User Behavior Profiling using Smartphones

Master Thesis

Onur Mat

August 26, 2011

Advisor: **Sacha Trifunovic**
Supervisor: **Prof. Dr. Bernhard Plattner**

Computer Engineering and Networks Laboratory, ETH Zurich

Abstract

Smartphone applications contemplate to increase the functionality of the devices in order to improve social and collaborative interactions that allow users to communicate, exhange data, and share opinions. Also, the applications should consider the surroundings (know the current location) of the user in order to provide a better, more personalized service. For future applications, it is crucial to understand properly the social relations and interactions as well as usage patterns of its users.

In this thesis we present **Collect&Trade**, an Android application that extracts smartphone data related to the user behavior. The application extracts data from various kind of data sets such as Contacts, SMS and Phone Logs, Geolocation and Applications. Additionally, we extract available Facebook information. The application was deployed among the members of the Communication Systems Group at ETH in order to test its functionality and collect an initial set of data. Some basic analysis was performed on the collected data.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this project.

First of all, I would like to thank Prof. Dr. Bernhard Plattner for giving me opportunity to work in his group.

I am indebted to Sacha Trifunovic for his supervision, help and endurance. He has been a great advisor and has been very helpful in every manner.

I am obliged to Ajita Gupta who has worked with me on this project. She has been hard-working, helpful and patience during the project.

I would like to thank to the participants of the Test who shared their personal data and gave suggestions.

I am very grateful to my family for their endless support and encouragement.

Finally, I would like to give my special thanks to my love Seda for her patience and support that made me to complete this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Smartphones have become increasingly popular in recent years. The technological improvements and the huge variety of possible smartphone applications greatly enhanced the user experience, fueling this trend. Users aquire more and more applications which allow for collaborative interactions, communication and exchange of data and ideas. Generally, smartphone application integrate more and more a social layer, taking advantage of the direct access to the user and his/her communication partners and social network. Ideally, applications adapt to the user's behavior in order to decrease his/her required interaction and maximize the quality of his/her experience.

Nevertheless, developing user behavior aware applications are not an easy task. It is necessary to analyze and understand the user's smartphone usage patterns and social interactions in order to be able to use them for the user's benefit. In order to classify behavior patters, behavior related data needs to be collected from many users.

For this purpose a data collection application on the users'smartphones is required. Nevertheless, users are usually not willing to give up sensitive data for free. To cope with this, an incentive should be provided, such as winning prizes, a fun game, or any other beneficial service, such as providing the user interesting information about his/her behavior.

1.2 Problem Description

The goal of the project is to develop such a data mining application, that collects and analyzes smartphone data related to users'usage and communication behavior. Since this project overlaps with the semester thesis **Social Data Mining on Smartphones** the the tasks and analysis of both projects are similar. This project can be divided into the following tasks.

The first task is to find a relation between the user's current activity (work, traveling, home or social activity) and the application usage and traffic patterns.

The second task is to analyze the different social contacts of the user. Additionally, we want to correlate the users'phone contacts with their social network on Facebook.

The third task focuses on the communication patterns of the user, such as phone calls, SMS, and Facebook message interaction. We want to examine the relation between the communication patterns with the type of the users'relationships.

In order to achieve these goals a location based game called **Collect&Trade** is developed as an incentive for the user to let us collect data. This game is implemented in an Android application and contains services that collect data in the background. Finally, a field test was performed at the Communication Systems Group at ETH to evaluate the games functionality as well as to gather a test set of data that can be analyzed.

Chapter 2

Related Work

In this chapter three other studies that has similarities with this project will be described in details. The differences of the studies with my project will also be explained.

2.1 Stumbl

Since Facebook is very popular most of the data mining systems like Stumbl target collecting data from Facebook. The goal of Stumbl is to collect social data and user input in order to correlate Facebook interactions with actual social relations.

There is a Facebook application with the same name, Stumbl that collects data automatically as well as periodic user input. Stumbl tries to find answer to three main issues. The first one is *How is the type of the social tie related to the duration and frequency of meetings?* The study shows that the meetings are longer and more often with family members however meetings with colleagues are short and very frequent. Second issue is *How are different relationships related to the rate of communication on Facebook?* As it is expected the highest rate of the communication is between friends and family members. The percentage of communication between colleagues is really low. Final issue is *Are we mostly communicate with the friends that we have strong mobility relations or with the ones that we less likely meet during social life?* The study concludes that the communication between the ones we have strong relations is ten times higher than the normal Facebook friends that we rarely meet.

In this project, we do not only concentrate on Facebook data but we also collect contacts, sms, collocation and communication patterns. In addition to that, Facebook messages are also collected to in order to extend the analyze. Another difference of our project is that since the data collection is automatic and is not depended on user input the data will be more accurate. The data is taken directly from the phone database so there is no chance of user intervention. As a drawback, our application is implemented for the Android so there is a limitation on number of users. However, Stumbl is a Facebook application independent of platform so it has large number of users.

2.2 Device Analyzer

Device Analyzer is an Android application that collects data at background and gives statistics about the phone usage of the user. The data is periodically or manually uploaded to a remote server. The aim of the application is analyze the collected data to use on improvement of smartphones. Also recommendations are given to users in order to use the device more efficient. For example the best mobile connection plan to reduce the cost the user pays every month.

The application extracts information about user location, running processes, CPU usage, GSM Cell IDs and some basic phone data like call duration or sms statistics. Our project combines the data sets that Device Analyzer with more social data like Facebook.

2.3 Social Data Mining on Smartphones

This project is directly connected with our project. An Android application called *SocialMine* is developed in order to collect data from the device and as well as from Facebook social data of the user. The details of the collected data can be seen below:

- Contacts: All the contacts data on the phone
- SMS: Both incoming and outgoing sms
- Geolocation: Coordinates of the user
- Calls: All call history
- Facebook: All Facebook data except the contents

I also worked in this project with Ajita Gupta to develop *SocialMine* application. After the alpha test that we run together, I developed **Collect&Trade** game to be incentive for the user to install the application. According to the test results of *SocialMine* I updated the services, add new data extraction units in order to consume less battery and to be more efficient and accurate on data collection.

Chapter 3

Design

In this chapter, I will explain the design details of my application **Collect&Trade**.

3.1 Overview

The application consists of two major parts one of which is the game and the other one is the background services as shown in Figure 3.1. The game module interacts with the user directly. It is a location based game with the goal of collecting objects while visiting places like restaurants, bars, etc. The objects belong to different sets, depending on the type of place the user visits. The user can trade the objects with other users to complete a set. Also, the game history is displayed in a separate tab to give information about progress of the game.

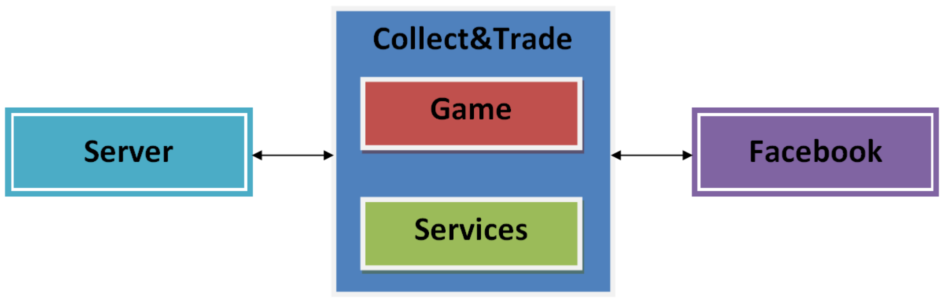

Figure 3.1: General Structure

The background services are designed for data extraction. Since the goal of the project is to find a relation between communication patterns of the user and social interactions, I created different modules for each data set which are **SMS**, **CallLogs**, **Contacts**, **Running Applications**, **Facebook** and **Location**.

In addition to the main application, another important segment of the project is the **Server** which serves as a permenant data storage system. The services collect the data from different kind of modules and then store it on the phone. Then, the service that is responsible for sending data to the Server, establishes connection over WiFi, data is transferred and the local copy on the phone is erased in order to free storage. The service repeats its work periodically unless it exceeds lifetime. Another interaction of the application is with Facebook. In order to get Facebook data of the user, another service establishes connection with Facebook after granting itself by *access token*.

3.2 Architecture

Now I briefly introduce the architectural design of the **Collect&Trade** application shown in Figure 3.2. It consists of 4 parts, a game, data extraction units, data sender and service management.

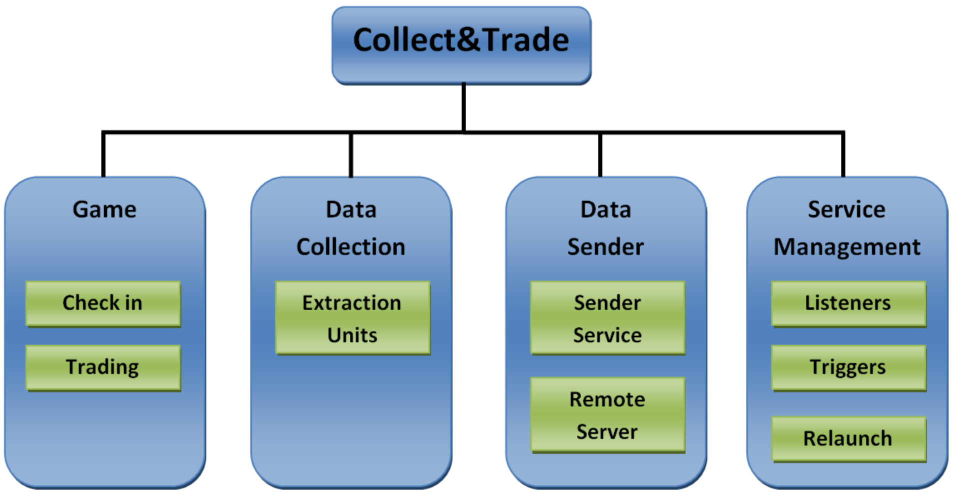

Figure 3.2: Architecture

The **Game** is the unit that interacts with the user. It allows users to check into (register at) places they are currently at, collect items by doing so and trade them with other users. The main interface displays the places in proximity, the collected items, as well as a check in and trade history. The **Data Collection** unit contains several services which extract data from the phone (sms/call logs, contacts, application usage) as well as from Facebook (message/wall-post logs). After extracting the data, the services filter and anonymize the data before storing it in the local phone database. The **Data Sender** is responsible for sending the collected data to a Linux Server and erase the local copy on the phone if the data is successfully sent. The **Service Management** unit is responsible to specific events from the Android operating system. Depending on the type of event, new services are started or some of the running services are stopped. Also, this unit is responsible to to make sure the services keep running in case of an application crash.

Chapter 4

Implementation

This chapter explains detailed information about the architecture of **Collect&Trade**. First I explain the general structure of the application and then I go into more detail about each major unit of **Collect&Trade**.

4.1 General Structure

Collect&Trade is developed in Java platform for the Android Platform 2.2 (API Level: 8)[1]. It contains three activities and ten services. One service communicates with Facebook to collect social data and one service is responsible for connecting data collection database in order to send the data. There are two layers of communication between **Collect&Trade** and server's database. The phones communicate with PHP server and the PHP server has a connection to MySQL database server.

4.2 Game

Collect&Trade is a location based game which serves as an incentive for the user to allow us collect and analyze personal data. The goal of the game is to collect total of 30 objects in 3 sets as fast as possible. By using network connection the location of the user is determined and a list of nearby places is shown to the user. The user gets objects by checking in one of the listed places. In addition, users can trade objects in order to complete a set faster via Bluetooth. So the user increases his/her chance to win the prize which is an additional motivation.

When user installs the application and launches it for the first time a dialog with the license agreement is shown to the user. If he/she accepts, a small manual that clarifies how to play the game is displayed. After skipping the manual, a new dialog appears in order to grant the application access to Facebook data. If the user skips this dialog, it will keepshowing up each time the user enters the game. Finally, a dialog appears to save the email address of the user for notification if he/she wins the prize.

[1] `http://developer.android.com/sdk/android-2.2.html`

After the initialization process the game is ready for playing. When the application starts the location listener is initiated to get the coordinates of the user by using WiFi or Mobile Connection. The main user interface consists of three tabs which are *Places*, *My Collection* and *Game History* explained in the following.

4.2.1 Places Tab

The *Places* tab displays nearby places according to the three categories, Restaurants, Nightlife and Shops. The three categories are assigned to three buttons. When user clicks on a category button the coordinates of the user are sent to **foursquare**[2] to get the information about nearby places for the corresponding category.

When user selects a place *PlaceActivity* is called. In the *PlaceActivity* the user has some information about the place like the distance in meters and the objects that can be received when the user checks in. There are a total of 30 objects equally divided into 3 different sets according to the category of the place. At each place a user can receive an object that is randomly selected of a subset of 5 objects belonging to this category's set. For example, for a place which is in the *Restaurants* category there will be 5 randomly distributed objects which are from the *Food* category of the collection. The available objects for each place is stored in local database to be consistant for future check ins.

In addition, when the user checks in the time of the check in is recorded in the local database. The user is not allowed to check in in the same place on the same day more than once. Another constraint for check in is the range limit. A user cannot be further than 100m from a place in order to be able to check in.

4.2.2 My Collection

The *My Collection* tab displays the types of objects one can collect, as well as how many objects have actually already been collected. There are 3 categories which are **Food**, **Drinks** and **Secret**. Each object category corresponds to a places category, i.e. Food to Restaurants, Drinks to Nightlife, and Shops to Secret. So when user checks in at a Restaurant he/she will get an object only from the Food category. The objects in the Secret category are not visible to the user until they were actually collected.

