Hovedopgave

Master i Informationsteknologi
linien i Softwarekonstruktion

Social Network Extender (SNE)

Af
Lars Nielsen Lind
Jesper Madsen

Den 15. juni 2010

Lars Nielsen Lind, studerende
Jesper Madsen, studerende

Niels Oluf Bouvin, vejleder
Contents

1. Abstract ......................................................................................................................7
2. Readers Guide ..........................................................................................................9
3. Introduction ........................................................................................................... 11
   3.1. Motivation ...................................................................................................... 11
   3.2. Problem Statement ...................................................................................... 13
   3.3. Method ............................................................................................................ 14
4. Related Work ......................................................................................................... 17
5. Analysis .................................................................................................................... 21
   5.1. Technical Analysis ........................................................................................ 21
       5.1.1. Android ................................................................................................... 21
       5.1.2. Bluetooth ................................................................................................ 25
       5.1.3. Matching ................................................................................................ 39
       5.1.4. Findings in Relation to Technical Analysis ................................... 43
   5.2. Security and privacy ..................................................................................... 45
       5.2.1. Security ................................................................................................... 45
       5.2.2. Privacy ..................................................................................................... 55
       5.2.3. Findings in Relation to Security and Privacy ................................ 68
   5.3. Service ............................................................................................................. 70
       5.3.1. Service Design and Context Awareness ............................................. 71
       5.3.2. The Craft of a Context-Aware Service .......................................... 75
       5.3.3. SNE Service Design ............................................................................. 77
       5.3.4. Our findings in relation to Service Design................................. 96
6. System Overview ................................................................................................... 99

7. Software Architecture ........................................................................................ 101
   7.1. Architectural Requirements ........................................................................ 101
      7.1.1. System Qualities .............................................................................. 102
      7.1.2. Business Qualities .......................................................................... 103
      7.1.3. Architectural Qualities .................................................................... 103

    7.2. Evaluation of the Architecture ................................................................. 104

    7.3. Architectural Description ......................................................................... 107
       7.3.1. Component & Connector View-point ............................................ 108
       7.3.2. Module View-point ......................................................................... 118
       7.3.3. Allocation View-point ..................................................................... 124

8. Prototype Implementation .................................................................................. 127
   8.1. Functional Limitations .............................................................................. 127

   8.2. Key Coding Experiences .......................................................................... 128
      8.2.1. Development Environment ............................................................. 128
      8.2.2. Bluetooth Communication .............................................................. 129
      8.2.3. Phone Contact List .......................................................................... 130
      8.2.4. User Interface .................................................................................. 131

9. Evaluation ............................................................................................................. 133
   9.1. The Scientific Method .............................................................................. 133

   9.2. Experiments ................................................................................................. 134

   9.3. Experiment Results .................................................................................... 135
      9.3.1. Bluetooth Range and Distance Impact ............................................ 135
      9.3.2. Number of Devices Impact on Performance .................................... 138
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3.3.</td>
<td>Noise Impact</td>
<td>140</td>
</tr>
<tr>
<td>9.3.4.</td>
<td>Walking Scenarios</td>
<td>141</td>
</tr>
<tr>
<td>10.</td>
<td>Conclusion</td>
<td>143</td>
</tr>
<tr>
<td>11.</td>
<td>Future Work</td>
<td>147</td>
</tr>
<tr>
<td>11.1.</td>
<td>Bluetooth and Wi-Fi in combination</td>
<td>147</td>
</tr>
<tr>
<td>11.2.</td>
<td>Other Mobile Platforms</td>
<td>147</td>
</tr>
<tr>
<td>11.3.</td>
<td>Contact List Filtering</td>
<td>148</td>
</tr>
<tr>
<td>11.4.</td>
<td>Real life usage experiment</td>
<td>148</td>
</tr>
<tr>
<td>11.5.</td>
<td>Privacy Interview</td>
<td>148</td>
</tr>
<tr>
<td>11.6.</td>
<td>Privacy Feedback</td>
<td>149</td>
</tr>
<tr>
<td>12.</td>
<td>References</td>
<td>151</td>
</tr>
<tr>
<td>13.</td>
<td>Appendix</td>
<td>155</td>
</tr>
<tr>
<td>13.1.</td>
<td>Prototype Code</td>
<td>155</td>
</tr>
</tbody>
</table>
1. Abstract

This thesis examines the needs of a context-aware mobile service, which enable users of mobile phones to discover other physically nearby users, based on each other’s mutual relations in the phone contact lists of the mobile phones. When users are discovered the service enables users to make contacts in various ways – e.g. adding each other as phone contacts tagged with context information like the physical location and current calendar event.

The thesis analyses the requirements for the service and presents a service design and software architecture. Based on this a prototype is developed, which is evaluated in order to find out if it is possible to develop a seamless service, which is based on Bluetooth networks and Android phones, without any dependency to external internet-based resources like servers and social network accounts.
2. Readers Guide

The thesis is organized into according to the following main sections:

Introduction
This section contains the motivation for the thesis, the problem statement and a description of the method used for working out the thesis.

Related Work
This section compares key areas from the nature of the SNE service with those identified in related work, and furthermore discusses related projects in general.

Analysis
This section contains an analysis of the problem statement with focus on identifying which fundamental requirements and qualities are most central when developing the SNE service. The analysis is divided in 4 main subsections – one focusing on the technical aspects (Android, Bluetooth and matching), one focusing on security and privacy aspects and finally a section focusing on the design of the SNE as a service including the aspect of context awareness.

System Overview
This section gives an overview of the SNE.

Software Architecture
With the findings of the analysis as a base, this chapter examines the software architecture of the SNE service.

Prototype Implementation
This section describes the experiences gained when implementing/coding the SNE prototype. Furthermore, the functional scope of the prototype is described.

**Evaluation**
This section describes how the developed prototype of the SNE service will be evaluated and what the results are.

**Conclusion**
This section summarizes the thesis and concludes whether the problem statement of the introduction section has been satisfied.

**Future Work**
This section lists the future works in relation to SNE.
3. Introduction

3.1. Motivation

People all over the world make heavy use of Social Networking in their daily life. With Internet applications like Facebook, LinkedIn and Twitter, they create and maintains contacts in different ways, but with the same purpose, socializing.

When people make a relation it is possible for others to see that relation and the relations of relations in various levels. This can be used both private and commercially. If we say person A knows person B and person C, and person C wants to be introduced to person B, then person C can contact person A asking him to make a prior introduction. Of course person C can contact person B directly and use the knowledge of the relationship between person A and person B to initiate conversations.

The more opportunities offered in levels of relations and the difference in the depth of these levels by various applications means a lot of more work to be done. Doing just a simple manual traversal of a network profile may be inadequate and may impose a heavy workload in time.

Another limitation of traditional social computing is that it does not help people to extend their social network based on their physical context. Imagine a situation where people for a while are at e.g. a conference colocated among a lot of strangers. Wouldn’t it be nice just to pick up the mobile phone and see if there is any match between existing social networks to start off conversations and later on, if new contacts are added to the social network, be able to remember the physical context of the contacts?
There is a need for a system, which enables people making a social match to initiate conversations with outset in their social contacts and that system has to be just as mobile as the people using it. And it has to be secure. Therefore this thesis will evaluate how the already existing capabilities of today’s Smart Phones, such as phonebook, calendar, GPS, compass, Bluetooth in combination can help people use- and extend their social network beyond the possibilities of traditional use in front of PC’s.

More concrete this thesis will in general evaluate the problem area through the following use case where persons, who do not know each other, are within a close physical range. The description below uses 2 persons (A and B) as example:

1. The phone of A discovers the presence of B’s phone

2. Phone A compares the content of its own phonebook with the content of phone B’s phonebook. If any contact matches are found the matching contacts are displayed on phone A to indicate to person A that she/he might have friends in his/her personal social network common to person B.

3. To help person A identify and locate person B the following information is displayed on phone A:
   a. Information about Person B
   b. The physical direction from person A to person B (using compass and location information)

4. The “match” leads to a direct physical conversation between person A and person B.

5. Using the mobile phones Person A and person B save the new relationship between each other in their social networks – annotated/tagged with information about the physical location of the meeting and pre-created calendar information.
The use case is using the phonebook as an example of a base for calculating a match between person A’s and B’s social networks. However, several other existing resources can be used as base for matching people’s social networks – e.g. Facebook, Twitter and LinkedIn.

As it appears from the use case above several personal data needs to be sent between the user’s phones – even that the users might not know each other. Normally people do not accept computer systems to communicate private personal data to other people or systems without their knowledge and acceptance. These security and privacy issues will also be evaluated in this thesis.

3.2. Problem Statement

The main problem that this thesis tries to answer is:

“Is it possible to develop a pervasive application for mobile Android devices, which enables physically co-located users to make physical contact in a secure way based on matching phone book contacts and with the use of existing smart phone capabilities like Bluetooth, GPS and compass, and with no dependencies to internet-based resources?”

The main problem can be split into the following sub-issues or questions:

- Is a Bluetooth network feasible for discovering nearby mobile phones and for communicating social network data between mobile phones, with no dependencies to internet-based resources?
- Is it possible to build security and privacy into the system and still maintain a good performing and user-friendly system?
- Is it possible to use the mobile phone phonebook as a source for matching personal social networks?
• Is the location sensors GPS and compass feasible to help people localize each other regarding the users physical direction?

• Is Android a suitable platform for development of pervasive mobile applications, which are utilizing the standard components of a mobile phone like phone contact list, calendar, location sensors (GPS and compass) and network communication possibilities (Bluetooth)?

The construction of the graphical user interfaces are not the focus of this thesis and will not be explained.

### 3.3. Method

The problem area will be attacked according to the process described below. The steps of the process are followed in an iterative way – meaning that each step will be visited multiple times.

1. The problem area is defined by formulating concrete problem hypotheses.
2. A working plan is developed in order to structure our way of working. Furthermore it will secure continuous feedback from our supervisor and make us able to deliver on time.
3. To be able to analyze the problem area and to reuse others knowledge we will seek for and read relevant literature and related work.
4. Using the knowledge gained from the relevant literature and related work the hypotheses will be analyzed with the focus on finding the system requirements.

5. A software architecture will be designed based on the results of the analysis.

6. To be able to evaluate the hypotheses of this thesis we will develop a prototype of the service running on Android-based smart phones.

7. Tests are performed in scientific way to make the evaluation.
4. Related Work

The SNE service tries to make use of resources that are available through the user’s device and only those resources thereby omitting external servers and the need for access to the Internet when facilitating person-to-person communication to the users. Other related work like for instance WhozThat [58] and Secure Social Aware [12] makes use of existing online social network accounts (e.g. Facebook) and has the needs for external servers and access to those social networks when the users make use of their services. MobiClique [7] is the related work which is closest to the network architecture of the SNE service, as it has minimized the need for server communication to the bootstrap phase of the system. Hereafter the devices exchange the retrieved user information directly using Bluetooth.

Common for almost all our related work and the SNE service is that they try to adapt the service provided to the individual person. In order to do that the services makes use of context (e.g. location data) and personal information (e.g. preferences) that may raise security and privacy issues.

The Secure Social Aware [12] service main purpose is to let a system make use of the user’s location and their preferences without compromising their privacy and make it possible to identify a user. It provides this by implementing an Authentication server, which hides the Facebook ID from service providers (e.g. a context aware music service). Compared to Secure Social Aware the SNE does also not compromise privacy of the users by exchanging identity numbers (e.g. Facebook ID), but the nature of the SNE service requires the users to show their personal information (e.g. defined by themselves as user information) in order to be social. They want to be identifiable by name and company.

Different kinds of resources can be used to match the service users. Compared to other related works, the SNE usage of the phone contact list
for social matching is to our knowledge unique. The SNE service makes use of the phone contact list, instead of existing online social network accounts, and makes a relation whenever there is a match between sensed devices contact lists. The Social Serendipity [13] service tries to narrow the matches by making use of user provided weights and similarity scores.

In order for the services to provide a way for the devices in context of relevance to sense and afterwards communicate with each other, the services make use of networking technologies. To sense devices the SNE service, just like for instance Social Serendipity [13] and CenceMe [2], makes use of Bluetooth. Where the SNE service tries to sense the proximity for devices non-stop, the CenceMe service makes use of duty circles to preserve power. The Social Serendipity service makes use of a unique Bluetooth identifier when storing sensed devices. The SNE service does the same, but do not as Social Serendipity make use of a centralized server.

All of the related works are like the SNE service context-aware. The SNE service sense if there are users in the proximity with mutual relations. It also tags the matched users with location data (e.g. who was attending the same event as oneself). MobiClique [7] also sense the users in the proximity with outset in relationship matches, where the user can define what kind of persons are of interest (music and hobbies etc.). The relationships are though different from the SNE service pre-defined by the users. The CenceMe [2] service sends information back to the online social network about where the users are and what they are doing. To sense what they are doing the CenceMe service let the users make use of the mobile phone as a sensing device to make customized tags (e.g. the phone moves indicates a user action).

To our knowledge none of the social-aware related works mentioned above have based their prototype implementations on the Android platform, like the SNE service. Instead they are based on other platforms like Windows Mobile and Java Micro Edition.
Finally we will relate to the Badge2match. It is an example of how new technologies can be used in social networking when people attends events to create a physical product.

Figure 1 - The Badge2Match

Instead of a mobile device they have developed a badge. The badge is an interactive physical badge the users wear around their neck like a neck less. Making use of wireless technology the badge indicates with sound and blinking lights that the person in front of you is of interest, taking outset in the user’s predefined criteria for the matching process, a process they integrate with existing registration system.

The SNE service fulfills the same mission - telling the user that there are persons of interest in the area, but does it in another way, and with outset in the relations instead of predefined criteria’s.

---

1 http://www.badge2match.com/
5. Analysis

The purpose of this section is to find the requirements of the SNE system according to the problem statement in section 3.2. The technical, the security and privacy, the service and the context-awareness aspects will be analyzed.

5.1. Technical Analysis

The technical analysis will include the following areas:

- Android
- Bluetooth
- Matching

The Android platform will be shortly analyzed in order to find the given possibilities in relation to the problem statement in section 3.2. Secondly, it will be analyzed how Bluetooth fits the nature of the SNE system. Finally, the technicalities of the matching function will be analyzed.

5.1.1. Android

The purpose of this section is to analyze the following:

- Overall platform characteristics
- Access to phone contact list
- Access to phone calendar
- Access to sensors (Bluetooth, GPS and compass)
- Possibilities for persistency

The following sections are divided in two sections. The first section include overall platform characteristics and the rest are included in the second section about SDK API’s.

5.1.1.1. Overall platform characteristics

Android has its own optimized Java Virtual Machine (JVM) for handheld devices, the Dalvik Virtual Machine (VM). In the book Pro Android [xxx] it is mentioned that a projects generated Java class files are combined into one or more Dalvik Executables (.dex) files thereby reducing the size by half from traditional .jar files at compile time. Hence, it is not possible to directly execute Java byte code. By using a different kind of assembly-code generation there is an approximately performance gain of 30% fewer instructions. Hence, Android is optimized for handheld devices, and as they say in the book [xxx], the performance paranoia extends into the rest of the Android SDK, that exhibits the most of the characteristics of a full-featured desktop framework. Android SDK support most of the Java Platform (standard edition) except for the Abstract Window Toolkit (AWT) and Swing, which has been replaced with an extensive modern UI framework.

As shown in figure 2 - The Android Software Stack (below), the SNE application resides in the Application Layer, and makes use of the Application Framework by using the Android SDK.

