TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science, Institute for Systems Architecture, Chair of Computer Networks

Masters Thesis

Dynamic Group Generation based on Interests

Eduardo Jacobo Miranda Ackerman
Born: 15. 06. 1977, Tijuana Mexico

Adviser: Dipl.- Medieninf Maximilian Walter
Advising Professor: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Date: 10. 12. 2009

Dedicated to my dear children and the memory of my father.

Acknowledgements

I would like to thank Karina my wife for supporting and helping me in this long process. The Mobilis group for their help and support Dr.-Ing. Thomas Springer, Dr.-Ing. Daniel Schuster and specially Dipl.-Medieninf. Maximilian Walther for the patience and guidance he showed during the development of this work. I would also like to thank Prof. Schill for strong direction and motivation to finish this work.

Contents

List of Figures

List of Tables

Glossary

API Application programming interface; an interface that defines the ways to request services

MSD Mobile smart devices; capable of communications and computer processing to run specialized applications

PC Stationary desktop computer

Smartphones Mobile phone with PC like functionality

tweets to send a brief message to the twitter social network

Chapter 1

Introduction

The Internet, a system developed to interconnect computers, has become readily accessible to Smartphones. With the development of Web 2.0, more services and applications are able to move off the personal computer and into hosted services, but these services are not always accessible to smartphones. Many services have not been adapted to work within the constraints of mobile devices such as a limited interface and limited processing power. Although there have been many improvements in Web design and Web development that facilitate interactivity, there are still many services lacking this improved interactivity(1) for Smartphones.

One of the tools propose to facilitate interactivity on a Smartphone, for a group of participants, is dynamic group generation. Because there can be many factors that define which participants should conform a group, the particular attribute of a common interest has been chosen to generate the groups. Dynamic group generation is not something new. Smartphone users are able to contact multiple parties via conference calls or messages to multiple recipient, these mecahnisms may be elaborate and cumbersomb for the user . The innovation item presented here is the mechanism by which the participants in a groups are selected. Instead of selecting participants one by one(2), the groups participants are selected from a central server that allows users to come together who might not otherwise know each other thus facilitating group creation.

According to (3), there are already more mobile web users in the world than PC based Internet users, there are also predictions that by 2015 more than 50% of the world will have an Internet connection, much of this growth coming from mobile phones. This

may be due to a higher number of services being available to users at a lower cost. If trends continue, Smartphones will have a lower cost and will present more sophisticated technical capabilities. The technical developments pioneered in the PC market will aid in the development of the smartphone, making the developments process faster. An example of this is the Android [1] smartphone that is built upon an operating system originally designed for stationary PCs.

There are many aspects that make smartphones attractive to users, such as enhanced reality, device mobility and location awareness(4), which are features that are not functional on PCs. There is also a current trend of interactive sites to turn mobile, this is to acces the services via a smartphone, allowing more features to be available to the smartphone's user. Some of these sites, such as social networking sites, personal blogs and image repositories allow content to be posted from the smartphone. This allows individuals to communicate to many others but provides little or no feedback communication.

Some of the caveats to smartphone functionality are inherent to its features: the compact design limits screen size and interfacing controls and the portability limits the power available for processing. Although there are attempts to improve application navigation and performance with features such as touch screens and voice commands, there are still many shortcomings in the ease of use compared to a PC. This is one of the reasons there is a need to develop tools to facilitate interaction between the user and the the services available to the smartphone, either by providing what the users may expect of the system or by changing what the user can expect of the system eather way enhancing the user's experience. This work proposes that dynamic group creation does just that, enhance the user experience by changing the way groups of participants establish communication.

1.1 Aim

The main goal of this thesis is to dynamically generate and manage groups of participants based on a common interest for the Mobilis(5) platform, extending a service similar to those provided by social networking services, where individuals with similar

[1] http://www.android.com/about/

interests or commonalities converse and connect with each another. The functionality would extend the services provided by AndroidBuddy[1], an application based on the Mobilis platform, by registering the users interests and creating a group when sufficient commonalities are found.

Although the main activity generated for the group is a multi user chat service, the system could present other group activities, such as coordinating tasks and environmental awareness. The design of the system should be modular to facilitate functionality extension and portability even though the implementation in multiple platforms is not within the scope of this work.

By dynamically creating groups the the burden of creating a group, inviting participants and administrating the group is shifted from the user to the server. This is particularly useful when the participants do not know each other and it is also very useful if there is a predefined group of people with which communication is established, such as informing friends of current events or organizing a sporting event. Studies show it is very common for teens and young adults to update friends via text message(6). If the tasks of setting up the group and selecting the participants were automated, it could only improve the users experience.

1.2 Scenario

This thesis will employ a tourist scenario to provide particular uses cases and give an example of the requirements. The tourist scenario has already provided use cases for other Mobilis (5) applications and has proved to be very useful because it provides flexible and dynamic use cases(7). An example of the flexibility in the use case is one where a tourist agency schedules activities for the participant and participation is voluntary, so the tourist can decide to go or not. Use cases may present themselves dynamically such as when a group of participants is touring the city and they find street performers, it may be more interesting for the participants to watch the street performers than to continue with the scheduled activity. It is also important to note that some activities may provide a better user experience when completed in groups(8) even if the members of the group do not know each other.

[1] http://141.76.40.90/ wiki/index.php/ANDROIDBUDDY

Most use cases in the tourist scenario are of voluntary participation, it is not obligatory to complete all activities; this is notable because it provides a reason for why the final application to be very user friendly. The participation should be of a group of people with a shared goal. If more people participate it should be easier to reach the common goal. There could be many goals accomplished, be it sharing information about the tourist area or getting together to socialize or even to find some security by being with others. These common goals provide a motivation (9) to participate in the group.

To provide a context for the tourist scenario the following points should be considered

- The participants are in a location where they have not been before or do not know very well.
- The usual home resources are not readily available, an example of this would be the lack of a PC.
- There is help and advice available but not always accessible, an example of this would be a tourist information kiosk where they do not speak a common language.

To Summarize the tourist scenario we should consider the participants to be in a new location away from home resources where a mobile device would be very useful, there must be a motivation(10) to collaborate with others because participation is not obligatory and we should also consider that the participants may not known each other, therefore a dynamic way to create the group is advisable.

1.3 Further Chapters

The following chapters will present some works and tools that aid in the development of this thesis. There is a description of the tools that provide a base upon which the implementation is developed. During the study of the material required to develop the work similar or related works were encountered and a brief summary of these works is presented.

The next few chapters provide an outline of the system from various points of view. The requirements section presents the system from the point of view of the customers

of the system by stating what is required or desired from the system at different levels. The design chapter exposes the point of view of the architecture of the system, how it is modularized and combined for best performance and extensibility. The implementation provides a some examples of the implementation from the developers stand point. Finally the conclusion and further works will complete the material provided for any further development.

Chapter 2

Foundations

This chapter presents some of the tools and technologies related to the development of the system. The tools and technologies that provide the application's basis also provides a suggested implementation style or architecture. This is particularly important for mobile devices where the system development should be constrained to the limitation of the device.

The main components upon which this work was developed are Android[1] for the client, XMPP[2] as the communication platform and the Mobilis(5) middleware to access services that have previously been developed.

2.1 Android

Android is a platform comprised of many elements including mobile devices, an operating system and the SDK . Initially, Android was a small start up that made mobile phone applications. In 2005 Google acquired the company and developed the operating system for two years. In 2007 the Open Handset Alliance[3] along with Google presented the free and open source platform for mobile phones, called Android. The Open Handset Alliance is a group of companies dedicated to developing open standards for mobile devices, building better phones for customers and providing joint direction for development. The members of the alliance span many fields, including mobile op-

[1] http://source.android.com
[2] http://xmpp.org/
[3] http://www.openhandsetalliance.com/

erators, software companies, commercialization companies, semiconductor companies and handset manufacturers.

Android offers middleware and key mobile applications that include many useful APIs to help develop applications. Because so much of Android is open it has been called "the first free, open source, and fully customizable mobile platform" [1]. Custom applications can run as well as preinstalled applications. Other advantages of being free and open is that it makes it well suited for academic development purposes and there are many sources for support, such as online documentation, sample open source code and theSDK System Development Kit. The kit contains tools for developing and debugging applications and designing user interfaces. There is also an emulator to test applications on a PC.

There are many devices that are able to run the Android operating system. By the end of 2009 there will be at least 18 phone models using Android[2]. Each mobile phone may present different feature configurations. There are hardware dependent features such as telecommunication support for EDGE, 3G or GSM, networking via Bluetooth or WiFi and location awareness based on GPS, accelerometers or compass. Features can also be implemented via software, such as memory management and process control that is optimized for mobile devices. There are several layers to the software stacksee figure 2.1 [3] on an Android device. At the lowest layer is a Linux kernel with special patches for mobile devices. This layer is the buffer between the hardware and the rest of the software stack. Upon the kernel are the libraries, mostly written in C/C++ that provide support for the Android runtime environment and the application framework. Atop the stack are the applications that can use the resources provided by the application framework or other applications.

2.1.1 Software Stack

Android is based on a Linux 2.6 kernel as an abstraction layer between the hardware and the rest of the software stack. It provides system services such as security, memory management and process management. The services support multiple threading and low level memory management. This allows multiple applications to be run at the same

[1] http://source.android.com/
[2] http://en.wikipedia.org/wiki/Android (operating system)
[3] Source: http://sites.google.com/site/edwardcraighyatt/projects/custom-android-phone

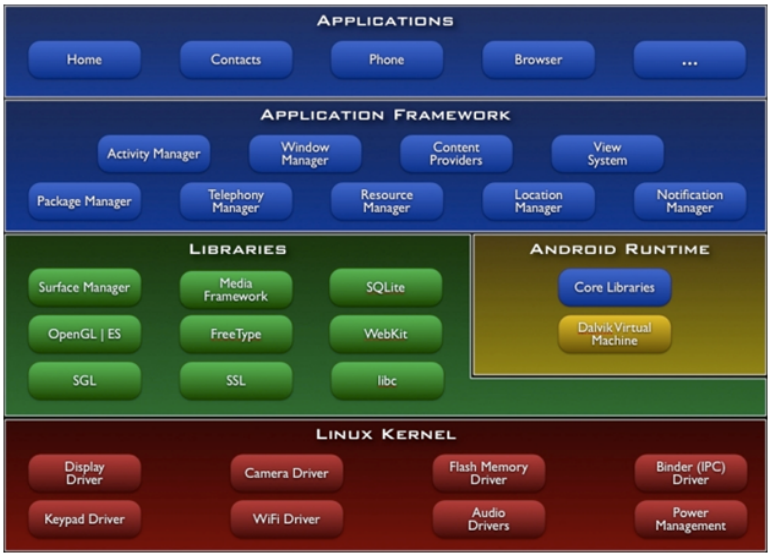

Figure 2.1: Android software stack -

time. The services also manage the network stack and device drivers that are tuned to the hardware configurations of each device. The libraries running on the kernel provide most of the functionality available in the core libraries of the Java programming language [1]. They include the surface manager that handles the display subsystem and composes graphic layer from multiple applications, libraries that support multiple media playback and recording and SQLite, a powerful and lightweight database engine.

On the same level as the libraries is the runtime environment. This environment is composed of libraries and multiple instances of the Dalvik virtual machine. There are multiple instances because each application runs in its own process with its own instance of a virtual machine. There are many advantages in keeping each application in its own process. It improves concurrency and security by making applications independent of each other, it also simplifies memory management and an application crash should not affect other applications running. Because there are limited resources on the phones, the Dalvik virtual machine has been designed so that a device can run multiple VMs efficiently [2] by minimizing the memory footprint required to run the system.

Before the final application layer, the application framework layer is presented. This layer exposes the programs that manage the phone's basic functions. They include an

[1] http://developer.android.com/guide/basics/what-is-android.html
[2] http://developer.android.com/guide/basics/what-is-android.html

activity manager that handles the application's life cycle and provides a common navigation backstack, content providers that allow access to data for multiple applications and tools to build the user interfaces. These programs were written in Java and designed to simplify component reuse. Therefore they are accessible to all applications and can also be replaced or enhanced.

The following describes the Android Application model. This model presents the components of an Android application and how these components access each other. The building blocks for an Android application are activities, services, content providers and broadcast receivers.

2.1.2 Android Applications

Android applications are composed of four basic building blocks: activities, services, content providers and intent broadcasters. These components are bundled together with the resource files the application requires to form the aplication. The final bundled file is compressed to an APK file. Each APK file is considered one application and is used to distribute and load the application onto the phone. Because each component can be called individually the application is started when any component of the aplication needs to be executed, and shuts downs when no longer needed or system resources are required by other applications.

Because each application runs within its own virtual machine, each application is assigned its own Linux user ID so applications are isolated. It is possible for two applications to share the same ID to would allow separate applications to see each other's files and they could possibly run within the same VM to save resources. This is not the only way for applications to call each other. Permissions can be set to share data between processes and applications can call one another using intents. The use of intents and broadcast receivers can provide extended functionality in response to an application call by adding functionality to the call or replacing the functionality altogether.

Activities present the graphical display of to the application. They provide a user interface and react to user inputs or events. Although activities are independent, they can work together and communicate to make one cohesive user interface. The activity windows display user interface controls such as buttons and lists, these components are

known as views. There can be an overlay of views that provides a complete perspective of the user interface. There is typically one activity marked as the first one to display when an application is initiated and from there activities can call other activities. This provides the user a means to navigate through the application.

Services are components that do not have a user interface and are usually run for an extended amount of time. Examples of the services available in the Android framework are the network monitor or a media player. While activities are used to start services, it is also possible for activities to connect to running services through an interface the service exposes and services can also start activities in case a special event occurs. If activities and services are running concurrently, it is possible for the service to spawn a new thread in order to avoid blocking the activity, this is very useful in cases such as the media player where the user requires access to the controls such as play and pause while the music is heard in the background uninterrupted.

Intents and Broacast Receivers are messages for components of an aplication to communicate between different virtual machines and within the same virtual machine. These are received by Broadcast receivers. An example of a broadcast may be low battery warning or an incoming message. Activities and services can be started using intents and broadcast receivers. The intents describe the action that should be performed and can carry additional information, called extras, to provide required information to start an action. Various components can react to a single broadcast, this allows the system to add functionality to the response or replace the activity that responds to a broadcast. For example a broadcast can be received by the notification manager to notify the user of a new event by flashing the backlight and vibrating and also an application can start to respond to the same broadcast. Components can publish intent filters to describe what type of intent each component is able to process to avoid redundant message passing.

Content Providers exists, to simulate a common storage area for all Android applications, . They manage access to persistent data such as databases and images trough a content resolver. The content resolver serves as a broker between the component and the data available by the content provider. Examples of the content providers are the

contacts stored in the phone or the images stored in memory. These can be required by various applications for different purposes and require a standardized way to access them.

Another component that should be mentioned is the manifest file. This file is where the permissions and intent filters are registered and it is also used by the operating system to tell which components exist in the application and additional information of the application.

2.2 XMPP

The Extensible Message and Presence Protocol (XMPP) was developed initially by the Jabber open source community as an XML-based protocol for near real-time instant messaging and presence information. In 2002 it was formalized by the IETF [1] resulting in the publication of the XMPP base protocols. These include the Jabber client and server protocols as well as the presence and IM session establishment and security protocols.

Currently, XMPP has implemented many extensions including multi-party chat, voice and video calls, collaboration tools, content syndication and remote systems monitoring. These extensions are developed through an open standards process by the XMPP community and commercial developers, which produce a wide variety of XMPP-based software. The open standards process is run by the XMPP Standards Foundation that is in charge of documenting and managing the XMPP protocols. There have also been developments by Google for XMPP. Initially the Android application framework presented an XMPP service but Google changed its name because as it developed it was no longer compatible to the XMPP standard.

The XMPP protocols are free, open, public, and clearly documented, multiple implementations of this standard exist in the form of clients, servers, server components, and code libraries. One implementation that runs on Android is called Smack. It is written in Java and implements the core XMPP protocol and also provides an architecture to extend features by handling custom messages.