When the user chooses a category, the objects are shown in a list. If the user clicks on an object, the *ObjectActivity* is started. This activity is designed for trading object among users in order to finish the game faster. Trading is based on **Bluetooth**[3] connections. When two users click the *Trade* button in the *ObjectActivity* a connection is established and the objects are exchanged.

In Android, Bluetooth establishes client-server type of connections by design. One device listens for the connections and the other devices attempts to connect the server device. In order not to annoy the user by requiring him/her to

[2]`http://developer.foursquare.com`
[3]`http://developer.android.com/guide/topics/wireless/bluetooth.html`

choose whether to be the client or the server, in **Collect&Trade**, both devices are designed to be server and client at the same time. When the *Trade* button i clicked, first of all the application checks whether Bluetooth is enabled, enabling it if this is not the case. Then a Bluetooth server socket is created and it starts listening for incoming connection attempts. Additionally, the device is made discoverable so other devices can connect to it. The phone will be discoverable for 300 seconds.

After creating the server and being discoverable, the device starts searching for other phones to connect. The search takes 12 seconds and all the addresses of found devices are recorded. When the search is over, the phone tries to connect to the devices that where found. For each device a client socket is created. A predefined **Universally Unique Identifier UUID**[4] is used for both server and client sockets. This makes it possible to detect and connect to the correct device.

Since both devices are server and client at the same time, the one that attempts to connect first will be the client. Also if a connection is established, other connection attempts and searching devices are all cancelled in order to decrease the load of the *BluetoothManager* which is a special class responsible for Bluetooth connection.

When the connection is established, the name of the object that the user would like to exchange is sent to the other phone. The user is then prompted to approve the exchange of these specific objects. If user confirms the trade then the approval message is sent to the other device and the phone waits for the response. If both users approve the trade then it is succcessfully completed and the *ObjectActivity* is terminated.

Unfortunately, the Bluetooth connection is not reliable and it often crashes because a device is both server as well as searching for other devices in range at the same time. In order to solve this problem an *EmergencyThread* is created. After 60 seconds, if the trade is not completed then the trade is cancelled and all valid or invalid Bluetooth connections are cancelled. Then, a dialog is displayed to the user to retry the connection attempt or terminate the *ObjectActivity*.

4.2.3 Game History

The *Game History* tab shows the progess of the game. It keeps the track of check ins and trades. For each check in or trade it displayes the object that is taken with a timestampt. The user can distinguish trading history from check in history by the font color. The text color of trading history is red.

4.3 Data Collection

The goal of the project is to analyze smartphone usage, communication and mobility patterns of the user. So the Data Collection unit is a central part of the application. There are 10 services in **Collect&Trade** and these services can

[4]`http://developer.android.com/reference/java/util/UUID.html`

be categorized into **Triggered Services** and **Periodic Services** as shown in Figure 4.1. The reasons behind categorizing the services are some of Android's limitations and optimizing the battery and CPU usage.

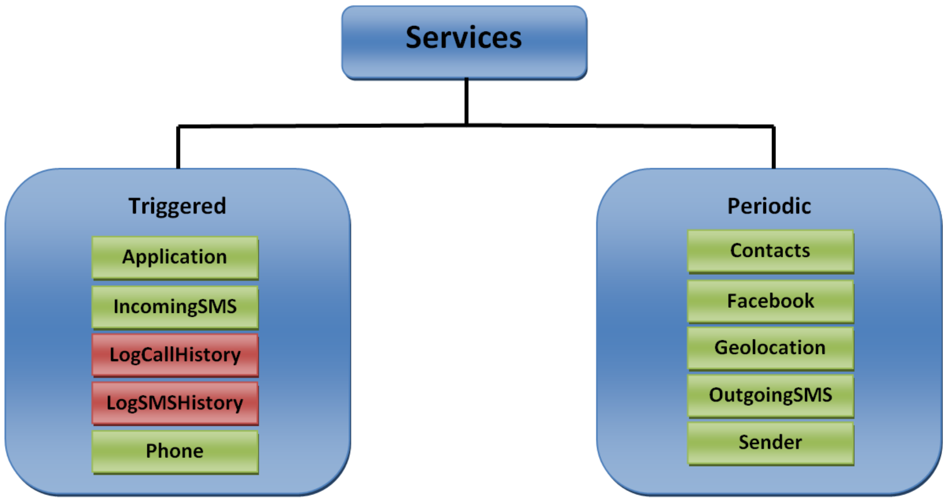

Figure 4.1: Service Categories

For the **Triggered Services**, I create special listeners for certain actions called Broadcast Receiver. This mechanism allows to listen and register callbacks when certain messages are sent by the Android System. Messages could be *SCREEN_ON*, *POWER_CONNECTED* etc. The services in red in Figure 4.1 are killed when they finish their jobs. They collect all the call logs and sms logs that exist on the phone before the installation of the application. However, due to the limitations of Android we do not have this kind of mechanism for certain actions like *SMS_SENT* so it is unevitable to use periodic services.

The **Periodic Services** can also be separated into two groups, one of which keeps track of the latest recoreded data and collects only the newest data. The other one collects all the data whether it is already recorded and sent to server or not. For some of the data I can extract the timestamp to record only the newest part of the data but for some of the data Android does not support such a mechanism. For example, in the Contacts database the system sends message *CONTACTS_DATABASE_CHANGED* but there is no information which contact has been changed or a new contact is added. Also, searching and keeping track of contacts to find out the changes is a very costly operation for the phone. So it is more convenient to record all the entries for some of the data sets.

4.3.1 Application

The *ApplicationService* is a periodic service which runs every minute. Although it runs periodically, this service is activated only when the screen is on. I used a BroadcastReceiver for listening system messages related with the screen called

ScreenReceiver. So if the Android system sends the message *SCREEN_ON*, the *ApplicationService* is initialized and it will keep running until it receives *SCREEN_OFF* message.

By using *ActivityManager*, the service gets the list of current running applications on the phone. The details of the collected data can be listed as:

- Process Name: The package name that is defined by the developer
- Process ID: The unique ID given by Android system
- Timestamp: Current time that application runs

4.3.2 Contacts

The *ContactsService* is one of the periodic services which runs once a day. Although Android system sends a special message called *CONTACTS_DATABASE_CHANGED*, it is impossible to get information about what is changed in the database. System could not distinguish whether a new contact added or a previously saved contact data has been changed. In addition to that, if user enables the synchronization of contacts with Google or Facebook contacts, in each synchronization session Android system sends the message *CONTACTS_DATABASE_CHANGED*. So I decided to make *ContactsService* periodic rather then a triggered service. In each run the service collects all the data related to contacts whether it has been saved and sent to server or not. The details of the data extracted from contacts database is shown below:

- Contact Names
- All Contact Phone Numbers (Mobile, Work, Home, etc.)
- All Email Addresses (Mobile, Work, Home, etc.)
- Notes (Remarks, Link between Social Network Friends & Phone Contacts)

4.3.3 Facebook

Facebook is one of the most important services of the **Collect&Trade** because it extracts most of the social content that we use during the test. Since Facebook is very popular and users communicate through Facebook and share content it is crucial to collect data.

In order to extract data from Facebook, the concent of the user should be taken. Since Facebook SDK uses **Single-Sign-On** as an authentication method if I grant the permission once it will be enough for taking all the data. As explained in section 4.2, after user opens the application a warning is shown in order to grant permission. If user agrees on that, a GUI is displayed for user to login to Facebook. When user gives permission, an *access_token* is sent to the application as a response from Facebook. After that, the *access_token* is saved in order to use afterwards. The *access_token* has a lifetime of couple hours as a default but there is a special permission called **offline_access** to extend the lifetime and this permission is included in the application in order not to bother user each time for accessing personal data on Facebook.

There are two different APIs in Facebook. The new **Graph API**[5] provides

[5]`http://developers.facebook.com/docs/api/`

a simple and convenient view of Facebook data and the relations. **Graph API** is used in most of the cases but since the Facebook has not been finished transferring every functionality to this API, older version called **REST API**[6] is used for extracting some of the data sets. For example, if **Graph API** is used to extract messages, it will return only last 25 messages. But if **REST API** is used, this limit can be set to any value to get all the messages.

The *FBService* periodically collects data with HTTP requests in every 4 hours. All data except Facebook messages is saved again in every session because the changes in Facebook data cannot be determined. However, for messages since there is timestamp for each message the timestamp of last message is recorded and for the next session the messages only after this timestamp are taken with using a special variable in **REST API** called *"since"*. Since most of the Facebook data is coming from messages, it reduces the work of *FBService*. The details of the collected data is shown below:

- User Profile: General Information, Work & Education History
- Friends: Name, ID
- Common Friends: ID
- Messages: ID, Participants, Replies, Timestamps
- Wallposts: ID, Likes, Comments, Timestamps
- Hobbies: Activities, Interests, Books, Music, Movies - with respective Timestamps, ID and Categories
- Pokes: Poker, Pokee, Timestamp
- Likes: Name, Category, Timestamp
- Groups: Name, Group-Position
- Events: Name, Timestamps, Location

4.3.4 Geolocation

The *GeolocationService* collects location and available network information of the phone with a period of 2 minutes. It uses WiFi or Mobile Connection to locate user over internet. Latitude, longitude and accuracy of the location is recorded.

In addition to location information, I get the base station information of the phone. By using *TelephonyManager*, I extract id of the current cell tower called Cell-ID and the Local Area Code of that cell tower. Since there is no information about the accuracy of the location, it is hard to predict the exact location of the user but this information is useful when there is no internet connection available to locate the user.

Finally, if WiFi is enabled, I get the latest successful WiFi scan results by using *WifiManager*. The Basic Service Set Identifiers (BSSID) and the signal

[6] `https://developers.facebook.com/docs/reference/rest/`

strength is saved. As in cell tower case, this information allows us to discover the location of the user but it will not be so accurate. The details of data collected by location service is shown below:

- Coordinates: Latitude, Longitude, Accuracy, Timestamp
- Cell Tower Information: Cell-ID (Unique ID of current cell tower), LAC (GSM Local Area Code), Timestamp
- WiFi Networks: BSSID (MAC address of access point), Signal Strenght Level, Timestamp

4.3.5 SMS

The SMS service contains 3 different services which are *LogSMSHistory*, *IncomingSMSService* and *OutgoingSMSService*. First two services are triggered services and the remaining one is periodic service.

The *LogSMSHistory* service is a triggered service which runs only when the device is connected to a power supply. The *PowerReceiver* is a Broadcast Receiver which listens for *POWER_CONNECTED* and *POWER_DISCONNECTED* messages. When it receives connected message, it starts the *LogSMSHistory* service. This service collects all the incoming and outgoing sms that exists on the phone even before the installation of the application. This service is designed as triggered in order to save battery because it will take too much time to collect all the messages if the sms database has considerable amount of data. So when the receiver gets a disconnected messages it stops the service. The unique ID of the last saved message is stored locally in the phone in order to continue extracting data when the phone is connected to power supply again. Once it collects all the SMS on the phone it saves a variable to local database and then it will be never started again.

The *IncomingSMSService* is also a triggered service. Another Broadcast Receiver called *SMSReceiver* is assigned to this service. When SMS is received, Android system sends *SMS_RECEIVED* message and after that the broadcast receiver starts the *IncomingSMSService*. The service gets the newest SMS from database and extract data.

The *OutgoingService* is a periodic service that runs in every 4 hours. This service is started if the *LogSMSHistory* service finishes collecting all the sms'. The service extracts only the newest outgoing sms. The details of extracted data is shown in below:

- OutgoingSMS: Phone number of Recipient, Message Length in characters, Timestamp
- IncomingSMS: Phone number of Sender, Message Length in characters, Timestamp

4.3.6 Phone

The Phone service is a combination of 2 services, *LogCallHistory* and *PhoneService*. Both services are triggered.

As in the *LogSMSHistory*, the *PowerReceiver* is used to initialize the *LogCallHistory*. This service collects all the call history of the phone until the receiver stops it. The unique ID of each call log is saved in order to continue from where it was before. The service will never run again if it records all the call log.

Another receiver in this module is *PhoneServiceReceiver* that listens for messages related with calling like *STATE_IDLE*, *STATE_RINGING* etc. After that it starts the *PhoneService* and this service extracts the newest entry in call logs and record it to our database. The details of the information that is collected is shown below:

- Calls: Phone number of Caller/Callee, Name of the Caller/Callee, Timestamp, Call Duration, Call Type

4.4 Data Sender

The Data Collection unit extracts all the data that is needed. In order to free the database and analyze the data, it should be sent to the server. There are two important components of data sending, *SenderService* and *Remote Server*.

4.4.1 Sender Service

SenderService sends all available data to remote server in every 4 hours. In every session the service checks if WiFi connection is available or not. If it is, then sending process begins.

Android does not have any mechanism to communicate directly with a remote database. In order to send the data I need an intermediary to communicate between phones and remote database. So PHP server is used as an intermediate layer between phones and remote database. For each table of phone database a PHP file is created to communicate directly. The phone should authenticate itself to PHP server in order to send data. There is a pre-shared authentication key both in phones and in PHP server so only the phones can communicate with the PHP server. After the authentication the data is sent to PHP server row by row. If PHP server sends data to remote database successfully then a message is returned to the phone to erase the local copy of the data to free storage for new data.

In addition to authentication, for confidentiality reasons the data is encrypted before it is sent with a symmetric 128-bit block encryption scheme called **Advanced Encryption Standard (AES)**[7].

