticated mechanisms to share these credentials. This section will describe how to share the credentials after having created a network. How to use them please see section A.6.

QR code: Once the conversation is set up, the credentials for the conversation can be shared using a QR code by tapping on the according icon in the action bar on top of the conversation screen (figure A.2). A new screen containing a QR code will be launched. This QR code can now be captured of other devices, or you can tap on the menu button (usually on the bottom left on most devices) which brings up a “Share” button. From here you can share it as a picture, for example sending it to a computer and display the code on an overhead projector.

NFC tag: The credentials can also be written to a writable NFC tag. Of course this feature is only available on NFC capable devices. After tapping on the NFC share icon a dialog will pop up asking you to tap a writable NFC tag on the back of your device (see figure A.3). After having done that, a dialog indicating the success of the write operation. The newly written tag can now be passed along the participants who want to connect their devices.

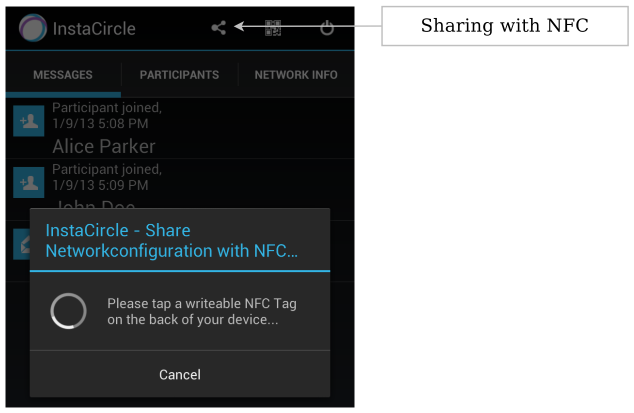

InstaCircle
Sharing with NFC
MESSAGES
PARTICIPANTS
NETWORK INFO
Participant joined, 1/9/13 5:08 PM
Alice Parker
Participant joined, 1/9/13 5:09 PM
InstaCircle - Share Networkconfiguration with NFC...
Please tap a writeable NFC Tag on the back of your device...
Cancel

Figure A.3: Writing NFC Tag

A.6 Joining a conversation

There are three possibilities to join into a conversation which will be explained in the following section.

Typing the password: The process of connecting to a conversation using just the password is basically the same as creating a new conversation. After tapping on the according network in the start screen (figure A.1) you'll just have to enter the password which has been given to you from the conversation issuer in the password dialog. You'll be redirected to the conversation screen (figure A.2) and you'll be able to exchange messages with the other participants.

QR code: If your devices has a camera, you can also use a QR code which is presented to you. This QR code contains the SSID as well as the password for this conversation. In order to initiate the capturing of the QR code, just tap on the "Connect using QR Code" icon in the action bar on the top of your start screen (figure A.1). On the new screen you'll see the picture as seen from your camera's perspective. You'll now have to center the QR code in the clear area of the screen. The software captures automatically the QR code as soon as it is readable and initiates the connecting process. If the connect was successful, you'll be redirected to the conversation screen (figure A.2).

NFC tag: If you have a NFC capable device, the only thing you need to do is tapping the NFC tag on the back of the device. You should get this prepared NFC tag from the issuer of the conversation. Make sure that NFC is enabled on your device, it won't work otherwise. In order to connect to a conversation using an NFC device, it is not even necessary to launch the InstaCircle application manually. Just tap the tag on the back of the device and InstaCircle immediately launches and initiates the connection process. After a successful connect you'll find yourself in the conversation screen (figure A.2).

A.7 The conversation screen

The conversation screen has three views which are accessible through the navigation tabs on top at the bottom of the action bar.

Messages: In this tab you can see all the messages which have been exchanged so far during the conversation. You can also see if other participants joined into the conversation or when participants leave the conversation. This is indicated with a different icon and a different message text. In this view you also see the messages which have been sent directly to you by another participant. At the bottom of the screen there is a text field where you can compose and send your own broadcast messages.

Participants: In this tab you can find a list of all involved participants of this conversation. A participant can have the state active and inactive, which is also indicated in this view. A participant becomes inactive as soon as he/she leaves the conversation. When tapping on the participant, a detail screen as displayed in figure A.4 will be launched where you can see further information (IP address, sequencing number, etc.) of this particular conversation. Further down there is a text field which can be used to compose messages which will only be seen by this particular participant (unicast messages). You can navigate back to the previous screen by tapping the back button of your device, or just by clicking on the InstaCircle icon on the top left of the screen.