[1] http://www.ietf.org/

2.2.1 Smack and Openfire

To better understand how XMPP works a small example is provided. For this example a running XMPP server, such as Openfire, is accessible via its URL. Because XMPP is open anyone can setup their own server and connect via Internet to other servers. In this case the server will be called localhost, and an external server will be called jabber.org, to show a connection established between the two servers, see figure 2.2.

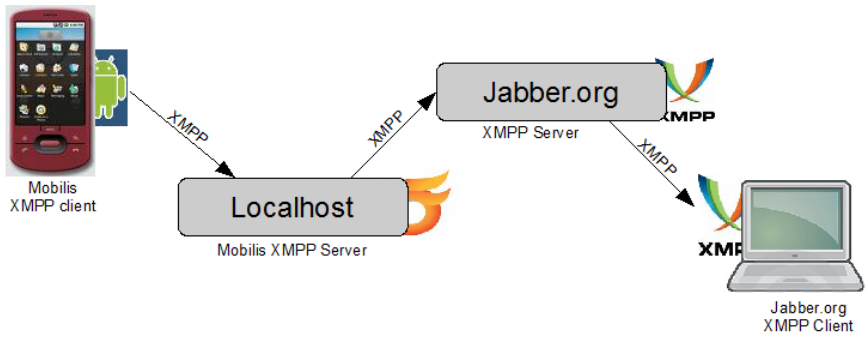

Figure 2.2: Server to server communication connecting clients -

In this example the Client has already setup an account with the name JSmith, his account includes his password and a list of his contacts called a roster. When the connection is established to the server the client can specify his connection configuration. This may include message compression and authentication settings, the client can also specify a resource name to specify where the connection comes from, for example "home" or "mobile", see figure 2.3. Each connection must have a specific resource name to illustrate the user Jsmith@localhost would have the full address jsmith@localhost/mobile.

Once a connection to the XMPP server is established, the client can login and send descrete units of information called stanzas, like messages and configuration. Stanzas are XML based and used in XMPP for all communications.

To illustrate the concepts of the initiation protocol the figure see figure 2.4 is provided based on the smack library, in Smack the stanzas are called IQs.

Below the code using the Smack library to send a message and the resulting stanza is presented:

```
1: Message message = new Message();
2: message.setTo("tom@jabber.com");
```

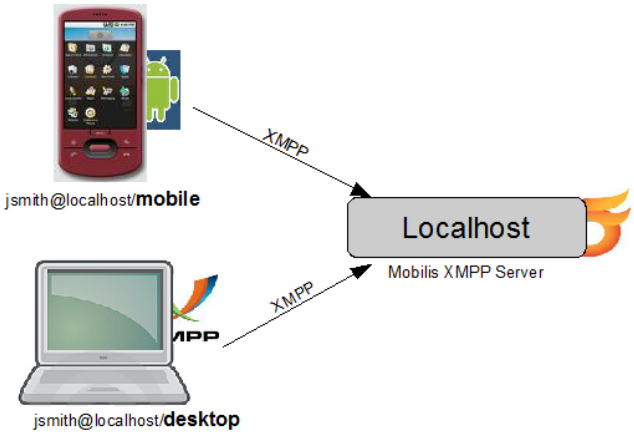

Figure 2.3: Client login in from phone and desktop, with configuration information -

```
3: message.setSubject("Server down");
4: message.setBody("Hello, Hal. Do you read me, Hal?");
5: connection.sendPacket(message);
```

To produce the following stanza in XML:

```
1: <message from-'tom@jabber.com/laptop' to='jsmith@localhost' xml:lang-'en'>
2: <subject>Server down</subject>
3: <body>Hello, Hal. Do you read me, Hal?</body>
4: <message>
```

In Smack messages can also provide special properties additional to the standard content of a message that may be a very powerful tool to provide additional data in a simple format. Currently supported types of adittional data are: integer, long, float, double, boolean, string, and Java-objects.

```
1: Message message = chat.createMessage();
2: // Add a Color object as a property.
3: message.setProperty("favoriteColor", new Color(0, 0, 255));
4: // Add an int as a property.
5: message.setProperty("favoriteNumber", 4);
6: chat.sendMessage(message);
```

Having provided some information on how stanzas are sent, the processing of incoming messages will be presented. In Smack there are two classes that receive messages,

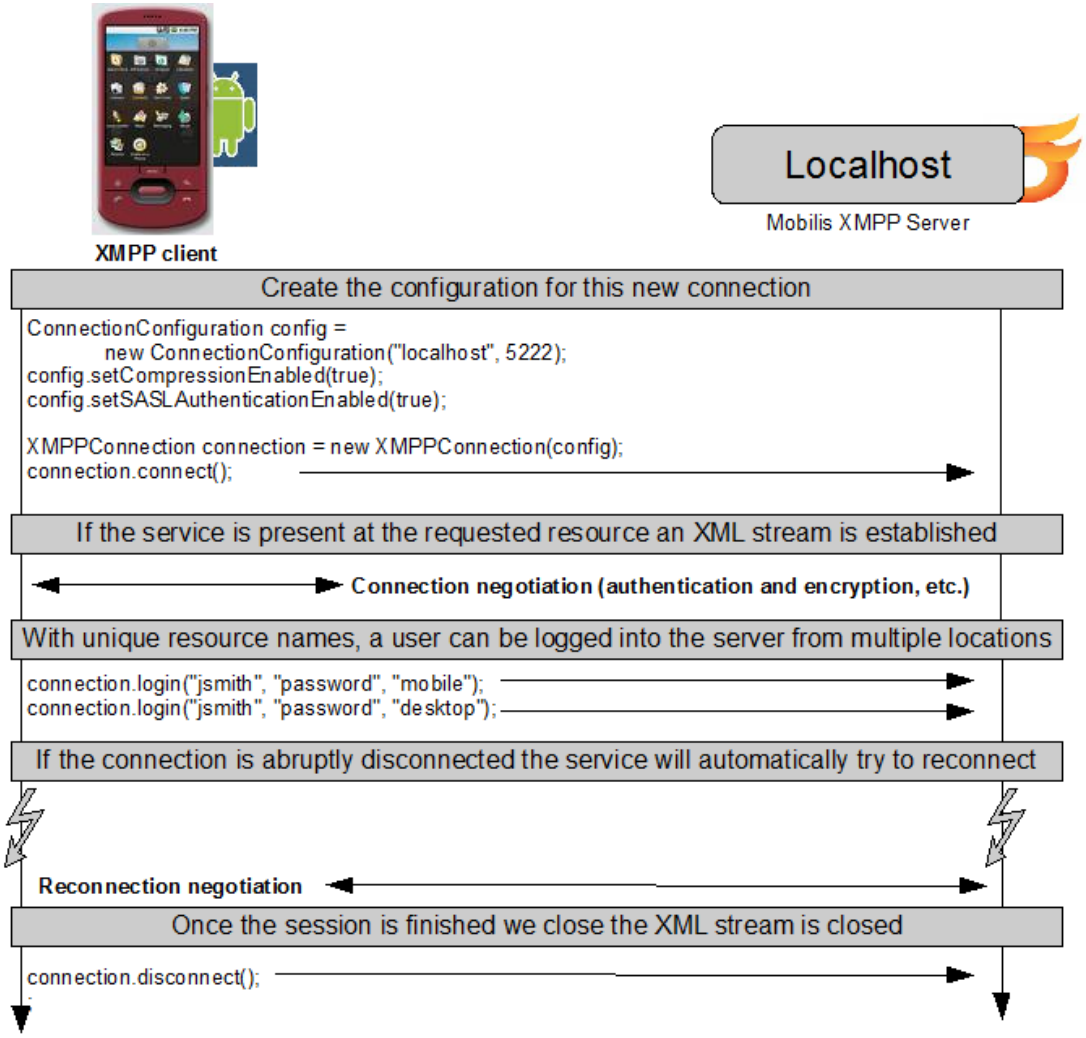

Figure 2.4: Smack connection -

packet collector and packet listener. The packet listener reacts to all stanzas coming in so it is good for event style programming. The packet collector has a result queue of packets so it is useful for waiting for a specific packet to arrive.

These classes can then distribute the packet information to the classes able to process them. This mechanism of receiving packets of a certain type and passing them over to a class that can process them is similar to the mechanism used in Android with intents and broadcast receivers. By evaluating the namespace of the stanza the packet listener can forward the packet to an IQ provider, used to parse the message into Java objects that can then be processed.

Some of the extensions that are already included in Smack are in table 2.1.

Table 2.1: Smack extensions

Name	Description
Private Data	Manages private data.
XHTML Messages	Allows send and receiving formatted messages using XHTML.
Message Events	Requests and responds to message events.
Data Forms	Allows to gather data using Forms.
Multi User Chat	Allows configuration of, participation in, and administration of individual text-based conference rooms.
Roster Item Exchange	Allows roster data to be shared between users.
Time Exchange	Allows local time information to be shared between users.
Group Chat Invitations	Send invitations to other users to join a group chat room.
Service Discovery	Allows to discover services in XMPP entities.
File Transfer	Transfer files between two users over XMPP.

Smack provides an architecture to develop extensions and process them in the same manner as base protocols by generating custom IQs, providers and packet listeners.

2.3 Mobilis

Mobilis (5) is a mobile collaboration middleware that provides various services, such as geo-location and context management services. Developed by the Computer Networks chair at the Informatics Faculty, TU Dresden, it has been used for various applications that include a car sharing application and a location aware trading application.

One application that uses the Mobilis middleware is AndroidBuddy. It brings social networks to an Android enabled device, not only for contact information but also location awareness.

AndroidBuddy follows the client server model, it uses XMPP communication to access the Mobilis server and the client is designed to run on Android smartphones.

The Mobilis server is accessed by all the users, it handles client registration, database storage, group chat administration and a publish subscribe mechanism used to update location information.

Mobilis is accesed through an XMPP communications server that provides basic messaging and presence features. It is also able to handle some extensions to the protocol or convey the extended protocol stanzas to be processed by Mobilis. For the AndroidBuddy system the XMPP server used is Openfire [1].

The client runs on the Android smartphone and can deal with multiple accounts on Facebook[2] and on Mobilis. It has a context service that handles location updates and a social networking service that handles updates to Facebook accounts. The system also accesses information from the smartphone, specifically the contacts. These contacts are integrated with the contacts on the Facebook and Mobilis accounts to manage one global list of contacts. It also provides various other services such as a multi-user chat and notifications.

2.3.1 AndroidBuddy Composition

A review of each of the relevant components on the server, the client and the communications protocol is presented.

On the Mobilis server there are three main components: the XMPP communications server, the AndroidBuddy Agent and the Session coordinator. Both the AndroidBuddy Agent and the Session coordinator are configured and managed by the Mobilis Manager.

The XMPP communications server used for the AndroidBuddy system is Openfire. It was developed by Ignite Realtime [3], the developers of Smack[4]. The server stores account information for each user and provides the standard features for the XMPP

[1] http://www.igniterealtime.org/projects/openfire/
[2] www.facebook.com
[3] http://www.igniterealtime.org/
[4] http://www.igniterealtime.org/projects/smack/index.jsp

protocol, such as presence information and IM. All communications between the client and server are done trough this communication server.

The AndroidBuddy Agent is responsible for processing the extended functionality of the XMPP protocol, such as the following:

- The buddy finder service
- Social network integration service
- Context service

The buddy finder service uses the location information of each user to send proximity alerts if users are nearby one of another. It sends the location information using a stanza that was specially developed for the AndroidBuddy application.

The social network integration service provides a proxy for the user to update account information in a social networking site from the AndoridBuddy application. The system is enabled to process Facebook account information using a specialized stanza. This stanza contains the name of the network that is being updated, the name of the service and the updated value. It can also be used to update Mobilis acount information.

The context service manages user context information such as location updates and notifies other services of the updates.

There are four XMPP packet extensions developed for AndroidBuddy, see figure 2.5 (11). In order for Mobilis to process the specialized stanzas, the services that converted the stanzas to objects, called IQ providers, are a registered along with the packet listener to forward the messages to the according service.

The Session Coordinator see figure 2.6 (11). is responsible for initializing and managing session agents. Therefore it is not responsible for administrating groups directly. The session agents do however administrate the groups that are created. Some of the responsibilities of the session agents are to manage group membership and the affiliation rights of each participant in the group and handle the administration of the multi-user chat instance in the XMPP server.

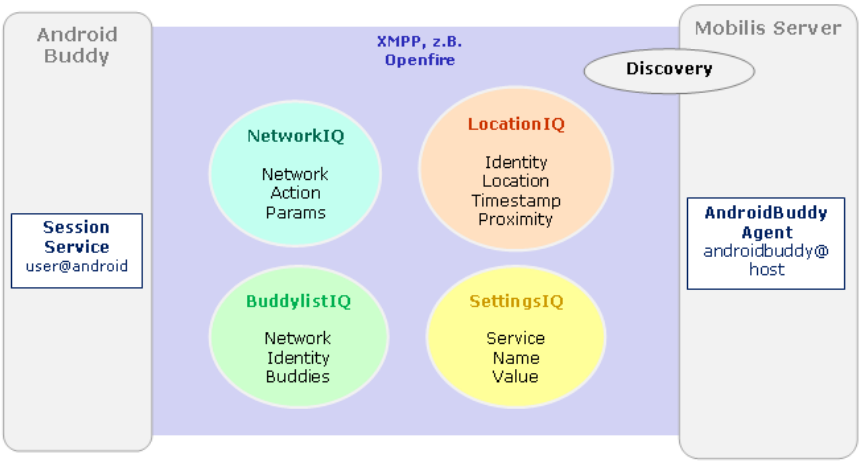

Figure 2.5: Extensions Developeded for AndroidBuddy -

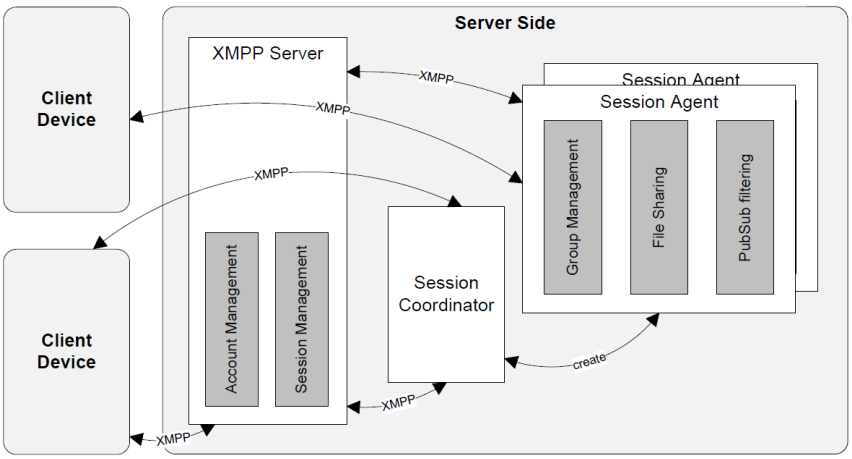

Figure 2.6: AndroidBuddy composition -

2.3.2 AndroidBuddy Client

Because the AndroidBuddy Client is an Android application, we can divide the system into four basic types of components: user interface activities, services running in the background, information repositories and additional tools.

User Interfaces The map screen, see figure 2.7 (11) is the initial user interface, it shows the user's current location and the location of nearby contacts. The map screen may also show activities that can be started with nearby contacts. The same contact information is available in list form on the Buddy list tab. This simplifies lookup for the user. Another user interface is the chat screen. This screen is automatically created also when the user receives an invitation to chat by the Mobilis server. There are several account and network connection configuration screens. Many of the screens are accessible trough the "'menu"' button on the device. This is a context menu only available in the main map screen.

Services The client activities are supported by several services running on the client. The central service is called the session service. It initiates and coordinates all other services, including the Smack library that establishes the communication with the XMPP server. Although the Android architecture indicates the use of intents to call components, in order to minimize code complexity a direct referencing of Java objects was used for method calls.

The session service is where services capable of processing XMPP extension stanzas are registered. In the following sections we will examine some of these services. Most of these services have a counterpart in the Mobilis server, see figure 2.8 [1].