4.4.2 Remote Server

The PHP and MySQL[8] servers are configured in a Linux Server in the server room of CS Group. In MySQL server, there is a predesigned database called

[7]`http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`
[8]`http://www.mysql.com`

SocialMine that contains 38 tables for corresponding data extraction units as shown in Table 4.1.

For security reasons, the port that is used for communicating with the server

Module	**Table (Columns)**
UserData	UserData (IMEI, PhoneNumber, GoogleAccount)
Game	GameData (IMEI, Data)
Applications	ApplicationLogs (IMEI, ApplicationName, ProcessID, Date, Time)
Location	GeoLocation (IMEI, Latitude, Longitude, Date, Time) CellInfo (IMEI, Cell-ID, LAC, Date, Time) WifiInfo (IMEI, BSSID, Strength, Date Time)
Phone	PhoneLogs (IMEI, Date, Time, Name, Phone Number, Duration, Call-Type)
SMS	IncomingSMS (IMEI, Date, Time, Thread-ID, Sender, Recipient, SMS-Length) OutgoingSMS (same as above)
Contacts	ContactName (IMEI, Contact-ID, Name) ContactPhoneDetails (IMEI, ID, Phone Number, Number-Type) ContactEmailDetails (IMEI, ID, Email, Email-Type) ContactNote (IMEI, ID, Note)
Facebook	FBGeneralProfile (IMEI, Name, ID, Gender, Last Update, Birthday, Websites) FBEducationHistory (IMEI, School Name, Education Type, Graduation Year) FBSchoolClasses (IMEI, Class, School Name) FBSchoolConcentration (IMEI, School Name, Concentration) FBClassMates (IMEI, Name, School Name) FBWorkHistory (IMEI, Employer, Description, Start Date, End Date) FBProjects (IMEI, Name, Description, Start Date, End Date, Employer) FBColleagues (IMEI, Name, Employer) FBEvents (IMEI, Name, ID, Starting Time, End Time, Venue) FBGroups (IMEI, Name, ID, Position) FBPokes (IMEI, Recipient, Sender, Time) FBFriends (IMEI, Name, ID) FBLikes (IMEI, Name, ID, Category, Created Time) FBInterests (IMEI, Name, Category, Created Time) FBActivities (IMEI, Name, Category, Created Time) FBMovies (IMEI, Name, Category, Created Time) FBBooks (IMEI, Name, Category, Created Time) FBMusic (IMEI, Name, Category, Created Time) FBMessages (IMEI, Message-ID, Name, Last Update, Total Participants) FBParticipants (IMEI, Name, Message-ID) FBReplies (IMEI, Thread-ID, Replier, Time) FBWallposts (IMEI, Post-ID, Name, Created Time, Updated Time, Type) FBWallComments (IMEI, Post-ID, Name, Created Time, Number of Likes) FBWallLikes (IMEI, Name, Post-ID)

Table 4.1: *SocialMine* Database Structure

is changed to a value rather that 80. All the configuration of PHP and MySQL servers are done according to the specified port number. So the server does not accept any connection from anywhere rather than given port.

In MySQL server, a special user is created that has only INSERT permissions for SocialMine database. The credentials of this user is used in PHP files to communicate with MySQL server. This makes our server more secure since no deletion or any change on the database is allowed for this special user.

4.4.3 Local-Remote Communication

This section explains the whole path of the data that is taken from local phone database and sent to the remote server.

On the phone:
First of all, the data is extracted from the phone database. After that, all of the values that will be sent are concatinated to one string. In order to separate the values in the server side easily, I put "###" string between each value. In addition to that, the authentication key and the unique IMEI is appended to the string. So the final value of the string will be:

sendData = IMEI###authentication_key###value1###value2...;

Now the data is ready for encryption. AES block encyption method is used to encrypt the data. After the encryption, the encrypted data is sent to PHP server to run the corresponding PHP file on the server. So the address of the HTTP request will be:

address = http://SERVER_IP:PORT/filename.php?encrypted=ENCRYPTED_STRING;

Now the phone waits for the response from the PHP server. If the response is 1 this means data is successfully inserted to remote database so it is safe to delete the local copy from the phone. But if the response is 0 then the rest of the sending is cancelled and local copy is not deleted.

PHP Server:
When PHP server receives a requests, first of all it decrypts the text with the preshared keys. After the decryption the string should have the same structure as it had on the phone before sending:

decryptedString = IMEI###authentication_key###value1###value2...;

After that, decryptedString is splitted by using "###" to extract the authentication key and other values. Then the authentication key is checked. If it matches, it is safe to insert values to database. Other values are also extracted and PHP server attempts to connect and send data to MySQL server with credentials of special user that is defined in previous part. Finally, PHP server waits response from MySQL server. If the response is *true*, *1* is sent to the phone and if response is *false*, *0* is sent.

MySQL Server:
If the values are inserted to the database successfully, *true* is sent as a response to the PHP server and *false* is sent otherwise.

4.5 Service Management

In order to make **Collect&Trade** resilient to crashes additional mechanisms are implemented. Since the aim of the project is to collect data continuously so the application should have a structure to restart the services if something goes wrong. The only service that runs continuously with a short interval is Geolocation service. So I put all the listeners for triggering the services into the Geolocation service. To make sure that it is running all the time, I implement a check to the onCreate() method of the main activity. When user opens the application it is checked if Geolocation service is running or not and if not, it is restarted.

In addition, sometimes application does not crash but Android system kills the background services due to the lack of free memory. After the system kills the services, it always reschedule the services to be created again in the future. But the services are only created and they are not restarted. In order to make them restart when they are recreated, I put a special code in to onCreate() methods of the services to force them start. So if the system kills the background service of **Collect&Trade**, they will restart for sure.

Another important issue is to restart the services after phone reboot. *BootUpReceiver* is used to restart the services which are periodic after the reboot. Also, the listeners are initialized for triggered services in *BootUpReceiver*.

4.6 Anonymization

The collected data does not contain any content but in order to find correlation between contacts, SMS and Facebook data it is necessity to record the names, Facebook IDs and phone numbers. Also, the privacy of the user should be protected. So hashing is used to anonymize the private data.

In order to hash the private data a special security class called *MessageDigest*[9] is used. **Secure Hash Algorithm-1 (SHA-1)**[10] is chosen as an algorithm for hashing. So when hashing is applied to the private data, it creates a unique hexadecimal string with the length of 32 characters.

There is a challenge on hashing of phone numbers because they are stored differently in Call logs, SMS database and in Contacts. So there should be some standardization on the representation of the phone numbers. To achieve this, all the special characters like "-" are eliminated from the phone numbers.

Another issue is country codes. When the phone receives a call, the representation of the number is depended on the service provider. Sometimes the country code is added and sometimes it is not added to the number. To solve this problem, the number is standardized in a special method. The details can be seen below:

[9] `http://developer.android.com/reference/java/security/MessageDigest.html`
[10] `http://en.wikipedia.org/wiki/SHA-1`

- Eliminate "-" and empty spaces: 076-123-45-67→0761234567
- If starts with "00": 0041761234567→+41761234567
- If starts with "0": 0761234567→+**41**761234567
- If starts with "+": Do nothing

It is assumed that all the numbers without country code belongs to Switzerland. So "+41" is added to the beginning of the number. For future work, the corresponding country code can be added by analyzing the number.

Chapter 5

Test

This chapter gives an introduction to the first test our Android application, **Collect&Trade**.

5.1 Test Details

Since the project is related to the semester thesis *Social Data Mining on Smartphones*, this test can be counted as the second test because in the first test that is called *Alpha-Test* we only test some of the background services. After implementing the game as an incentive and developing additional data extraction units, it was crucial to test the application again.

In order to attract more people, we assign a prize to the test. So one of the participants that completes the collecting total of 30 objects in 3 different sets will have chance to win a Samsung Galaxy 10.1 tablet.

The application was deployed with an email invitation that can be found in Appendix C.1. The total number of participants in the test is 12 what contains CSG members, project colleagues or some arbitrary people willing to participate.

The test began on 09.08.2011 and ended on 21.08.2011. The application will continue colleting data until the life time of 30 days exceeds.

5.2 Test Data

By examining the data, we can conclude about the efficiency and stability of our application **Collect&Trade** as well as information about participants.

Since in each session of the services a timestamp is added to the database we can compute the running times of the background services of **Collect&Trade**. Services are killed by Android system due to the lack of free memory. Although the services that are killed are rescheduled to restart after some time defined by the system, the data until restarting is lost.

Chapter 6

Data Evaluation

In this chapter, the test results of **Collect&Trade** will be investigated in details. The data of the test was filtered using MySQL commands that is available in Appendix A.2. The software that was used in order to calculate mean, variance or standard deviation values is Microsoft Excel. The plots were drawn by using Microsoft Word.

The collected information can be evaluted in four main areas and the results can be categorized in four social dimensions which are **Contacts**, **Interaction Graph**, **Communication Patterns** and **Phone Usage Graph**.

Finally, the issues about the background services and the game will be investigated in detail. The bugs that occurred during the test and possible solutions will be explained.

6.1 Contacts

This social dimension contains information about the people that the user has connection. The contacts can be categorized into two one of which is phone contact and the other one is Facebook contact.

The average number of phone contacts is 620 and the median is 170. This value is high since one of the candidates has 3852 contacts. So the standard deviation is 1110. Since the number of participants is really low the average values are not accurate. The average number of Facebook contacts is 252 with a standart deviation of 143 and with the median 172. The average value that is displayed by Facebook Statistics is 130 that is lower than the average of our participants. The details can be found in Figure B.1 and Figure B.2.

When Facebook contacts and phone contacts are compared with an average of 81 contacts are common. Due to the hashing of the data, the average value is not reliable. If the names of the same person are different on phone and on Facebook then it will not be counted as a common contact since hashing of the names will be totally different. The result shows that there is a high amout of separation between real contacts and online contacts since to save a person to phone contacts you should have a close relationship then adding him/her on Facebook.

6.2 Interaction Graph

Collect&Trade has four data extraction units for different communication channels which are *Facebook*, *Geolocation*, *Phone* and *SMS*. In order to show the correlation between these four channels we created the **Interaction Graph** shown in Figure 6.1.

Each node represents the communication channel and each arrow shows the interaction between these channels. The numbers inside nodes indicate average number of people that the user communicates. The numbers on the rows represents the average number of overlap between data extraction units.

From the Figure 6.1, we conclude that users communicate 21 different people

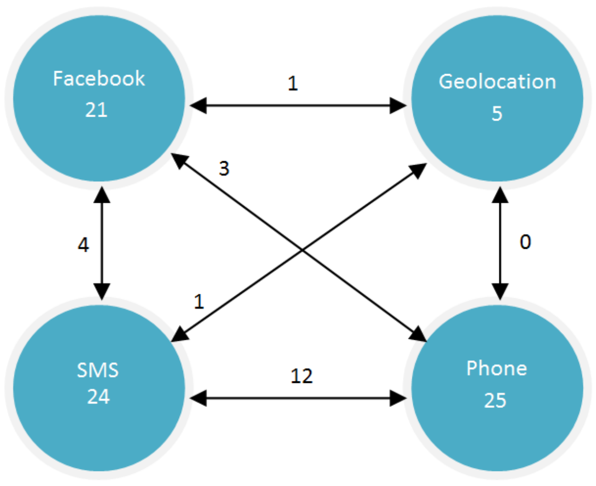

Figure 6.1: Interaction Graph

on Facebook on average. Also, 4 of them are communicated by sending or receiving SMS. Finally, on average users meet 1 person out of 24 to whom he/she communicates with SMS.

These interactions can be used in order to detect the social ties of the users. We conclude that people that the user meets and has contact over phone are close to the participant however, people that user does not meet and communicates only through Facebook are remote friends.

6.3 Communication Patterns

This result contains detailed information about the social and communication behavior of the participant.

By using the data collected by 4 different extraction units, we try to conclude some statistics about the communication patterns of the users. From the results we try to find answers to the following questions:

- Phone: How many calls does the user make per day and what is the average duration?
- SMS: How many SMS are sent and recieved in every day? Is there a relation between lenght of the SMS and social ties of the users?
- Facebook: How does the user communicate on Facebook? Is the communication depended on the level of closeness?
- Location: Where does the user spend his/her time? How often do the participants meet?

6.3.1 Phone

According to the results, on average the participant makes total of 6.4 calls including 1.3 incoming, 3.2 outgoing and 1.9 missed calls. The median values are 1.2, 3 and 1.5 respectively. The average duration of incoming calls is 2 minutes and 55 seconds and median is 2 minutes where as the average duration for outgoing calls is 1 minute and 12 seconds and the median is 1 minute and 2 seconds. The details are in Appendix B.2.1. Finally, Figure 6.2 shows the daily phone call traffic of the users.