In order to develop applications to the Android platform and thereby making use of the Android SDK, its fundamental concepts must be known. According to [20] such concepts are: Activity, View, Intent, Content provider, Service and the AndroidManifest.xml.
An Activity represents a single screen usually containing one or more views but doesn’t have to. The View forms the building blocks of a user interface in a hierarchical and self-drawing form and there are lots of UI elements to choose from. The Intent defines an “intention” to do some work. For instance start services and launch activities. Intents loosely couples the action and action handler, and are initiated both by the application and the system. The Content provider is used to expose data, for instance a list of contacts, and have an application using data from other applications via a standard mechanism without exposing the underlying storage, structure and implementation. The Service includes two types of background processes: local and remote services. Local services are accessed by the application that is hosting the service. Remote services are accessed remotely by other applications running the device. The AndroidManifest.xml defines the contents and the behavior of the application and it lists, for instance, the applications activities, services and the permissions.

Figure 2 – The Android Software Stack [yyy]
The permissions must be set in order to make use of for instance the Sensors, the Contact List and the Calendar. When the permission for read and write is added for the Contact List, then the relevant items in the API are available for the developer in the project.

![Figure 3 - the Android application lifecycle](image)

Figure 3 - the Android application lifecycle

The lifecycle, depicted in figure 3 – the Android application lifecycle, is strictly managed by the system. In order for the developer to build a stable Android application the developer has to have an understanding of the lifecycle.

In the book [20] the following note is stated:

“Android runs each application in a separate process, each of which hosts its own virtual machine. This provides a protected-memory environment. Moreover, by isolating applications to an individual process, the system can control which application deserves higher priority. For example, a background process that’s doing a CPU-intensive task cannot block an incoming phone call.”
5.1.1.2. SDK API's

The Android SDK [55] offers the following according to the needs of the SNE system:

- Access to phone contact list
- Access to phone calendar
- Access to sensors (Bluetooth, GPS and compass)
- Possibilities for persistency in flat files or in a SQLite database.

5.1.2. Bluetooth

The purpose of the technical analysis of Bluetooth is to examine how it fits the nature of the SNE system. Due to the nature of the SNE system the following items need to be satisfied:

- Location context awareness
- Exchange data between devices
- Make phone calls and send SMS text messages
- A minimum of dependencies to the physical surroundings
- Secure communication
- Performance and scalability
To analyze the listed items above the following sections are divided in six sections. The first section gives an overview of mobile network architectures. The second section describes the basics of Bluetooth. The third section clarifies the above mentioned first six items needed and the fourth section analyses how these six items fits into the SNE system by the usage of Bluetooth. The fifth section examines the need for service discovery and the sixth section examines the need for a point-to-point communication protocol.

5.1.2.1. Bluetooth and Mobile Network Architecture

Modern smart phones are typically equipped with the wireless radio capabilities mentioned below:

- Bluetooth radio
- WiFi radio
- 2G/3G radio

These network capabilities open up a world of alternatives when it comes to architectures for mobile communication [41]. Roughly speaking communication architectures can be grouped into the following 4 alternatives:

- Point-to-point Communication
• Meshed Communication (ad hoc network)

• Cellular Communication

• Cooperative Communication

Point-to-point communication (see figure below) is characterized by mobile devices communicating directly without any infrastructure equipment like base station or hotspots. Typically today’s mobile phones offer point-to-point communication using the short-range radio’s Bluetooth and Wi-Fi. Several other technologies exist, but are not widely available in mobile phones today.

Meshed communication is an extension of the point-to-point communication architecture. The single devices of the point-to-point networks are capable of operating as relays for communication between devices which are not within radio range.

Figure 4– Point-to-point [41]

2 http://en.wikipedia.org/wiki/Mobile_ad_hoc_network
Cellular Communication is defined as the traditional mobile networks (see figure below) where a mobile device connect through a base station, which operates as a relay for the mobile device to connect with other mobile- or fixed devices. Typically today’s mobile phones offer cellular communication by providing access to the 2G and 3G networks.
Cooperative Communication is defined as a hybrid of the other communication architectures (see figure below), which makes use of point-to-point-, mesh- and cellular communication in a combined way.

5.1.2.2. Bluetooth Basics

Bluetooth is a Wireless Personal Area Network (WPAN) characterized as the following [25]:

- The network is based on radio technology operating in the 2, 4Ghz ISM band

- Bluetooth is a short-range network typically having a 10-100 meters range – depending of the class of device. However, in practice mobile phones typically have a range around max. 10 meters.
• Devices in a Bluetooth network form a so-called Piconet, which is an ad-hoc computer network (see figure below). A Piconet consists of a master device and up to 7 slave devices, where the first device attending the network is the master device.

![Figure 8 – Piconet](image)

• Devices of a Piconet are using a dedicated radio channel for communication. To minimize the risk of clashing with other devices using radio channels within the 2,4 GHz band and to enable co-existence with static (non-hopping) ISM systems, a so-called frequency hopping pattern is used. The hopping pattern is “hopping” between 79 frequencies within the 2,4 GHz band 1600 times per second. To implement the frequency hopping pattern the devices of a Piconet are synchronized to the clock of the master device, which is used to time the swap to a new radio channel.

• To enable devices to communicate with multiple devices in parallel, a device is allowed to be part of multiple Piconet’s in
simultaneously. When multiple Piconet’s are connected, it is also called a Scatternet (see figure below). This way a device can be a slave in several Piconet’s, but only be a master in one.

**Figure 9 – Scatternet**

- The bandwidth is approximately 1 Megabit per second for basic rate or 2-3 Megabit for enhanced data rate (EDR)
- A Bluetooth device can have 60 simultaneous Bluetooth connections

### 5.1.2.3. SNE Requirements to Bluetooth usage

Location context awareness in the SNE system means the possibility for a mobile phone to discover the physical nearby presence of other mobile...
phones. Clearly this advocates for a short-range point-to-point network technology, which can be used to “sense” the nearby presence of other devices.

After the nearby presence of a device has been “sensed”, it must be possible to exchange data between two devices using the network – e.g. social network data and user profile data about the phone owners. Furthermore, a requirement to the network is to have the possibility of make phone calls and send SMS text messages.

The network technology must have minimum of dependencies to the physical environment of the SNE system in use. This means that physical objects between communicating devices like humans, walls, furniture etc. should not prevent the devices from communicating. Furthermore the network must not require the presence of e.g. a Wi-Fi hotspot or similar equipment to make the system work – i.e. the network must work in a point-to-point fashion.

The chosen network technology is furthermore required to support general security mechanisms in order to secure the data of the user and privacy. Security- and privacy is covered in details in section 5.2.1 and 5.2.2.

Another important requirement for the SNE system is performance and scalability:

- When a SNE device has to discover other devices in a location with many nearby devices it depends heavily on the network, which puts high requirements on the signaling- and data through-put performance of the network.

- The network must scale with many SNE devices communicating simultaneously.

- A device must be able to communicate with multiple devices simultaneously.
5.1.2.4. How Bluetooth fits SNE

This section evaluates how Bluetooth fits the network requirements specified in section 3.1. Note that requirements regarding security are treated later in dedicated sections.

Location context awareness

Bluetooth being a short-range (approx. 10 meters for typical mobile phones) radio technology means that devices will only be able to communicate when placed in the same physical location. This property of Bluetooth fits very well with the concept of the SNE system, as the short-range radio can be used to “sense” the nearby physical presence of other SNE users.

Whether 10 meters is too little of course depends on the environment in which the SNE service is deployed. 10 meters fits the initial intention that the system is deployed in quite crowded environments like business exhibitions and conferences where people are located very closely together. Put it in another way, if the range instead is 100 meter, then the SNE system would probably be overloaded if it had to operate in a very crowded environment. However, if the SNE is deployed in less crowded environments, 10 meter might be too short.

An interesting issue for the SNE service is if it is possible to perform discovery and matching of mobile users within the short range of 10 meters.

No dependencies to the physical surroundings

As described above Bluetooth operates as an ad-hoc network, which is based on point-to-point communication between network devices without the need for any other equipment like e.g. a hotspot or base station.
Furthermore, as Bluetooth is a radio based network technology, it does not require line-of-sight like e.g. IR based technologies do. This means that Bluetooth perfectly lives up to the requirement of having no dependencies on the physical surroundings.

*Exchange data between devices*

Bluetooth supports the requirement of communicating data between devices. The question is though, whether a data rate of 2-3 Mbps per second is acceptable for the SNE system. Compared to the 3G mobile network the data rate is a bit better (max. 2 Mbps) and 3G furthermore might prevent users to use the SNE system due to the cost using the network.

Compared to Wi-Fi it is significantly lower (approx. 10 Mbps) [44]. So why not just use Wi-Fi instead of Bluetooth then? Wi-Fi seems to be significantly more power-demanding during network communication. In the paper *Experimenting with Opportunistic Networking* [4] tests are showing that using Wi-Fi for continuously discovery devices significantly drained the device battery compared to Bluetooth. Another reason for not using Wi-Fi is that Android 2.1 [45] does not support using the IEEE 802.11 in “ad hoc” mode, which is needed for devices to communicate point-to-point without the need for a Wi-Fi access point/router [46]. At the time of writing, this however seems to be changed with Android 2.2, which allows Android devices to run as Wi-Fi hotspots [47].

Due to these issues Bluetooth currently seems to be the best choice for the SNE system. However, when it gets possible to run Android in Wi-Fi “ad hoc” mode, then an interesting possibility could be to combine Wi-Fi and Bluetooth, where Bluetooth is used for device discovery and Wi-Fi is used for point-to-point communication. This idea was also presented as part of the article [4].
Make phone calls and send SMS text messages

Obviously Bluetooth cannot cover this requirement, as phone calls and SMS text messaging are only features supported by the 2G/3G network. This requires the SNE network to be a mix of Bluetooth point-to-point networks and the 2G/3G cellular networks – i.e. a cooperative network.

5.1.2.5. Service Discovery

SNE is a highly distributed system, where each mobile device at the same time both functions as a client and a server. The server-part of the mobile device offers a service for clients of other mobile devices to connect to. A fundamental requirement is then that devices need to be able to discover each other remotely without any help of a centralized server regarding how to locate the service.

Service discovery can in general be defined as the following [43]:

“Discovery is the process by which an entity on a network (a client) is spontaneously notified of the availability of desirable services or devices on the network (resources).”

Basically the SNE system needs for the service discovery:

- Be able to find devices within Bluetooth network-range
- Be able to exclude/filter devices, which are not offering the SNE service. Many different kinds of Bluetooth devices might be discoverable using the Bluetooth device discovery Protocol, but only devices providing the SNE service are relevant.

Several alternative protocols exist to be used for Service Discovery: Jini, Salutation, UPnP, Bluetooth SDP, SLP and Bonjour. The protocols are described in [42] and will not be described here.
As Bluetooth has been selected as the point-to-point network, it is natural to consider the Bluetooth Service Discovery Protocol. However, a huge disadvantage with the Bluetooth Service Discovery protocol is that it is not part of the device inquiry process. Instead the service discovery process requires the establishment of a point-to-point connection with each device found in the inquiry process. This is likely to have a bad influence on the service discovery process of the SNE application – especially in a crowded environment with many Bluetooth devices.

Let’s take an environment containing 30 Bluetooth devices, but with only one of them offering the SNE service. In a worst-case scenario, this will require establishing 29 unnecessary point-to-point connections (including service discovery) before we get to the relevant device. The paper [4] confirms this concern with experiments, which show that the service discovery process adds several seconds per device.

The same problem will apply with the other service discovery protocols mentioned above, as they also will require the establishment of point-to-point connections after the device discovery process have been performed.

Instead of using any of the service discovery protocols it would be nice if it was possible to incorporate the service discovery as part of the device discovery process. Bluetooth v2.1 contains an interesting feature called the [Extended Inquiry Response (EIR)](https://www.bluetooth.com/specifications/technology/bluetooth-core-specification-v2-1-edr) [57, p 1306], which extends the existing [Inquiry Response (IR)] with [Service Class UUIDs](https://www.bluetooth.com/specifications/technology/bluetooth-core-specification-v2-1-edr), which is exactly what is needed for the SNE service. Unfortunately, this is not supported by the current Android API, even that the devices are based on Bluetooth v2.1 + EDR. Currently, this forces us to analyze the possibility for using the basic IR. Let’s see what device information’s are obtained during the device discovery process:

- Device address
- Device class
- Device name (friendly name)
The big question is whether an application will be allowed to attach information to one of these attributes in order to indicate the availability of the SNE service.

The device address is used to uniquely identify a device in a Bluetooth network and is a 6-byte hardware address hard-coded from factory, so this is obviously not modifiable from an application. The value of the device class indicates the class of device – i.e. phone, headset etc. Whether the class of device is modifiable from an application is unclear at the time of writing.

However, the value of the device name has no restrictions apart from the length of the string, which is 248 bytes. This string is typically modifiable for both users and applications running of a device, which actually means that it might be used for indicating the availability of the SNE service and thereby getting a way to exclude/filter the devices not providing the SNE service.

Using the device name comes with some limitations:

- The user might have specified her/his preferred device name manually, so she/he might not like that an application is also modifying the device name.

- Multiple applications using the same approach cannot co-exist on the same device, as there is a risk that they might overwrite each others strings.

Being aware of these limitations it is still proposed to use the device name to indicate the service availability, as it is expected to optimize the service discovery performance.

The article Using Bluetooth Device Names to Support Interaction in Smart Environments [16] is an example of other work using the device name as way of communicating. Here users use the device name to send commands to the system by manually entering commands in the device name field.
5.1.2.6. Point-to-point Communication

To handle the point-to-point communication between two devices, we need to decide on a protocol for sending data.

To be future proof, we need a protocol which is platform independent. Choosing e.g. Java RMI would limit the SNE system to only run on Java-based device platforms. However, the future might require the SNE system to run on multiple platforms, so this would not be acceptable.

As the data/messages are text-based we have been decided to go for a simple text-based protocol running directly on top of Bluetooth, because it is simple to implement and it is platform independent.

Regarding the way of structuring the text-based data, we have decided to used JavaScript Object Notation (JSON), which is a standard often used for implementing Web Services. Another common standard for structuring text-based data is XML, but as JSON is simpler and more compact than XML, it is a better match for the SNE requirements.

Android has built-in API’s for handling JSON structured text, which makes it easy to implement.

To avoid re-inventing the wheel we have chosen to structure protocol data according to JSON-RPC.

---

3 http://json.org/

4 http://json-rpc.org/
5.1.3. Matching

In order for the SNE system to work there has to be a matching function with the purpose of matching users with outset in their mutual relations.

To support a seamlessly use of the SNE system as a service for the users, we want to investigate if it is possible to make use of their contact lists, as outset for this matching function. Thereby we will avoid the login to an account each time the users make use of the application, as would have been necessary if the matching function have used the user’s social network accounts e.g. Facebook and LinkedIn. In fact by doing it this way there will be no need for an authentication on a remote server every time a match has to be performed and no user login data has to be saved in the SNE system, to make a seamlessness authentication.

To analyze the needs of the matching function the following aspects are examined:

- Matching and privacy
- Data to be used for matching
- Multiple contact lists and calendars
- Search algorithm
- Mapping of non-readable Phone Number to Contact ID
- Versioning of non-readable contact lists
- The matching process
5.1.3.1. Matching and Privacy

Using the users contact list content as source for the exchange data between the user’s devices must be done with attention to privacy issues. The data may not be readable in an understandable way by others under transfer between the devices and it must not be placed on other users devices in a readable form. A cryptographic hash function to make data non-understandable is described in section 5.2.1.

Another issue is contact list filtering, as a users contact list may include entities that the user don’t want to make use of as outset for a new relation. It could for instance be a counterpart in a court case. In order to take privacy into account (see privacy section 5.2.2) it must be possible for the users to indicate users the matching process must omit when generating the lists.

5.1.3.2. Data to be used for Matching

The matching function doesn’t need all the data about the entries in a contact list, but exactly what data is best fit to use for the purpose of matching?