The group management service holds the memberships to the groups the user is affiliated to, so it is the counter part to the Session agent on the server. It is also used to query available MUC groups and handle invitations to participate in a MUC group.

The context management service handles presence and location notifications from other users and collects them in a list to trigger updates on the main map activity.

The places management services implements an internal database to store rich location based data that includes a proximity alert and a timestamp. Because the data

[1] http://141.76.40.90/ wiki/index.php/ANDROIDBUDDY

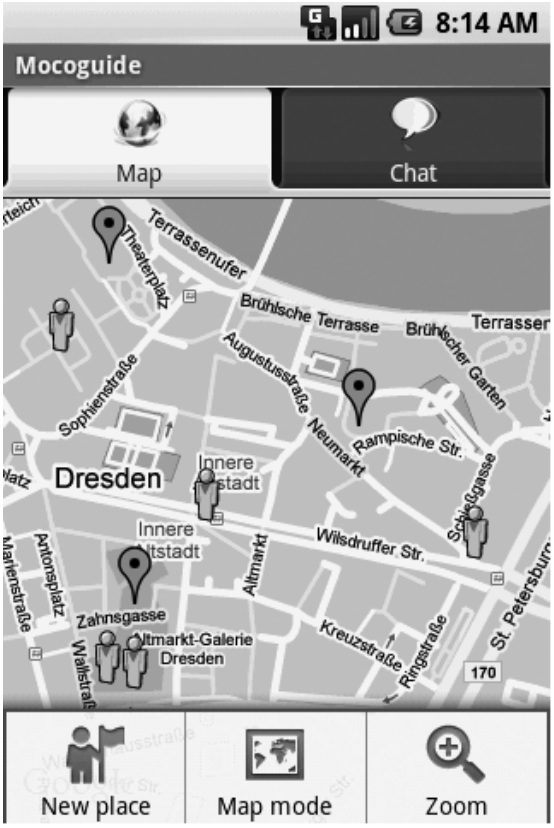

Figure 2.7: AndroidBuddy protype -

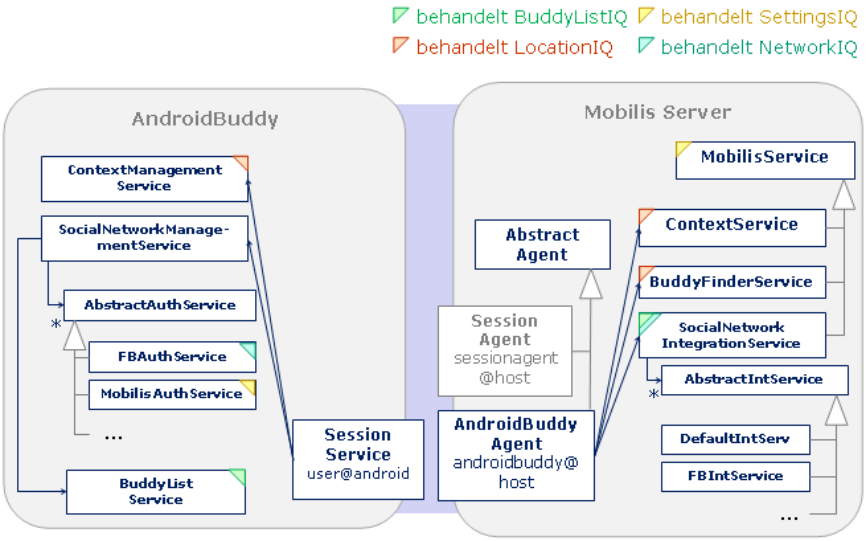

Figure 2.8: AndroidBuddy Services -

is not frequently updated, this service uses the publish subscribe mechanism to acquire the information sporadically from the server.

The social networking management service is used for logging in and administrating connections and accounts to the various networks, namely Facebook and Mobilis.

The buddy list service integrates the smartphone's contact list and the mobilis contact list and publishes the result to the Mobilis server. This allows an integrated management of the contacts with additional contacts from Facebook.

The map enrichment Service adds elements to the map screen, it employs a lightweight data structure to differentiate between people and places and also posts location coordinates on the main map activity.

2.3.3 Information Repositories and Additional Tools

The AndroidBuddy system utilizes a content provider from the Android operating system to access the contact information stored on the smartphone, and an object oriented database implementation stores account credentials and other information local to the application. The database is accessible to all components within the application to store and access pertinent information.

One additional tool provided by the AndroidBuddy project is the KLMSimulator. This is a small Java application that can communicate to the XMPP server as though it was a client. This allows easier testing of sample code evaluation and it is also useful for testing the interaction of multiple client instances.

2.4 Conclusion

The applications on which this work is built upon are well supported for development. The Android platform and the XMPP protocol both have good documentation and example code. The AndroidBuddy system does not have as much online support, but since it is built in house there are some developers with a good working knowledge of the system that are willing to provide clarification or commentary as to the function of the system.

Chapter 3

Related Works

The material studied to prepare this work spans many fields. This is partially due to the scarcity of materials dealing with dynamic group initiation(12) with the further refinements of a mobile environment and the use of interests.

The documentation read could be categorized into three major fields. The first field comprises systems that enhance the contact list information already available to the users by providing an analysis of how contacts are selected to form a group. The next field is of systems that provide suggestions to build a community additional to the user's contacts. The final field is that of papers dealing with sharing interests in a group(13) or coordinating activities as a group.

3.1 Contact List Enrichment

This section presents works that intend to aid in the selection of contacts to participate in a group. Some of these systems enrich the information available for the user to select a contact, others provide a suggestion based on social network analysis.

3.1.1 Cluestr

Cluestr(12) presents a community-aware mechanism that allows the efficient selection of contacts to select them to form as a group. This is achieved by providing a recommendation from the contact list to add to the group communication. The mechanism by which the recommendations are selected is the following:

1. The user categorizes each contact such as family of classmate.

2. The category information is used to build a social network graph that includes contacts of all the participants in the system known as the community.

3. Clusters of participants are deduced from the social network, these clusters represent groups such as colleagues or a group of friends.

4. Finally when the user initiates a group and selects a contact the application provides a recommendation based on the selected contacts, with each additional contacts the application refines the clustering and provides the refined recommendations of contacts to add to the group.

To illustrate a simple use case is provided where a user has a contact list of A, B, C, D and E. Each contact is a member of one or more clusters, represented here by a number 1,2,3,4 and 5.

- A = 1,2,3
- B = 1,2
- C = 4
- D = 2,3
- E = 3

in this simple use case the initiator first selects A to participate, based on the contacts' clusters the engine recommends B, D and E. The contact C is not recommended because it does not match any cluster of A. Then the initator invites E to participate, based on this additional information the engine refines the participants to one single cluster and recommends D. The contact B is no longer recommended because it is not part of the refined cluster (3).

This mechanism does not only help to reduce the time to setup a group. It also helps the user to remember what contact to invite to participate.

To avoid tasking the user with maintaining the category information for all contacts a social network analysis generates a relationship types between each one of the users contacts. The social network information is obtained from Facebook[1], a social networking service where users can establish relationships with other users.

[1] www.Facebook.com

The figure 3.1 presents an ego-alter graph where the ego at the center represents the user and the alter nodes represent the contacts. The links represent the relationships that are established between the participants. A small cluster of two nodes(a_9, a_{10}) is evident in the lower part of this graph. Other overlapping clusters are also evident, these clusters represent relations such as family or sports team. The collection of all nodes and relations is called the community.

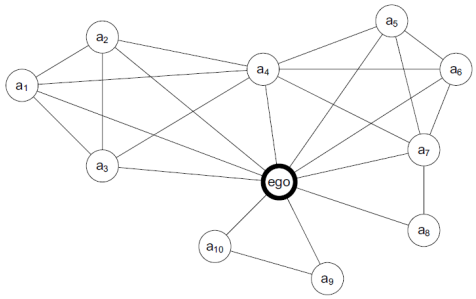

Figure 3.1: Visualization of sample ego-graph -

Mathematical analysis is used to derive the clusters of each contact this information is used by the engine to rank each contact for recommendation. The ranking is based on the clusters of the selected contacts, the more matching clusters a contact has the higher the ranking. In figure 3.2 the selected contacts are presented on the left and the recommendations on the right. In the selected contacts cluster C_1 appears 3 times, cluster C_2 appears 3 times and cluster C_3 appears once. This information is used to score the rankings, for example the highest score is 6 because the contact is a member of cluster C_1 that scores 3 and the cluster C_2 that scores 3 in total the score is 6. The lowest ranking is 1 because the contact is a member of cluster C_3, this contact is also a member of cluster C_4 but that cluster is not in the selected contacts therefore it does not add points to the score.

The system was evaluated based on several criteria including how fast a group setup for three scenarios:

- Scenario 1: The group consists of all members of one community.
- Scenario 2: Only a part of a community's members should be invited.
- Scenario 3: A random sample of contacts, regardless of their community affiliation should be selected.

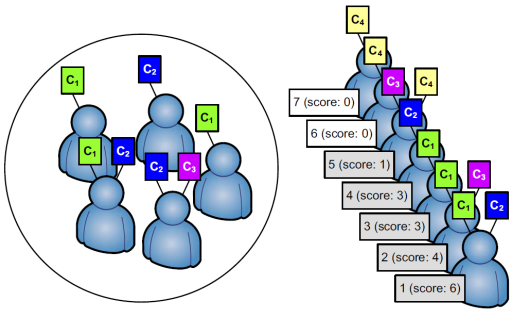

Figure 3.2: Visualization of sample ranking -

To provide a frame of reference each scenario was tested using three methods of presenting the contacts, in the traditional alphabetic order, contacts in a cluster without ranking and the recommendations system top 5 ranked contacts.

The results, provided in the figure 3.3, show that the recommendation systems fairs better in both scenario 1 and 2 in the case of random selection recommendation performance decays. In all three scenarios if the user did not find the intended contact the selection method was reverted to alphabetic, in the third scenario. Overall the system performed well for what would be most real life scenarios where a user would select members of a particular group.

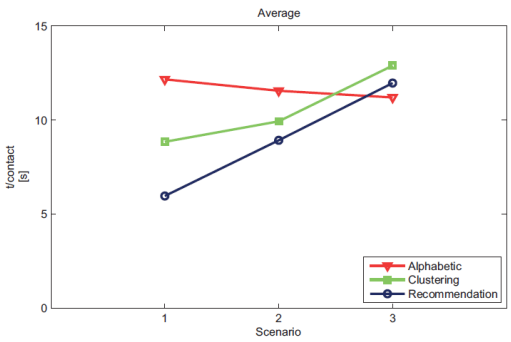

Figure 3.3: Average time required to initiate a group for in Cluestr -

3.1.2 Friendlee

Friendlee(14)provides contact recommendations based on context information. The system analyzes the user's call and messaging activity to rank the contact list, providing a higher ranking for those that are contacted more often or for longer times. The

system presents context information including location, weather and phone status such as on hold or busy for each contact to providing an insight into the users environment or availability. One of the main concerns for the users of this system is lack of privacy in sharing context information with others. To address this issue the system proposes a category-based privacy model providing more and finer grained information for certain categories of contacts, such as family or collegues and more general or less information for others.

3.1.3 ContextContacts

ContextContacts (15)provides context information for the user's contacts, this includes phone usage and time spent at a location. Each element of information that composes the contacts' context is called a "cue". The system does not provide a recommendation. Instead the user is expected to use the "cues" to deduce the contact's situation or availability and thus select participants. A sample of the cues provided to the user and the relevance in making social judgments are presented in the figures 3.4 and how these cues are presented to the user can be see in figure 3.5 (15).

Cue	Significance for social judgments	Representation in ContextContacts
Location	Hints of current place, which hints of activity and task	City (e.g., Helsinki or HKI) and district (Kumpula) / user given description (text)
Time spent in the current location	Hints the temporal extent and phase of the current activity	Hours and minutes spent in the location (text)
User-selected alarm profile	Conveys user-decided interruptability and desired communication channel	Icons for audio and tactile alarm on/off
Phone manipulated recently	Hints proximity and responsiveness to communication attempts	A hand icon turning from gray to red if the phone has or is being used
# of unknown BT phones nearby	Hints of the type of current social activity	If more than 0, a green person icon appears; the number is expressed textually next to the icon
# of contact lists' BT phones nearby	Hints of the presence of friends, and of the type of current social activity	If more than 0, a yellow person icon appears; the number is expressed textually next to the icon

Figure 3.4: Situation cues and relevance in making social judgments -

These approaches have little or no support for extending the social network, but provide an insight(16) into how participants in a group are selected.

3.2 New Contact Recommendation

Systems that provide suggestions to add to the contact list are presented next. Many of these systems are centralized and have access to not only a global list of participants

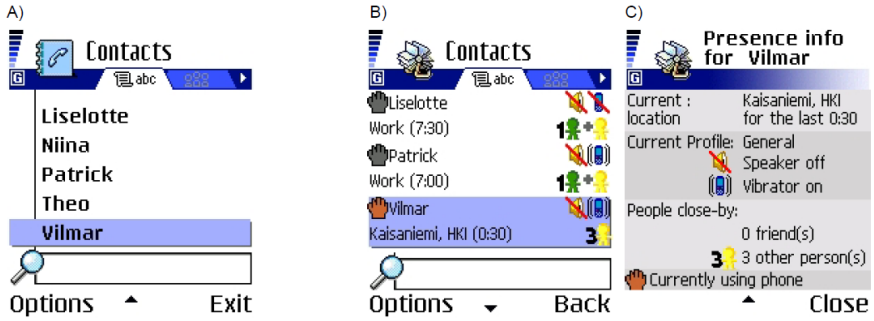

Figure 3.5: A)standard contact list, B)ContextContacts, C)ContextContacts, detailed information -

but also related data that is used to propose the additional contacts. One in particular presents an Ad-hoc approach to present potential contacts.

3.2.1 Do you know

(17) Do you know is a desktop widget developed to recommend contacts within a company's social network. The widget is based on SONAR(18) a system for collecting and aggregating social network information across an organization. The relationships are deduced from sources that include organizational chart, co-authorship of various documents like papers, patents or wiki pages and mutual contacts in various social networks. The system is mainly directed for large companies where employees may not have the opportunities to establish contact with coworkers of interest and access to reference information is available.

SONAR assigns a score to each relationship based on the activity within the different sources such as document co-authorship or contacts in common within a social network. The scores are aggregated and used to produce a weighted recommendation list. This list is then used to create the final ranked list of recommendations provided to the user. Figure 3.6 (17) presents the widget screen with recommendation criteria.

The resulting analysis of the performance of the system was very positive, in usage and value of the information provided. Although the system recommends the top 100 participants most users only reviewed the top ten recommendations and rarely reviewed more than twenty recommendation. Most users found the evidence for recommendation, such as co-authorship, very useful and invited other participants to join the system.

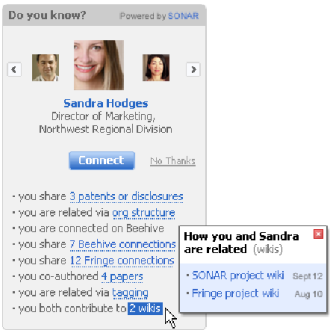

Figure 3.6: Do you know widgit screen -

3.2.2 Nokia Sensor

Nokia Sensor: From Research to Product (19) is envisioned to moves the personal web page to the smartphone. The application uses bluetooth communication to share personal digital identity with others including a personal Web page like presentation screen.

"The Nokia Sensor application allows users to express themselves and discover new things about others nearby in a novel and simple way. This means that the value of using mobile phones and devices is expanded to connecting with those who share the same space at the same time, acquainted or unacquainted," says Hannu O. Nieminen, Vice President of User Experience Unit at Nokia.

This application seeks to enhance existing social behaviors, practices, and experiences between people by being able to identify a potentially interesting person, discover invisible connections, and find out more about a person's interests, history and preferences with the aid of data exchange between Smartphones see figure 3.7 (19).