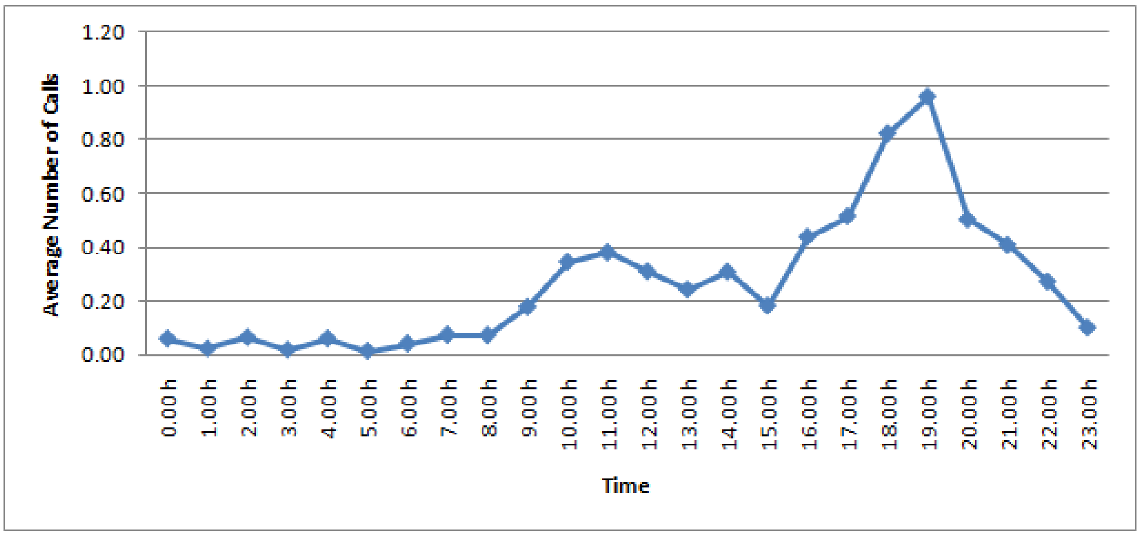

Figure 6.2: Daily Phone Call

6.3.2 SMS

User receives 3.4 SMS per day with an average length of 80.4 characters with the median values of 3.7 SMS with 78.2 characters. Also, user sends 2.2 SMS per day with an average lenght of 71.5 characters. The median values are 2 SMS per day with 67.3. The detailed statistics is shown in Appendix B.2.2. Also, like in Phone Calls a graph is created in order to show daily SMS traffic or users as in Figure 6.3.

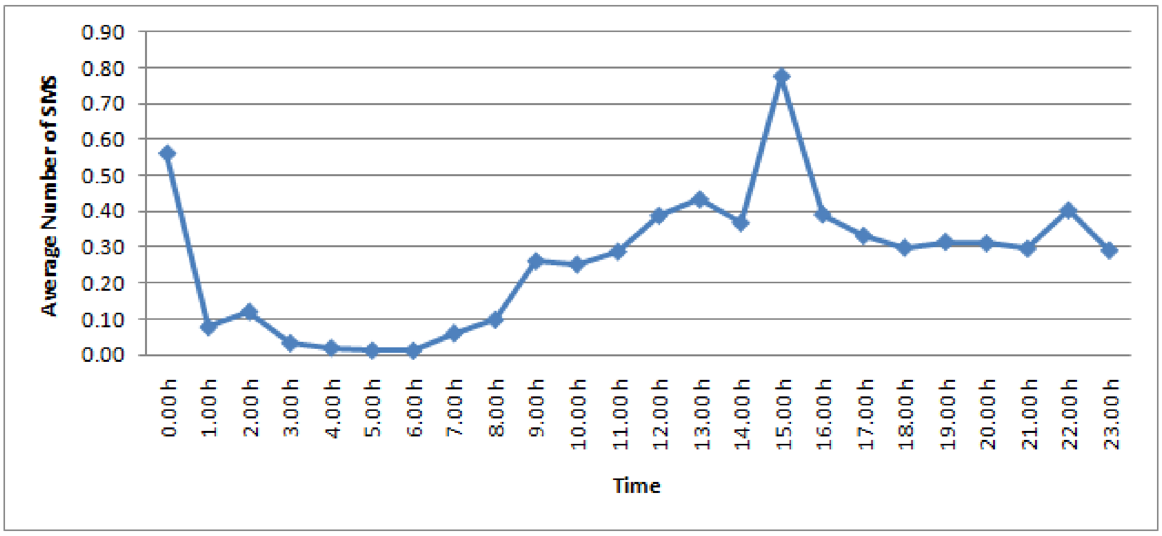

Figure 6.3: Daily SMS

6.3.3 Facebook

After analyzing Facebook data, we conclude that user has an average of 344 Facebook Messages, 19 Wallposts, 16 Likes, 0.5 Pokes, 33 Groups and 1.3 Events that he/she will join. The medians are 118 for Facebook Messages, 21 for Wallposts, 6 for Likes, 0 for Pokes, 21 for Groups and 1 for Events. Since it is not mandatory to login Facebook in order to play the game, we have limited number of participants that logged in to Facebook as shown in Appendix B.2.3. Additionally, a daily Facebook Messages traffic is shown in Figure 6.4.

6.3.4 Location

From the graph shown in Figure B.18, on average users spend 6.9 hours at work, 11.3 hours at home and 1.9 hours for meal. The remaining time is considered as travelling which is 3.9 hours. In addition to that, we analyze the pairing of users. It is assumed that two participants meet if they are in the range of the same WiFi network. As a result, user meets 4.83 colleagues per day and spends 1.23 hours per each colleague. The details can be found in Figure B.19 and B.20.

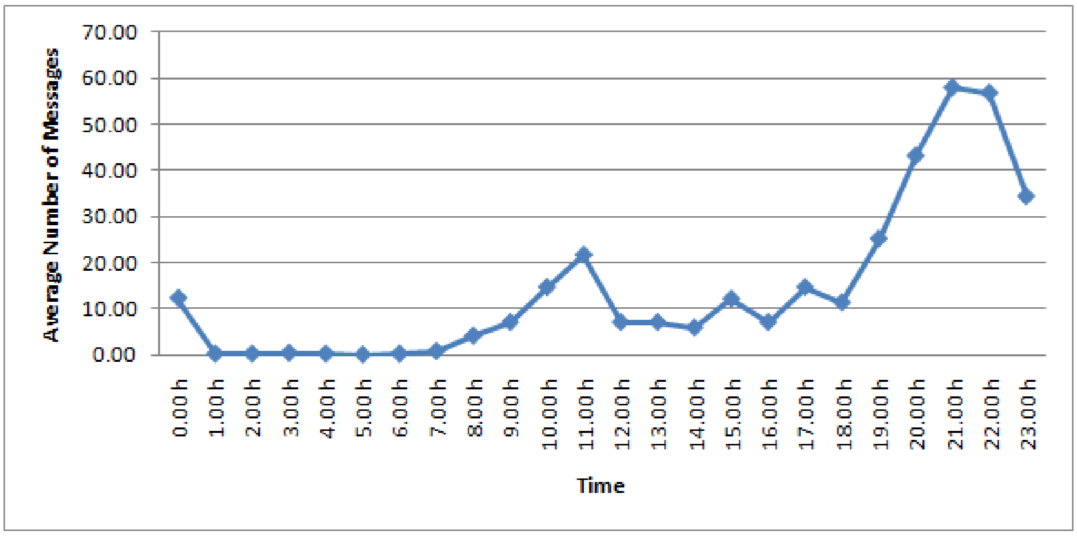

Figure 6.4: Daily Facebook Messages

6.4 Phone Usage Graph

The *ApplicationService* records the running applications when the user turns on the screen and it keeps running until the screen is turned off. So by analyzing the data that the service extracted we can conclude about phone usage behavior of the users.

According to the results participants use their phones for 73.08 minutes per day on average. It is the duration when the screen is on. The details are shown in Figure B.21. In addition to that, we create **Phone Usage Graph** that shows how the participants use their phones during whole day on average. As it is expected, the phone usage increases after work and it is minimum at morning. The details can be seen in Figure 6.5.

6.5 Data Statistics

During 13 days the total amount of the data collected is 71.5 MB for 12 participants. So the size of the data that is 458 KB per participant per day is reasonable. If we take this value as base, for future tests that contains 100 participants with 6 months of test period the size of the data will be 8.3 GB which is acceptable for such a large scale users.

The results show us that maximum amount of data comes from Facebook Mutual Friends table. In this table the names of the common friends of the participant is recorded. Since the names are hashed and each name contains 32 characters, the size becomes so large. The *GeolocationService* extrated second largest amount of data since it records the information about all available WiFi networks in range with a period of 2 minutes. The third largest amout of data belongs to Application Logs. The reason is that, *ApplicationService* records all

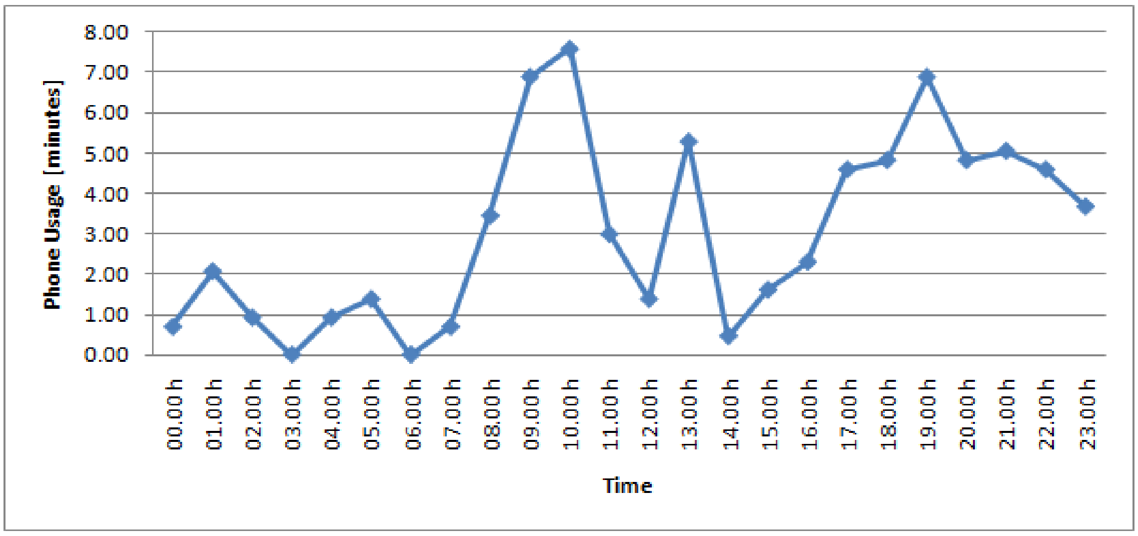

Figure 6.5: Phone Usage

the running processes which includes background and system services. The details about the sizes of the data collected for each service can be seen in Table 6.1.

Data Extraction Unit	**Size [KB]**
Application	64
Contacts	57
Facebook	147
Game	0.35
Geolocation	10
SMS	5
Phone	3

Table 6.1: Amout of Data per User per Day

6.6 Application Evaluation

According to the data that was collected during the test we realized some bugs in the application. We can categorize these bugs into two, **Background Services** and **Game**. The problems and possible solutions will be explained in each section separately.

6.6.1 Background Services

After the test, we realized that some of the background services like *ApplicationService*, *SenderService* have some problems. The bugs will be explained in each service.

Application Service: This service records all the running processes with a period of one minute when the screen is turned on. So it will record all the processes including Android system processes and background services of other third party applications. By examining the data that this service extracted it is not possible to detect which applications are really accessed by the user. The service can not distinguish whether a process is on foreground or not. So it is not possible to detect which applications are used by the participant. The phone has a feature that shows the CPU usage of applications. Also, there are some applications in the market in order to display the CPU usage. So by using different mechanism the actual applications that are used by the participant can be distinguished and recorded. The amount of the data that *ApplicationService* extracted is considerably high. Using another mechanism will also solve this problem.

Sender Service: The *SenderService* is responsible for sending the collected data from phone database to the remote database. It uses only WiFi connection for sending. During the test we realized that if WiFi connection is lost during the sending session, the service keeps running and tries to send the data. Since it sends row by row for each row the service waits for timeout error. So the service keeps running for hours and this leads to increase on battery consumption. In order to solve this problem, a *BroadcastReceiver* can be created for listening Android system messages telling that WiFi connection is lost. After that the *SenderService* is killed in order to save battery. Moreover, if WiFi is not available in one session of the *SenderService*, it should not wait another 4 hours to try sending again. It should listen for WiFi connected message and when it receives it should immediately start sending data.

6.6.2 Game

If the Android phone is rooted[1] the user can have access to the database of applications. Since the database of **Collect&Trade** is not encrypted the data can be accessed by the user. In addition, the user can manipulate the database by inserting valid data to the tables. Moreover, he/she can insert check in data in finish the game faster and win the prize. So the database should be encrypted in order to prevent fake data.

Trading via Bluetooth is not reliable on our game. Due to the work load of *BluetoothManager* it crashes frequently. There are two types of Bluetooth connection in Android, secure and insecure. Secure connection is not as resilient as insecure connection to the failures. Although, insecure connection is more stable it is implemented after the Android version 2.2[2]. So in order to use insecure connection to be more reliable the operating system of the phones should be updated recently.

[1] `http://en.wikipedia.org/wiki/Rooting_(Android_OS)`
[2] `http://developer.android.com/sdk/android-2.2.html`

Chapter 7

Discussion

This chapter explains the issues that are confronted during the project, *Limitations*, *Privacy* and *Incentive*.

7.1 Limitations

First of all, **Collect&Trade** is implemented for Android phones. Recent researches show that share of Android in global smartphone market is increasing. According to one of the most popular IT research company, Gartner the share of Android is 22.7% for 2010[1]. *Symbian* is dominant in smartphone market by the end of last year. Due to this drawback, it is hard to find participants for the test. For our test, almost all the partipants were work colleagues so it is hard to conclude precise statistics.

Another issue about the Android is the background service management. There is not any option to give priority to the background services in order for system not to kill the service in case of lack of free memory. So if the amount of free memory decreases the Android system kills the background services. This causes to data loss which induces to insufficient data for reliable analysis.

7.2 Privacy

Smartphones contains considerable amount of personal data like emails, contacts, sms, locations, credentials etc. The applications can extract the private data of the user without user concent like in one of the famous IPhone games called *Angry Birds*. The application collects the location data of the user with IMEI and sends to marketing companies. The user is not aware of this issue [2].

