Wi-Fi P2P Chat

Distributed Systems – Project Proposal

Alexander Weber, Sammy Jäger
14-915-797, 12-935-631
weberale@student.ethz.ch ,
sajaeger@student.ethz.ch

Peter Grönquist, Rubén Fischer
13-935-317, 13-928-742
petergro@student.ethz.ch,
rubfisch@student.ethz.ch

Michael Seeber, Bastian Morath
15-941-198, 15-916-786
seeberm@student.ethz.ch,
morathba@student.ethz.ch

1. INTRODUCTION

Messaging apps have been around for quite a while. Most of them have one big problem: They require the devices to be online, that is having network access, which can be problematic in airplanes as well as highly crowded areas like festivals where the cellular network might not be able to handle the high load.

Our goal is to exploit the opportunities modern mobile devices and operating systems provide, to create a messaging app that works without network access. The app sets up a fully offline peer-to-peer connection between two or more devices, using the Wi-Fi P2P capabilities provided by Android phones.

2. SYSTEM OVERVIEW

2.1 User Interface & UI Logic

The structure of the app follows more or less the approach of a simple messaging application. Figure 2.1 gives an overview of the app Activities needed.

The app starts in the *MainActivity*. Here the user can set a username that identifies him. After entering a name, the app will switch to the *BroadcastReceiverActivity* through the press of the Join-Button. This activity lists all users that are within reach, and thus can be messaged. The user can start a new chat with one of the listed users by clicking on the entry. This will bring up an alert, telling him that he must wait for the chat partner to accept his request in order to start messaging.

As soon as the chat request gets approved, the app switches to the *ChatActivity*, where the users can hold a conversation like in every other standard messaging app. The user is also able to add other users to an already active chat. For this, he must press the plus sign in the top right corner, which will bring up the *InviteUsersActivity*. This screen lists all the active users within reach that can be added to the chat. As before, they receive an alert, which allows them to either accept or cancel the request to join an ongoing conversation.

2.2 Software Architecture

The most important part of our app will be the Wi-Fi peer-to-peer (P2P) API, which allows Android 4.0 or later devices to connect to each other via Wi-Fi without an intermediate access point. [1]

We propose the following structure for the P2P part of our app: [2]

2.2.1 *Connection Establishment*

In order to establish a connection between the devices we need several components, the main ones being a *WifiP2pManager* and a *BroadcastReceiver*. The *WifiP2pManager* allows us to connect the application to the Android Wi-Fi P2P framework, while the broadcast receiver responds to intents broadcasted by the Android system, which allows us to handle chat requests, changing peers and so on. As soon as we obtain a *WifiP2pManager* and set up a broadcast receiver, the app can make Wi-Fi P2P method calls and receive Wi-Fi P2P intents.

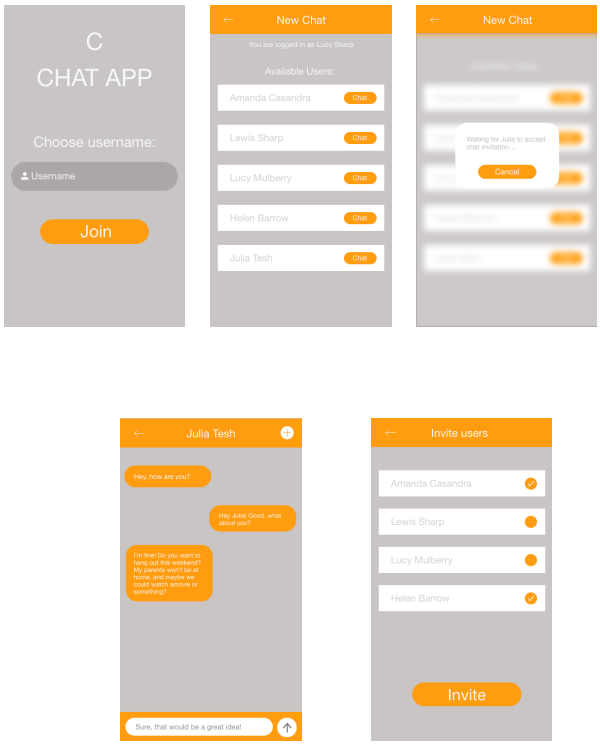

Figure 1: Idea of the final UI design

- Discovering Peers:

  To discover peers, we call the *discoverPeers()* function on our Manager. If this call succeeds, the system broadcasts the WIFI_P2P_PEERS_CHANGED_ACTION intent, which we can then listen for in our Broadcast Receiver to actually obtain the list of currently available peers and to update the list of possible chat partners accordingly.

- Connecting to Peers:

  When the user has figured out with whom he wants to chat, i.e. which device to connect with, the app needs to call the *connect()* method with an *WifiP2pConfig* object that contains the information of the device. The connection success or possible failure can be handled by an *ActionListener*.

2.2.2 Transfer & Information Exchange

Once a connection is established between two devices, the messages are transferred via sockets. The Wi-Fi P2P framework selects a *GroupOwner* that accepts connections using a server socket and then spawns client sockets for every client. We thus propose having a *ClientSocketHandler* and a *GroupOwnerSocketHandler*, respectively. Further we propose having a ChatManager that handles reading and writing of messages with socket buffers. The main steps are thus:

1. Create a server socket at the *GroupOwner*. This socket waits for the connection from the Clients on a specified port and blocks until it happens, so this is done in a background thread.
2. Create a client socket at all clients. They use the IP address and port of the *GroupOwner* socket to connect to the server device.
3. The *ChatManager* then handles the messages via the output- and inputStream of the socket to write and read, respectively, and posts them back to the UI Activity to display.

3. REQUIREMENTS

1. Use the *Wi-FI P2P* API to enable offline communication. This includes the following:
   - Receive a list of available devices
   - Connect to a specific device
   - Transfer data, i.e. chat messages between at least 2 devices
2. Each user can have only one active chat at a time.
3. Users can do the following:
   - Create or join a chat, either by sending a chat request to another device within reach or by accepting one
   - Send messages to all members of the active chat
   - Receive messages from all the members of the active chat
   - Add additional users to the active chat
   - A user should be able to disconnect from an active chat
4. People who want to chat with each other need to be within range of their Wi-Fi signal.
5. Each device needs to support Wi-Fi Direct, i.e. runs at least Android 4.0 and has a working Wi-Fi adapter

4. WORK PACKAGES

- **WP1**: *Main Activity*: Create Sign-In view
- **WP2**: Implement *WifiP2pManager* & *BroadcastReceiver*
- **WP3**: *BroadcastReceiverActivity*: Display peers
- **WP4**: Connect to a selected peer & send a "chat"-request
- **WP5**: *ChatActivity*: Create UI for exchanging messages
- **WP6**: Data-transfer component
- **WP7**: *InviteUsersActivity*: Create view where the user can invite more users to an already existing chat
- **WP8**: User should be able to disconnect from a chat

5. MILESTONES

Workpackages Distribution:

- Alexander Weber: WP1
- Peter Grönquist: WP2
- Rubén Fischer: WP3
- Sammy Jaeger WP4
- Bastian Morath: WP5
- Michael Seeber: WP6
- Group: WP7, WP8

Schedule:

- Week 1: After the first week, the main structure of the Activities should be there.
- Week 2 & 3: In the following two weeks, the core of our app, the *Wi-Fi P2P* API should be implemented and be working.
- Week 4: The last week acts as a buffer and is here for bug fixes and optimizations

6. REFERENCES

[1] Android Wifi P2P. `https://developer.android.com/guide/topics/connectivity/wifip2p.html`.
[2] WiFiDirectServiceDiscovery DemoApp. `https://github.com/aosp-mirror/platform_development/tree/master/samples/WiFiDirectServiceDiscovery`.