If an entry name is chosen there may be a high degree of inconsistency. The first, middle and surname can be created in various ways. For instance a contact with the name Hans Peder Jensen could be entered into a contact list like this:

- Hans Peder Jensen
- Hans Jensen
- Hans Peder
- HPJ
- HP
- H.P.Jensen
The entry name may be created with a large amount of diversity between the users. There is also a high degree of duplicate names. In Denmark for instance there are approximately 5023 persons with the name Jens Jensen [34] and is it the same Jens Jensen as a person that is the mutual relation between two devices? The matching function will not be able to find out of that with outset in the entry name alone. And some entries might be created with the use of a nickname. Hence, it is not an option to make use of the entry name from the contact list as a factor in the matching process.

If the phone number is chosen there still is a possibility for inconsistency. Like with the entry name a phone number can be created in various typical ways.

Another option could be to use an entry’s e-mail address. It is, just as the phone number is, unambiguous. We are aware of that a couple can share their e-mail address and mobile phone, but today an e-mail address and a mobile phone is a personal accessory. We chose to make use of the phone number because in Android devices the Gmail contacts are added to the users contact list, and the Gmail contacts may be created without any contact information besides the essential e-mail address. If it is a serious contact it is very possible that the user will create it with all the relevant contact information including a phone number. Thus an entry in the contact list with a phone number might be more relevant to the purpose of the application with outset in the application nature.

In the application there will be a need for showing the users some data about their mutual relation when there is a match. These data are already present on their devices in their own contact list, and by using their own entries, the way they have created them; the users can make an advantage of the diversity in the ways they name their contacts because they are displayed with for instance a familiar naming.
5.1.3.3. Multiple contact lists and calendars

It is possible for the users to have more than one contact list and calendar on their Android device. Therefore the SNE system must make it possible for the users to define which to use with the SNE system. If the wrong contact list are used the relevancy of the matches might be decreased, and if the wrong calendar are used the geo-tagging will indicate a wrong or no event location at all.

5.1.3.4. Search algorithm

The matching process has to make use of some sort of search technique to make a match between the contact list data. We do not know the number of entries in the lists used for matching, but the search algorithm will always be linear because all of the mutual relations in two lists have to be known in order to display them for the users. Therefore all entries in the lists have to be compared against each other. If the exact identity of a mutual relation was not to be displayed then other and faster search algorithms may have been used, for example Binary Search and Hashing, ending the search when the first match has been found.

5.1.3.5. Mapping of non-readable Phone Number to Contact ID

When the lists for matching are generated the phone numbers will be made non-readable. As the non-readable phone number is the only data received it must be possible to map it to the contact ID of the equivalent phone number on the local device. Thereby it will be possible to extract the contact information about the mutual relation (e.g. name, address etc.).

To avoid re-generating the non-readable phone numbers at each matching, the non-readable phone numbers must be generated once at application
startup and cached in an in-memory mapping table, which maps between the non-readable phone number and the contact ID. This table must be updated when contacts are added to the phone contact list.

5.1.3.6. Versioning of non-readable Contact Lists

In order to optimize the matching process, the non-readable contact lists must only be exchanged when necessary (e.g. new contacts added). By adding a version number to the lists it is possible to only exchange the lists when the version number differs.

5.1.4. Findings in Relation to Technical Analysis

Android

In relation to Android the following has been found:

- The Android SDK supports the need to access the phone contact list.
- The Android SDK supports the need to access the phone calendar.
- The Android SDK offers the possibility to access phone sensors (Bluetooth, GPS and compass).
- The Android SDK offers the possibilities to save flat files persistently.

Bluetooth
The following has been concluded in relation to the network analysis:

- Bluetooth is suitable for location awareness (sensing nearby devices), as Bluetooth has build-in support for device discovery. The Bluetooth range of 10 meters is suitable for limiting the physical area when the SNE system is deployed in crowded environments like conferences.

- The Bluetooth bandwidth of 2-3 Mbps is enough for the relatively small data sizes send between SNE devices.

- Sixty simultaneous Bluetooth connections for a device is enough for the SNE service.

- Bluetooth is less power-demanding compared to Wi-Fi.

- The built-in support for service discovery in Bluetooth does not fit the nature of the SNE system. Instead the device name can be used as a way of filtering devices not running the SNE service.

- An interesting alternative for future work, could be to gain the best from both Bluetooth and Wi-Fi by using the two in combined solution. Bluetooth for discovery and Wi-Fi for communication.

**Matching:**

The use contact list data must take privacy into consideration. The data must be non-readable, which can be done by use of a cryptographic function.

In order to avoid being matched with unwanted contacts in the phones contact list, it must be possible for the user to filter this list.

The phone number has been chosen as the data used for matching, as it is most unique data of the contact list.
There can be more than one contact list and calendar on the users Android device, this has to be taken into consideration.

The search algorithm to be used for matching must be linear, as all contacts must be compared.

In order to extract contact list data about mutual relation, the non-readable data has to be mapped in an in-memory table to the Contact ID. To avoid regenerating the mapping, it must be generated at application startup.

To optimize matching of already matched devices, a version number must be used for the generated lists to avoid unnecessary exchange over the network.

5.2. Security and privacy

As mentioned in the problem statement, security and privacy is a key concern of the SNE system. The analysis is divided into a section about security and one about privacy.

5.2.1. Security

The security is divided into 3 sub-sections. The first section has the focus of putting SNE security in perspective according to computer security in general. The second section analyses how the security mechanism provided by Bluetooth fits the general security requirements stated in the first section. The last section uses the output of the two first sections to analyze what the security needs of the SNE system is.
5.2.1.1. Security in general

Traditionally security has been a major concern in distributed systems with wired networks of computers. However, as earlier mentioned the nature of the SNE system is quite different compared to traditional wired computer networks, as it is deployed in a mobile network where the mobile devices communicate in a point-to-point fashion.

So, the question is how the security needs of the SNE system differ compared to the experiences gained from traditional wired computer networks. The paper *Mobile Security and trusted computing* [8] argues that the basic security requirements are the same, but the approached needed to accomplish the requirements are different. The basic security requirements are listed:

- **Authentication**
  A device may need a way to verify the claimed identity of a remote device.

- **Authorization**
  A device may need a way to control which local resources can be accessed by a remote device. Maybe a remote device only has the rights to access parts of the services offered by the local device.

- **Accountability**
  If a remote user/device gets authorized to access to a service offered by the local device, the remote user/device must be held accountable for all action made with the local service.

- **Confidentiality**
  It is important that sensitive data is protected, so that unauthorized remote users/devices cannot get access to it.

- **Integrity**
  The integrity of data must be secured. It must not be possible for
unauthorized users/devices to manipulate data by re-ordering, deletion or replication.

When it comes to the approaches for support the basic requirements the following challenges compared to traditional wired networks:

- Being inside and outside of the corporate network
- Being in a managed security environment

In traditional wired networks the notion of being inside or outside the corporate network is used, where firewalls etc. are used to protect close corporate networks. In ad hoc networks and point-to-point networks like the SNE system, this notion does not give any sense. The networks can be established anywhere at anytime without the possibility to protect it using traditional mechanisms like firewalls.

Furthermore, it does not give any sense to have the notion of being in a managed security environment, where it is controlled which type of devices is part of the network and who owns the devices. In networks like the SNE system all kinds of Bluetooth enabled devices owned by any person or organization can be within Bluetooth network range, and are thereby candidates to participate in the SNE network.

5.2.1.2. Bluetooth Security

The security architecture of the Bluetooth specification supports three main areas of basic security [6]:

- Authentication
- Confidentiality
- Authorization
Device implementations can conform to the basic security areas in different degrees depending on the type of device and the needed level of security. In order to handle this diversity the Bluetooth Security architecture defines 4 Security Modes to which devices can comply [6]:

- **Security Mode 1**
  Security Mode 1 simply implies no security functionality at all – i.e. no authentication and no confidentiality. This security mode is only supported in Bluetooth v2.0 + EDR (and earlier versions). Mode 1 is not supported by the Android API [48].

- **Security Mode 2**
  Security Mode 2 implies authentication and encryption. Users are authenticated through a so-called pairing-process where each user is asked to enter a PIN code – the process is successful if the same PIN code is entered on both devices. After the PIN code has been accepted secret key information is exchanged to establish an encrypted connection between the devices (confidentiality). The authentication is performed at the “service level”, which is at the L2CAP layer of the Bluetooth protocol stack. This enables the support for authorization, where a centralized security manager maintains policies restricting which of the different services provided can be accessed and which cannot.

- **Security Mode 3**
  Security Mode 3 and 2 basically specifies the same kind of authentication and confidentiality (encryption). The difference is when the authentication is performed. Compared to mode 2 the mode 3 performs the authentication at a lower layer in the Bluetooth protocol stack (LMP), which means that the notion of authorization is implied. In mode 3 a device will either have access to all- or no services.

- **Security Mode 4**
  As part of Bluetooth 2.1 + EDR the Security Mode 4 has appeared,
which claims to be more secure and to be more user-friendly compared to mode 2 and 3. Like mode 2 the security functionalities of mode 4 is performed at the service level, but it comes with an improved way of pairing and authenticating devices - Secure Simple Pairing (SSP). SSP improves security by adding Elliptic Curve Diffie Hellman (ECDH) public key cryptography for protecting against eavesdropping and man-in-the-middle attacks (MITM) during the pairing process between two devices. SSP improves user-friendliness by providing new models for pairing devices, depending on the input- and display capabilities of device types. The model relevant for the SNE system is the Numeric Comparison model, which is designed for pairing the device types used in the SNE service, both have the possibility of showing a six digit number and allowing the users to enter “yes” or “no” to allow/disallow the pairing.

Regarding confidentiality 3 modes/levels of encryption is defined:

- **Encryption Mode 1**: No encryption is performed on any traffic.
- **Encryption Mode 2**: Traffic which is related to traffic on a individual point-to-point connection between 2 devices is encrypted using the link key produced during the pairing process. Broadcast traffic is not encrypted -
- **Encryption Mode 3**: All traffic is encrypted using an encryption key based on the master link key.

### 5.2.1.3. Security needs of SNE

#### 5.2.1.3.1 Authentication

As mentioned earlier the SNE system is intended to rely entirely on point-to-point communication between the mobile devices. This nature of the
SNE contradicts with the requirement of authenticating users due to the following reasons:

- There is no presence of any network infrastructure.
- A key concept of the SNE application is that users base their decision on whether to establish a physical social connection on the commonalities between their social networks (phone books). This obviously requires the matching function to exchange information between the devices, while the users remain anonymous.

In traditional wired network systems user authentication systems are often based on the presence of a statically located authentication server. However, without any network infrastructure in the SNE system an authentication server solution is not a possibility to help implementing a user authentication system.

Looking at the general headlines about Bluetooth security, it seems that it supports the general security requirement about authentication. However, when you dive into the details it becomes clear, that Bluetooth security conflicts with the nature of the SNE system, where users remain anonymous during contact list matching.

As the pairing process requires the users of the devices to physically agree on a PIN code before a secure point-to-point connection can be established. This means the pairing process needs to be performed before matching information can be exchanged, which clearly makes the pairing process very much against the nature of the SNE system and the SNE system will be required to bypass this process somehow. According to the four Security Modes of Bluetooth this leaves us with the following alternative ways to follow:

1. Base SNE on Security Mode 1 and bypass the security functionalities offered by Bluetooth
2. Base SNE on Security Mode 2, 3 or 4 and use PIN codes hard-coded by the SNE application. This way the PIN code dialog will not appear of the devices of the users.

Alternative 1 will require the SNE application to implement security functionalities in the application layer – i.e. above the Bluetooth protocol stack. However only device types based on Bluetooth v2.0 (and earlier) might support this security mode – i.e. newer- and future devices will probably be based on v2.1 or higher. Furthermore, Android 2.1 devices are based on Bluetooth v2.1 + EDR and thereby only support Security Mode 4 with backwards compatibility to Security Mode 2 devices.

This leaves us with security mode 4 in alternative 2, which requires using the pairing process. Apparently the SNE system is in conflict with the specification of Bluetooth, and it seems like the only way to use Bluetooth is to somehow hard code the PIN code used in the pairing process.

So far we can conclude that while the SNE system cannot rely on any help from the network infrastructure and users need to remain anonymous, then user authentication cannot be implemented in SNE.

However, instead of authenticating users at runtime when using the SNE system, the authentication model could be applied at the SNE application installation time. If the Android Market [49] is used for deployment of the SNE application, the Android Market should authenticate the user by registering the Gmail account- and the Bluetooth device address of the user. This way the user of a SNE device can be identified based on the device address and this might prevent malicious users to get hold of the SNE application. A Gmail account is probably not very good as identification – a more secure way to identify users would be to use personal digital certificates e.g. Digital Signatur [50]. Authenticating users at installation time will not be part of this thesis, but might be a solution to investigate in future works.
5.2.1.3.2 Confidentiality

To secure confidentiality in the SNE system 2 problem areas need to be secured:

- Data exchanged on the network
- Sensitive data stored on remote devices

Data exchanged on the network

Data exchanged on the network must not be sniffed and misused by malicious users. All data exchanged between the mobile devices must be encrypted during the network transfer, so data sniffed on the Bluetooth network cannot be misused by malicious devices.

As described earlier part of the Bluetooth pairing process is to setup a secure point-to-point connection between the mobile devices, with the purpose of securing confidentiality. The earlier security modes (mode 2 and 3), which are used in Bluetooth v2.0 (and earlier), are using the pairing pin code, entered by the users, to generate the keys used for encryption. This approach has appeared to involve some severe security flaws, where the pin code can easily by “cracked” [1]. As the pin code is used in the encryption of data this enables malicious users to decrypt and misuse data send on the network.

However, as mentioned in section 5.2.1.2 security mode 4 introduced in Bluetooth v2.1 EDR, has improved confidentiality using public key cryptography, which is not using the pairing pin-code as input to the encryption algorithms.

Alternatively, SNE could improve the confidentiality even more by adding an encryption layer on top of the Bluetooth layer – e.g. commonly used public key cryptography mechanisms like SSL. This is however not treated
any further in this thesis, as the Bluetooth security mode 4 is seen as secure enough for the SNE system.

**Sensitive data stored on remote devices**

Data exchanged about social networks (phone book contacts for matching) must not reveal privacy of users. As described above Bluetooth security mode 4 will secure that the data will be encrypted during the network communication, but what happens after the phone book contacts have been received by the remote device? If all phone book contacts are stored in a plain text manner it certainly opens up for compromising the user privacy.

This means that is it not enough to secure confidentiality data during network communication, data also need to be stored at the remote devices in a non-readable form. Hence, is it possible for the SNE system to make the social contact data for matching non-readable on the local device before sending it to the remote device?

A way to accomplish this is to use cryptographic hash functions like [52]:

- MD5
- SHA-1
- SHA-2

The details of the hash functions are not the focus of this thesis, but can be read in [52]. The difference among the 3 functions is the level of security, where MD5 is seen a not secure and SHA-1 is less secure than SHA-2. It is recommended to use either SHA-1 or SHA-2 in the SNE system, where SHA-2 is the preferred one in regard to security. However, SHA-2 values are longer than SHA-1 values (224 – 512 bits compared to 160 bits), so it will influence the performance of the social matching process.
5.2.1.3.3 Authorization

Bluetooth is offering a kind of authorization, for controlling access to Bluetooth services. The SNE system only offers one service though, which is available to all devices having the SNE system installed, so this kind of authorization is not needed.

However, another kind of authorization is needed in the SNE system in order to preserve user privacy. One example is whether remote users are allowed to add a “discovered” user as a phone contact. In this case is it configurable for a user whether the SNE system can allow this automatically, or whether the user wants to be asked (yes/no) to “authorize” this operation. This is just one example of user privacy - see the privacy section 5.2.2 for more details.