Social Data Mining Systems are built on the purpose of collecting private data. As stated in article *Privacy Preserving Data Mining* [3], the main issue is confidentiality of the private data. The goal of data mining systems is to give recommendations to user and be beneficial to the user. **Collect&Trade** has similar aim that is to determine the social relations and mobility patterns of the user.

The user should be aware of the collected data for any application. The developers should state exactly what type of private data will be collected and how the data will be used. After that, the decision of sharing the data belongs to the user.

7.3 Incentive

The main goal of the project is to extract the personal data of the users for analysis purposes. However, people are not willing to share their data unless they have benefits. So in order to have large set of participants there should be an incentive for users to run the application.

Designing an incentive is not easy process since you should come up with an original idea because if you develop an application that is similar to the ones that are already in market then people will not choose to use your application. Additionally, it will be more attractive for the users if they gain some valuable things by using application like in *foursquare*. If you are a mayor of a place, then you can have discounts in shops, restaurants or bars. So we added prize to our application to attract more people. We gave a Samsung Tablet one of users who completed all the objects.

Incentive or prize is not sufficient for having large number of participants. It would be a good option if users can recruit their friends to the game by sending invitation. Also, you can provide an incentive to the users to invite as well. This will increase the competition among users and the number of the users will increase. So the analysis will be more precise.

Chapter 8

Future Work

This chapter explains in detail how **Collect&Trade** can be improved in the future to be more advanced. Since developing a social data mining system is an iterative process and this is the first version of the application, the project can be updated in three main areas.

Game:

In the game, the objects and the places has a relation as discussed in section 4.2.2. However the objects in one category is randomly assigned to the corresponding place. So user can get a *Pizza* object from a restaurant that has no Pizza in their menu or user can get *Doner* object from McDonald's. As a future work, this problem can be solved by creating a separate database for objects and places by analyzing the detailed information about the places.

Another upgrade can be done on trading. The trading is based on Bluetooth but it is not reliable and crashes frequently. Since the devices acts as both server and client this gives heavy load to *BluetoothManager*. Moreover, the device is searching for other devices at the same time so it could not manage this much load. As a solution, being server or client can be randomized to decrease the working load of the manager. So trading will be more reliable.

There is a prize at the end of the game. So if the users can see the scores of other users this will increase the competition and will make the game more fun for the users. We can create a remote database and send the scores to the server. So the devices update the local copy from the server in order to show top list to the user.

Finally, the incentive is not sufficient to attract more people. An option should be added to the incentive for users to invite their friends. This will increase the competition among users and increase the number of participants.

Data Extraction:

In the application *ContactsService* collects all the contacts data with a period of 24 hours whether the database is changed or not. So this service can be modified in such a way to detect the changes and run only if something changes on the contacts database to decrease the run time and amount of data that is sent. In addition to that, some of the smartphones has a special mechanism to merge contacts from different types like Facebook, Gmail or real SIM contacts. If this information could be extracted, it will make easier to relate the contacts.

Facebook service should be upgraded like above to detect the changes and run only if something changes on the Facebook data. This will reduce the amount of data that is collected because most of the data comes from Facebook. Another solution would be to move the Facebook data extraction unit to server side. Since the data is extracted from Facebook by HTTP requests and with *access_token*, this work can be done on server side by just passing the token to server side.

Sender Service:

In the *SenderService* the data is sent to the server row by row and for each row the data is encrypted on the phone, decrypted on the PHP server and after the authentication the data is recorded to remote database. So this confidentiality and authentication for each row make the process to run slower. The sending can be modified in such a way that the data is sent table by table or whole database can be sent to the server with only one session in order to decrease the working load of the service.

For sending data to the remote server, only WiFi is used in order not to annoy the user by cost of the network traffic. However, if the time periods that WiFi is avaliable do not overlap with the period of *SenderService* then the data can not be sent to the server. So one option can be added to the menu to allow user to select between Mobile Connection (3G or EDGE) and WiFi. So for sending, the service will not wait for the WiFi connection. Another solution could be to create a *BroadcastReceiver* to listen for the message that Android system will send when WiFi is connected.

If the issues that are mentioned above are added to **Collect&Trade**, the application will be more reliable and this will lead to a larger scale of users which install the application and let us to collect their personal data. So the statistics that is concluded will be more accurate.

Chapter 9

Conclusion

We presented an Android application called **Collect&Trade**, a Social Data Mining System for smartphones. The applications collects social and mobile data by six data extraction units which are *Application*, *Contacts*, *Facebook*, *Geolocation*, *SMS* and *Phone*.

We implemented a location based game to be an incentive for user to install the application and allow us to analyze personal data. After the implementation we deployed the application and run a test among 12 participants for 13 days.

The **Contacts** shows the relation between the real contacts of the user and his/her online contacts. As a result, user has strong social ties with the ones he/she communicates over phone than the ones he/she is connected over Facebook.

Using **Interaction Graph** we can conclude the relationship of the user with the corresponding participant. The analysis is based on the interactions over four different communication patterns which are Phone, SMS, Geolocation and Facebook.

Analyzing the records that is collected with data extraction units, we have concluded the analysis in four different graphs in order to expose user behavior and mobility patterns. The general statistics of the users like the average number of calls per day, the lenght of the sms, information about where the users spend their time or Facebook usage are shown in **Communication Patterns**. By using the results, we can conclude the level of social ties for each person that user communicated.

In the **Phone Usage Graph** we give the active phone usage results. Since we record the screen on/off times, we conclude a daily phone usage schedule. This gives us information about when the users actively use their smartphones and which purposes.

Social Data Mining on smartphones is in its early stages but will be increasingly important to research.

Appendix A

Collect&Trade Instruction Manual

A.1 Running the Application

In order to run the application with apk file, the application should be signed on Eclipse. The crendentials of signing are shown in Table A.1.
In order to install the application, you should enable the option **Unknown**

Existing keystore	debug.keystore
Password	*android*
Alias	androiddebugkey
Password	*android*

Table A.1: Signature Credentials

Resources from the Application Settings.

A.2 Linux Server Configuration

First of all, PHP and MySQL Servers should be installed to the linux server by running following commands:

- sudo apt-get install apache2
- sudo apt-get install php5-mysql
- sudo apt-get install libapache2-mod-php5
- sudo apt-get install mysql-server

Since the received data is encrypted and should be decrypted, *mcrypt* module should be installed for php5 with the command:

- sudo apt-get install php5-mcrypt

During the installation of MySQL server, it will ask for *root* password. You should provide a strong password for user *root* for security reasons.

After installing both servers, a new site should be created at location "/etc/apache2/sites-available". The default site can be copied with a different name and then the site should be configured for listening a specific port number rather than 80 like below:

- sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-available/mysite
- sudo nano /etc/apache2/sites-available/mysite
- Change 80 to PORT_NUMBER
- sudo nano /etc/apache2/port.conf
- Change 80 to PORT_NUMBER

Now the PHP Server is ready for mysite configuration to listen connections from port PORT_NUMBER. The site should be enabled and the server should be started by following commands:

- sudo a2ensite mynewsite
- sudo /etc/init.d/apache2 restart

In order to check if the server is running or not open a browser and type: *http://SERVER_IP:PORT_NUMBER*. You should see a web page telling you that the server is running. After that, the PHP files should be copied to *"/var/www/"* and the database should be created in MySQL by using the script **socialmine.sql** that is available at repository.

Both servers are ready for receiving the data. After configuring the IP and Port number of the server in the application, **Collect&Trade** can send personal data to remote server.

The data is filtered using MySQL commands that are shown in Table A.2

```
\\Create Database \\
create database MYDATABASE;

\\Select Database \\
use MYDATABASE;

\\Delete Database \\
drop database MYDATABASE;

\\Create Table With Two Columns (First with String-, Second with Number-Value) \\
create table MYTABLE(id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
MYCOLUMN1 VARCHAR(30), MYCOLUMN2 INT)")

\\Show All Tables In Selected Database \\
show tables;

\\Insert String-Value Into Table Column \\
insert into 'MYTABLE' ('MYCOLUMN') values('"MYSTRINGVALUE"');

\\View Table \\
select * from MYTABLE;

\\Filter Table To Show Specific Column \\
select MYCOLUMN from MYTABLE;

\\Filter Table To Show Distinct Column-Values \\
select distinct MYCOLUMN from MYTABLE;

\\Delete Entries Inside Table \\
delete from MYTABLE;

\\Delete Table \\
drop table MYTABLE;
```