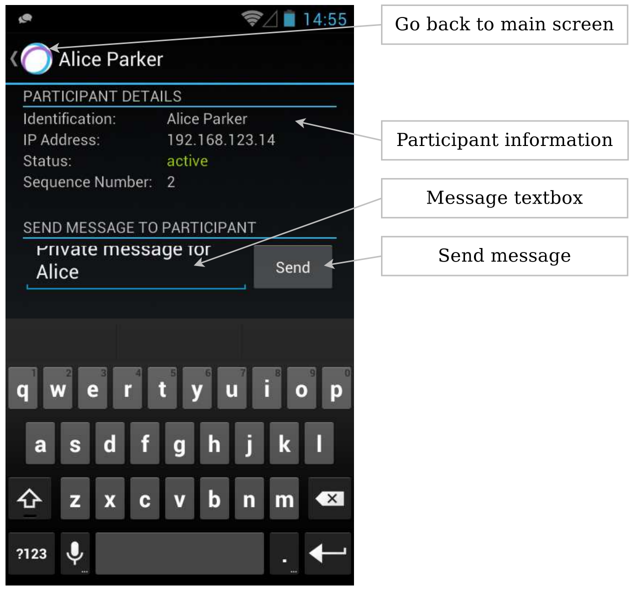
Network Info: In the last tab you can find information concerning the network configuration or you as a participant.

A.8 Leaving the conversation

In order to leave a conversation, it is not enough to just close the InstaCircle application. Until you explicitly leave the conversation, the application runs in the background and is able to receive messages even if the application is not visible on your screen. If you want to leave the conversation, you have to tap on the according icon on the top right of the conversation screen (figure A.2). After having confirmed your leave in the following dialog, you will be redirected back to the start screen where you can either initiate a new conversation, connect to another conversation or just quit the application. At this point you are no longer able to receive any messages of the conversation.

A.9 Rejoining the conversation

After having left a conversation, you can always rejoin a conversation just as you would connect for the first time. In order to keep you updated, The application will ask all other participants to resend you the messages which were exchanged during your absence.



PARTICIPANT DETAILS

Identification:	Alice Parker
IP Address:	192.168.123.14
Status:	active
Sequence Number:	2

SEND MESSAGE TO PARTICIPANT

Private message for Alice

Figure A.4: Participant Details

References

[1] Android, *Android.* `http://www.android.com/`, 2013. [Online; accessed 12-January-2013].

[2] Apple iOS, *Apple iOS.* `http://www.apple.com/ios/`, 2013. [Online; accessed 12-January-2013].

[3] BFH E-Voting Group, *E-Voting Group.* `http://e-voting.bfh.ch/`, 2013. [Online; accessed 12-January-2013].

[4] F. HAO, D. KHADER, P. Y. A. RYAN, and B. SMYTH, *A Fair and Robust Voting System by Broadcast*, (2012).

[5] The Bouncy Castle Project, *Bouncy Castle.* `http://www.bouncycastle.org/`, 2013. [Online; accessed 12-January-2013].

[6] Wikipedia, *Barcode — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Barcode`, 2013. [Online; accessed 12-January-2013].

[7] ——, *Bluetooth — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Bluetooth`, 2013. [Online; accessed 12-January-2013].

[8] ——, *Near field communication — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Near_field_communication`, 2013. [Online; accessed 12-January-2013].

[9] ——, *QR code — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/QR_code`, 2013. [Online; accessed 12-January-2013].

[10] ——, *Transmission Control Protocol — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Transmission_Control_Protocol`, 2013. [Online; accessed 12-January-2013].

[11] ——, *User Datagram Protocol — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/User_Datagram_Protocol`, 2013. [Online; accessed 12-January-2013].

[12] ——, *Wi-Fi — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Wi-Fi`, 2013. [Online; accessed 12-January-2013].

[13] ——, *Wi-Fi direct — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/wiki/Wi-Fi_Direct`, 2013. [Online; accessed 12-January-2013].

[14] Windows Phone, *Windows phone.* `http://www.windowsphone.com/`, 2013. [Online; accessed 12-January-2013].