5.2.1.3.4 Accountability

As SNE users are anonymous (see also the section about authentication) during the service discovery and social matching process it is obviously not possible to hold user accountable/responsible for their actions with the SNE system.

However, the SNE system must log the user actions together with the associated user info data. This way there is chance that user and the users actions can be seen.

5.2.1.3.5 Integrity

The SNE system does not provide any services which allow remote SNE users to change any local resources. However, the SNE system also has a responsibility to secure that remote devices cannot get access to any other
resources on the local device. It must e.g. be secured that when the SNE system turns on the Bluetooth adapter, it must not open up security “holes” which can break the integrity of the data placed on the device.

The SNE is of course not able to control the other Bluetooth services installed on a mobile device, but it must warn the user about possible danger when the Bluetooth adapter is turned on, for instance in some sort of help pages.

### 5.2.2. Privacy

The SNE service makes use of “event-tagging”. By “event” we mean the device user’s location combined with an extract from the device user’s calendar and an auto-generated timestamp. The “event” is attached to the remote user’s entry in the users contact list and is logged. The tagging with location data is also known as the phenomenon ‘geo-tagging’. Thereby the remote user’s whereabouts are recorded to another user’s device. This kind of functionality may trigger privacy concerns and may pose a threat to the efforts trying to make user’s use the application. A newly posted blog in the psychological domain goes the other way and makes it an advantage to show others that the person has been attended the right places, and the name for the phenomenon is ‘geo-social’ [32].

Another source to concerns might be third-party entities placed in the contact lists that are used for matching and how these contact list data are exchanged and stored. When there is a match between two users there is a need to see some information about the mutual relation. But what if the relation for instance has a secret phone number or for instance is the mistress and also is the wife’s best friend, and which of the contact list data are to be displayed?
And how much data about one self has to be exchanged to other users before an exchange of contact data is authorized?

The following sections focus on the possible privacy concerns and how to avoid they become a threat for the application. First a definition of privacy is examined and afterwards we look at privacy preferences, re-identification, and third-parties, to make up one’s mind about privacy concerns.

5.2.2.1. Privacy definition

According to the *Privacy and Human rights: Overview 2003* [37], privacy is divided into four separate but related aspects:

- Information privacy (collection and handling of personal data)
- Bodily privacy (peoples physical selves, for instance genetic and drug testing)
- Privacy of communications (security and privacy of communications, for instance telephones and e-mail)
- Territorial privacy (limits on intrusion to domestic and other environments, for instance video surveillance and ID checks)

that are protected by four models for privacy protection:

- Comprehensive laws (general law)
- Sectoral laws (opposite to general law)
- Self-regulation (codes of practice)
- Technologies of privacy (individual users use of technology)

In [37] they state that:

"Of all the human rights in the international catalogue, privacy perhaps is the most difficult to define. Definitions of privacy vary widely according to context and environment. In many countries, the concept has been fused with data
protection, which interprets privacy in terms of management of personal information.”

Hence, the term privacy, as indicated by the list above can be cluttered. In relation to the SNE service we use the following aspects of privacy:

- Information privacy (collection and handling of personal data)
- Privacy of communications (security and privacy of communications, for instance telephones and e-mail)

### 5.2.2.2. Privacy preferences

According to the article *Understanding and Capturing People’s Privacy Policies in a Mobile Social Networking Application* [15], people have expressed concerns and are apprehensive about the privacy implications associated with software that allow users to locate one another. The article explores technologies that empower users to more effectively and efficiently specify their privacy preferences (or ‘policies’) to address these concerns adequately. Technologies such as User Interfaces for specifying rules and auditing disclosures, and machine learning techniques to refine user policies based on their feedback are taken into consideration. The purpose is to better understand people’s attitude and behaviors towards privacy as they interact with the application. Their studies, performed as lab and field studies, show that people are not good at articulating these preferences and the accuracy only marginally over time increases unless they are given tools that help them better understand how their policies behave in practice. The studies confirm that privacy preferences tend to be complex and depend on a variety of contextual attributes (e.g. relationship with requester, time of the day, where they are located).

In a newly posted blog article with the title (translated from Danish): *Facebook is about to make location technology mainstream. Consequences incomprehensible* [32], written by Anders Colding Jørgensen, Cand. psych Internet, in the psychological domain, states that one of the next big things
for the users of Facebook is to be concerned about the status of the location where they are recorded. Is it a fine restaurant or is it a fast food place, is it the “right” shops. Instead of trying to hide ones whereabouts, the trick is to reveal it. Some of the comment to the blog entry takes the idea even further, and indicate that in the future people will hire other people to attend the right places for them. These people are a kind of location managers.

The blog entry in [32] has the direct opposite attitude towards privacy in relation to location data as opposed to the article in [15] and [10] below.

This YouTube video recording of a CNN feature on rising Facebook privacy concerns [38], debates that Facebook users profile data and friends are exchanged with another web site and when a user visit that web site the user can see for instance how their friends has interacted with that web site, thereby revealing third party actions.

Senator Al Franken states that this is a consumer protection issue and the users have to opt in. Senator Mark Begich states that the problem is that the users didn’t accept that kind of user information exchange at the time they signed up to Facebook.

The problem is that the user has to opt out instead of opt in and that opt out procedure is a complicated process for the user, and therefore the users has less control, Senator Charles E. Schumer states in this YouTube CNN feature video [39]. Therefore he has initialized political steps and has asked the Federal Trade Commission to come up with new rules regulating how private information is shared between social networking sites. In relation to [37] this is an attempt to introduce sectoral laws to enforce rules on user privacy.
Facebook wants to try out its new service with the: Docs, Yelp and Pandora web sites. After this political and media storm about the concerns for the users privacy Mark Zuckerberg the Facebook founder has announced in his blog [40] that they have implemented a new privacy model where the users interest are taken into consideration and it is much simpler to control the privacy settings. The users though still have to opt out. This action is thereby in relation to [37] an attempt by Facebook to make usage of self-regulation.

Is it possible for us to prevent that our use of location data becomes a threat to the user’s use of the application by implementing the experiences in [15] and in [32] in some sort of way and take notice of the above mentioned Facebook handling of privacy problems. And which of the experiences can we use in our application taken its service nature in to consideration to make the users feel comfortable in regards to privacy when using the application?

We will not make use of machine learning techniques to refine user policies based on their feedback, and we will also not make use of auditing disclosures. Therefore this will not be further analyzed.

First step is to examine the basic nature of how preferences are implemented in Android. And how, if it is possible, to implement our preferences in a way, that gives the users a possibility to better understand
the preferences and the functions they represents. The second step is to examine if it is possible for us to make use of User Interface technology to make it possible for the user’s to specify rules. The third step is to examine if the high-level perspective grouping of privacy can be used in our application.

The question about which preferences we need in the SNE application in order to make it possible for the users to specify their rules related to privacy, and the preferences initial state, will be attained below.

### 5.2.2.2.1 User Interface Aspects

**Android preferences**

In *Pro Android 2* [20, ch. 11], the Android preferences are examined and their work is used as basis for the development a simple prototype in order to gain an overview of the Android Preferences.

We expect the Android users to be familiar with the Android user interface from their previous use of other Android applications. They know how to use the menu button that is built into the device casing to open the menus available to the active application. They also know that when the menu Settings is activated then the settings available to the application are presented. Users that are new to the Android User Interface will quickly learn how to navigate to and make use of the available settings in the SNE application by their use of other applications installed to their device, if we make use of a uniform way to implement this functionality, the “Android built-in way”.

60
In an Android application the user make use of the menu button in the device casing to open the application menu. When the button related to the settings in the menu is activated (figure 11) the available settings are displayed. They can be shown as a group of settings (figure 12) or simply just as the available settings (figure 13). If grouping of the settings are used as in figure 12, then if one of the items in the group is activated then its related grouped settings are shown as in figure 13. Notice that we have omitted the ListPreferences and the RingtonePreferences because they are not used in the prototype.

As mentioned in [15] a tool to better understand an applications policies may in practice be useful. Maybe the use of an Android Dialog that displays their present privacy settings at application startup could be such a tool (figure 14 below). Though it is extremely simple the user quickly gains an overview of their current privacy settings.

The use of Android Dialog could also be used at first time use of the application to enforce that the users commit oneself to their privacy settings (figure 15 below).
Another issue to take into consideration, regarding the preferences in relation to better let the user understands them and defines their own rules, is the possibility for applying some sort of helping text to all the items in the settings.

As shown in the prototype figure 16 it can be done by adding more text to the summary of an item. The text that was added for the purpose of analysis was a text that is normally used as dummy text for typographical work in relation to web development and design [30]. The text consists of 1200 characters with space.

As shown in figure 16 (horizontal) and in figure 17 (vertical) the text is shortened. Horizontal view allows for more text than the vertical view, but nor the horizontal or the vertical view allow for display of all the text.

It can be a possibility to display text about the preferences before they are shown but in the same view, as it can be seen in figure 18 below. Notice that this figure is a snapshot from an Android device which makes use of the Danish language.
Another option could be to implement a menu item with access to a web site containing a very informative explanation of the SNE application as a service and its use of functionalities that might raise user concerns (maybe a list of pros and cons when revealing ones whereabouts), and its preferences to control these functionalities. Of course in conjunction with the initiatives mentioned above.

An option could also be to implement a menu item with access to a view containing help in relation to the privacy settings as shown in figure 19 and in figure 20 (below).
Of course the view has to be nicely formatted and it must be a scrollable view, but as it can be seen all of the text now is available for the user. This way the user can get help when they are attending an event and need instant help with their privacy preferences, without leaving the application.

Specify rules

Both article [15] and the blog entry [32] triggers that the users must be able to specify some rules in relation to privacy. Article [10] mentioned below about re-identification also includes issues of relevancy for the rules. When will the user be visible and when will the user be invisible. We must let the user decide when they want to be related to for instance a particular place or a particular event, or wants to be visible for other users (future work).

The overall nature of the SNE service also triggers the need for some rules. The application makes use of the users contact list on their Android devices in order to make a match. But what if the contact list contains a mistress or a person that you have been part against in court proceedings? Do the users
want to initiate a conversation with other unknown users with outset in those persons? We don’t think so.

The application will make it possible for the users to contact a matched user either by phone call, SMS messaging and add to contacts request. Some might not want to be disturbed this way. And all actions done by the user or the system during the user’s use of the application will be logged to history log with event-tagged values. Some might not be interested in that too.

When the application starts for the very first time what are the initial settings for the rules? We believe that all rules which can relate to privacy concerns must be defaulted as not enabled. Then it is up to the users to make the decision about how much of their whereabouts they want to reveal.

5.2.2.3. Re-identification

In the article I Know What You Did Last Summer: risks of location data leakage in mobile and social computing [10], re-identification based on the user’s movements and publicly available information is the primary privacy of concern. One of the main issues in the article is:

“How much could be discovered by analyzing a regularly updated dataset of location records over a period of time and how location data can be misused.”

Re-identification is, in the article [10] described as a process where a user is matched in two datasets using some linking information (e.g. name and location) in order to disclose for instance the user’s identity, and find the user’s details and interests. Public available information from social networks like Facebook, LinkedIn and Twitter may be used, as well as ordinary Google or other search engine searches may be used to make a cross reference. Techniques like location data mining can be used to discover relation between the user and place, and can be combined with pattern recognition in daily movement records. A successful re-
identification may not only violate the user's privacy but may also be a threat to the security.

One of the concerns could be burglars. If the user is attending an event in another town and that location of the user is posted real time to a public accessible place then everybody might know that nobody's home. The user will take this into consideration because recently there has been a lot of focus on that issue.

Another concern could be the theft of the users identify and afterwards misuse it in various ways.

The event-tagging used in the SNE application does not record the user's whereabouts to a public central server so there is no chance of a continuous recording and therefore no fine-coarse pattern matching can take place. That said the users do exchange some basic user information and they do allow, if add to contacts are enabled, other users to tag in their contact list the event and event location they have attended when adding each other to their contact lists.

### 5.2.2.4. Third-party entities

The SNE makes use of the users contact lists in order to make a matching. To do that contact list data has to be exchanged between the devices. This can be a threat to the privacy of the third-party entities if the contact list data is transported in an insecure way and if the contact list data is readable for other users from their devices. These security issues have to be dealt with (5.2.1).

### 5.2.2.5. Device Owner User Information

Some users might see it as privacy issue that the application automatically
exchanges the local user information (name, title, company and description) with remote devices, but the nature of the SNE service demands for some user information to be visible for other users prior to requests for add to contact list. The user information must be seen as a sort of advertisement of that particular user and is used to trigger some sort of interest by the other users, and hopefully some use of the applications contact functions. Either by direct phone call, SMS or a request to add to contact list.

5.2.2.6. SNE preferences

From the SNE application nature we find that there is a need for preferences in relation to:

- Communication
- Logging
- Contact List Filtering

The privacy preferences must consist of Phone call, SMS and Add to contacts. If they are enabled then it is possible for other users to directly via the Social SNE application to make direct phone calls and send SMS messages to the user. It is also possible for other users to initiate an Add to contacts request. If a user accepts such a request the contact is added to the contact list with event-tagged data constituted with data from the calendar, the GPS sensor and the application.

The logging preferences consist of Logging and if it is enabled then history logging is turned on. When turned on all activities registered by the application will be logged as geo-tagged data and the user’s interaction with other users will be included in the other user’s log. When turned off no activities will be logged and the user will not be included in other user’s log.

The contact list filtering preferences consist of Contact List Filtering. When enabled it must be possible to select entries in the contact list to be omitted
when the system generates the prepared contact list used by the matching function (section 5.1.3 Matching).

5.2.3. Findings in Relation to Security and Privacy

Security

The following has been concluded in relation to the security analysis:

- The Bluetooth pairing concept contradicts with the nature of the SNE service. To develop a seamless SNE service this must be bypassed.
- It is not possible to implement authentication, when users need to communicate anonymously and without the help of a centralized server.
- The confidentiality gained in Bluetooth 2.1 for data exchanged on the network in good enough for the nature of the SNE system.
- It is not enough to secure confidentiality for data exchanged on the network. To preserve user privacy the contact lists exchanged must be stored in a none-readable form on remote devices. SHA-1 and SHA-2 are suitable cryptographic hash functions for this purpose.
- The SNE does not need to enforce any authorization rules automatically, but users need to authorize each other to allow being added as a phone contact on a remote device.
- It is not possible to ensure accountability when no authentication is present.

SNE must warn users regarding the possible dangers when having the Bluetooth radio turned on.
Privacy

The SNE service usage of location data and exchange of contact lists may trigger privacy concerns for the users and third party entities (person in the contact lists). The privacy concerns might be the consequences of re-identification.

We have to kinds of users – users with privacy concerns and users without privacy concerns. The main focus must be to make the users with privacy concerns comfortable using the service, by using of tools such as dialogs, helping texts and preferences.

We have found that the SNE application has a need for preferences in relation to:

- Communication
- Logging
- Contact List Filtering

The users must be able to manage the above mentioned preferences in order for the users to realize the potential by using the application. If a user wants privacy the user disables the preferences and if the user wants less privacy the user can select that level of privacy.

As shown in the Android Preference prototype (above) we believe the use of tools to let the users get a better understanding of the preferences available to them and thereby get a higher level of preference usage, will be a good design choice. The use of Android dialogs to display for instance the current privacy settings or the initial privacy settings will be useful information for the user (maybe they can turn the show of dialog on/off in the settings). Also the help view in relation to the settings and other helpful information will be useful for the user, because they don’t have to leave the application, and they get a better understanding of the privacy issues.
We do not have a need for the user to add new rules dynamically. Thereby we mean that the user can’t dynamically add new preferences or groups of preferences to the application. At the moment we also do not want to make use of any sort of feedback feature in relation to preferences and privacy concerns. It would be a nice way to stay in touch with the users movements in the privacy concern domain and therefore it might be relevant for us in future works.