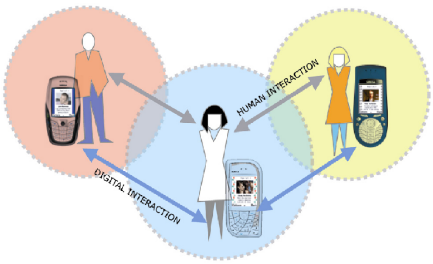

Figure 3.7: Sensor augmented interaction -

Social behavior studies provided the basis for several design principals introduced

for the development of this system, these are the following:

- Supporting the acquainted as well as the unacquainted to establish a social connection.
- Identity expression and social curiosity; Expressing ones identity is done trough a variety of ways, ones clothing, gestures and personal homepages provide an insight into one's self and the curiosity to have an insight into others are fundamental social acts. The application proposes augmenting the principle of "see and be seen" by providing a similar mechanism in the digital realm.
- Focused vs. unfocused encounters; Focused encounters refer to those where the participants interact such as a face to face meeting, in contrast unfocused encounters are where people do not interact but acknowledge each others presence. One of the targets of this application is to provide legitimate reasons for people to move from unfocused to focused interaction by revealing interesting and hidden connections.
- Social memorabilia; Is the trace left behind by an encounter, like exchanging business cards or scribbling down a phone number. These provide a way to re-encounter or a reminder of a social contact. The system should provide digital versions of this.
- Identification and traceability; These could be represented by a personal telephone number or personal image to identify users during repeat encounters.
- Optimal use of mobile data; Smartphones offer a wide array of mobile data from audio and image files to personal calendar and contact information. This data should be used to provide an identity expression and provide data to find correlations between users.
- Pervasiveness; In this system it means to rely on other users in order to provide the core experiences.
- General tool; The application should provide tools used for socializing in professional and personal lives.

The results were two development systems that provided the core of the final Sensor application, DigiDress and Scent.

DigiDress a digital personal 'expression page' including text and imagery, that could be scanned by other DigiDress users in the proximity. This asymmetrical 'peek view' could also allow the 'onlooker' to discover common interests with or connections to the page owner thus encouraging the unfocused encounter to a focused one. Users could save any found identity expressions on the phone for future reference.

Scent focused on providing various levels of social interaction possibilities by taking the users' mobile data, such as contacts, and comparing it with others, the users are then presented the matching entries. The users could then keep a record of this encounter as social memorabilia. Both Scent and DigiDress also provided a guestbook. Contents of the guestbook were publicly available to any other (proximate) application user.

To find and browse proximate identity expressions a process referred to as scanning is used. Users were not allowed to scan for others' identity expressions before they created at least a minimal version of their own although scanning for and finding others' identity expression does not require a request/consent process. Other processes such as sharing mobile data do require consent.

Sensor combined the two approaches and provided additional features. The figure 3.8 (19) presents the major features of the Sensor application. There a presentation of the introductory page inspired by DigiDress, some of the additional feature were media sharing and a popularity feature.

The system was released as free software by Nokia with some features dropped due to privacy and technical issues such as personal data comparison and phone to phone "viral" distribution. Overall the development system provided good experience as a learning tool for the developers.

3.2.3 Twitviz

(20) Twitviz is an interesting approach to exploring social networks based on interest, it is a desktop application. One of the problems the system addresses is the lack of support for initial group formation in social networking services, in this case Twitter. Twitviz helps the user on this initial group creation by allowing the user to explore new participants that may share the same interests.

Exploring new participants is done trough a graphical tree display where the user is the central node. From there a branch is established that represents an interest and

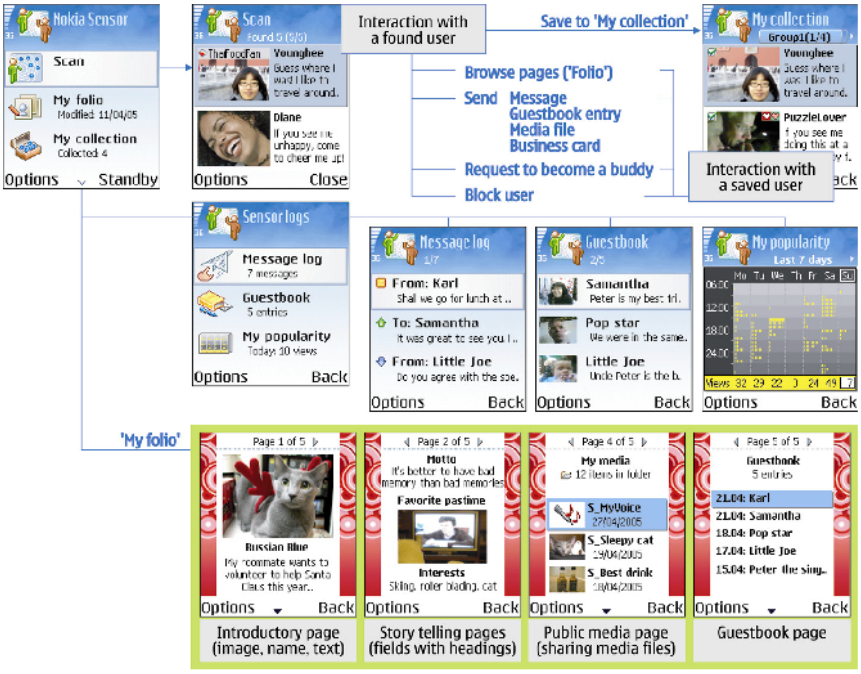

Figure 3.8: Sensor screen map -

the leafs that expand from the branches provide contact information for participants using the term of interest in tweets . In this way the system provides a collection of users who may share the same interests. There is also an egocentric graphical display of participants reciprocating within the social network, see figure 3.9 these are the participants that follow the users tweets and vice versa. Contact information is also provided for the reciprocating participants. By combining these two graphs the user may visually discover new potential contacts by exploring a hierarchical structure of contacts. There are contacts of contacts, where the participants use the term of interest. This provides some level of security in selecting new contacts.

Figure 3.9: Twitviz network view -

One of the drawbacks of the system is that general terms would provide a subset of participants that do not share the same interest. Placing the term within a context was not possible in the application, so for example if the user could be interested in the fruit apple and get results for apple computers.

3.2.4 PeerHood

(21) PeerHood is an implementation of a Peer-to-Peer neighborhood and communication middleware run on a Linux or Symbian OS, developed in the Communications software laboratory of Lappeenranta University of Technology in cooperation with Nokia. The PeerHood middleware was extended by a social networking application(22), developed by Bishal Raj Karki. The approach used to join groups is based on dynamic group discovery in accordance to common user interests. Because PeerHood is peer to peer, the devices must be within range to create a network with other users and the logic to create a group must be minimal in order to accommodate the smartphone's capability. The implementation had no visual interface and the main focus of the work was being able to add the functionality to the middleware in contrast to how well the implementation worked. One of the interesting conclusions of this work is that it recognized the need to find the semantic network of terms in order to register the same interest. For example the term "football" and "soccer" could be considered within the same semantic network and so as the same interest.

3.3 Groups and Interests

Some papers were reviewed to evaluate activities for a group such as a groundswell (23) that is defined as "a spontaneous movement of people using online tools to connect, take charge of their own experience, and get what they need–information, support, ideas, products, and bargaining power–from each other.". The term groundswell is a good representation of what a virtual group activity could be. Other group activities such as a Textmob (24) that presents the used or text messeges to form protest groups dynamically. Other papers were reviewed to explore how interests are used in a group, for example how interest are derived from visited web pages (25) or how they are used in tagging communities (26).

3.4 Conclusion

During the search for reference materials and similar works, there was no system found in direct competition for dynamic group creation. This is not to say this work is the first to develop dynamic group creation based on interests, but the variety of platforms and means by which a group is selected differs greatly between each implementation. There are some proprietary dynamic group creation systems implemented for social networking sites, but because of their closed nature an in depth analysis is not accessible. One item that was consistent between many of the works is that within the development process the conceptual part had more of an impact than the technical part. This is to say there was more effort in how the system would perform than the actual performance. It is also very helpful to view similar problems from different points of view to have a better understanding and contrast as to what is being developed, for example systems that ranked the contacts to form groups or papers that describe the way a group functions.

Chapter 4

Requirements

This chapter will present the different requirements expected of each level in the developed system. These requirements will include items that may not be necessary to complete the core functionality of the system but would enhance the user experience and could be beneficial for potential further development. The different levels of the system will be divided in the following sections.

- related to AndroidBuddy
- related to dynamic group generation based on interests
- related to a good user experience
- overall requirements

The requirements will be presented along with a specific use case, the tourist scenario, to provide a motivation or overview of why the requirement is present. At the end of each subsection a table containing the specific items and additional data will be presented. The additional data will indicate the relevance as an indicator of how important it is to reach this requirement. Not all requirements present the same importance to be implemented. A high relevance indicates it is indispensable for that requirement to be met for the system to function properly, a medium relevance indicates the presence of the defined feature would improve the system and should be implemented, and a low relevance indicates the requirement can be present but is not required for the function of the system. Because the system being developed is based on an application that has various features, there is an implemented column that indicates if the requirement

has been meet by the previous developers or if it has been partially completed, having presented some work but not the complete feature required by the system.

4.1 AndroidBuddy

The requirements stated here are to reiterate and itemize those stated in the task description. The system is expected to expand the features available on the AndroidBuddy system. This fits into the tourist scenario in the sense that there are various features available to the users by AndroidBuddy as mentioned beforehand.

Because the system has been built to handle protocol extensions it is expected to develop extensions to handle the interest information required to generate the groups, the continued use of XMPP as the communications protocol and the Mobilis services such as client accounts [AR-1.0], presence and messaging and multi-user chat. This also implies that the extension to the system should keep a familiar nomenclature [AR-1.1] and architecture of the system as related to AndroidBuddy, to facilitate further development.

Table 4.1: Basic Requirements

ID	Description	Relevance	Implemented
AR-1.0	user account information	High	Partial
AR-1.1	follow similar nomenclature and architecture of system	High	

4.2 Dynamic Group Generation Based on Interests

This section will define the requirements of the user for the additional features to be presented in the system. Because the groups are to be formed dynamically based on interest [DGR-1.0]it is required that the user registers and unregisters interests[DGR-2.0] in the system. As with any manual data entry it is expected to have some issues such as bad spelling or the use of inflections in the interest term, this could be corrected by a spell checker[DGR-2.1] and system to find the root of the term being used, and because there is no standard vocabulary for interest a system to provide similar terms is suggested. The similar term [DGR-2.2]system should be able to find a relation between

two terms that are not necessarily synonyms but are within the same grouping of words. An example could be the terms party and dance, as they are not synonyms but they are somehow related. The interests that are provided to the system may be accompanied by some additional data to further define the interest[DGR-3.0], such as private or only to be shared with known contacts[DGR-3.1]. This would provide an added level of security, especially when we are proposing to generate groups with participants unknow to the user.

To make the system more intuitive for the users, the user could request some recommendations(27) from the system[DGR-4.0]. The recommendations could follow several criteria based on the interests collected from all the users such interests that are most popular[DGR-4.1], the interest that are active(28) in the geographical region[DGR-4.2], interest that the users' contacts have and the users himself does not[DGR-4.3]. Other recommendation criteria could help in standardizing the interest vocabulary by proposing a term that is similar[DGR-4.4] to the interest provided and is also already found in the list of interests from all participants. This recommendation type should be extensible[DGR-5.0] to accommodate additional context information or different criteria(29), for example the criteria of some related works (12)where groups were generated based on relationship type[DGR-6.0]. This would require a relationship type tag be added to all contacts.

The activity that is started[DGR-7.0] as a group should be group chat[DGR-7.1] (30)but should not be limited to this, so extensions to the triggered activities should be possible so the user is invited[DGR-7.1.1] to participate in other types of activities. Some example of additional group activities could be participating in a poll[DGR-7.2] (31) or survey[DGR-7.3]. This would require a standardized query[DGR-7.2.1] and results[DGR-7.2.2] compilation so as not to confuse the participants in a tourist case this may be a poll to see if the large group transport is required or if there are enough participants to warrant a price reduction at a certain event. Another possible group activity could be a shared task (32) list[DGR-7.4] in a tourist case scenario. This could mean that a group has participants provide particular food items to have a communal meal. This would also require a near real-time update to the list.

Because the participants potentially have various activities [DGR-7.5]that can prompt for participation, settings to autorun [DGR-8.0] activities should be provided to min-

Table 4.2: Group Creation Requirements

ID	Description	Relevance	Implemented
DGR-1.0	Generate groups dynamically based on interests	High	
DGR-2.0	Users register interests	High	
DGR-2.1	Term correction (spelling, synonyms)	Medium	
DGR-2.2	Similar terms alredy registered	Medium	
DGR-3.0	Record interest metadata	Low	
DGR-3.1	Define sharing level (with buddies, with all)	Low	
DGR-4.0	Request/display server suggestions	High	
DGR-4.1	Server suggests popular interests	Medium	
DGR-4.2	Server suggests interests based on location	Medium	partial
DGR-4.3	Server suggests Buddies' interests	Medium	
DGR-4.4	Server suggests Similar interests	Medium	
DGR-5.0	Server suggests are extensible	Medium	
DGR-6.0	Server initiates a group based on relationships	Low	

imize the effort of the user to participate in a group. These settings could also be proposed by the Mobilis service simplifying selection for a standard configuration.

Other miscellaneous requirements would be to notify users when there are not enough participants to maintain a group and destroy that group[DGR-9.0]. It may also be suggested by the system the interests that are not in use for some time and if they should be eliminated from the interest list[DGR-9.1]. The user may also decide not to input the interest manually instead to select from a list of all interests[DGR-10.0] or only those with an open group[DGR-10.1].

4.3 Good Users Experience

To provide an overall pleasant user experience[UIR-1.0] it should be noted that a seemingly fast[UIR-1.1] and stable network connection is available, in case there is a weak connection caching and messages resend would mediate the problem. Simple and clear user interfaces [UIR-1.2] make the system more intuitive for the user, it should also be simple for the user to find the interfaces that are required. This could be archived by

Table 4.3: Group Activity Requirements

ID	Description	Relevance	Implemented
DGR-7.0	Server provides a group activity	High	
DGR-7.1	Main group activity is multi user chat	High	partial
DGR-7.1.1	Use invitations to request participation	Medium	partial
DGR-7.2	Group activity is polling	Low	
DGR-7.2.1	present the users a standard query format	Low	
DGR-7.2.2	present the users query results update	Low	
DGR-7.3	Group activity is a survey	Low	
DGR-7.4	Group activity is a task list	Low	
DGR-7.5	Group activity is extensible	High	
DGR-8.0	Configure activities to start automatically	Low	
DGR-9.0	Destroy unused groups	High	
DGR-9.1	Detect unused interests	Medium	
DGR-9.2	Eliminate unused interest	Medium	
DGR-10.0	User can select interest from list	Medium	
DGR-10.1	Select interest from active groups	Medium	

auto starting or inviting to start an activity, having a central place to access related activities or a map of options would also assist the user.

The user should feel the application is friendly[UIR-1.3] and not invasive, even though it may prompt activities continually. Some kind of motivation [UIR-1.3.1] may be in order to promotes participation. The user may also find it helpful to access the system as an anonymous guest[UIR-1.4]. The system may also seem more accessible if the user can switch back and forth between different activities[UIR-1.3.2].

4.4 Overall Requirements

The users may not be aware of some overall requirements, although they may be demanded at some time, for example communications security[OR-1.0] over the network and within the application(33) or service adaptation to resources available to the device, including the network strength. The service adaptation[OR-2.0] could mean disallowing

Table 4.4: User Experience Requirements

ID	Description	Relevance	Implemented
UIR-1.0	good user Experience	High	
UIR-1.1	Fast, no lagging	High	
UIR-1.2	Intuitive navigation of user itnerface	High	
UIR-1.3	Addresses the human factor	High	
UIR-1.3.1	Participation motivation	Low	
UIR-1.3.2	Screen switching	Low	
UIR-1.4	Run as guest	Low	

or lowering the frequency with which some messages are sent.