Table A.2: MySQL Commands

Appendix B

Test

B.1 Contacts Plots

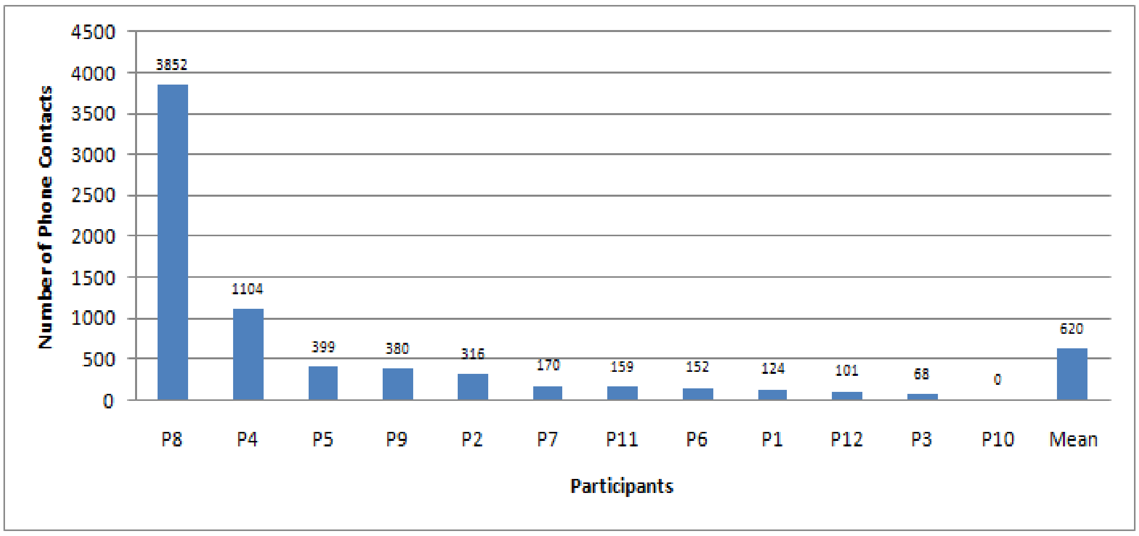

Figure B.1: Phone Contacts Graph

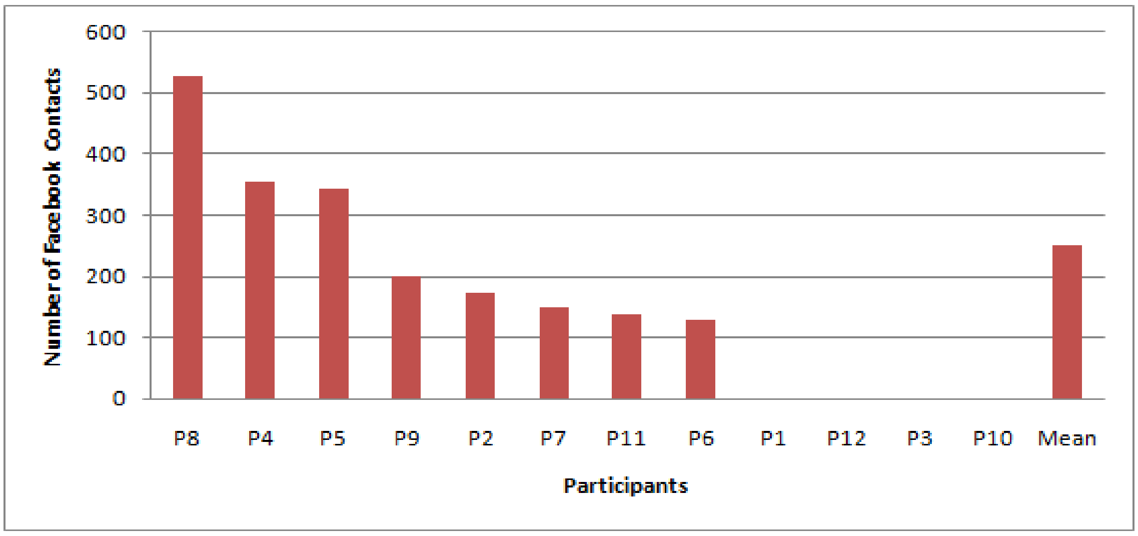

Figure B.2: Facebook Contacts Graph

B.2 Communication Patterns Plots

B.2.1 Phone

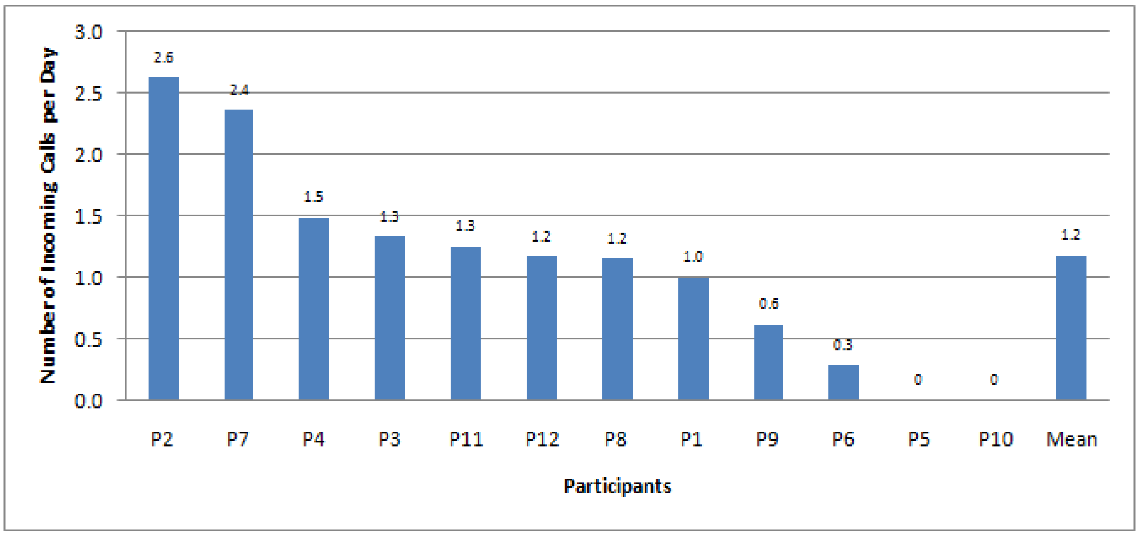

Figure B.3: Incoming Calls per Day

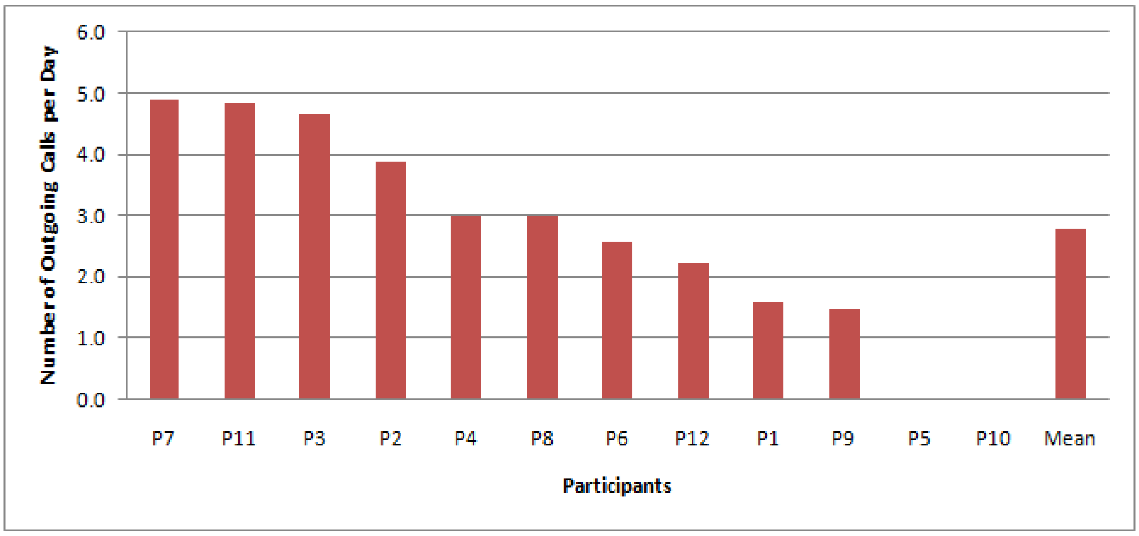

Figure B.4: Outgoing Calls per Day

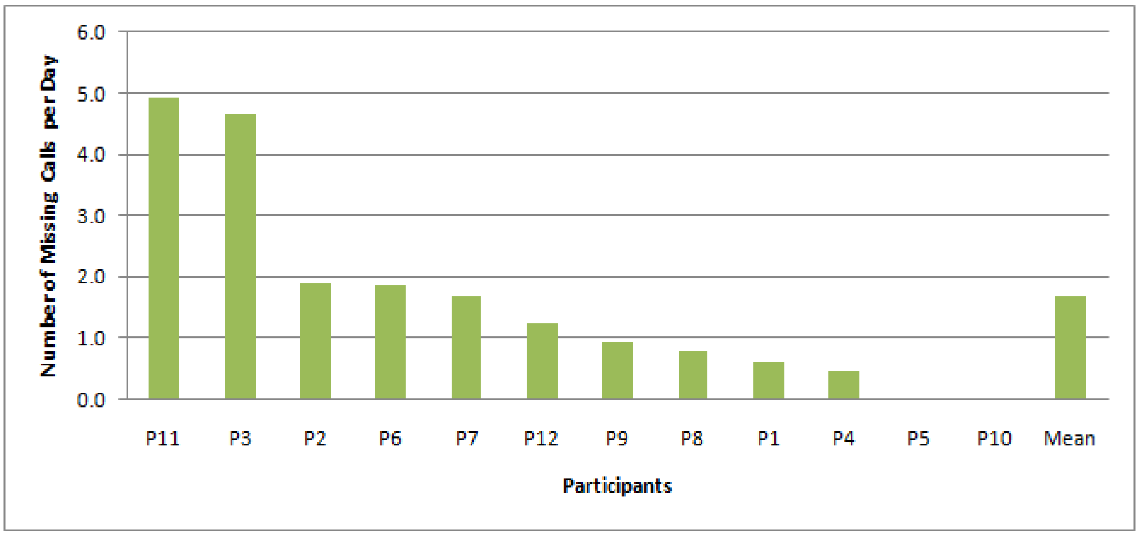

Figure B.5: Missed Calls per Day

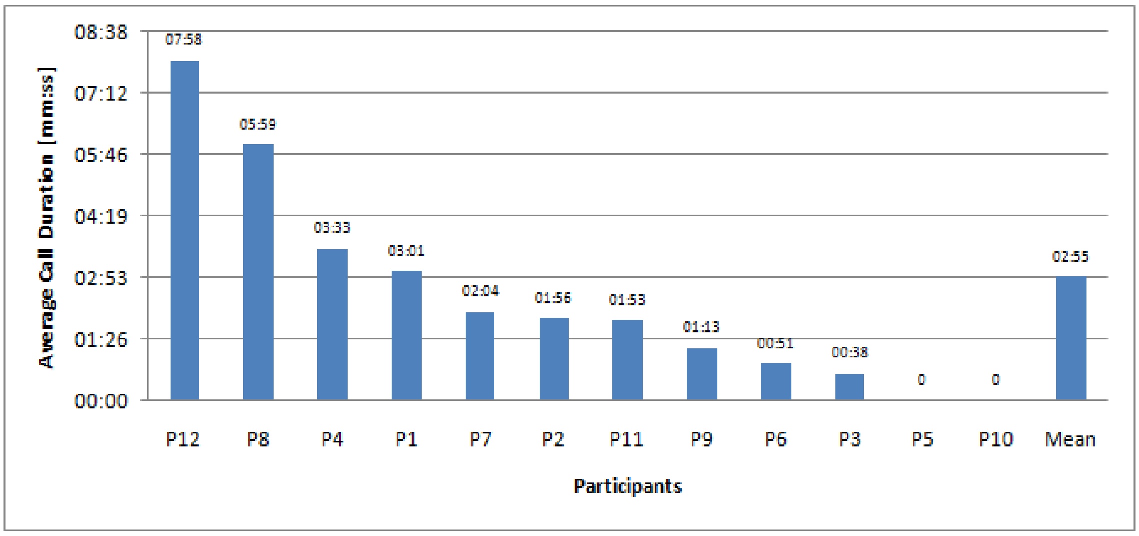

Figure B.6: Incoming Call Duration

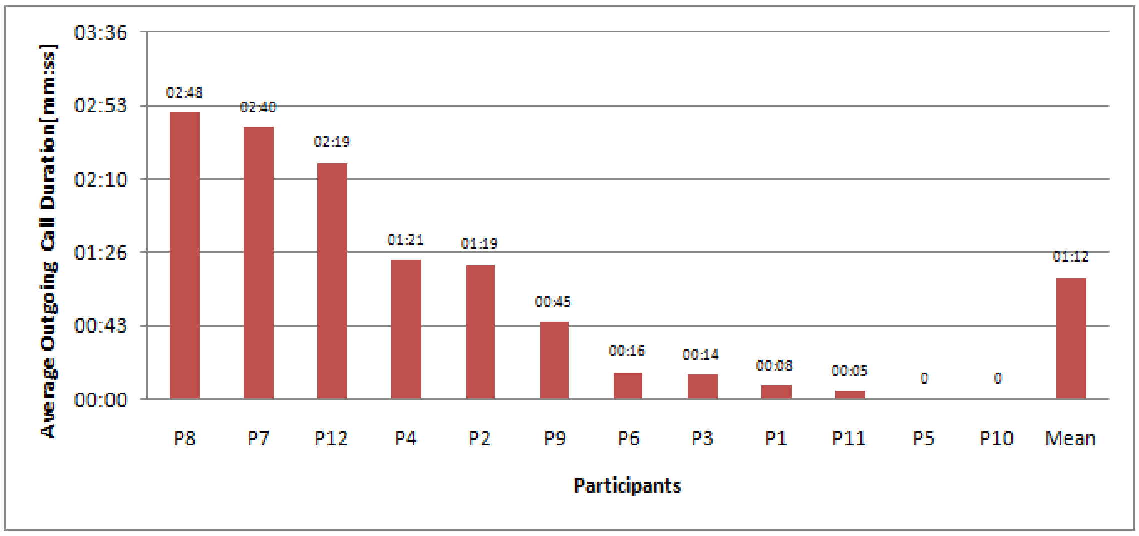

Figure B.7: Outgoing Call Duration

B.2.2 SMS

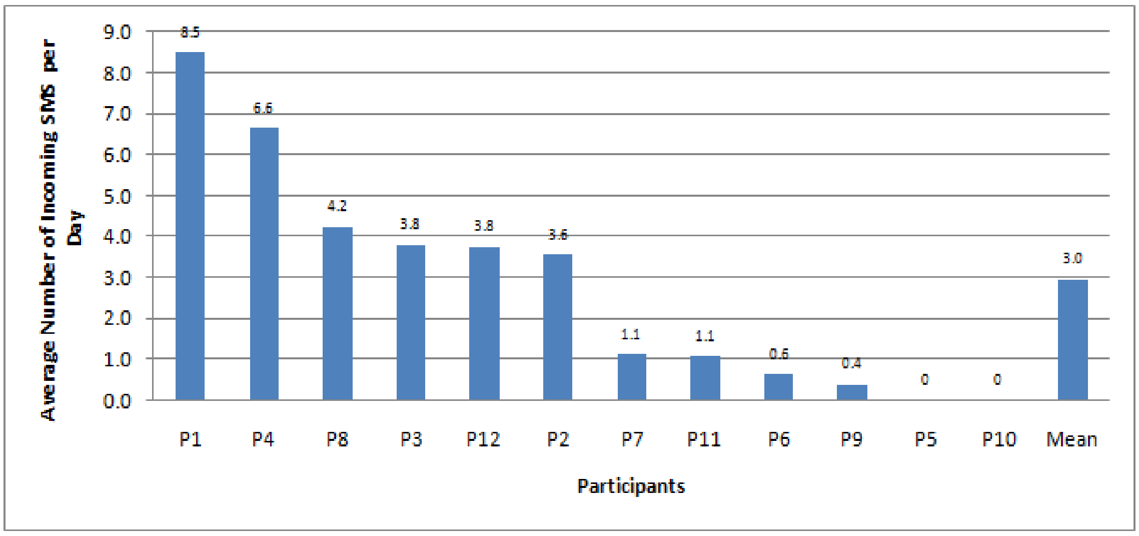

Figure B.8: Incoming SMS per Day

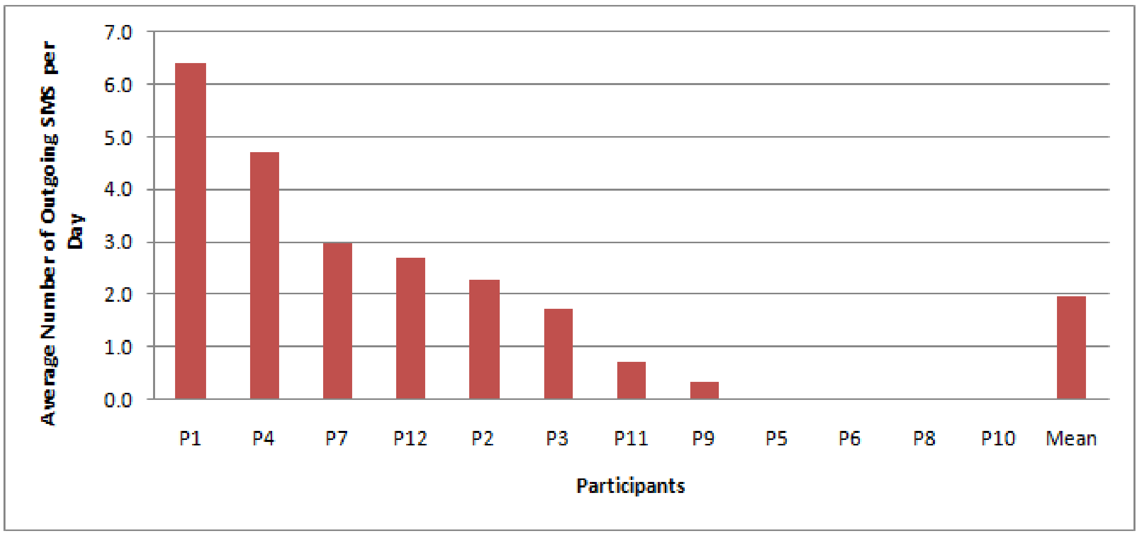

Figure B.9: Outgoing SMS per Day

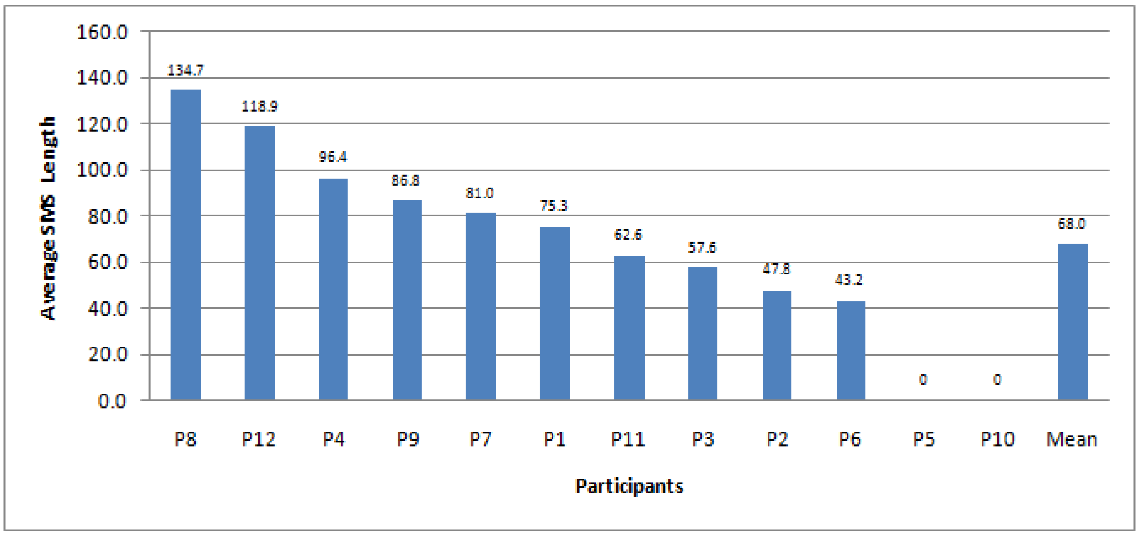

Figure B.10: Incoming SMS Length

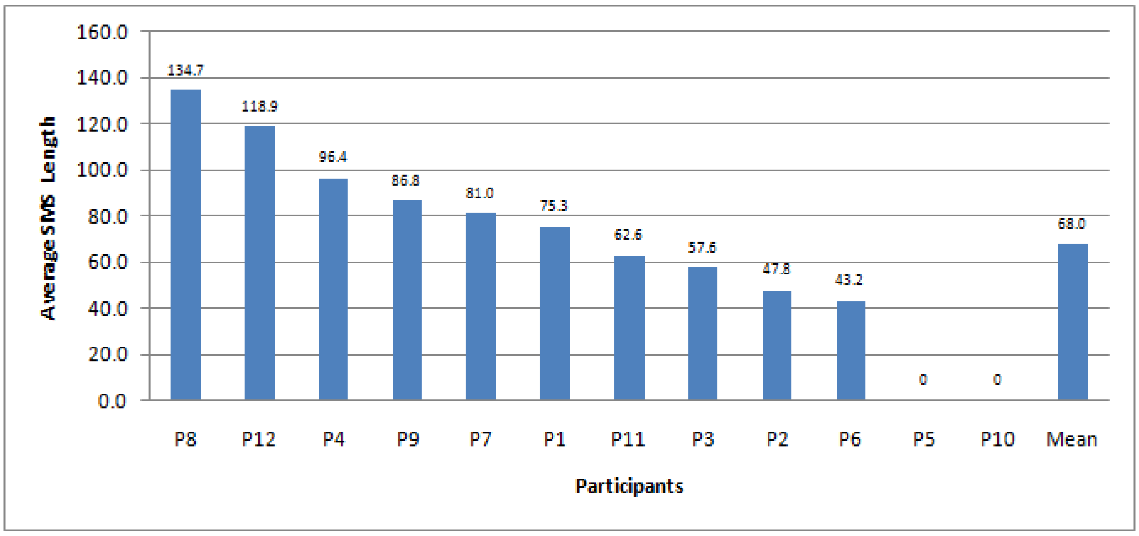

Figure B.11: Outgoing SMS Length

B.2.3 Facebook

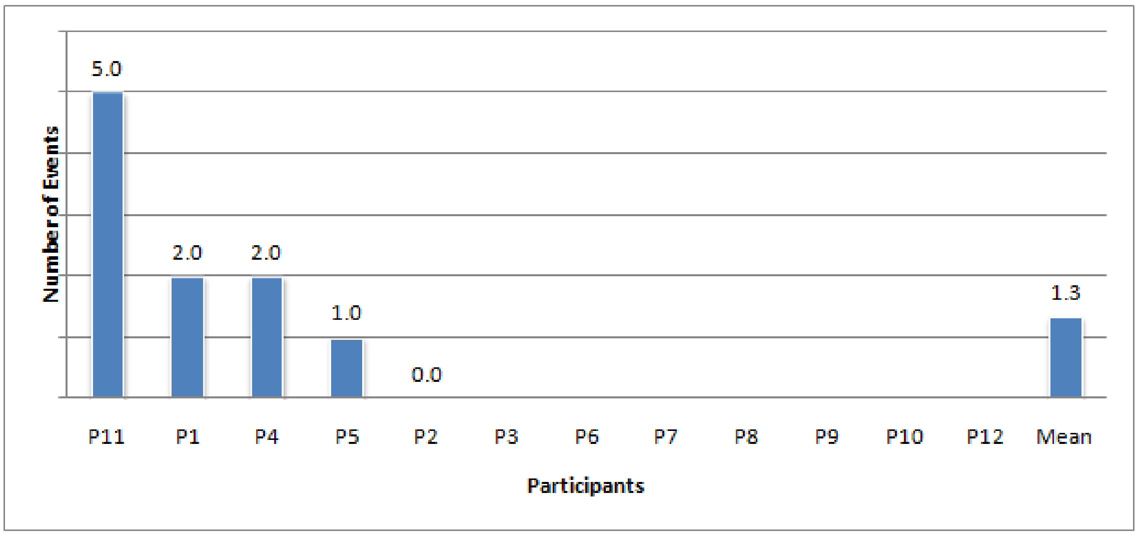

Figure B.12: Facebook Events

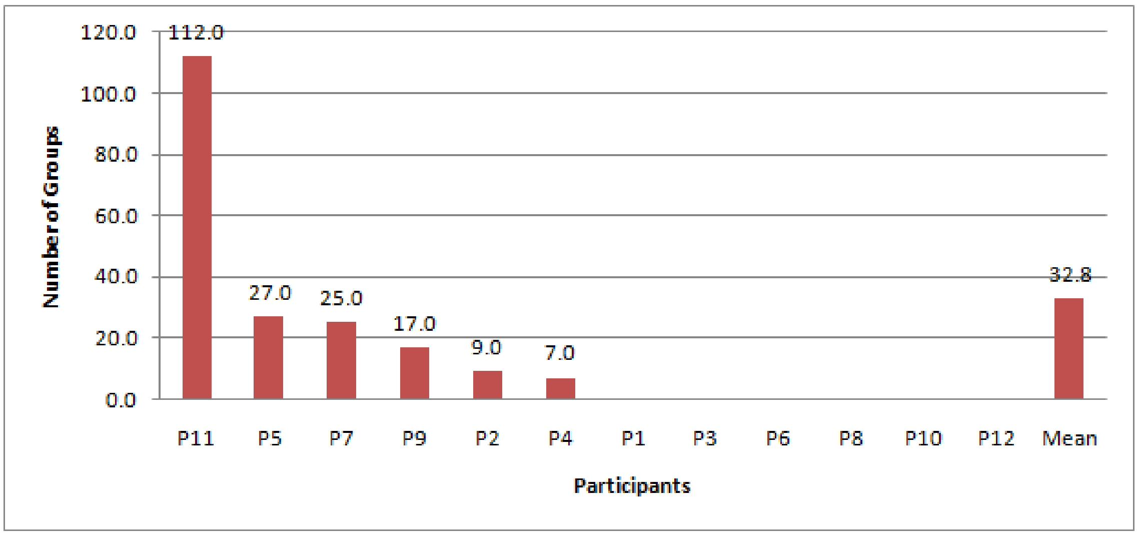

Figure B.13: Facebook Groups

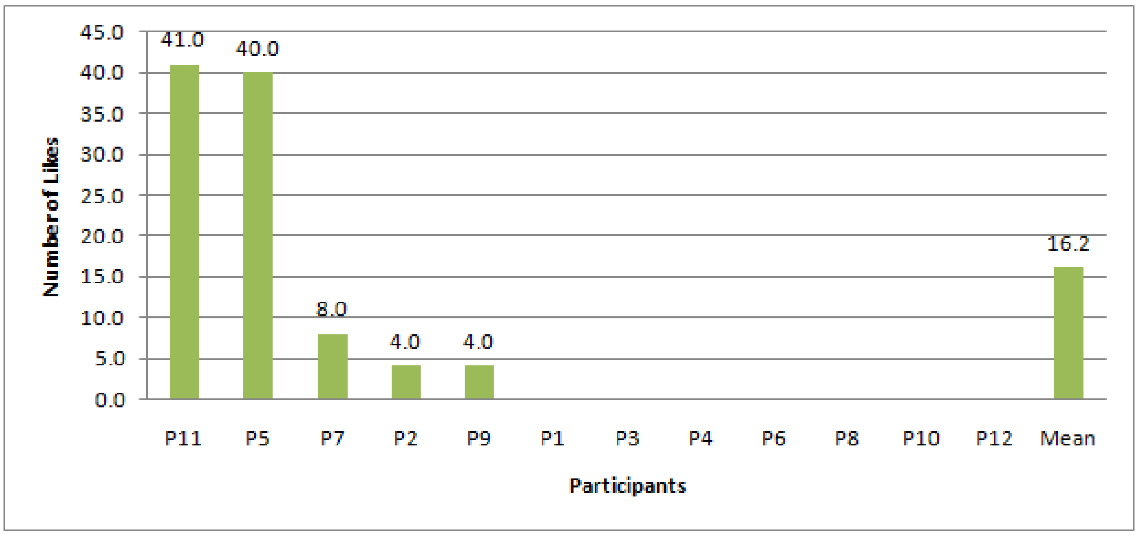

Figure B.14: Facebook Likes

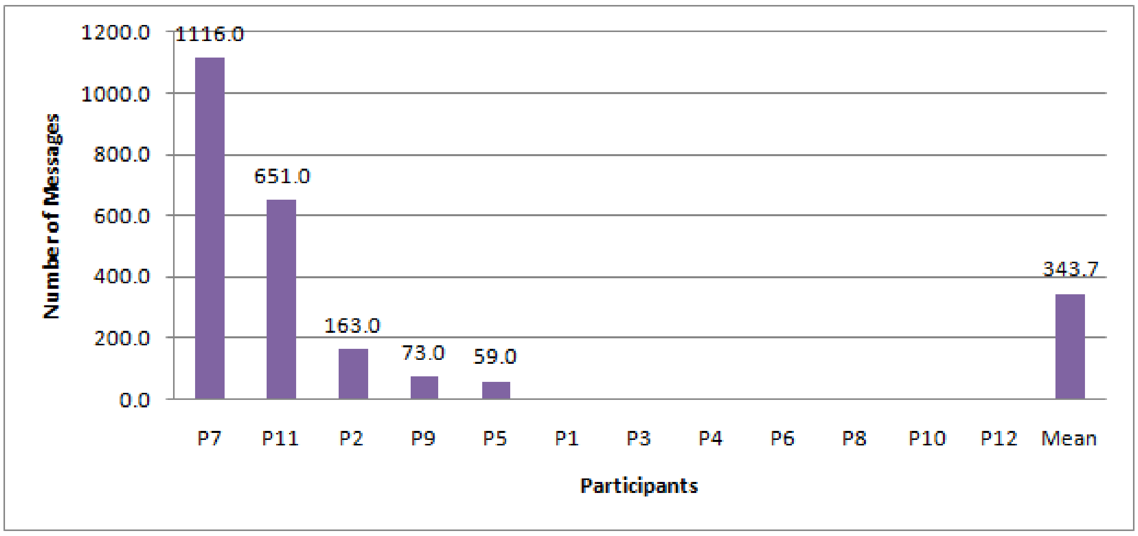

Figure B.15: Facebook Messages

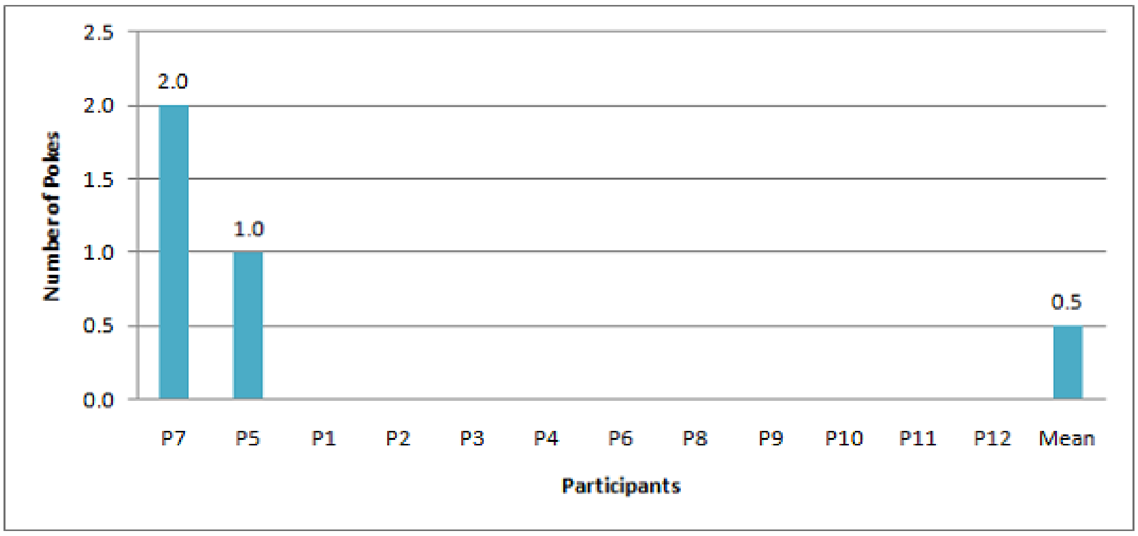

Figure B.16: Facebook Pokes

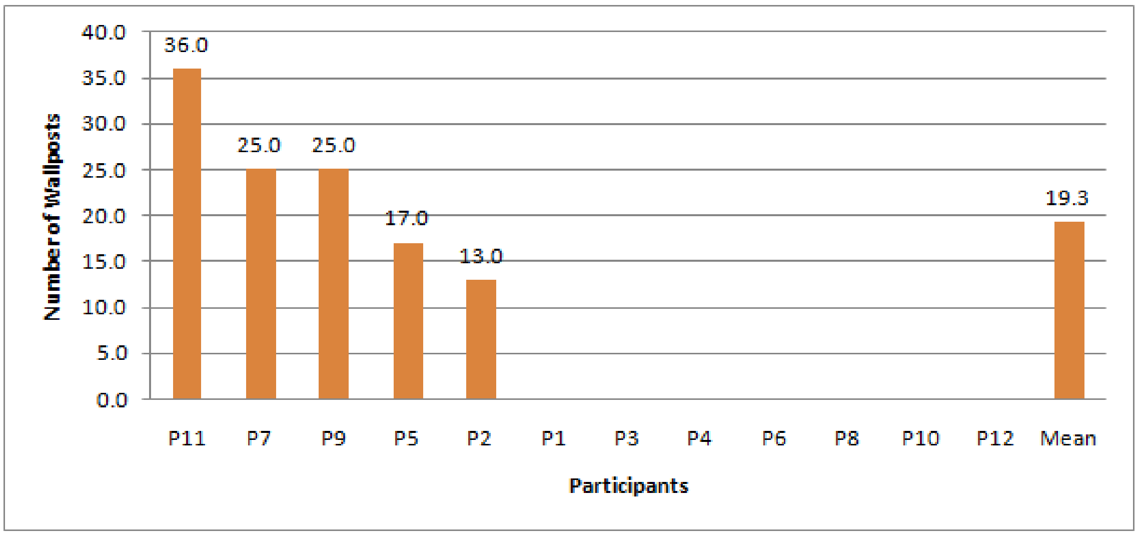

Figure B.17: Facebook Wallposts

B.2.4 Location

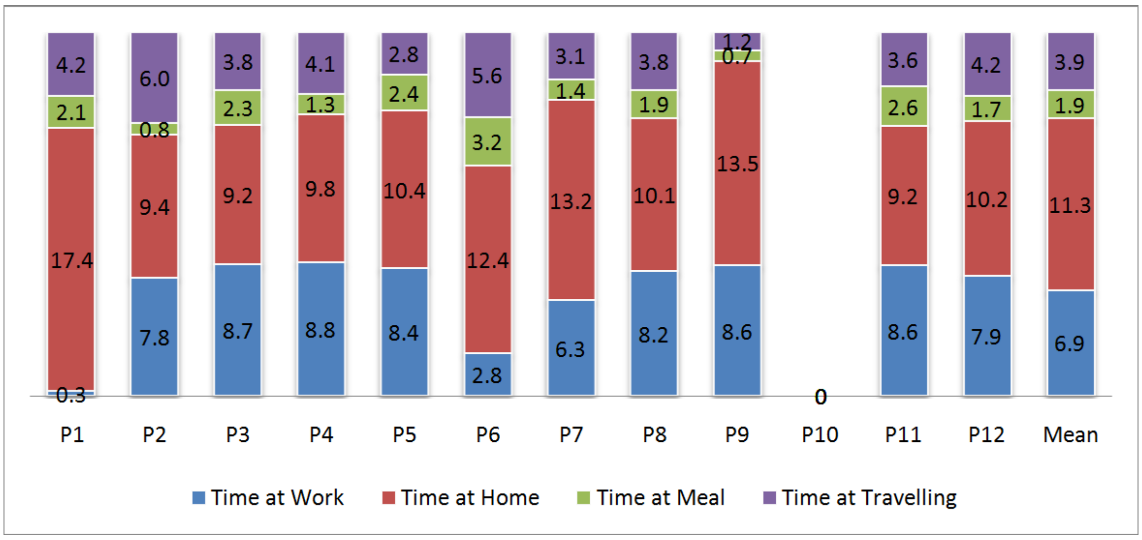

Figure B.18: Daily Time Schedule