In order to protect the privacy of third-party entities (contact list items) only the entity’s name must be revealed when there is a match between the users and it must be the name from the local contact list. The sharing of the contact list has to be done in a secure way in order to maintain third-party privacy. This trigger some security issues (see section 5.2.1 about security).

In order to advertise the user to remote users, the nature of the SNE service requires the user to make the user information (name, title, company and description) public to remote users.

Finally we are aware of that the application only will be a success if the users are willing to use it in a relaxed privacy mode, but we can’t force the users to make use of the application in our preferred way. Facebook tried to do that by self regulation, and now they are partly revising their privacy policy [38] after intense media and political criticism [39] [40], that might result in legislation.

Therefore in the Social Network Extender application all the preferences have to be defaulted to not be enabled. And as mentioned above, the user must be in control their level of privacy and has to opt in.

5.3. Service

The overall purpose of this section is to design the SNE system as a service.
In order to gain an understanding of the terminology used the first section focuses on defining what service, service design and context awareness is all about. The second section will examine the method to be used for designing the SNE service. The last section will end up using the defined method for designing the concrete SNE service.

5.3.1. Service Design and Context Awareness

The new frontier of interaction design, according to [27], are services where the focus is on context in the ways of performing tasks with the purpose of designing the system as a whole (service design), and the system is the service. In traditional design the focus is on the relation between the user and a product. In the book The Craft of Service Design, Chapter 8, Service Design [27], the author try to give an explanation of what a service is, why a service must be designed and the methods used to craft services.

In the article Service Blueprinting: A Practical Technique for Service Innovation [28], a service is described as dynamic, and frequently co-produced in real time by users and technology, and is modeled by a technique called Service Blueprinting with the focus on the user in order to visualize the service processes. According to the article one of the most distinctive characteristics of a service is its process nature:

“Unlike physical goods, services are dynamic, unfolding over a period of time through a sequence or constellation of events and steps. The service process can be viewed as a chain or constellation of activities that allow the service to function effectively.”

Hence, it is vital for us to gain knowledge of the service processes, and Service Blueprinting, as described in [28], can help us with the challenges of service process analysis and design. Service Blueprinting is also a step in the method described in [27] to craft a service, which consists of the steps
Process Map and Service Blueprint. Like in [27] its focus is on the way the user over time interacts with the service at many different touchpoints.

According to the article *Understanding and Using Context* [29] context is poorly used as a source of information and there is an impoverished understanding of what context is and how it can be used, to increase the communication in human-computer interaction and thereby make it possible to produce more useful services. In order to make use of context effectively there must be an understanding of what context is and how it can be used.

In [29] they give the following definition to context:

“Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”

The entity is similar to a touchpoint in [27] where a touchpoint typically are the environments, the objects, the processes and the people. But that’s not the only similarity. In [27] a service consists of multiple touchpoints and focus in relation to service design has to be on these multiple touchpoints and the user’s interaction with them over time. It’s that moments that are of interest.

According to [29] it is of most importance that we get an understanding of the context and how the context can be used in the development of the SNE service, because this will help us to determine what context-aware behaviours to support. In order to get that understanding we can make use of interaction design in the form of Service Design to get an extended knowledge about the application as a service and the context where it interacts.

In [29] they give the following definition to context-awareness:

“A system is context-aware if it uses context to provide relevant information, and/or services to the user, where relevancy depends on the user’s task.”
According to Alex Küpper [33] context can be grouped in two categories, primary and secondary contexts as shown below.

Primary:
- Time
- Objects nearby
- Network bandwidth
- Orientation
- Light
- Sound
- Temperature
- Pressure
- Physical location
- Other objects

Secondary:
- Social situation
- User activity

The secondary contexts are more abstract and can typically be derived by a combination of the primary contexts. The combination of the primary contexts could for instance be information obtained by the device sensors e.g. GPS, compass and Bluetooth, and the device calendar and contact list, all together used to define the users actual social situation.

The context will be used in service processes and in Service Design a service process is defined as how the service is acted out; how it is ordered, created and delivered. As stated in [27]:

“Designers can’t control the entire experience. However, interaction designers do have to define and design at least some of the pathways through the service. These pathways contain service moments - small parts of the experience - which, when hung together, constitute the service and its experience.”
By thoroughly looking at all the moments along the user’s pathway through the service, we can get the necessary knowledge about our system as a system that adapts its behaviour according to the actual context.

According to the article Note about Context-Awareness [30], the challenge for us as developers of a context-aware system, will be to make it possible for the system to adapt to the actual context, and as shown in figure 21 below, there are four types of behaviour: Tagging, Displaying, Helping and Execution, that thoroughly must be considered as a combination of the three factors: Accuracy of Context Information, Confidence in Action and Consequences.

The tagging action saves the context information for later use. The Display action is when a system shows the context information without taking notice of any actions carried out. Both the helping and the executing actions take into consideration which actions to perform. The difference between them is that the helping action only prepares the system to the action while the executing action without user involvement automatically performs the action.
We can incorporate the “context-awareness model”, as shown in figure 21 above, into the Service Design method for crafting a service, by adding it to the step Service Blueprinting, and rename the service design method to The Craft of a Context-Awareness Service.

The purpose of the method change is to get a thoroughly understanding of context and context-awareness issues.

### 5.3.2. The Craft of a Context-Aware Service

The SNE service seen as a service, where the service and its characteristics, as defined in [27], is the chain of activities that forms the process and has value for the end user, has the following characteristics:

- Intangible
- Provider ownership
- Co-created
- Flexible
- Time based
- Active
- Fluctuating demand

and is crafted using an interaction design process that includes the following steps (after our changes):

- Environment Description
- Stakeholder Description
- Touchpoints
- Service Blueprint
  - Service Moments
    - Execution of the “context-awareness model”
and consists of multiple touchpoints. Notice that we have moved the step Process Map and added it to the step Service Blueprint so that the step is aligned between [27] and [28]. It’s the Service Blueprint step that can be used to examine how to make the service context-aware, now even more thoroughly after the incorporation of the “context-awareness model”. The Company Branding and Perception step originally included in [27] is omitted because the focus is on the service and its support for context-awareness and not on branding.

The first two steps, Environment Description and Stakeholder Description describe the overall scenario and its users. This description can also consist of for instance photos, videos and so on. The next step, Touchpoints is generated as a list. That list is used in the step Service Blueprint, both in the overall description and in the Process Maps. The Service Blueprint includes the step Service Moments, which are composed of a set of discrete moments (events) that can be designed with outset in the used touchpoints. If there are multiple paths then there are multiple designs [27]. We omit sketches and storyboards in relation to the moments, but each moment should show what service elements that are affected, e.g. the touchpoints [27]. For each moment found during this service design process, the “context-awareness model” is run. The step Service String is usually used to show the big idea about the service in the form of writings and storyboards that include one or more moments (events). We include the Service String in our Service Blueprint because the resulting prototype shows the big picture together with the Process Map. Again we omit the storyboard. The step Process Map results in the creation of a visual overview that shows the high-level view of the overall experience, the boundaries and the touchpoints at each stage of the process [27]. The last step Service Prototype will result in a service prototype, constituted by all the individual moments and collection of moments prototypes created during the service design process. We simply have chosen to make use of
prototypes instead of creating storyboards or for instance sketches. The Service Prototype will result in a prototype created with the use of screen dumps from the Android device.

To round off the service characteristics, mentioned above, it is intangible because the service itself is ephemeral. It provides provider ownership because the service cannot be purchased by the user; only a usage fee can come into consideration later on. The service is co-created because the service can’t exist without the involvement and engagement of the users that delivers the data used by the system and the users themselves. If there are no users then the service has no content and other users won’t use it. The service is also flexible because the service can adapt to the users new situations when the users set their level privacy in the preferences (5.2.2). Finally the service has a fluctuant demand which means that the service may vary in time according to the events and the number of guests attending the events and their social mood.

5.3.3. SNE Service Design

5.3.3.1. Environment Description

The application will be located at the users Android devices and will be used physically at the places of the events they attend to and where they together with other users make use of the application are thereby are parts of the co-creation of the service.

The SNE service can operate in many different environments. We have no control of where the service is established between the users. The following need to be taken into consideration:

- Indoor/outdoor usage (satellite coverage)
- Noise from other electronic devices (radio frequencies and magnetic disturbances)
- Number of SNE devices

5.3.3.2. Stakeholder Description

There are two types of stakeholders [27]:

- Obvious
- Non-obvious

where the obvious stakeholder is a user that uses the application in a relaxed privacy mode, thereby making full use of the application and the functionalities it provides as a service, and the non-obvious stakeholders is a user that uses the application in a strict privacy mode and the persons figuring on the contact lists of the users (third parties). There are no prioritizing between the obvious and the non-obvious users.

5.3.3.3. Touchpoints

The environment is described above. We have no control of the environment because the users can use the service where and when they want to. The environment hence is therefore typically an event with the possibility of making some sort of social networking. In order to constitute a service with the possibility of a match at least two Android devices using Bluetooth with the application running must be present.

The objects are theses Android devices with the Social Network Extender application installed and running, and thereby making use of the service. It is also the device calendar, the device contact list, the GPS sensor, the Bluetooth adapter and the Compass sensor.
According to [27] the process is how the service is acted out; how it is ordered, created and delivered, and it can be designed by mapping the pathways to moments. That we will do below.

The people or as we say - our users, are those that have the application installed and running on an Android device. They are essential for the service and there are, according to [27], three types of users to design for:

- The users
- The employees
- Third parties (entries in contact lists)

In the SNE application as a service there is only the user’s to take into consideration. The employees don’t make actions at service runtime that interacts with the users, so there are no co-creation of the service between the employees and the users in real time.

List of touchpoints:

- Physical location (event address)
- The Social Network Extender application
- Device GPS sensor
- Device Calendar
- Device Contact list
- Device Compass sensor
- Device Bluetooth sensor
- Users

The listed touchpoints will be used both in the Process Map and in the Service Blueprint.

5.3.3.4. Service Blueprint

In this section the SNE application as a service will be blueprinted. The
segments of users in focus are all users that have the SNE application installed and running on their Android device.

The Service Blueprint, as a visual view, in the form of a Process Map typical consists of five components, according to [28]:

- Customer Actions
- Onstage/Visible Contact Employee Actions
- Backstage/Invisible Contact Employee Actions
- Support Processes
- Physical Evidence

but as mentioned above there are no interactions between the employees and the users. In [27] the Process Map at page 191, Figure 8.12, consists of 3 components:

- Phase
- Actions
- Problems

We will make use of: Customer Actions, Support Processes and Physical Evidence, where Phase is identical to Physical Evidence and Actions are identical to Customer Actions. Finally we will make use of both Problems and Support Processes. Then it will be possible to map the user actions with the physical world, where it is relevant, and also map the user actions to some necessary support processes. Do problems arise then we can describe them.

Before we make our Process Map we first make a list including the moments in the service.

The SNE has the following Service Moments:

1. User starts the application
2. User sets the user information
3. User sets preferences
a. Privacy  
b. Logging  
c. Contact List Filtering  

4. User is matched  
5. User selects a match  
6. User contacts a matching device user  
   a. Contact for add to contact list  
   b. Contact via direct call  
   c. Contact via SMS  
7. User receives a request to Add to contact list  
8. User receives a direct call  
9. User receives a SMS  
10. User minimizes / maximizes the application  
11. User read the log  
12. User gets GPS notifications  

With outset in the list above we analyze and design those moments, and run the “context-awareness model” in the moments of relevance. We have omitted the deployment of the application in this work and focus on the user usage.  

1. User starts the application  

The user starts the application and the first thing that happens is a supporting process that checks for the initial application states (settings) to be correct. First there is a check for the presence of a Bluetooth adapter in the device. If there is no Bluetooth adapter the only option for the user it to shutdown the application (figure 22). If there is a Bluetooth adapter in the device the supporting process checks if it is enabled. If it is not enabled then the user can chose to enable it or to shutdown the application. If the user chooses to enable it the support process enables the Bluetooth adapter and continues as if the Bluetooth adapter was already enabled (figure 23).
Now the supporting process checks if the user info (2. User sets user info) and the preferences (3. User sets preferences) has been set (figure 24, 25 and 26 above). When they both are set the supporting process continues by displaying a status view of the privacy settings. The status view will always be displayed at startup (figure 27 above). When the user clicks on the OK button the application starts as a service.
2. User sets the user information

The user starts the application main menu by pressing the menu button on the device casing and doing it from the main view. In the menu the user selects and clicks on the My Info button (figure 28). In the My Info view the user sets the user info and press the Update button. The user can press the Cancel button to undo an update (figure 29 and 30).

3. User sets preferences

The user activates the main menu and select and press the Settings button.

As seen in figure 31. This opens the Settings view containing the group of preferences: Communication, Logging and Contact List Filtering (figure 32). If the user presses the menu button on the device casing the menu: Help to Settings, as specified in section 5.2.2 Privacy, appears (figure 33). If the user clicks on that menu a view that contains help in relation to the settings of preferences, privacy concerns and advantages are shown to the user (figure 34 below).
If the user selects Communication in the Settings view (figure 32) then the Communication view (figure 35) will be shown to the user. This group of preferences includes: Phone call, SMS and Add to contacts, and define the rules for the other users when they want to communicate with the user.

If the user selects Logging in the Settings view (figure 32) then the Logging view (figure 36) will be shown to the user. This group only contains one preference: Logging, and define the rule for this device and other user’s devices in relation to log the user’s history of usage.

Figure 34, 35 and 36 are shown below.

If the user selects Contact List Filtering in the Settings view (figure 32) then the Contact List Filtering view is shown (figure 37).
This view contains all the items in the devices Contact List for the user.

It is here possible for the user to select those items in the contact list that will be omitted by the systems matching process.
4. User is matched

Every time the service main window is activated by the user, either at start up or when the user returns from another view, the supporting process matching function is activated to generate a list of other users in which the user has a mutual relation with (figure 38 and 39). The users are shown with a compass indicator to show the last known direction of the users in relation to the user and an indication of the users present status: active (green) or passive (red) e.g. inside outside range of the Bluetooth network.

![Figure 38](image1)
![Figure 39](image2)

In order to find the mutual relations between the user and the other users the supporting process discover function first must find all the other users with the Social Network Extender application as a service running within range. Then for each user found the matching function exchange a list of extracted data from the user’s device contact list and make a match between the contents of the lists.

The user will be notified every time a new match is found (figure 39).
A supporting logging process logs all matches to history log if logging is enabled.

**Context-awareness**

According to the context awareness model the SNE service is context aware according to the following:

- **Showing remote users with matching contact lists:**
  The SNE application is automatically performing the matching *(executing)* and showing the matched users *(displaying)* without any involvement of the user.
  
  o The context used are “nearby SNE devices” and “phone contact lists”, which are used to infer the social matches *(social context)*. The accuracy of these contexts is very accurate as the hashed phone number are unique and the Bluetooth device discovery is very reliable.
  
  o We are pretty confident that the displayed matches will indicate a social relationship between the users, but it assumes that the user is using the contact filtering to filter out non-social contacts.
  
  o The consequences of a wrong match can result in privacy issues, where the user is related to persons to be filtered out. The can result in rejected conversion.

- **Logging remote users with matching contact lists:**
  This is very similar to showing the matches, but the action is *tagging* instead of *displaying*. The bullets below describes the difference.
  
  o As an extention the “physical GPS location” is used when logging a match.
The accuracy of the GPS location very much depend on whether the SNE service is used indoor or outdoor, as GPS satellite coverage is best outdoor.