Table 4.5: Overall Requirements

ID	Description	Relevance	Implemented
OR-1.0	Secure system communication	Low	partial
OR-2.0	Service adaptation to device resources	Low	

4.5 Conclusion

The requirements presented here provide the basis for the design of the system, they provide what will be expected of the system not only by the user but by what a potential future users may require. And it also present what requirements have been partially or fully meet by previous developers.

Chapter 5

Conceptual Design

This chapter presents a detailed view of the components of the system and how they are put together. Many factors are taken into consideration to develop the general architecture from the specifications that are derived from the requirements. To develop the design the core requirements provided the general structure and the additional requirements are adapted to function within this general framework.

5.1 General Architecture

The overall architecture of the system is dictated by the Mobilis architecture, this is a client server model, where the additional features are added to the session service of the client and the server. Therefore all the extended features either add data structures to be communicated by the AndroidBuddy services or the services are registered in the session service to handle custom messages developed to accomplish the requirements.

The sequence diagram 5.1 is presented to provide an overall illustration of the services and datastructures of the system. It represents on the client side a user requesting a suggestion, then starting the interes service with the suggestion as the term of interest and participating in the group activity based on the interest. On the server side it represents a request for a suggestion that is provided by the suggestion service, a request for an interest that triggers a group creation and the group activity that is initiated.

In the further sections of this chapter we will explore in more detail the services and messege types that support the functionality of the client and server for dinamic

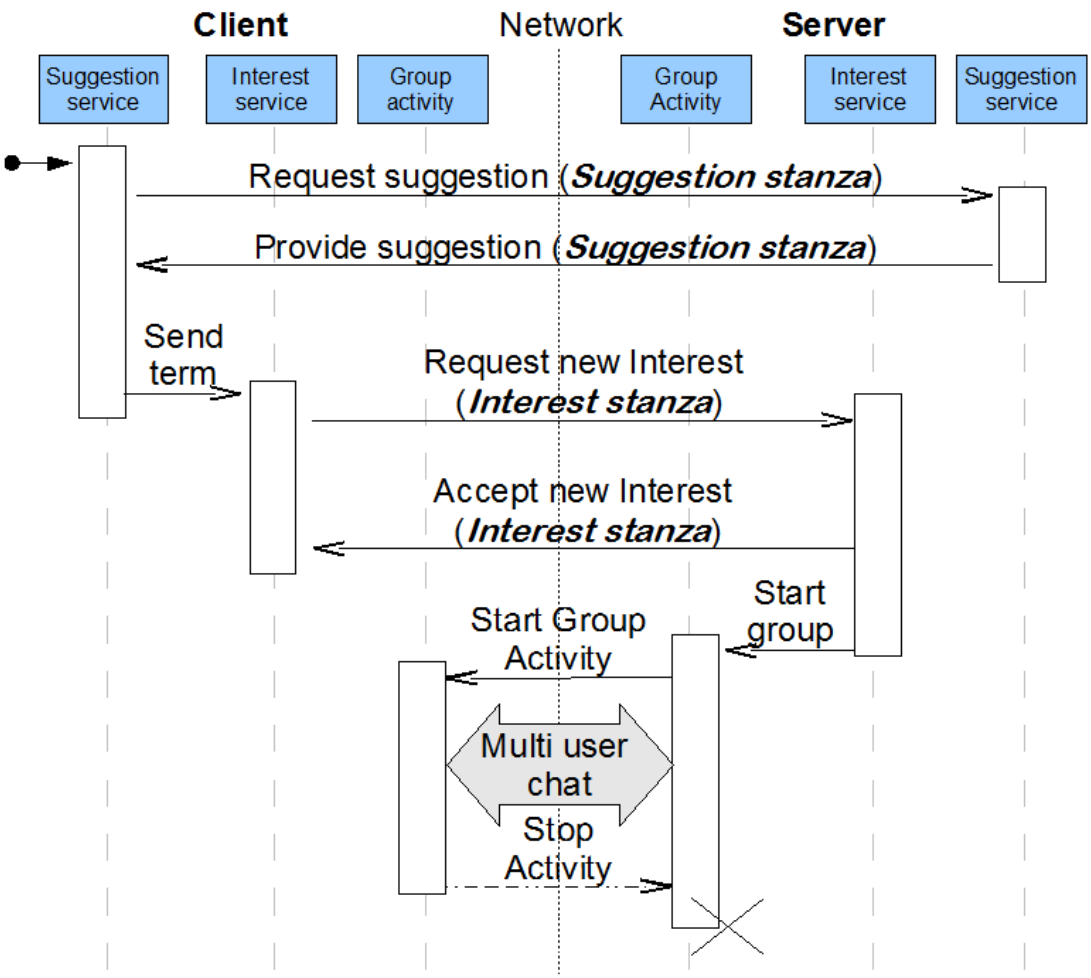

Figure 5.1: General sequence diagram of the developed system -

group generation, but first we will see how the additional feature are embedded into the AndroidBuddy system.

5.1.1 The Client

the client, see figure 5.2, makes use of the established communication from the session service running on AndroidBuddy, and uses the Android application framework to access the services already present in AndroidBuddy, namely intents and broadcast receivers. The client also requires persistent data storage. For this the database functionality from AndroidBuddy is extended and for less frequently accesses data storage such as settings, the editable preferences from the Android framework are used. The various components that provide the extended functionality include the interest service, the group activity and the suggestions service.

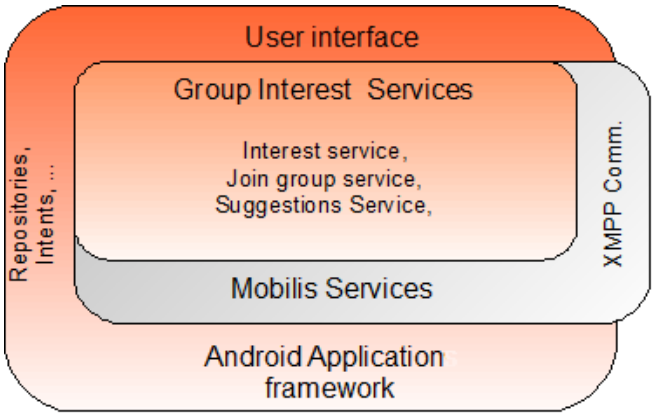

Figure 5.2: General client structure -

5.1.2 The Server

The server, see figure 5.3, makes use of the established communication from the AndroidBuddy agent and makes extensive use of the services provided such as location information available from the context service and the settings service that is extended with additional data structures that are required by the services presented here, namely the recommendation service. The services developed here present the same structure as other Mobilis services, therefore the services are registered and initiated by the Mobilis manager, group activities such as multi user chat are initiated though this manager.

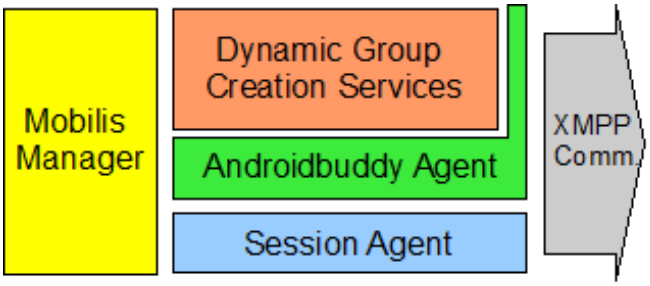

Figure 5.3: General Server structure -

5.1.3 The Communications

The communications use the XMPP protocol, to be specific the Smack[1] implementation of this protocol is employed. Smack provides many features to handle unknown data structures, the capabilities of XMPP to have extended protocols is available in Smack in the form of classes that extend the IQ class. Based on the requirements an interest stanza, suggestions stanza and group activity will need to be communicated between the client and server. Therefore the protocol for these stanzas are proposed in addition to the current stanzas, see figure 5.4.

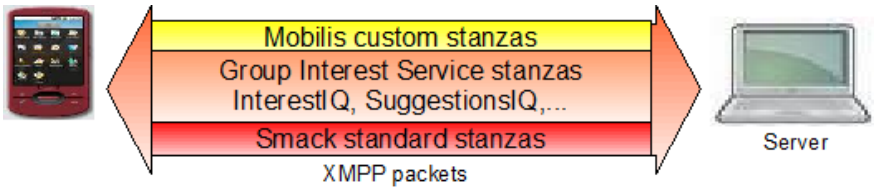

Figure 5.4: XMPP communications -

5.2 Communications Data Structures

The internal structure of the developed XMPP extensions stanzas is presented here.

5.2.1 Interest Stanza

Dynamic group generation is based on the interests that are shared among the user, therefore they are an important part of the developed features. For the system to

[1] www.igniterealtime.org/projects/smack/

function at its full potential the services require more than the term of interest itself, this additional information will also aid in avoiding some of the problems presented in similar works, such as the lack of context (22) and security issues (33).

The interest stanza is composed of the following:

The Interest term this is the general term that is used to be standardized or taken to represent the interest. To standardize the term is to put it in a form that is better suited for sharing with other or to find more occurrences of matches to other users' interests. To provide an example "oranges" and "Orange" may represent the same interest but require to be standardized to "orange" in order to provide a textual match.

The category field provides a small but significant context for the term of interest, with this field we are able to avoid interest confusion from a similar term. To provide an example take "apple" as the interest, this could refer to the computers or the fruit. The category field will allow to distinguish different interests having identical text. This field can also be used to provide a context particular to the user, such as "My family" or "my friends", where the term of interest would indicate other users, in this way a predetermined group of users can be established. The users particular context can be used to develop a service similar to that proposed in (12).

The share level field will define with which subgroup of the community of users the interest can be shared. This is a security field to address the users' concerns with sharing personal information. Suggested levels are share only with buddies or with relation to a certain the degree, where the X_i Degree represents the level of separation from the immediate social network, this is to say the interest can be shared with the buddy of a buddy or the second degree of separation, the first degree are the users' own contacts.

The notes field provides additional information for the item of interest to share small insights into the item of interest.

The inner type is used to send a location stanza or buddies stanza as an interest's additional data.

To provide an illustration of when intrest stanzas are used and the services that use them the flowchat 5.5 is presented.

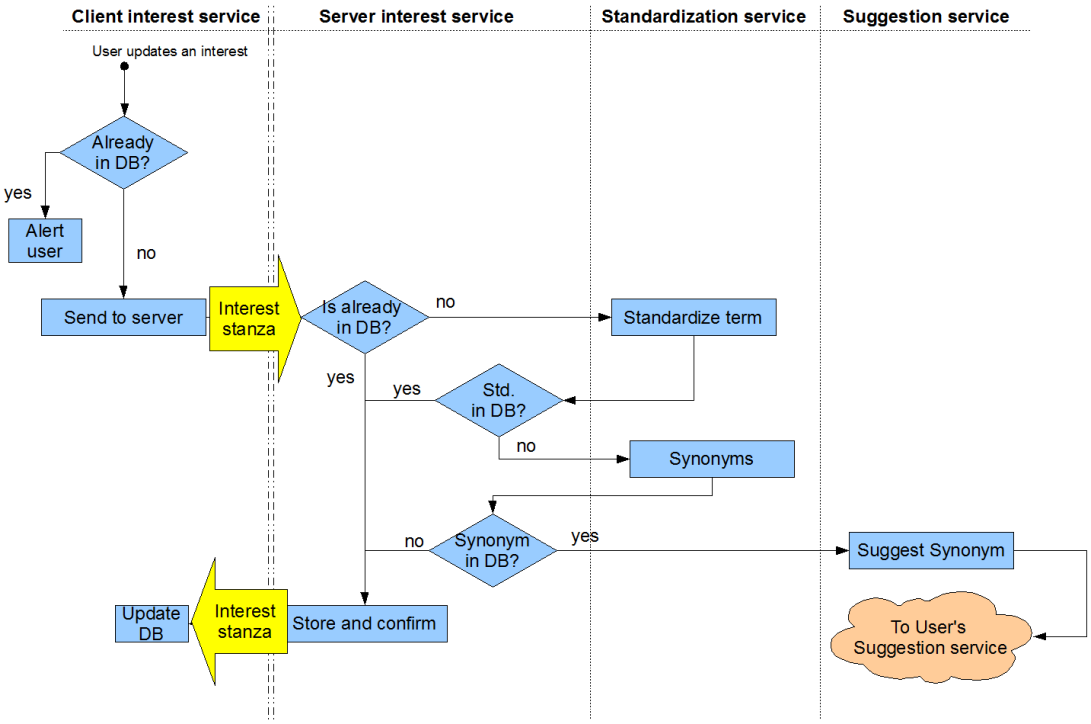

Figure 5.5: Interest information flowchart -

5.2.2 Suggestions Stanza

One of the tools provided for the user to help broaden the interests and in this way promote group generation, is the suggestions. The suggestion provided can be based on many factors such as popular interest or interests of users thar are near by. Suggestions are also used to promote term standardization and to avoid mistakes from the users manual input of the term of interest. The suggestions are also used for the users particular context to send an invitation to a user to join another user's predefined group. To illustrate a short example is provided:

```
User Alice registers the interest term "Bob" in the category "My friend".
```

```
The system then sends user Bob a suggestion for the interest term
 "Friend of Alice" and accepts the interest.
Once Alice selects the interest "my friends"
The group of user that have registered the interest "friend of Alice"
are invited to participate in a group activity.
```

The suggestion stanza is composed of the following:

- A list or single term of suggestion, that can be accepted to become the term of interest.
- The source of the suggestion, to indicate if it is a recommendation from the system or an invitation from another user. This field would also indicate if it is a filtered(34) recommendation, this is to say the recommendations are filtered trough various suggestion criteria.
- The category and ranking for each term suggested if one is available.

This stanza should also be used to request the suggestions from the system.

5.2.3 Additional Stanzas

The system will also require various other stanzas such as nudge that is used to prompt information to the user. This is a one-way communication used to motivate participation. It is provided as a proof of concept for one directional group activities because a nudge is generated based on the interest information of the user community. The questionnaire stanza is used to provide a question answer mechanism for the group, such as polls or common task lists for a group. This stanza would require the following fields:

- A title and/or instruction.
- A set of questions and possible answer type, the type of answer could be yes/no, done, or an open answer.
- A field to indicate if the questions are still open to be answered or the answer is provided. This is especially useful for a group task list.

- And revision field to maintain the most current questionnaire, if some questions have already been closed.

The multi user chat, invitation and settings stanzas are already part of the extended protocols supported in Mobilis and are used in various services.

5.3 Underlying Mobilis Services

In this section we will present some of the services from Mobilis that provide support for the additional features presented here. A brief description of how and where they are used is presented.

As mentioned in the general architecture the additional features make use of the communication services in Mobilis. The session service is the component that maintains the user's connection to the XMPP server. It is also the component where the Smack library is implemented to process specialized incoming messages, therefore all the stanzas developed for this system must be registered in the session service and the mechanism to process the specialized stanzas must follow the suggested Smack architecture, this is done by generating a class that extends the IQ class from the Smack library, generating an IQ provider that parses the XMPP stanzas into objects that can be passed between classes, and an extension of a packet listener that filters the specific IQ types to be processed. All these components must be registered in the session service.

The users' location and proximity to others is part of the information available from the context service. This information is accessed by the suggestion service, to provide location aware suggestions. Another service accessed by the suggestion service is the users' roster to provide suggestions based on the interests of the users' contacts. The user information and stored settings are also accessed through the session service.

5.4 Client Services

In this section we present the components that add features to the client, in figure 5.6 we can see the relation of the main components. The communication between the components is done following the Android software stack architecture, this is trough intents and broadcast receivers, except for the XMPP communications that are done

trough the session service. The functionality of each component can be controlled by the settings to provide functionality adapted to the particular desires of the user or available device resources.