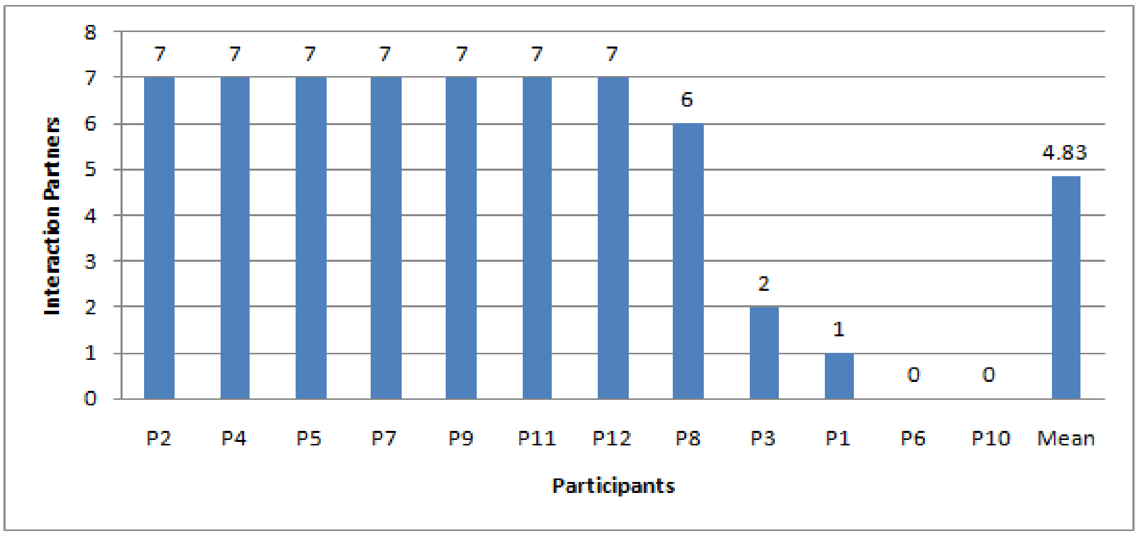

Figure B.19: Interaction Partners

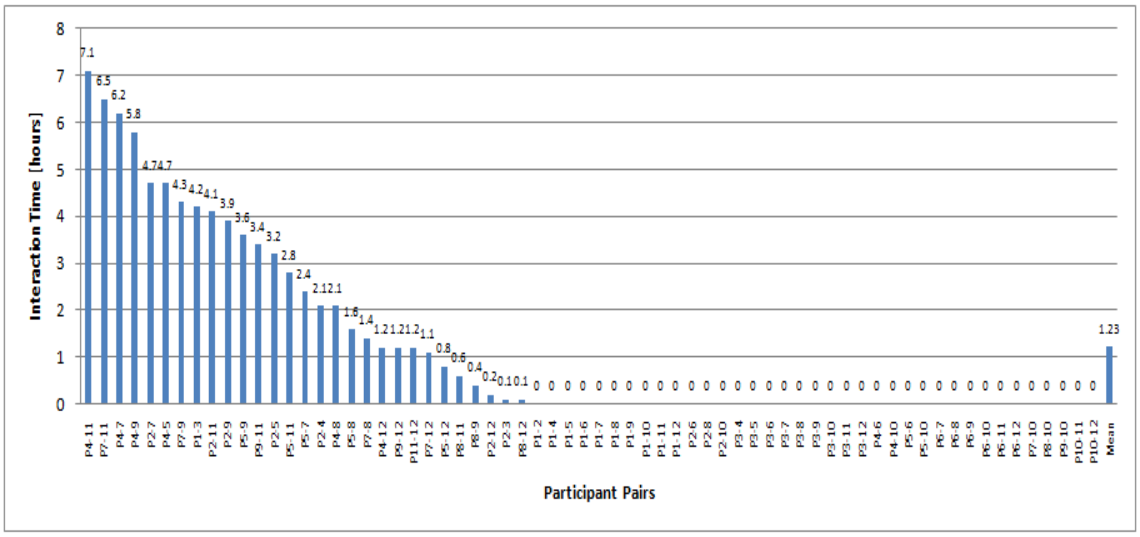

Figure B.20: Duration of meetings per Participant

B.3 Phone Usage Plots

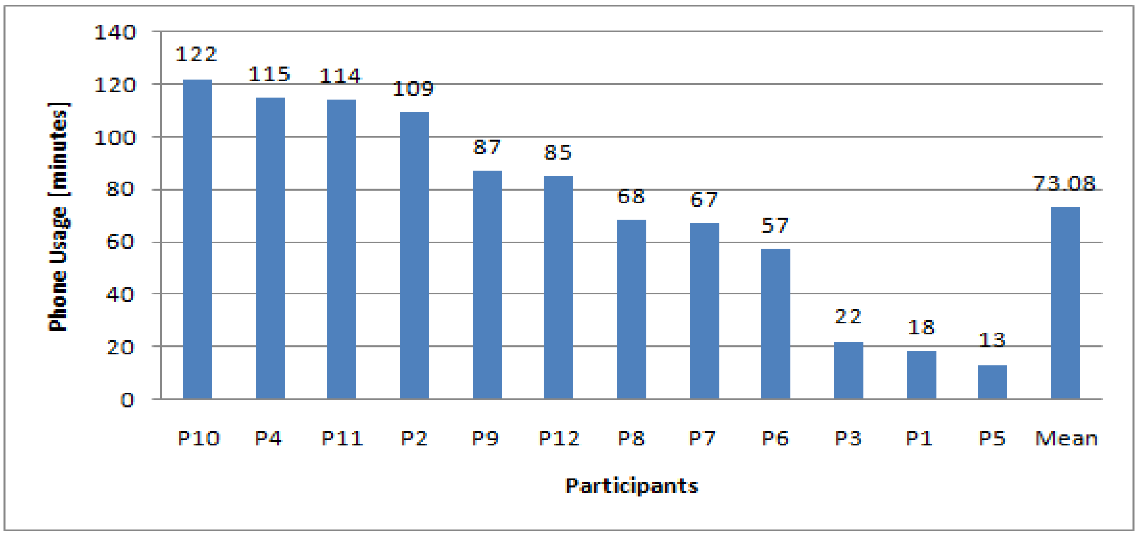

Figure B.21: Phone Usage per Participant

Appendix C

Invitation Email

Dear all,

Short version:
""""""""""""""""

We need your help with a research experiment. If you have an Android phone you can help us by installing and playing the attached Game. You might even win a Samsung Galaxy 10.1 tablet. It would be extra helpful if you get your friends to play the game as well.

Detailed version:
"""""""""""""""""""

As part of a master thesis at the Communication Systems Group (CSG) at ETH, we are doing an interesting experiment. Our goal is to analyze and compare different forms of communication: face-to-face (co-location), phone calls, SMS, Facebook interaction. You can participate in this experiment by using our Android application 'Collect & Trade'.

'Collect & Trade' is an Android based game which allows you to collect items from places (by checking in) and trade them with other players. The goal is to complete all 3 sets of items as fast as possible. For each completed set you can win a Samsung Galaxy 10.1 tablet. The faster you finish a set, the higher is your chance of winning. By trading duplicate items you can increase the speed of completing a set.