The consequence of bad GPS coverage is simply that the match will not be tagged with the physical location.

- **Showing state and physical direction of remote user:**
The SNE application is automatically continuously updating the physical direction and state of the remote user (displaying).

  - The context used are “physical location”, “physical direction of local device (orientation)” and “is device nearby”. As mentioned earlier the GPS accuracy depends on indoor or outdoor environments. The accuracy of the compass (to calculate direction) depends on the influence of other magnetic nearby objects.

  - The accuracy of the compass could result in significant deviations.

  - The consequences of a wrong direction can make it difficult for users to locate each others.

5. User selects a match

When a user wants to know more of the users and/or want to contact the users in which the user has a mutual relation with, the user selects and click on that user from the list of matches in the main window (figure 40). An entry in the list is constituted by the users name, title, company and present status (active/passive).
In the contact person view (figure 41 and 42) the user can see the user info of the selected user, and who their mutual relation(s) are. It is also possible to initiate digital contact to the user in three ways: Add to Contacts, Call and SMS, but only if it enabled in the remote users preferences.

**Context-awareness**

According to the context awareness model the SNE service is context aware according to the following:

- *Showing communication possibilities of a remote user:*
  
  Based on the preferences of the remote user the SNE application automatically shows (*displaying*) communication possibilities (add to contacts, call and SMS).

  - The context used is “remote user preferences”, which are used to find the communication possibilities to offer in the user interface. The accuracy of the context is assumed to be high.
We are pretty confident that the communication possibilities are according to the wishes of the remote user, as the preferences of the SNE application is easy to use (see privacy section 5.2.2).

The consequences of a not wanted communication possibility could be that a remote user is unwillingly contacted.

6. User contacts a matching device user

The user sends a request to a matching user by clicking the Add to Contact button (figure 42 above). If the matching user accepts the request or (figure 43) the matching user rejects the request (figure 44), then a message is shown indicating that.

If the matching user accepts the request then the supporting process creates an entry in the users contact list and adds information about the social
status e.g. the location and the event, by use of the device GPS sensor and the user’s calendar. The data is also written to history log if Logging is enabled.

If the request is rejected no actions are taken by the supporting process.

The user can also contact a matching user by phone call or by SMS. If the user click on the Direct call button (figure 42 above), then a normal phone call is initiated. If the user click on the SMS button (figure 42 above), the supporting process sends a default SMS message to the matching user, telling the matching user that the user are interested in contact.

*Context-awareness*

According to the context awareness model the SNE service is context aware according to the following:

- **Add to contacts request:**
  When the user presses the “add to contacts” the SNE application prompts the remote user to accept or deny the request (*helping*). The context used can be several kinds of information about the situation of the remote user – e.g. mood, busy/not busy etc., which cannot be automatically sensed by the SNE service.

- **Accepted add to contacts:**
  When the remote user has accepted the add to contacts request the SNE application add the remote user as contact (including physical location and event) in the contact list of the local device.
  - The context used are “physical location”, “time and “phone calendar” As mentioned earlier the GPS accuracy depends on indoor or outdoor environments. The accuracy of the
event information put on the contact depends on how the user is using the phone calendar.

- If the calendar item found using the time context attribute does not tell anything about the event that the user is attending, then event awareness is useless for the user.

- The consequence of a wrong awareness is simply that the saved contact contains irrelevant information.

7. User receives a request to Add to contacts

When the user receives a request to add to contacts by another user a message is shown with information about the requesting user and who the mutual relation(s) are (figure 45).

If the user accepts the request then the supporting process creates an entry in the users contact list and adds information about the social status e.g. the location and the event, by use of the device GPS sensor and the user’s calendar. The data is also written to history log if Logging is enabled.
Context Awareness

Similar to the context awareness in the section above. The only difference is that the situation is seen with the eyes of the remote user.

8. User receives a direct call

The user answers the phone as usual.

9. User receive a SMS

The user receives the default SMS the normal way (figure 46).

![Figure 46](image)

10. User minimizes / maximizes the application
If the user wants to make other things on the device while the Social Network Extender as a service is running, the user can choose to minimize the service. To do that the user selects and clicks the Minimize menu item in the main menu (figure 47 below). Then the system minimizes and an application icon appears in the device notification area (figure 48 below).

To maximize the application the user slides down the device notification and selects the Social Network Extender (figure 49).

11. User read the history log
The user selects and clicks on the History menu item in the main menu to read history logging (figure 50 above). In the history logging view all the user’s usage in relation to this device, the matches found and all the request to add to contacts and the users action are displayed (figure 51 above).

12. User gets GPS notifications

The user receives notification by the supporting process if the GPS sensor is turned off (figure 52) or if the satellite coverage is bad and no location data are received (figure 53).
5.3.3.5. Service Prototype and Process Map

Designing the moments in the Service Blueprinting has resulted in a prototype. The prototype has been implemented on a User Interface level and all the functionality, e.g. matching and so on are sketched.

The Process Map has been worked out in hand according to the prototype description. As the process map is very simple it is not shown.

5.3.4. Our findings in relation to Service Design

Service Design and Context Awareness

From the SNE service nature we know that it has the most of the characteristics of a service. We also know that it is a context-aware application making use of location- and social-awareness
By looking at the definitions of context, context-awareness and service design we find similarities. By extending the service design method with the context awareness model we have defined a good method to design the SNE service. As a result we will get a prototype of the service with focus on context awareness.

**SNE Service Design**

Using “The Craft of a Context aware Service” method we found the user paths through moments of the service. By finding the touchpoints of the service we have found the entities relevant for the interaction between the service and the users in the service moments.

In each of the service moments it has been considered how to make the SNE service context aware according to the context awareness model. For each of the context aware situations the accuracy of the contexts, the confidence in performing the correct action and the consequences of a wrong action has been considered. Based on this the most appropriate service action has been selected.
6. System Overview

The SNE service overview depicted below includes four Android devices: A, B, C and D. The black arrow depicts various type of communication between devices making use of the SNE service.

![Figure 54 – The SNE service overview](image)

Device A and device B are within Bluetooth range and can discover, match and exchange, thereby making use of the SNE service. They can also communicate by SMS and by call (phone). Device D are not in range with any devices at the moment but it has already earlier been discovered and matched by device B and therefore they can communicate via SMS and call (phone) when outside Bluetooth range. Device C is like device D also not within Bluetooth range with any device and it can’t communicate with any devices via the SNE service.

From figure 54 the SNE service might seem quite simple, but it is an example of a physical environment consisting of only 4 devices. However, a real-life environment could consist of many devices, which all want to
discover- and exchange data between each other. Image a conference having 100+ co-located devices running the SNE service.
7. Software Architecture

This section examines the software architecture of the SNE system according to the process it has been developed, which is reflected by the following steps:

1. Architectural requirements: The first section will state the requirements put on software architecture, by examining the quality attributes of the SNE system.

2. Evaluation of the architecture: Quality attribute scenarios are used to make the requirements concrete and measurable.

3. Architectural description: The third section will describe the actual architectural design, which is going to support the requirements defined in the first section.

7.1. Architectural Requirements

Before we start building the software architecture of the SNE system, it is important to know what the architecture is going to accomplish – i.e. what are the requirements put on the architecture.

Given the functionality of the SNE system the architecture you might think that the requirements are clear. However, considering the functionality of the system alone, the architecture can be designed in almost any way. It does e.g. not indicate anything, which can help the architect to find the most suitable way to structure the system into software components and how to let the components communicate together. An example in the SNE
system is the functional requirement that it must be able to discover nearby
matching users. This alone does not tell us that the architecture actually
must secure, that this function executes in very fast way.

Instead we need to know about the desired quality of the discovery and
matching function. E.g. many devices must be discovered simultaneously
and the devices might be mobile and thereby quickly get out of network
range – i.e. the performance of this function is an important quality. Put in
other words, to help finding the most suitable architecture we need to know
about the quality attributes of the system that needs to be constructed
[Software Architecture in Practice; [53]].

Quality attributes are classified the following way according to [53]:

- **System qualities**: e.g. availability, modifiability, performance,
  security, testability and usability.

- **Business qualities**: e.g. time to market

- **Architectural qualities**: e.g. conceptual integrity, build-ability,
  correctness and completeness.

### 7.1.1. System Qualities

The following system quality attributes has been identified as being
significant for the SNE system:

- **Performance**: Specifically the performance of the device discovery
  and the phone contact list matching need to perform very fast, as
  the devices might be mobile and thereby quickly can move out of
  network range (see section 5.1.2 Bluetooth).
• **Scalability:** Many devices must be present in the environment of the SNE system. This requires that multiple devices can be discovered simultaneously and that a device must be able to communicate point-to-point with multiple devices simultaneously (see section 5.1.2 Bluetooth).

• **Security and privacy:** The security and privacy of the SNE users are significant quality attributes (see section 5.2.2 privacy and section 5.2.1 security).

• **Modifiability:** It must be as easy as possible to extend the SNE system to use alternative network technologies like Wi-Fi (see section 11 future work). Furthermore, it must be easy to reuse parts of the SNE architecture in other pervasive applications, which are based on similar Android-based technologies.

• **Usability:** It must be easy for the SNE users to change privacy preferences using the graphical user interface (see section 5.2.2 privacy)

### 7.1.2. Business Qualities

As the business possibilities are not focus of this thesis, the business qualities are obviously not the focus of the SNE architecture. If the SNE system was developed as part of a commercial project, then we of course might have had another situation.

### 7.1.3. Architectural Qualities
As the SNE system is a relatively small system, the architectural qualities are not the prime focus of this thesis. It is of course strive for conceptual integrity, where the similar things in different components are designed in similar ways, but when the system is small enough to be implemented by a couple of developers, there should be no need to apply specific mechanisms or tools to enforce the integrity.

### 7.2. Evaluation of the Architecture

In order to make the architectural requirements concrete and measurable/testable the following sections provides the relevant quality scenarios [53] for each of the quality attributes described above.

The actual test of the architecture is documented as part of the evaluation section 9, where the quality attribute scenarios are mapped to experimental hypotheses.

#### 7.2.1.1. Performance Quality Scenarios

<table>
<thead>
<tr>
<th>Source</th>
<th>Two SNE users (A and B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>B enters the network range of A</td>
</tr>
<tr>
<td>Artifact</td>
<td>The SNE system</td>
</tr>
<tr>
<td>Environment</td>
<td>A’s device is discovering devices</td>
</tr>
<tr>
<td>Response</td>
<td>B is shown in the list of matched users on A’s device</td>
</tr>
<tr>
<td>Response measure</td>
<td>The operation succeeded within 10 seconds.</td>
</tr>
</tbody>
</table>
7.2.1.2. Scalability Quality Scenarios

<table>
<thead>
<tr>
<th>Source</th>
<th>Four SNE users (A, B, C and D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>B, C and D enter the network range of A</td>
</tr>
<tr>
<td>Artifact</td>
<td>The SNE system</td>
</tr>
<tr>
<td>Environment</td>
<td>A’s device is discovering devices with B, C and D as moving devices.</td>
</tr>
<tr>
<td>Response</td>
<td>B, C and D are shown in the list of matched users on A’s device</td>
</tr>
<tr>
<td>Response measure</td>
<td>The operation succeeded before B, C and D moved out of network range.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>50 SNE users (A and 49 others)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>49 devices are within the network range of A</td>
</tr>
<tr>
<td>Artifact</td>
<td>The SNE system</td>
</tr>
<tr>
<td>Environment</td>
<td>A’s device is discovering devices with 49 statically (not</td>
</tr>
</tbody>
</table>
7.2.1.3. Security and Privacy Quality Scenarios

<table>
<thead>
<tr>
<th>Source</th>
<th>Four SNE users (A, B, C and D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>B, C and D presses the “add to contacts” button simultaneously</td>
</tr>
<tr>
<td>Artifact</td>
<td>The SNE system</td>
</tr>
<tr>
<td>Environment</td>
<td>B, C and D has detected matches with A</td>
</tr>
<tr>
<td>Response</td>
<td>A, B, C and D have exchanged contact data</td>
</tr>
<tr>
<td>Response measure</td>
<td>The operation succeeded within 5 seconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Two SNE users (A and B) and a man-in-the-middle (MITM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>A is discovering and matching B, while the MITM is sniffing the hashed phone numbers.</td>
</tr>
<tr>
<td>Artifact</td>
<td>The SNE system</td>
</tr>
<tr>
<td>Environment</td>
<td>The MITM is sniffing the hashed phone numbers send between A and B.</td>
</tr>
<tr>
<td>Response</td>
<td>The hashed phone numbers send between A and B.</td>
</tr>
</tbody>
</table>
| Response measure | The phone numbers cannot be extracted from the
hashed phone numbers.

7.2.1.4. Modifiability Quality Scenarios

<table>
<thead>
<tr>
<th>Source</th>
<th>Developers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus</td>
<td>SNE can discover and communicate over a longer distance.</td>
</tr>
<tr>
<td>Artifact</td>
<td>The communication technology must be based on Wi-Fi instead of Bluetooth.</td>
</tr>
<tr>
<td>Environment</td>
<td>When the SNE system is running.</td>
</tr>
<tr>
<td>Response</td>
<td>The network discovery- and communication components must be modified.</td>
</tr>
<tr>
<td>Response measure</td>
<td>The change can be done without modifying the components concerning the user interface and domain logic.</td>
</tr>
</tbody>
</table>

7.2.1.5. Usability Quality Scenarios

The nature of the SNE system is very simple and will not have any significant impact on the structures of the software architecture.

7.3. Architectural Description

The concern of this section is how the software architecture must be structured to support the requirements defined above in the most
appropriate way. According to An Approach to Software Architecture Description Using UML [54] the structure must be seen from the following different view-points:

- Component & connector view-point
- Module view-point
- Allocation view-point

The sections below describe the SNE software according to these three views.

7.3.1. Component & Connector View-point

The focus of this view-point is how to structure the run-time functionality of the system into a set of components, which together form the aggregated functional responsibility of the system. According to [54] a component is defined as “a functional unit that has a well-defined behavioral responsibility” and a connector is defined as “a communication relation between components that defines how control and data is exchanged”.

The diagram shown below shows how the SNE software architecture is divided into components:
The SNE application is executing as a single Dalvik JVM process in the Android operating system. The diagram above shows 2 instances of the SNE process (DeviceA:SNE and DeviceB:SNE) in order to visualize the
communication between the components executing on different mobile phones. The BluetoothAdapter component is an Android-specific process and is not in the power of the SNE application. It is just shown to illustrate the communication of the DiscoverySensor component.

The explanation of the figure above is divided into three sections. First the responsibility-split between components is explained. Second the communication modeled by the connectors is explained and finally key scenarios showing the interaction between the components are explained.

### 7.3.1.1. Components

The following explains how the responsibilities of the SNE application are divided between runtime components and how component execution is divided into threads:

- **UserInterface**: Responsible for providing graphical user interfaces for displaying and modification of the SNE domain data (see GUI's in section 5.3.3 regarding *Service Blueprints*). The UserInterface is executing in a separate thread due to the nature of Android.

- **DomainModel**: Responsible for:
  
  - Keeping track of the SNE domain data in an “in memory” domain model and in persistent storage. The data are “User info of local device”, “Contact lists”, “Privacy preferences” and “Matched remote devices”.
  
  - Handling of the “add to contacts” functionality – including event-tagging of new contacts. As this is an expensive task, which requires network communication, it is executed in a separate thread to avoid blocking the GUI to long.
  
  - Continuously refreshing the direction of remote devices.
• **Matching**: Responsible for matching the contact list of the local device with the contact list of other devices. Each matching of a remote device is executed in a separate thread in order to optimize performance by matching multiple remote devices simultaneously.