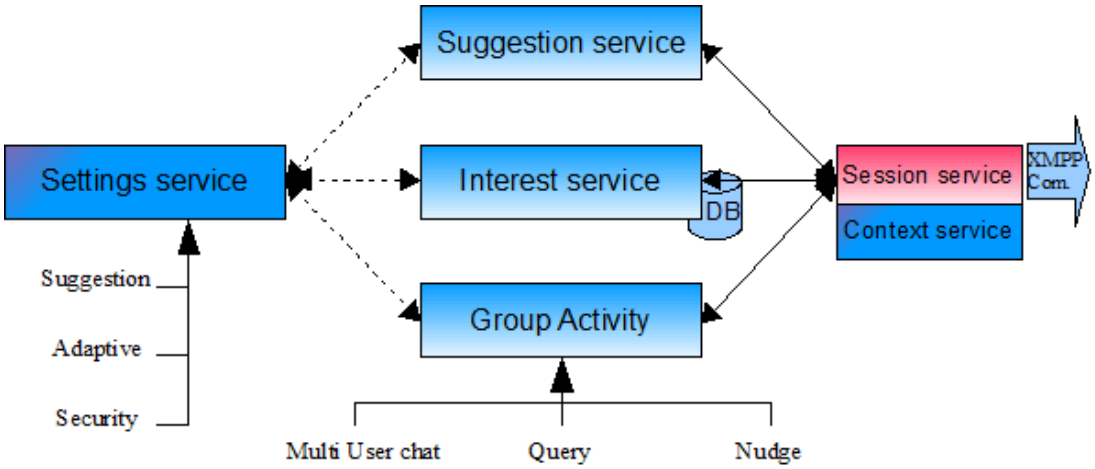

Figure 5.6: Main client components -

5.4.1 Interest Service

The interest service component is responsible for storing, retrieving and updating the user's interest information, it follows a model view controller design pattern, where the model is the repository that stores the interests information. The views are those presented to the user, the stanzas to send to the server or the intents to share interest information between the components in the client. The controller is the editing screen that updates the interests. The editing screen may be started by the user or by the suggestion service.

Because this is the central service it may present a map to the options for the user to easily navigate through the different screens and where a few instructions may be presented.

This service is also responsible for handling the user's particular categories, this is the "my family" or "my friends" categories, to start a group of predetermined participants for the user.

5.4.2 Suggestion Service

The suggestion service is responsible for presenting the suggestions obtained from the server to the user and passing this information to the interest service to register an interest for the user. It is also in charge of requesting and storing suggestions from the server. There can be various criteria to provide a suggestion, but four main types of suggestions to present to the user are the following:

- Single criteria suggestions such as by location or popularity.
- Multiple criteria or filtered suggestion such as by location and popularity.
- Term standardization suggestion. This is to send a term of interest and receive the suggestion of a similar term better suited for the repository of user interests.
- User's particular category suggestions. This is to receive a suggestion to add the interest "Friend of" or "Family of" from another user.

This service is also responsible for retrieving, storing and displaying the various suggestion criteria available at the server.

5.4.3 Group Activity

The group activity service is responsible for handling the activities, such as multi-user chat, for the client. With the aid of the users' settings to respond to activities in a predetermined manner, it is responsible for handling the start of applications or components for the group activity.

The responsibilities for this component include the following:

- Notify the server if it is not able to process an invitation.
- Process invitations to start activities such as multi-user chat or polls.
- Start the requested activity and process messages if necessary.
- Close the activity when required.
- Notify the server when an activity is closed
- Maintain a register of the activities the user is participating in.

The activities that can be started are those provided by the Mobilis platform.

5.5 Server Services

The server presents the counterparts to the client services. Most of the logic is executed in the main Interest management service, the suggestion management service and the group activity service, see figure 5.7. The communications are established trough the AndroidBuddy agent. The Interest service manages the interests and triggers the group activity service that in turn starts the required group activity. This is the core of the system. A valuable feature is the suggestion service that takes the interest information to generate a list of interests that may be suitable for the user.

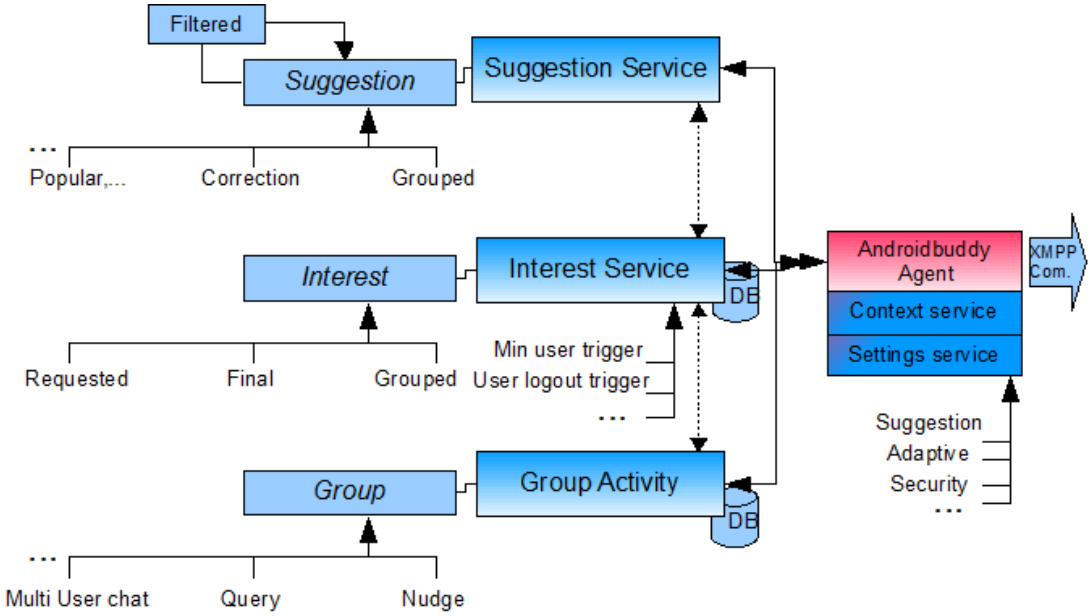

Figure 5.7: Server Composition for developed system -

5.5.1 Interest Management Service

The interest management service at its core takes interests from the user and triggers a group activity to be started. To accomplish this task it requires many services. Once an interest term is received it must be standardized. This is to correct the spelling, find the root of the term of interest removing any inflection such as the plural form, converting to all lowercase letters, and triggering the suggestion of a similar term if one is already found in the interest repository, this may include synonyms. If the term of interest from the user, this is the requested interest and the standardized term of

interest are different the standardized term of interest should be sent as a suggested to the user to correct the entry. To accomplish this the suggestion service is employed, if there is no need to notify the user it can be considered a final term of interest.

The Interest management service may register various triggering mechanism such as minimum number of active users to start a group chat or minimum number of participants to trigger closing a group. Once a trigger is reached the appropriate service should be notified. Other triggers can be time dependent, for example if a term of interest has not been used in a determined period of time a nudge could be sent to suggest the removal of the interest by the user.

This service also stores and retrieves the interest terms for each user. It also provides this information to other services such as the suggestion service or group activity. It is also responsible for handling special interests such as those of a guest user's and personal category interests where the management of the interest is strongly linked to the particular user. A guest users interests are not stored persistently, therefore the guest user is suggested to register interests to use the service and those interests are discarded once the user logs off.

Personal category interests are handled as very particular interests. A conversion from the interest service needs to be accomplished in two stages, they are the following:

Once an interest is received from the originator it is treated as a personal category interest if the category field begins with the keyword "my", The term after "my" is considered the subgroup or personal category, this may be "friend" or "family". The interest field should contain a participant's identification, if it does not a nudge can be sent to notify the user and the interest is discarded. If the user exists, the term of interest is generated in the following manner.

The interest is formed by concatinating of The subgroup term , the keyword "of" and the user's identification, for example "family of Bob". This interest is then stored for the originator and suggested to the identified participant. The final result is a state where participants have an interest in common as unique as the user's identifiers.

If an interest is received from the originator that contains "my" as the start of the category and the interest field contains the user ID. A group activity for all the users with this interest should be triggered.

The Interest service should also handle the share levels in an interest as described in the interest stanza, section 5.2.1.

5.5.2 Suggestion Service

The purpose of the suggestion service is to advise the user of interest that may be suitable, in this way promoting participation. In the previous sections a personal category suggestion and term correction have been illustrated, here we present suggestion by various other criteria that include popular interest, interest based on location, interests from buddies, interest similar to other users and filtered interest where the term must meet various criteria.

The suggestions management service may receive a request for recommendations of a certain criteria. The server should reply with the suggestions as presented in the suggestion stanza. The user may decide to add the suggestion as an interest, this would be handled by the interest management service.

Some of the recommendations strategies are presented below:

Popular Recommendation returns the interests that are the most popular among users in general. This implies that there may not be an open group from them at the time but it is probable to join one once more users are active. The algorithm to select the most popular is simple.

There should at least be enough users registered with this interest to form a group. Use the mode to provide a cutoff as to what is popular and what is not and provide only the top 20 elements as not to overwhelm the user with options.

Location Recommendation , this component uses the context service to detect what participants are within the radius of the users location established by the context service and provides a list of interests that participant close by have and the users does not.

Buddies Recommendation , this component also uses information obtained from the Mobilis platform in the form of the users' roster or contact list, the algorithm is as follows: Get all the user's buddies Compile and possibly rank the interest that they have Eliminate the interests the user already has Provide the list.

Similar Interests Recommendation , this component provides interest that may be to obscure to find in the popular interest and may not be found within the buddies' interest. The algorithm is as follows:

For each participant find how many interests are in common and Get top users. From the users find how many differences in interest, Provide the interests from the with most interests in common and least different, but at least one different.

This insures that if there is a user with a wide variety of interests because there are so many differences, that user will not necessarily be the match for similar

Open Groups , this recommendation strategy provides the catalog of open groups, so the user may be able to join a group immediately by adding the interest.

Each of the strategies may present a ranking, such as number of time a term is repeated or closest to current location. The ranking may be provided for the user to have a better overview of how interesting the suggestions are.

5.5.3 Group Activity

The group activity for the server has very similar functionality as that of the client, this includes maintaining a register of active groups, starting the activity and destroying it. The main activity that is started is multi-user chat, which is already partially supported in the Mobilis platform. Other activities include a questionnaire that may be presented in the form of a survey, poll or common task list, this activity will make use a specialized stanza as presented in the additional stanzas section 5.2.3.

5.5.4 Additional Services

Additional services in both the client and the server that are not at the core of the functionality of the system include the guest login service, were the user will be able to participate without using a personal account to maintain anonymity or test the service without having to register. The Adaptive service is responsible to constrain the functionality of all the components to limit the resources used on the client.

5.6 Conclusion

In this chapter presented the complete design of the system that includes the core functional of interest service to manage interests, group activity to start group activities and the suggestion service to enhance the compatibility of interests.

While not all the services are required for the system to function properly they provide a placeholder for further development and may enhance other features of the system.

Chapter 6

Implementation

In this chapter some of the experiences gained while developing the application and an insight into how the system is implemented is provided.

The development was completed in a series of stages that involved the complete system. Each stage or round presented a more refined set of challenges, from establishing the minimal functionality to providing the added features.

6.1 General Architecture

in order to implement the basic functionality of registering interests on the server a brief review of the existing XMPP extensions, develped by the XMPP Standards Foundation [1], to find a suitable match or to develop one from the design had to be conluded. An example of this could be XEP-0048 for bookmarks [2] that allows bookmarks to be shared in multi-users chats, URL of the bookmark could be modified to store the string of interst for the users to share interests in a group.

A review of several XMPP extensions provided various mechanisms to share interest information, but there were two main drawbacks to all of them. Stanzas were not explicitly designed for interest information so other types of information could be shared using the same extension; this could present confusion at processing time and there was limited or no possibility for adding fields to the stanzas, this presented design issues in how the information is to be interpreted and future extensibility.

[1] http://xmpp.org/extensions/
[2] http://xmpp.org/extensions/xep-0048.html

There were many indicators to develop a new interest type stanza for the system. A repository for the interest information of each user was developed following nomenclature and general structure of the components already present in the system. The structure that was followed was the creation of a bean class that stores the core of the information being handled. The bean is wrapped in a packet class that extends the IQ class from the Smack library to be able to send the information as a stanza. In order to convert a stanza back into an object a provider class that extends the IQProvider class from the Smack library is used. Finally the services that contain the logic to process the stanzas are created and registered in the Session Server along with the custom IQ class and the IQProvider, a graphical depiction is presented in figure 6.1. In this graph the custom components for interest information are presented below the Smack library classes and above the registration to the SessionService class.

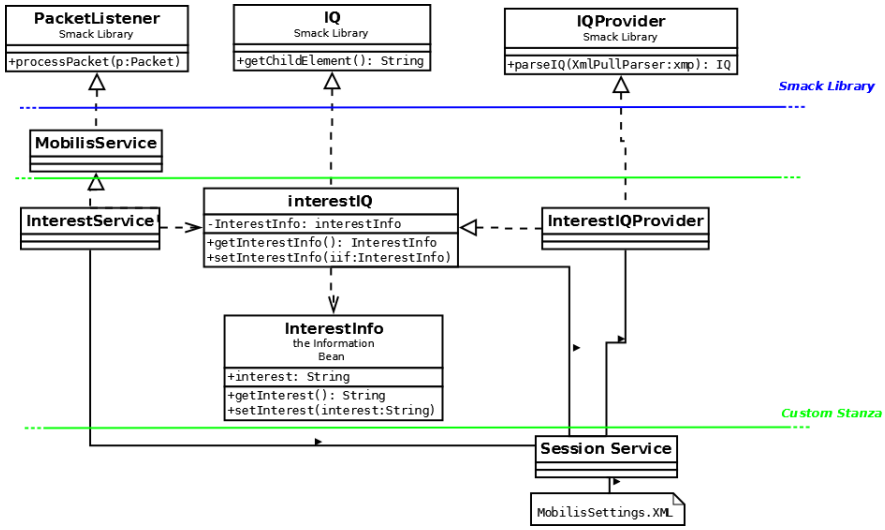

Figure 6.1: Composition of a custom stanza from the Smack Library -

The next stage was to generate a group and start a multi user chat with the participants that share an interest. This presented a few challenges due to the multiple components that required management, as the AndroidBuddy system presents the Session server and session agents to manage a group and initiate the multi user chat. The session agent in turn uses the invitation and multi-user chat extensions provided by the smack library.

The Session coordinator and multiple session agents presented several issues for the

implementation of a multi user chat, for instance the added layer of indirection to create a group did not allow specifying the name or topic of the multi-user chat. For these and other reasons it was decided to manage the groups directly using the invitation and multi-user chat capabilities of the smack library overriding the session agents. This allowed better control over the groups for multi-user chats. In order to maintain the architecture of the system the initation of groups was maintained in the session agent components bu the groups could be managed directly on the XMPP server. For the client component these changes presented little change.

The concluding stage of development was to add and maintain activities that are triggered by interests. The main mechanism is to have a minimum number of participants that share a common interest, to triggered the activity of a multi-user chat. This initial trigger presents the structure to followed by other triggers, the structure is an observer design pattern with the aid of an interface, where the model is the interest repository and the observer are the triggers, see figure6.3.

In the following code the process to create a new group based in users common interest is presented. the addInterest method in line 2 is the interface that is updated by the interest service and update to the triggers service local interest repository, line 8, is reviewed to evaluate if there are enough users to start a group, line 13, if so invitations to participate in the group are sent, line 15, eash time a new participant is interested in the group.

```
1: // method to observe when a new interest arrives in InterestService.class
2: public void addInterest(String iid, InterestInfo interestInfo) {
3:   String interest = interestInfo.getInterest();
4:     Set<String> users = mInterestsTable.get(interest);
5:
6:      if (users == null) // if a interest has no users create the table for user
7:         mInterestsTable.put(interest, users=new HashSet<String>());
8:      users.add(iid);
9:      MobilisManager.getLogger().info(
10:        "Added interest from: "+iid+" for "+interest+":"+users.size());
11:
12:     //if there are enough user create group and send invitations
13:     if(users.size()>=minGroupSize) {
14:        mSessionService.createMUC(interest);
```

```
15:         inviteUsers(interest);
16:     }
17:}
```

In the design chapter the interest service has many responsibilities. To minimize code complexity several functions were added as a separate class that is registered with the interest service. This improves modularity, extension and clarity of the code.