In the background we collect communication related information. All date is anonymized (stripped from personal information) and encrypted on your smartphone BEFORE being sent to our server. Additionally, we do NOT collect any content of messages, but only anonymized logs. The data will also not be published or given to third parties at any time.

The setup of 'Collect & Trade' requires minimal effort from your side. Please carry out the following steps:

1) Enable Unknown Resources (Settings –> Application)
2) Install CollectTrade.apk from the attachment
3) Open 'Collect & Trade', enter your email address and login to Facebook. The email address is only required to notify you if you win a Samsung Galaxy 10.1 tablet.
4) Send this mail to your friends with Android phones.
5) Play the game, collect and trade items, and win a tablet!

We appreciate your participation and are looking forward to suggestions and feedback for further improvement of our project.

Best Regards,
Sascha Trifunovic
Onur Mat

Attachment:Collect&Trade.apk

Table C.1: Invitation E-Mail

Bibliography

[1] Tuong Huy Nguyen CK Lu Annette Zimmermann Atsuro Sato Hugues J. De La Vergne Roberta Cozza, Carolina Milanesi and Anshul Gupta. Market share analysis: Mobile devices, worldwide, 4q10 and 2010. *Mobile Communications Worldwide, Telecom Equipment Europe, Mobile Devices Worldwide, Telecom and Internet Markets Asia/Pacific*, pages 1–12, February 2011.

[2] Scott Thurm and Yukari Iwatani Kane. Your apps are watching you. *Wall Street Journal*, December 2010.

[3] Benny Pinkas Yehuda Lindell. Privacy preserving data mining. *Advances in Cryptology*, pages 36–54, September 2010.