• **DiscoverySensor**: Responsible for discovering nearby remote devices. The DiscoverySensor is executing in a separate thread to be able to perform “background” device discovery in parallel with matching and other functionalities. The DiscoverySensor thread must not be blocked due to an ongoing matching of a remote device, as this will degrade the performance of discovering and matching of other devices.

• **PollingSensor**: Responsible for continuously checking the availability of remote devices and fetching the location data of the remote devices. This is executing in a separate thread.

• **Communicator**: Responsible for:
  
  o Keeping track of point-to-point connections towards remote devices.

  o Providing remote communication services to other components.

  o Listening for incoming connection request from remote devices (server functionality). This is executing in a separate thread.

  o Sending SMS text messages to remote devices.

  o Initiating phone calls to remote devices.
7.3.1.2. Connectors

Besides the splitting of responsibilities on components, another important aspect is the connectors between the components. The connectors model how the components communicate together in order to fulfill the needed functionality. The connectors are explained below:

- **UserInterface – DomainModel:** A standard Model-View-Controller pattern is used to coordinate the communication between the DomainModel and the UserInterface. The UserInterface has the role of view and controller, whereas the DomainModel has the role of model. A design goal here has been to decouple the model from GUI. Direct Java calls are used as communication protocol.

- **DomainModel – Communicator:** The DomainModel is using the Communicator as a service to communicate with remote devices. Direct Java calls are used as communication protocol.

- **PollingSensor – Communicator:** PollingSensor is using the Communicator as a service to communicate with remote devices. Direct Java calls are used as communication protocol.

- **DomainModel – PollingSensor:** PollingSensor is updating the DomainModel when the state of the remote device changes.

- **DomainModel – Matching:** When the Matching finds matching remote devices the DomainModel must be updated. To decouple the Matching from the DomainModel this is designed using a standard Observer pattern, with the DomainModel as Observer and the Matching as Subject. Direct Java calls are used as communication protocol.

- **Matching – Communicator:** The Matching is using the Communicator as a service to retrieve contact lists from remote devices. Direct Java calls are used as communication protocol.
• **Matching – DiscoverySensor:** When remote devices are discovered by DiscoverySensor the Matching must be informed in order to initiate matching with the remote device. To decouple the DiscoverySensor from the Matching this is designed using a standard Observer pattern, with the Matching as *Observer* and the DiscoverySensor as *Subject*. Direct Java calls are used as communication protocol.

• **Communicator – Communicator:** This connector is modeling the remote point-to-point communication between 2 instances of the Communicator component type. As indicated in the figure above the protocol is based on Bluetooth (BT), where both sides of the connector can have the role of *server* and *client*. As explained in section 5.1.2.6 JSON has been selected as the way of structuring the data exchanged in the communication.

• **DiscoverySensor – BluetoothAdapter:** This connector models the communication when discovering remote devices. The DiscoverySensor component has the role as the client, which initiates the discovery using the Bluetooth Inquiry protocol. The BluetoothAdapter (Android process – i.e. not SNE component) has the role as *server*, which answers the *inquiry request* with an *inquiry response*.

### 7.3.1.3. Key Scenarios

The following sections explain key system scenarios, which are significant to the architecture. Simple scenarios like making a phone call, sending a SMS and updating user info etc. are left out, as they do not have any significant impact on the architecture.

An important note to the diagrams is that the object instances reside on two different devices (mobile phones). The gray objects on the left-most side
reside on device A and the green objects on the right-most side reside on
device B.

### 7.3.1.3.1 Application initialization

The major steps in the initialization sequence of the SNE application is the following:

1. The DomainModel is loaded from persistent storage.

2. The DomainModel verifies that “user info” and “preference” has been setup by the user. If not the following steps happen – else continue in step 3.
   
   a. The DomainModel informs the UserInterface.
   
   b. The UserInterface shows a dialog to the user.

3. The DomainModel prepares the phone contact list by hashing (SHA) the phone numbers.

4. The Discovery component is started.

### 7.3.1.3.2 Discovery and matching of a device

The diagram below shows the sequence when a new remote device with matching contacts is discovered.

A pre-condition for the scenario is that the remote device is unknown to the system and that the contact list of the remote device has matching contacts with the contact list of the local device.
Figure 56 – Discovery and matching
7.3.1.3.3 Refreshing the availability and direction of remote device

The figure below shows the sequence when refreshing the availability and direction of remote devices.

Figure S7 – Device availability and direction
7.3.1.3.4 Add to contacts

The diagram below shows the sequence when a user requests to add a remote user to the phone contact list.

*Figure S8 – Add to contacts*
7.3.2. Module View-point

The package diagram gives a top level view of how the classes of the SNE architecture are modularized in packages.

Figure 59 – Package diagram
In order to make it easy to relate to the components of the components & connectors viewpoint, the top-level package structure to a great extend reflects the way the system responsibilities are divided into runtime components.

However, to separate the higher level domain concerns from lower level technical concerns the following packages cannot be mapped directly to runtime components:

- **repository**: The concern of persistent storage is encapsulated in classes of the repository package in order to separate the concern of persistent storage from the domainModel classes.

- **Sensor**: The sensor package is inspired from the ContextPhone [ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications] [22]. The purpose is to model the context information into decoupled sensors, which have the responsibility of “sensing” context changes (e.g. sensing nearby remote devices) and make them available to higher level components. The result is generic sensor abstractions, which are independent of specific domain logic – i.e. code in the domainModel package. The DiscoverySensor and the PollingSensor can of course be mapped directly to runtime component. However, the gpsSensor and the compassSensor are not, because they are not seen as active components running as separate threads or processes.

- **socialLogger**: The concern of logging system events into the history log is encapsulated in the classes of the socialLogger package.

The sections below visualized how the most significant of the packages from the diagram above is composed into classes.
7.3.2.1. Domain Package

The diagram below shows the class structure of the domainModel package.

The classes are structured according to the domain data of the SNE system. In order to simply access from other packages the façade pattern (see *Patterns in Java, volume 1* [56]) has been used to decouple client packages.

Figure 60 – Domain package
7.3.2.2. Repository Package

The diagram below shows the class structure of the repository package.

Three kinds of repositories exist, which provide the same interface towards the classes of the domainModel package. A variant of the Abstract Factory pattern [56] has been used to decouple the domainModel classes from the construction of the concrete repository objects.

![Diagram](image)

Figure 61 – Repository package
7.3.2.3. Matching

The diagram below shows how the matching package is composed of classes.

The Matching class is the key class, which plays the role as observer of new discovered devices. Furthermore, it acts as the subject, which informs MatchingObserver’s when new matching devices arrive.

Figure 62 – Matching

7.3.2.4. discoverySensor

The diagram below shows how the discoverySensor package is composed of classes.

The DiscoverySensor is the key class which is accessed from the out-side (domainModel). The DiscoveryObserver implement the observer role of the Observer Pattern [56]. The DiscoveryBroadcastReceiver listens for events from the low-level Android Bluetooth adapter.
7.3.2.5. Communicator Package

The diagram below shows how the communicator package is composed of classes.

The Communicator is the key class with is accessed from the out-side (domainModel and matching). The responsibility of listening for new incoming client-requests for connections is delegated to the ConnectionListener. The responsibility for requesting new connections towards remote devices and for using already established connections is the responsibility of the DeviceCommunicator, Connection and ConnectionCreator.
7.3.3. Allocation View-point

The focus of this view-point is how the components from the components & connectors view-point are mapped in the environment of the system.

The diagram below shows that the allocation of SNE components is very simple. The diagram shows that the components are allocated in a single process of the Android operating system, which is running on mobile device hardware node. Three instances of the mobile device are shown, to illustrate the distributed nature of the SNE system. All might depend on each other and communicate with each other simultaneously via Bluetooth.
Figure 65 – Deployment diagram
8. Prototype Implementation

The purpose of this section is the following:

- Describe the functional scope of the implemented prototype of SNE service compared to the service design (section 5.3.3).

- Describe the key experiences when implementing/coding the prototype.

- The mobile phones HTC Desire, Nexus One and G1 has been used for development. In order to use the G1’s, we have updated them to Android 2.1 from 1.6.

8.1. Functional Limitations

The prototype is in general implemented according to the service design and software architecture described in the previous sections. However, when implementing the prototype the key focus has been on the remote device communication capabilities involving Bluetooth discovery and data exchange (device discovery and matching). The prototype has the following functional limitations:

- Contact List Filtering and Add To Contacts are not supported in the final prototype, but small explicit coding experiments have been performed to prove that it is possible to access the phone contact list. Furthermore, a small coding experiment has been performed to test accessing the Android Location API and Calendar API.
• The user preferences are implemented in the user interface and are saved persistently. However, the preferences are not used by the application, as we don’t see any uncertainties in being able to implement this feature.
• Dummy values are used for user info of the device owner, as we don’t see any uncertainties in being able to implement this feature. The friendly device name is used as user name to get different user name for different devices.
• The Communicator and DiscoverySensor assumes that Bluetooth is always enabled and the “Discoverable” setting is manually set in the Android Bluetooth Settings. Small explicit coding experiments have been performed to prove that it is possible to trigger a GUI pop-up’s asking the user to enable Bluetooth and make the device “discoverable”.
• The GUI does support screen rotation. It only supports the portrait mode.

8.2. Key Coding Experiences

The coding experiences is divided in the following sections:

• Development Environment
• Bluetooth Communication
• Phone Contact List
• User Interface

8.2.1. Development Environment

We have used the following development environment:
• Eclipse IDE with the 5ADT plug-in has been used as coding environment

• 6DDMS (Dalvik Debug Monitor Server) of the Android SDK has been used for reading log messages, device file exploring, viewing threads etc.

• The emulator of the Android SDK has been used for small experiments – e.g. when testing the user interface. Unfortunately, the emulator does not support applications using Bluetooth, so we have not been able to use it much.

Our experience is that it is quite easy to code for the Android platform if you have experience with the Java programming language. The Eclipse IDE combined with the DDMS provides powerful tools, with makes it relatively easy to code using Java and debug on remote devices.

8.2.2. Bluetooth Communication

The following key coding experiences have been gained related to coding the DiscoverySensor and the Communicator (both using Bluetooth):

• The Android Bluetooth adapter informs about found devices multiple times within the same discovery cycle, so a filtering mechanism had to be implemented to filter away already known devices.

5 http://developer.android.com/sdk/eclipse-adt.html

6 http://developer.android.com/guide/developing/tools/ddms.html
• The idea of using the friendly device name to enable SNE service discovery does not work if devices are already paired before the SNE application is started. Apparently the Android Bluetooth adapter does not request the remote device name again, when re-discovering a device with which it is already paired.

• A Bluetooth client socket cannot connect (BluetoothSocket.connect()) while discovery is ongoing. A workaround was implemented to temporarily disable discovery while connecting the socket. An already connected socket can be used for communication when discovery is ongoing.

• It is only possible to make a device “discoverable” for 300 seconds. This constraint is against the nature of the SNE service, as it will be annoying to the users, when he/she is asked to enable “discoverable” every 300 seconds.

• The Android Bluetooth API of the SDK does not support automatically pairing of devices (with a hard-coded PIN code), which is required by the nature of the SNE service. However, the runtime class library running on the Android devices actually provides an API (Bluetooth.setKey(byte[])) for automatically pairing of devices. As the API is not supported by the SDK it obviously cannot compile within Eclipse, but we managed to use the API using Java Reflection. This way we managed to successfully pair G1’s with Nexus One, but somehow we never succeeded in pairing the HTC Desire with any of the other mobile phones.

8.2.3. Phone Contact List

The following experiences have been gained when accessing the phone contact list:

• The Android phone user can have multiple Gmail accounts – i.e. he/she can have multiple contact lists. Due to this we learned that the user needs to tell the SNE application what Gmail account to use for contact list matching. Furthermore, the contact list can be divided into groups, so the user also needs to select which group to
use. The ID of the mail account and the group needs to be used when accessing the contact list from the code.

- A user can have multiple phone numbers attached to a contact, each having a label – home, mobile, work etc., where one of them is marked as the primary number. When coding we never managed to retrieve all the phone number. We could only retrieve the primary number, which is used in the prototype.

- We successfully managed to geo-tag a contact with the location. The API supports that we define a new datatype (called contact MIME type), where the location is stored. However, the standard contact application of Android only shows the MIME types that it knows – i.e. the SNE specific location MIME type is not shown in the standard contact manager.

### 8.2.4. User Interface

The following experiences have been gained during development of the graphical user interface:

- The Shared Preference API of Android was an easy and quick way to implement the user settings regarding Communication and Logging preferences. However, we would have liked to use the same component to implement the Contact List Filtering, but the API does not support the dynamic behavior of adding new entries/preferences add runtime.

- We did not manage to make the list of matching devices clickable, while showing the graphical direction component.
9. Evaluation

This section describes how the developed prototype of the SNE service will be evaluated and what the results are. It is divided into 3 sections, where the first section defines the scientific method used. The second section defines what the experiments need to answer and the third section describes the results of the experiments.

9.1. The Scientific Method

The Scientific Method as defined in the article Scientific Debugging [11] is used to define, perform and reflect on our experiments in a systematic, structured and scientific fashion. With this method the way (natural) scientists work when establishing some theory about the universe are summarized. In a very general form, according to the article, the Scientific Method proceeds roughly as follows:

1. Observe (or have someone else observe) some aspect of the universe
2. Invent a tentative description, called a hypothesis, that is consistent with the observation
3. Use the hypothesis to make predictions
4. Test those predictions by experiments or further observations and modify the hypothesis based on your results
5. Repeat steps 3 and 4 until there are no discrepancies between hypothesis and experiment and/or observation

And the result of the process is described as follows:
“When all discrepancies are gone, the hypothesis becomes a theory. In popular usage a theory is just a synonym for a vague guess. For an experimental scientist, though, a theory is a conceptual framework that explains earlier observations and predicts future observations.”

9.2. Experiments

The overall goals with our experiments are to answer the following questions:

- What are the Bluetooth ranges of different Android device types?
- What is the performance of discovering and matching a remote device?
- How does the distance between users within range influence on performance?
- How does the number of devices within range influence on the performance (measure discovery time and matching time)?
- How does noise in the environment influence on the performance and scalability? E.g. other non-SNE Bluetooth devices (max 60 devices).
- What are the performance differences between different contact list sizes?

The discovery and matching process can be divided into the following four main steps:

1. D: Device discovery
2. C: Retrieve contact list from remote device
When measuring performance according to the questions above it is important to know how much time is spend on each of these steps. Especially the last step U is important, as it defines the point where the discovery and matching process has finished. This is important, because the remote user can now move out of range and still be on the matched list.

### 9.3. Experiment Results

The sections below describes the results of the experiments defined in the sections below. All time measurements are in seconds.

In general the time measurements for matching (M) are not shown in the tables below, as the measurements showed around 0-1 milliseconds.

#### 9.3.1. Bluetooth Range and Distance Impact

The table below shows the range of the different combinations of device types. The experiments are performed using the build-in Bluetooth settings GUI of the Android phones without having the SNE application started.

<table>
<thead>
<tr>
<th>Model (discover)</th>
<th>Model</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>G1</td>
<td>20-21 m</td>
</tr>
<tr>
<td>G1</td>
<td>Nexus</td>
<td>41 m</td>
</tr>
<tr>
<td>Nexus</td>
<td>Desire</td>
<td>41 m</td>
</tr>
<tr>
<td>Nexus</td>
<td>G1</td>
<td>41 m</td>
</tr>
<tr>
<td>Desire</td>
<td>Nexus</td>
<td>41 m</td>
</tr>
</tbody>
</table>

*Table 1 – Discover (device build-in functions without SNE running)*
Compared to the table above, the three tables below show measurements when the SNE application is running. The tables show the max range and the impact of physical distance. The time spend (D, C, U and total) is measured on three distances (10, 20 and 30 meters), where each distance is measured 3 times. Note that the measurements below include both Bluetooth discovery and Bluetooth socket communication, which might influence on the max range.