During development of the basic architecture several issues were found where the existing architecture could not be followed, due to various reasons including code bugs. The service that reads the settings for some components of the server did not function properly and the settings in some classes were provided not by the settings service but by the default value in each class. The new components manage the settings in a different to clarify that the settings service has a limited functionality.

This section provids an overview of the components that are required for the system to function, this includes building the user interfaces following the Android architecture and implementing some of the features presented in the design chapter.

6.2 XMPP communications

For the basic architecture an XMPP communication was established. Several custom stanzas were developed following the Smack architecture of an IQ class extension, a packet provider, and a packet listener that triggers the class responsible to process the information for the interest service and the suggestion service. The interest stanza contains the string fields interest and category as described in the design chapter, for the security aspect a boolean field to indicate if it is an interest to share with buddies only is also present. The subtype field was omited but can be implemented using the setProperty method from the Smack Library, with this method a java object can be sent from the client and cast at the server. The Suggestions stanza contains a single field to indicate the type of suggestion that is povided or requested and an ordered list of items and their ranking if there is one, or the suggestinos.

6.3 The Client

The components that compose the client can be broken down into four sections; these are the user interfaces, the activities that provide logic to the interfaces, the services running on the background to trigger activities and the database that is the main repository for persistent user information. In this section an account of concerns that developed during the implementation is presented.

6.3.1 Interfaces

The interfaces developed here were designed to follow a similar look of the ones already existent in AndroidBuddy, except for one to demonstrate an interface that could follow a similar structure to the current interfaces but may be considered more fluid or intuitive, while maintaining a very similar functionality. See figure 6.2.

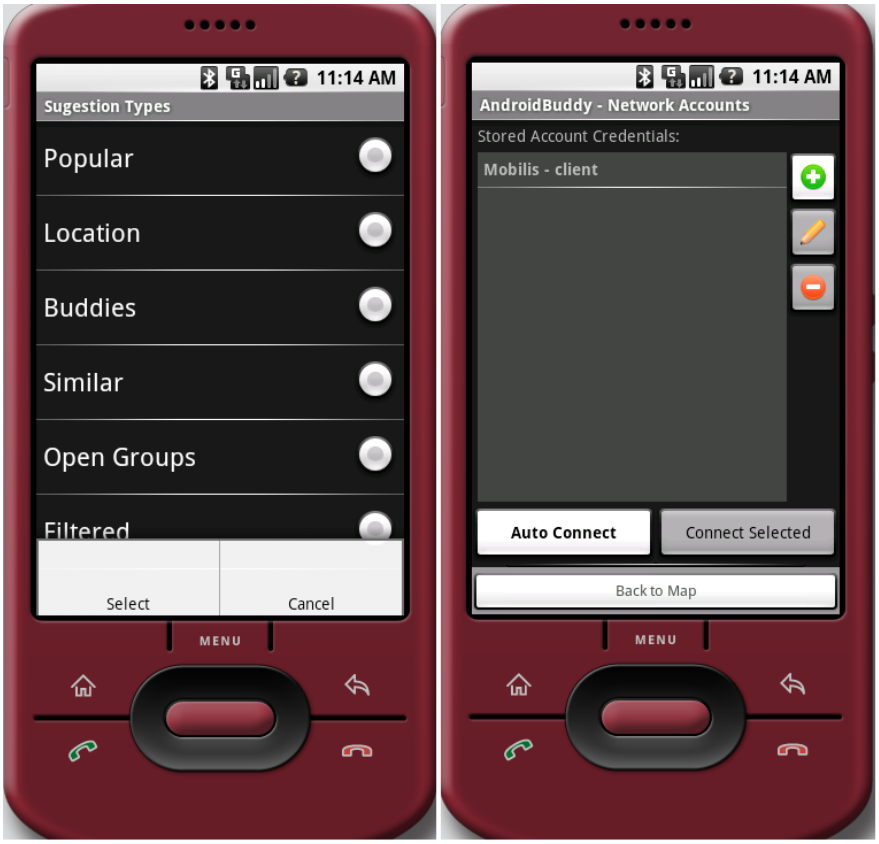

Figure 6.2: Left: UI with large list and collapsable menu; Right: Current list UI -

Many user interfaces contain activities to be perform to the items on a list and to return to the main user interface that is the map screen. The processes for the user is to select an item then apply the activity to that item and finally to select return to map. To illustrate the steps are as follows:

1. Start Interface
2. Select Item
3. Select activity for the item or cancel
4. Select return to map

This could be reduced to only providing the list of items upon which the activity is to be performed and possibly a contextual menu accessed through the "menu" button on the device, if the users selects an item from the list this automatically triggers the activity to be performed on this item and a return to the main screen. If the user does not wish to complete the activity for an item, the user may use the menu on the device to go back to the main screen. This makes the processing more fluid for the user. To illustrate the following sequence is presented:

1. Start interface
2. for an activity other than the default select from "menu"
3. Select Item

In step 3 after the user selects the item the user is returned to the map screen. If the user requires the default activity step 2 can be disregarded.

The Android application framework provides many tools for user interfaces including the preferences screen that is automatically generated based on preferences stored in an XML format as a resource file. This user interface is very useful but has some issues when the values are accessed by services with no context. The context may be provided by the current top activity or main activity that is always present while the application is running, but updating it from other activities may cause conflicts. This issue was encountered or the automated update of user configuration from a service, services do not have contexts and therefore cause conflicts. This service was not fully implemented.

6.3.2 Activities

The activities provide some logic for the user interfaces. This logic was mainly filtering the information to be broadcasted as an intent or to store the information in the local database. Accessing functions trough intents, that are short internal text messages, instead of the well known object referencing presented some challenges during debugging. The key-value structure of the intent messages, were the key was a simple text, was prone to simple spelling mistakes that would cause a component crash where the problem was not clearly evident.

6.3.3 Services

The structure of the added services differed slightly from the structures already in AndroidBuddy, mainly in the access to the database , because of the instantiation order of the services. The requirement of a context for the database to be initiated posed conflicts for the database to be accessed directly by the services that do not have a context. To maintain consistency between the existing system and the developing one the services maintained broadcast listener although they are not currently required by the system. many activities are triggered by incoming stanzas received trough the developed service. The services store information in the database and trigger the activities to take the information from the database and present the updates on the user interface. An example of this is the suggestion service where incoming stanzas trigger the update of the user interface.

6.3.4 Database

The database on the client is an object oriented database in contrast to the SQLite database that is part of the Android framework. While the performance is comparable the advantages of storing the information as an object are not fully present in the system. There are some drawbacks in additional object creation for comparisons and sorting. The additional objects require more memory and comparisons and sorting is more elaborate than a comparable SQLite database. Ultimately the decision to keep the object database rested on the fact that there are very few elements that are store in the repository. This minimized the disadvantages while making full use of the direct object instantiation and modification.

The following code sample is taken from the mainView activity of the client and presents the solution to the instantiation problem mentioned in 6.3.3. The main map activity is the initial activity triggered by launching the application. This activity in turn triggers the Mobilis service that in turn generates the interest service. Becuse context is required for database instantiation at the client, the trigger to start the database for the service is in the main map activity, because the interest service has no contextit can not initiatethe database so this responsability is shifted to the database creator this is the main map activity.

```
1:  private void initDatabase() {
2:        DBHelper.getDB(); //is started in the mainView.class (main map)
3:          // then it is started in InterestService.class
4:          // services are created before context ; db needs context
5:        sessionService.getInterestService().startDB();
6:   }
```

6.4 The Server

In the previous sections some of the aspects of the server have been presented, namely establishing the XMPP communication and handling the filters and stanza types. For the server implementation there are more issues regarding the classes that compose the services and how to make use of the information already available from the previous implementation.

The first service to be implemented was the interest service that receives and stores the users' interests. The data structure where the interests were stored was initially non-persistent because the interest information was stored persistently on the user's device. This presented the issue of a lack of initial terms of interest users could match. To illustrate take the use case of a participant requesting the most popular interests immediately after the server has been started, the client would not receive any suggestions from the server. Several possible solutions were evaluated such as filling the database with initial "seed" (35) interests that could be of a general categorization or requesting the interest information from the users upon startup. The simplest and most direct approach is to maintain the interest persistently avoiding costly communications or waiting for users to add the "seed" suggestions as an interest.

Because the interest service provides underlying information for the suggestion service,see figure 6.3, a more detailed analysis of the structure of the interest repository is presented in suggestion services section.

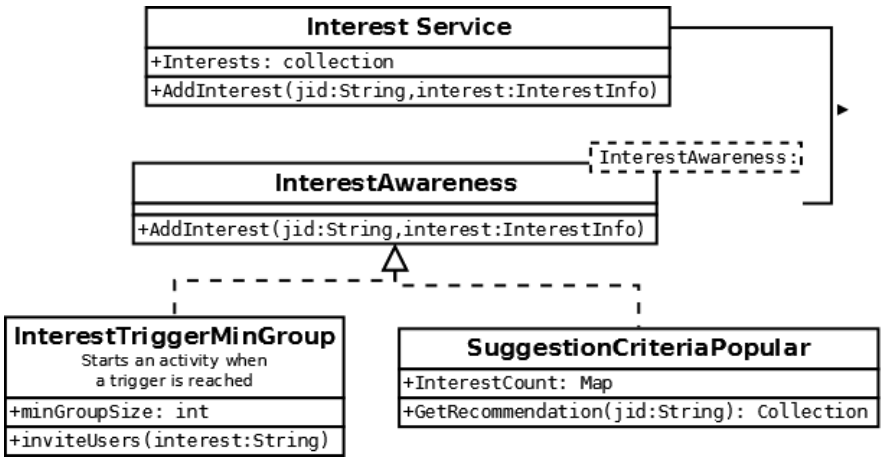

Figure 6.3: Observer design pattern used for the interest service -

One of the key services used by the interest service is the similar term service. This service provides a list of synonyms and words that are related to a certain term. This service is used to standardize the terms in the interest service. The intent of this is to reduce the set of terms that may represent a certain interest. To accomplish this many options were reviewed, mainly based on WordNet (36) a large lexical database developed at Princeton university. During testing finding synonyms performed very well, although finding related words did not function as well. To illustrate related words the example of the term "basketball" and the term "sports" may be seen as related words that are not synonymous. Other tools were searched to enhance the related word function. Documentation showed (26) that tagging communities provide strong links between related terms. Web scraping a site called the library thing[1] provided an excellent repository for related terms as pertaining to books. The web scraping consisted of taking the results of a tag cloud for books. The result was a list of terms that were related to one another and ranked as to how often the terms were found together in the tagging of a book. The results were very promising except for the propensity for a high ranking of terms related to the literature format itself such as "biography" or "novel". To avoid providing false matches to the user the use of the

[1]www.thelibrarything.com

web scraping was eliminated. Instead the use of a web API from Big Huge Labs [1] was used. This API is based on the WordNet system with additional features such as spelling corrections and a much faster response time as compared to the WordNet web API. Ultimately this was the system that was selected for the term standardization.

Below is a code excerpt from the class that accesses the web services to find similar terms.

```
1: public Collection<String> getSimilarTerms(String term){
2:  Set<String> result =  new HashSet<String>();
3:  URL wordlist;
4:
5:     try {
6:       wordlist = new URL(
7:            "http://words.bighugelabs.com/api/1/"+KEY+"/"+term+"/text");
8:       BufferedReader in;
9:       in = new BufferedReader(new InputStreamReader(wordlist.openStream()));
10:      String inputLine;
11:
12:      while ((inputLine = in.readLine()) != null){
13:         result.add(inputLine);
14:         MobilisManager.getLogger().fine(
15:             "INFO Term: "+term+" is similar to: "+inputLine);
16:      }
17:      in.close();
18:      } catch (MalformedURLException e) {}
19:  return result;
20:}
```

The suggestion service is composed of various classes, see figure 6.4, that access related information to provide the suggestion such as user location and the user's roster. The nature of the AndroidBuddy system where various accounts are taken into consideration such as Facebook and the contacts from the smartphone to compose the users contact list presented some challenges in generating suggestion based on the user's contact. The contact list needs to be generated after the user has logged in, this could be after a suggestion for buddies interests is requested, therefore the sequence in which

[1] http://words.bighugelabs.com/api.php

the user requests and provides information has a direct effect on the results that are provided. Because stanzas may be received in a different order from which they were sent this means the results are non-deterministic. This may affect the performance of the system.

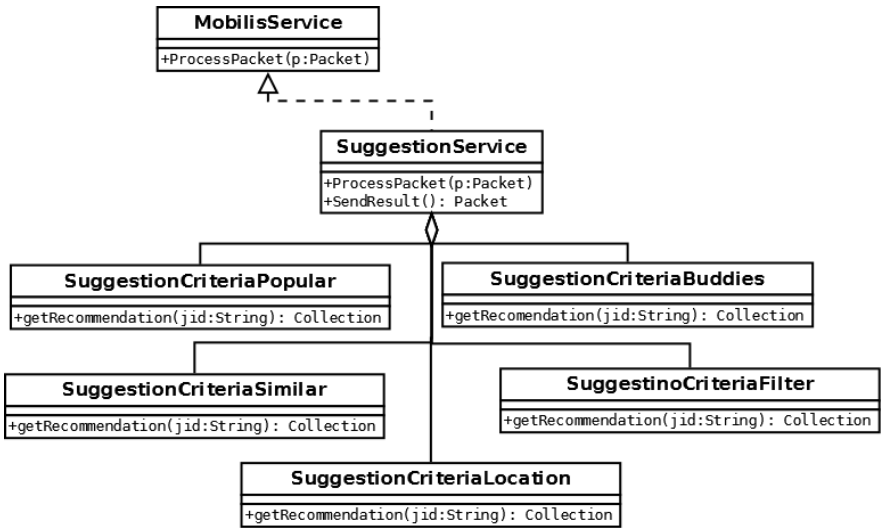

Figure 6.4: Facade design pattern used for the suggestion service -

6.5 Conclusion

While not all the features presented in the design chapter were implemented,such as the users particular context, service adaptation or multiple degree security for interest sharing. The resulting software is functional as a proof of concept and provides the structures to complete and elaborate on the system.

Chapter 7

Evaluation

To evaluate the system a short questionnaire was presented to a small group of users that used the system simultaneously. Although the system was slightly altered to increase performance for a small group, the results of the evaluation supported some of the design decisions of the system.

7.1 The Questionnaire

The questionnaire is composed of five sections, see figure 7.1. The general control section ; to establish a reference of the user's willingness to participate in a group. The interface section; to evaluate how accessible the user interfaces are and how clear the options are in the user interfaces. The navigation section; to evaluate the difficulty in changing between activities. the usefulness section; to register the services performance and the overall system. And the suggestions section to evaluate how well the suggestion service performed.

7.2 The Test System

The test systems were run on desktop computers running Linux and the emulator with the developed application, the server was run on one of the machines and the interest repository was filled with some values for the suggestion system to be able to provide some suggestions. A virtual location was given to each user, while not all users where within a close virtual location, all were in the same phisical location. This lead to some direct commenting on the shared interests between the participants. There was

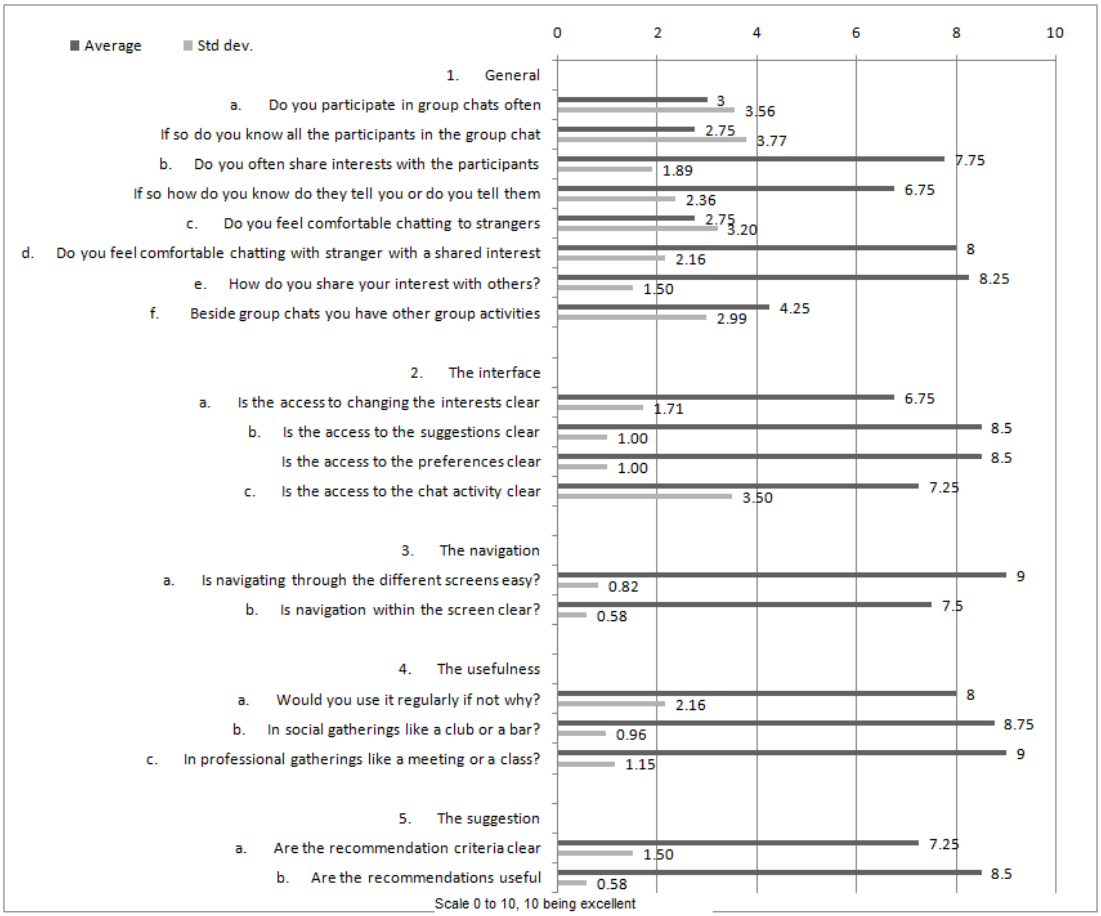

Figure 7.1: The questions and summary results of the questionnaire -

no tutorial on the navigation or function of the system, only a brief explanation of what is mainly expected of the users. This is to introduce personal interests and that suggestions are available. The evaluation was structured in the following manner:

1. The participants were given the purpose of the evaluation and a brief description of the system.
2. The control questions were completed.
3. The participant was instructed to use the system by introducing interest and requesting suggestions if necessary.
4. Once the participant stopped using the system the remainder of the questions were completed.

7.3 Observations

During the evaluation each participant was monitored for any notable activity. None of the participants had any prior experience with the Android emulator and often exited the application because of multiple selections of the "back" button on the emulator in an effort to return to the main screen. Most of the users searched for the activities to perform in the tabs of the main map screen as they are so prominent and would need to be instructed to use the "menu" button on the emulator. For some of the suggestion criteria the participants expected to receive the name of other users, namely in the location and the buddies recommendations. The participants expected to receive a list of participants close to the location of the user, not the interests of those participants. For the location suggestions and for the buddies the user expected a list of recommended contacts. This is notable because it supports the concept of systems to enrich the users contact list based on interests. None of the users selected the open groups or filtered suggestions. After asking the reason why the users stated that the name did not seem appealing and after presenting what the options actually do the user still did not see them as valuable features. The instruction given to the participants by the system in short notifications were not intrusive and useful for the participants. In general the activities within the system were not clear; the participants would often press the "menu" button on the emulator instead of the activities presented on the applications

user interface. For the initial task of entering user interests, one participant did not know why they should input a category and none of them were concerned about the security of the system and if the interest could be shared with any other participant.

7.4 Summary of Results

On average the participants took a passive role in participating in a group. The participants seldom chat and when they do it is mostly with acquaintances. Seldom do they feel comfortable chatting with strangers unless they know they have some interest in common. The participants are mostly happy to share their interests with other, even strangers and sometimes have group activities other than chat for example gaming. The participants found navigation through the different activities mostly clear but the options within each activity were not as clear. The chat activity and accessing the user's interests was not very clear. The suggestion system's criteria were also not very clear to the users. The suggestion the user focused on in the questionnaire and in observation were popular, location, buddies and similar. The filtered and open group suggestions where of little interest to the users.

Overall the system was considered good and useful in professional and personal environments. The participants proposed various applications for this type of system including some that have been presented in this work, for example in travel and in a new classroom to facilitate making acquaintances. Most users saw the feature of making new friends very appealing.

The summary evaluation of the questionare is presented in figure 7.2.

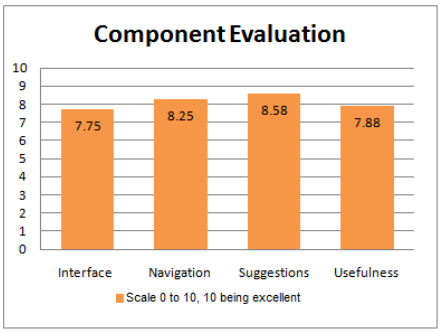

Figure 7.2: Summary of results from the questionnaire Sections -

7.5 Summary of User Comments

During the interview of each participant there were some notable comments regarding the system, some of these comments are summarized here to illustrate the users point of view.

There is a sense of security, even among strangers, as long as the group is of a consistent nature, for example if all the participants in a group are interested in computers the group feels consistent and if there are some other participants that are not as interested in computers but would only like to participate in some way the group does not feel as consistent and therefore not as safe. An extension to this is a comment from a user indicating it does not matter what the topic of conversation is as long as the participants are potentially interesting.

User enjoy having an alternative to whatever they are doing, having a group chat accessible even if they are not participating provides a sense of opportunity that enhances the users main activity, the last statements is based on a comment from a users that chats can be done in parallel to other activities to enhance another activity or destract form a boring activity.

All The participants agreed that the application is enjoyable and is a good tool to find friends.

7.6 Conclusion

The evaluation of the system was made based on the experiences of a small group of users. The server was populated with information to simulate the presence of many users and aid in the functionality of services such as the suggestion service. The results of the evaluation, registered in a questionnaire and in the notes taken during the evaluation, show that the functionality presented was useful or the users and that there are still some user interface issues that need to be improved. Users provided suggestions to improve the system including some elements that were presented in the design of the system although they were not implemented. Overall perception of the system was good and useful in formal and informal situations.

Chapter 8

Conclusion and Future Work

This short chapter presents a brief description of the potential future works and a final conlsion of this work.

Dynamic group generation based on interests is a valuable tool for users to enrich their social environment and use community knowledge to accomplish a shared goal. It is able to do this because it facilitates social interaction and overcomes, to some degree, the difficulties in engaging or starting a social interaction. The ubiquitous availability of social networking services based on smartphones makes the task of engaging in a virtual social network accessible by having so many potential participants available for communication the focus changes from finding other participants to finding the right participants to engage. This is evident in some related works that refined and enhanced the users contact list to select participants.The selection process is not centered on the user's current social network but on a more general compatibility point, that is the user's interests. Starting with a compatibility point that is not the user's current social network and instead personal interests allows the social network to expand and adapt more freely.

Dynamic group generation is not a tool that is found on many social services and may prove to be very attractive for users particularly when there are more users and the interest can be more refined, therefore allowing a closer match to the user's characteristics and thus a potential for better group activities.

The application presented in this work provides the proof of concept of the usefulness of dynamic group generation based on common interests between users and how it can develop and be combined with other tools, such as suggestions and standerdized interest

repositories to enhance the performance and meet the user's expectations. There is still research and practical analysis required to produce a consistent result to place the user in a group where the compatibility of all the users is very high, although the intermediate results that are currently provided are still valuable to the user.

System improvements and future works span a few fields from the technical to the social. Some of the basic aspects that proved evident here and can be considered for many other applications are the customization of XMPP to improve performance on mobile devices and to adapt to the changing resources of smartphones. This item could also be extended to provide a service for peer to peer communication that would still require a central server to enhance the security of the peer to peer connections by maintaining white or black lists and updateable security configuration for the users.

Other possible future works are related to how this system interacts with others for example possibly inducing the users interest based on usage patterns of Internet browsing or from key words used during discussion in the social networks, similar to those available in Twitviz (20) and (26) but applied to derive the user's interest. Another area that could be developed is a framework to provide dynamic group generation for composition with other services or applications. With additional work the dynamic group generation based on interests could be passed from the server to the client making it available to more services in a mashup-style service composition.

The design of this system was ambitious and was not fully implemented. Although the functionality that proved the concept of the system was present there are still items that could be developed from the design chapter. The completion of all the requirements and design aspects would meet all of the expectations of the system presented in the evaluation chapter. This would include having additional activities such as polling to facilitate initiation of group gaming, presenting items such as a location or a buddy as a suggestion or interest for the user and providing the contact information of participants in the group. Based on the experiences in the evaluation of the system there are many possible specific cases for dynamic group generation that could be developed, because there are so many activities that can be accomplished as a group with a common interest.

A good way to find the balance of features needed to present consistent results for dynamic group generation based on interest is the field testing, this provides valuable

input and highlights the most important features for the users, features that may not have the same importance for the developer.

As a result of the development of this system it was found that simple tools or features like the one presented in this work facilitate interaction between people at a basic level, in the same sense that blogs facilitated the personal web page, by making tools that are available for the users easier and thus more prominent. Simple tools like dynamic group generation facilitate the way a user accesses social networks, taking as much advantage as possible of the features in smartphones, such as location and availability and masking the limitation, such as a limited interface, as compared to a desktop computer to help adapt the features function on the smartphone to improve the users expierience.

References

[1] Athena Vakali Christos Zigkolis, Yiannis Kompatsiaris. **Information analysis in mobile social networks for added-value services**. *www.w3.org*, 2008. 1

[2] Toshiyuki Masui and Satoru Takabayashi. **Instant group communication with QuickML**. pages 268–273, 2003. 1

[3] Tomi Ahonen. *Tomi Ahonen Almanac*. eBook, 2009. 1

[4] Matti Rantanen, Antti Oulasvirta, Jan Blom, Sauli Tiitta, and Martti Mntyl. **InfoRadar: Group and Public Messaging in the Mobile Context**. 2004. 2

[5] Thomas Springer, Daniel Schuster, Iris Braun, Jordan Janeiro, Markus Endler, and Antonio A. F. Loureiro. **A flexible architecture for mobile collaboration services**. In *Companion '08: Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, pages 118–120, New York, NY, USA, 2008. ACM. 2, 3, 7, 16

[6] Rebecca Grinter and Margery Eldridge. **Wan2tlk?: everyday text messaging**. pages 441–448, 2003. 3

[7] Barry Brown and Matthew Chalmers. **Tourism and mobile technology**. pages 335–354, 2003. 3

[8] Guanling Chen Nan Li. **Multi-Layered Friendship Modeling for Location-Based Mobile Social Networks**. *MobiQuitous*, 2009. 3

[9] Ruth Kikin-Gil. **Affective is effective: how information appliances can mediate relationships within communities and increase one’s social effectiveness**. *Personal Ubiquitous Comput.*, **10**(2-3):77–83, 2006. 4

[10] Gerard Beenen, Kimberly Ling, Xiaoqing Wang, Klarissa Chang, Dan Frankowski, Paul Resnick, and Robert E. Kraut. **Using social psychology to motivate contributions to online communities**. pages 212–221, 2004. 4

[11] Istvn Koren. *Conceptual Design of a mobile collaborative Platform based on Android and XMPP*. Master's thesis, TECHNISCHE UNIVERSITT DRESDEN, 2008. 18, 20

[12] Reto Grob, Michael Kuhn, Roger Wattenhofer, and Martin Wirz. **Cluestr: mobile social networking for enhanced group communication**. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 81–90, New York, NY, USA, 2009. ACM. 23, 37, 45

[13] Brian Amento, Loren Terveen, Will Hill, Deborah Hix, and Robert Schulman. **Experiments in social data mining: The TopicShop system**. *ACM Trans. Comput.-Hum. Interact.*, **10**(1):54–85, 2003. 23

[14] Anupriya Ankolekar, Gabor Szabo, Yarun Luon, Bernardo A. Huberman, Dennis Wilkinson, and Fang Wu. **Friendlee: a mobile application for your social life**. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–4, New York, NY, USA, 2009. ACM. 26

[15] Antti Oulasvirta, Mika Raento, and Sauli Tiitta. **ContextContacts: re-designing SmartPhone's contact book to support mobile awareness and collaboration**. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 167–174, New York, NY, USA, 2005. ACM. 27

[16] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. **Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application**. pages 337–350, 2008. 27

[17] Ido Guy, Inbal Ronen, and Eric Wilcox. **Do you know?: recommending people to invite into your social network**. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 77–86, New York, NY, USA, 2009. ACM. 28

[18] Ido Guy, Michal Jacovi, Elad Shahar, Noga Meshulam, Vladimir Soroka, and Stephen Farrell. **Harvesting with SONAR: the value of aggregating social network information**. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1017–1026, New York, NY, USA, 2008. ACM. 28

[19] Per Persson and Younghee Jung. **Nokia sensor: from research to product**. In *DUX '05: Proceedings of the 2005 conference on Designing for User eXperience*, page 53, New York, NY, USA, 2005. AIGA: American Institute of Graphic Arts. 29, 31

[20] J Rodrigues V Ferreira D Ferreira, M Freitas. **Twitviz - Exploring Twitter network for your interests**. 2009. 31, 76

[21] Petri Hiirsalmi Jari Porras and Ari Valtaoja. **Peer-to-peer communication approach for a mobile environment**. *37th Annual Hawaii International Conference*, 2005. 33

[22] Bishal Raj Karki, Arto Hämäläinen, and Jari Porras. **Social networking on mobile environment**. pages 93–94, 2008. 33, 45

[23] Charlene Li and Josh Bernoff. *Groundswell: Winning in a World Transformed by Social Technologies*. Harvard Business School Press, Cambridge, MA, 2008. 33

[24] Tad Hirsch and John Henry. **TXTmob: text messaging for protest swarms**. pages 1455–1458, 2005. 33

[25] Dennis Muhlestein and SeungJin Lim. **A Common Interest Share Model for On-line Peer Communities**. pages 83–88, 2007. 33

[26] Elizeu Santos-Neto, Matei Ripeanu, and Adriana Iamnitchi. **Content Reuse and Interest Sharing in Tagging Communities**. March 2008. 33, 65, 76

[27] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. **Content-boosted collaborative filtering for improved recommendations**. pages 187–192, 2002. 37

[28] Frank R. Bentley and Crysta J. Metcalf. **Sharing motion information with close family and friends**. pages 1361–1370, 2007. 37

[29] Alexandre Spindler, Michael Grossniklaus, and Moira C. Norrie. **Development Framework for Mobile Social Applications**. pages 275–289, 2009. 37

[30] Scott Counts. **Group-Based Mobile Messaging in Support of the Social Side of Leisure**. *Comput. Supported Coop. Work*, **16**(1-2):75–97, 2007. 37

[31] Hirokazu Tomiyasu, Takuya Maekawa, Takahiro Hara, and Shojiro Nishio. **Social Network Applications using Cellular Phones with E-mail Function**. page 1253, 2005. 37

[32] Scott Counts and Karen E. Fisher. **Mobile Social Networking: An Information Grounds Perspective**. *Hawaii International Conference on System Sciences*, **0**:153, 2008. 37

[33] Alessandro Acquisti and Ralph Gross. **Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook**. *: Privacy Enhancing Technologies*, pages 36–58, 2006. 39, 45

[34] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. **Explaining collaborative filtering recommendations**. pages 241–250, 2000. 47

[35] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. **Methods and metrics for cold-start recommendations**. pages 253–260, 2002. 64

[36] George A. Miller. **WordNet: a lexical database for English**. *Commun. ACM*, **38**(11):39–41, 1995. 65

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from 1. 6. 2009 to 10. 12. 2009 under the supervision of Dipl.-Medieninf. Maximilian Walther at Informatics Faculty, TU Dresden.

Dresden,