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>C</th>
<th>U</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.8</td>
<td>3.6</td>
<td>0.1</td>
<td>5.5</td>
</tr>
<tr>
<td></td>
<td>4.4</td>
<td>4.7</td>
<td>0.3</td>
<td>9.4</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>4.4</td>
<td>0.6</td>
<td>5.5</td>
</tr>
<tr>
<td>20</td>
<td>8.2</td>
<td>4.6</td>
<td>3.0</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>2.4</td>
<td>28.9</td>
<td>3.3</td>
<td>34.6</td>
</tr>
<tr>
<td></td>
<td>28.9</td>
<td>3.8</td>
<td>0.1</td>
<td>32.8</td>
</tr>
<tr>
<td>30</td>
<td>7.6</td>
<td>8.4</td>
<td>0.4</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>4.6</td>
<td>0.2</td>
<td>6.7</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>3.5</td>
<td>0.1</td>
<td>4.1</td>
</tr>
</tbody>
</table>

*Table 2 – Nexus (scan) – G1 (discoverable) max. range 30 meters*
### Table 3 – Nexus (scan) – Desire (discoverable) max. range 40 meters

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>C</th>
<th>U</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.0</td>
<td>21.7</td>
<td>0.4</td>
<td>23.1</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>3.4</td>
<td>0.6</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2.1</td>
<td>0.4</td>
<td>3.3</td>
</tr>
</tbody>
</table>

### Table 4 – G1 (scan) – G1 (discoverable) max. range 20 meters

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>C</th>
<th>U</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6.0</td>
<td>3.7</td>
<td>0.1</td>
<td>9.8</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>4.0</td>
<td>0.1</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>2.3</td>
<td>2.3</td>
<td>12.4</td>
<td>17.0</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conclusions and observations:

- The SNE range when the G1 is involved is significantly lower than that the discovery range without the SNE application running. The Nexus – G1 did only succeed within a range of 30 meters compared to 41 meters, whereas the G1 – G1 succeeded within 10 meters compared to 20 meters.

- The SNE range of Nexus – Desire showed approximately the same range as the discovery range without SNE.

- The time measurements for using Nexus and Desire together shows a significantly better performance compared to when a G1 is involved.

- When retrieving both the contact list and the user info a Bluetooth socket is used. However, the difference is that creating the socket connection is part of the contact list measurement, whereas the
already connected socket is just reused when transferring the user info. This indicates that the Bluetooth socket connect procedure is a time expensive procedure.

- In general the measures of discovery vary a lot.
- The time for retrieving the contact list also varies, but not as much as the Discovery time.
- The time for retrieving the user info is quite stable, but with high peak values.
- The measures shows that the distance between the devices does not significantly impact on the performance of the SNE service.
- We observed that the Bluetooth socket connection used for retrieving the contact list and user info occasionally gets disconnected/dies, which did make the discovery and matching process fail.

9.3.2. Number of Devices Impact on Performance

The tables below show time measurements when a Nexus is discovering and matching a single G1 (first table), two G1’s (second table) and two G1’s and a Desire (third table).

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G1</th>
<th>G1</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>C</td>
<td>3.07</td>
<td>3.8</td>
<td>3.8</td>
</tr>
<tr>
<td>D</td>
<td>2.05</td>
<td>1.08</td>
<td>5.0</td>
</tr>
<tr>
<td>Total</td>
<td>5.13</td>
<td>4.98</td>
<td>8.9</td>
</tr>
</tbody>
</table>

Table 5 – G1
<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G1</th>
<th>G2</th>
<th>G1</th>
<th>G2</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>3.8</td>
<td>4.2</td>
<td>3.6</td>
<td>3.9</td>
<td>3.9</td>
<td>4.2</td>
</tr>
<tr>
<td>D</td>
<td>2.4</td>
<td>1.5</td>
<td>1.0</td>
<td>1.1</td>
<td>2.5</td>
<td>1.3</td>
</tr>
<tr>
<td>Total</td>
<td>12.4</td>
<td>9.8</td>
<td>12.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6 – G1 G2

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>D</th>
<th>G1</th>
<th>G2</th>
<th>D</th>
<th>G1</th>
<th>G2</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>C</td>
<td>4.3</td>
<td>3.7</td>
<td>4.0</td>
<td>4.2</td>
<td>4.4</td>
<td>2.6</td>
<td>4.2</td>
<td>5.0</td>
<td>2.3</td>
</tr>
<tr>
<td>D</td>
<td>1.9</td>
<td>0.3</td>
<td>1.3</td>
<td>0.7</td>
<td>1.1</td>
<td>0.4</td>
<td>1.2</td>
<td>5.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Total</td>
<td>15.6</td>
<td>13.6</td>
<td>18.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7 – G1 G2 Desire

Conclusions and observations:

- The time measurements show that the time per device is approximately the same. However, as we only have 4 devices for our experiments it is not a very reliable measurement to use for prediction of the influence of many devices.

- The measurements show that the curve for the aggregated discovery and matching time is linear with the number of devices involved (1, 2 or 3). This behavior can be explained due to the limitation described in the Prototype Implementation (section 8), where the Bluetooth discovery needs to be disabled while performing Bluetooth socket communication. This will result in a sequentially behavior where some devices will need to wait for a long time, which in turn can result in a missing match.
9.3.3. Noise Impact

The first table below show measurements of Nexus – Desire without any nearby devices make Bluetooth noise. The second table shows measurements with 10 devices with Bluetooth turned on. Five of the devices were discovering and 9 were discoverable.

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>C</td>
<td>3.8</td>
<td>2.5</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>D</td>
<td>0.9</td>
<td>0.7</td>
<td>0.6</td>
<td>0.6</td>
</tr>
</tbody>
</table>

*Table 8 – Nexus (scan) – Desire (discoverable) without noise*

**Nexus – Desire With noise**

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>1.6</td>
<td>1.8</td>
<td>1.9</td>
<td>1.4</td>
<td>1.8</td>
<td>1.8</td>
<td>2.0</td>
<td>2.2</td>
</tr>
<tr>
<td>D</td>
<td>7.0</td>
<td>0.05</td>
<td>2.8</td>
<td>0.4</td>
<td>2.6</td>
<td>0.7</td>
<td>2.2</td>
<td>0.07</td>
</tr>
</tbody>
</table>

*Table 9 – Nexus (scan) – Desire (discoverable) with noise*

Conclusions and observations:
• When having noise in the environments we observed many Bluetooth socket disconnects, which made the system very unstable. This could indicate that the Android Bluetooth adapter is bad at handling many simultaneous Bluetooth devices in parallel.

• It is hard to conclude whether the noise has any influence on time, but it seems like the discovery times increases when the number of discoverable devices increases.

• A strange observation is that the time for retrieving the contact list is faster with noise that without noise.

9.3.4. Walking Scenarios

Based on the conclusions made in the sections above it is interesting to evaluate whether the SNE application is capable of discovering and matching a walking person before she/he moves out of range.

To evaluate the walking scenarios we have measured that it takes approximately 9 seconds for at person to walk 10 meters.

Walking scenario 1

Device A user stands in a static place while device B user walks normally from one side of the range of device A straight through to the other side of the range of device A in a direct line.

When the Nexus and the Desire is used together it results in a walking distance of 80 meters (40x2), which gives the SNE application 72 (9x8) seconds to discover and match the walking person. According to the time measurements presented earlier, this is no problem if the person is the only person within range. However, as the influence of multiple SNE devices showed that the SNE application is discovering and matching the devices
sequentially, it means that with the presence of approximately 18 devices (18x4 seconds) we risk to miss discovering and matching the person.

A similar calculation can be done for Nexus – G1 and G1 – G1. The distance is 60 meters for Nexus – G1 and 20 meters for G1 – G1, which results in 54 (9x6) seconds and 18 (9x2) seconds for discovery and matching. We risk to miss discovery and matching with the presence of 14 devices (14x4 seconds) for Nexus – G1 and 5 devices (5x4 seconds) for G1 – G1.

Walking scenario 2

Device A user and device B user walks normally in a straight line from opposites direction in a way so that device A pass through all of device B’s range and vice versa.

This scenario results in half of the walking distances of scenario 1 and thereby the time for discovery and matching is also cut to the half.
10. Conclusion

In this thesis we have analyzed the requirements of a mobile social network service called SNE (Social Network Extender). SNE has been designed as a service using Service Design, which has resulted in a non-functional GUI prototype. Based on the analysis results and the service design, a software architecture has been designed, which has been the basis for the implementation of a functional prototype of the SNE service. Finally the functional prototype has been the basis for an evaluation.

The main purpose of thesis has been to answers the following questions:

*Is a Bluetooth network feasible for discovering nearby mobile phones and for communicating social network data between mobile phones?*

Our prototype experiments have shown that Bluetooth is feasible for discovering physically nearby devices. Android does not support that devices are “discoverable” for an unlimited time though, which prevents the SNE application from being seamless. The network range has shown to be better than expected (approx. 10 meters) with 40 meters for HTC Desire and Nexus One, and 20 meters for G1.

For point-to-point data communication Bluetooth has shown to only work partly. The problems listed below must be solved before it can be used in a real-world product:

- Android requires users to pair devices manually, which is very much against the nature of the SNE service, as user needs to be anonymous during the discovering and matching of remote devices.
- We have had problems using Android Bluetooth sockets. A socket can simply not be created while discovery is ongoing and we have observed several socket disconnects during our experiments. Furthermore, the performance of connecting a socket is not very good.

- It is currently not possible to filter away non SNE devices during device discovery. The only way to discover the presence of the SNE service is to try to connect a socket, which is an time expensive operation.

*Is it possible to build security and privacy into the system and still maintain a good performing and user-friendly system?*

The nature of the SNE service requires that the users allow information about them selves to be sent to remote users. However, it is still important to be aware of possible privacy concerns of the users. The SNE service is designed to provide user-friendly user interfaces for defining what information is provided to remote users and preferences for setting the level of privacy. The highest level of privacy is set as default, which requires the user to actively allow a lower level of privacy (opt-in).

As the SNE service is exchanging the phone numbers of the phone contact lists, it is important not to compromise the privacy of the third-party persons being in the contact lists. To avoid storing phone number in a readable format they are protected using cryptographic hashing. Furthermore, data are encrypted during network transfer using the Bluetooth protocol.

Due to the nature of the SNE service with no back-end server involved, it is not possible to provide the standard any authentication mechanisms.
Is it possible to use the mobile phone phonebook as a source for matching personal social networks?

Our experiments show that it is possible to access the phone contact list using the Android API, which answers that it is possible. The phone number has been chosen as the base for matching, as it is the most unique data attribute of the contact list. The phone number is normalized in order to be comparable.

Is the location sensors GPS and compass feasible to help people localize each other regarding the users physical direction?

Android provides easy to use API for accessing the GPS and compass sensors of the Android phones. Due to time constraints it has not been possible to finish this work. We have developed small coding experiments using the Android API, but it has not been included in the final prototype. But what we find that is of interest and has to be experimented with, is the smallest range between devices to see what is the limit for measuring the direction.

Is Android a suitable platform for development of pervasive mobile applications, which are utilizing the standard components of a mobile phone like phone contact list, calendar, location sensors (GPS and compass) and network communication possibilities (Bluetooth)?

In our experiments Android has been proven to be suitable for accessing the phone contact list. However, as it possible for the user to have multiple Gmail accounts, the user must set the Gmail account to be used in the user info menu.

Our experiments have also shown Android to be suitable for accessing items of the phone calendar based on a time stamp. Furthermore, Android provides an easy to use API for accessing the location sensors.
As described above when answering the Bluetooth question, it has been possible to implement the Bluetooth needs of the SNE prototype, but with some limitations as described above.

Is it possible to develop a pervasive application for mobile Android devices, which enables physically co-located users to make physical contact in a secure way based on matching phone book contacts and with the use of existing smartphone capabilities like Bluetooth, GPS and compass, and with no dependencies to internet-based resources?

Based on the conclusions of the questions above, the overall conclusion of our thesis is that it has not been possible to make a seamless real-life SNE service, due to the nature of Bluetooth on Android phones.
11. Future Work

11.1. Bluetooth and Wi-Fi in combination

It could be a solution to combine Bluetooth and Wi-Fi so that Bluetooth handles the discovery process and the Wi-Fi afterwards handles the point-to-point communication.

Using Wi-Fi for the point-to-point communication we get a higher network bandwidth compared to Bluetooth. However, the main advantage of this architecture is that we can bypass the pairing process required in Bluetooth, which contradicts with the nature of the SNE service.

We cannot use the normal way of using Wi-Fi today, as it requires the nearby presence of access points, which creates a bad dependency to the physical environment. Instead we need to use the ad-hoc mode of Wi-Fi, which will allow mobile phones to communicate point-to-point. We are aware that the ad-hoc mode is not supported in most mobile phones today, but we hope it will come in the future.

11.2. Other Mobile Platforms

In order to make the SNE service a commercial success it must be supported on other common mobile platforms – e.g. IPhone and J2ME.
11.3. Contact List Filtering

It is not possible to add new items to Android Preferences runtime. Thereby it is not possible to make use of Android Preferences to generate a list including all entries from a Contact List, and update it when new contacts are added, and thereby make it possible for the users to select entries from the contact list to be omitted by the matching function.

As a result this preference for the users has to be placed outside the most obvious place, the Android Preferences.

We have found two possible workarounds to examine further in the future. The first one adds a new Boolean mime type to the Contact List entries thereby letting the users use the device contact list as the place to select entries to be omitted. The second one includes a new menu item that opens up an ordinary scrollable list view with all the contact list entries and with the possibility to mark the users to be omitted.

11.4. Real life usage experiment

In order to test the SNE service in real life with a numerous numbers of Bluetooth devices over a longer period of time, a future experiment must be setup in a realistic event environment – indoor and outdoor.

11.5. Privacy Interview

The participants in the real life usage experiment can afterwards be
interviewed. Their experiences with the SNE service being foundation for a realistic prototype probe. The interview can be divided into a pre scheme and an after scheme, to be filled out before and after their participation in the real life usage experiment.

It will be of specific interest to know their experiences with smart phones prior to the experiment, and their attitude towards security and privacy in relation to the usage of location data. It will also be relevant to take the layout of the preferences in the SNE service into consideration when designing the pre scheme.

11.6. Privacy Feedback

In the future it might be interesting to ask the users about their privacy feedback via the SNE service in order to be aligned with the users of the service.
12. References


University of Colorado at Boulder.


http://www.bluetooth.com/English/Technology/Pages/default.aspx

[27] “The Craft of Service Design”, Chapter 8, Service Design (copy from AU)


[33] A. Küpper. "Location-Based Services". WILEY.


[37] “Privacy and Human Rights 2003”.
http://www.privacyinternational.org/survey/phr2003/overview.htm

[38] "CNN report on privacy concerns".
http://www.youtube.com/watch?v=ljfyoiUn7vQ

[39] "Facebook Privacy Issues".
http://www.youtube.com/watch?v=DYUlVJ92SKI

[40] Mark Zuckerberg, founder of Facebook,


[45] “android.net.wifi”.


13. Appendix

13.1. Prototype Code

The code of our prototyping work has been delivered in the code.zip file. To view- and try out the prototype:

- Install Android SDK and Eclipse IDE
  (http://developer.android.com/sdk/installing.html)

- The code delivered in code.zip contains several Eclipse projects, each containing experiments.

- The final prototype is named “SNE_prototype”

- To view the code, import the SNE_prototype project into Eclipse

- To run the prototype:
  - Connect an Android 2.1 mobile phone using USB cable.
  - Run the SNE_prototype from within Eclipse