

Marco von Arb, Matthias Bader

# veNETa: Mobile Social Networking



Semester Thesis SA-2007-18  
April to December 2007

Tutor: Michael Kuhn  
Supervisor: Prof. Dr. Roger Wattenhofer

### **Abstract**

Social Networking im Internet hat im letzten Jahrzehnt stark an Bedeutung gewonnen. Parallel zur rasanten Verbreitung des Zugangs zum Internet entwickelten sich verschiedene social networking-Portale zu intensiv genutzten Kontaktbörsen. Im Gegensatz dazu blieb für vergleichbare Ansätze auf mobilen Plattformen, beispielsweise Mobiltelefonen, der Durchbruch bisher weitgehend aus. Dieser Bericht beschreibt veNETa, einen neuartigen Ansatz für Mobile Social Networking auf Mobiltelefonen, der neben einem serverbasierten Betriebsmodus auch Friend-of-Friend-Funktionen im vollständig verteilten Betrieb unterstützt. Ausserdem gehen wir auf die zusätzlichen Möglichkeiten ein, die sich auf der Mobiltelefonplattform für einen Social Networking-Dienst ergeben, und beschreiben unsere Arbeit und Lösungsansätze bei der effizienten und sicheren Implementierung dieser Funktionalität.

### **Abstract**

During the last decade, in parallel to growing market penetration of internet access, online social networking has seen ever increasing interest. In contrast, apart from some notable exceptions, social networking on mobile platforms, such as mobile phones, could not live up to the expectation. This report describes veNETa, a novel approach to social networking on mobile phones, that supports friend-of-friend-features in a completely distributed mode of operation, as well as server-centric ideas. Additionally, we discuss the characteristics of modern mobile phone platforms that open new avenues for social networking services, and describe our work and the approaches for implementing this functionality in a secure and efficient way.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>9</b>
1.1	Motivation . . . . .	9
1.2	Dokumentstruktur . . . . .	9
1.3	Danksagungen . . . . .	9
<b>2</b>	<b>Mobile Social Networking</b>	<b>11</b>
2.1	Social Networking . . . . .	11
2.2	Mobiltelefonie . . . . .	11
2.3	Die Zukunft mobiler Social Networking-Dienste . . . . .	12
<b>3</b>	<b>Related Work</b>	<b>13</b>
3.1	Nokia Sensor . . . . .	13
3.1.1	Motivation und Fokus . . . . .	13
3.1.2	Leistungen und Resultate . . . . .	13
3.1.3	Einschränkungen . . . . .	13
3.2	Plazes . . . . .	13
3.2.1	Motivation und Fokus . . . . .	13
3.2.2	Leistungen und Resultate . . . . .	14
3.2.3	Einschränkungen . . . . .	14
3.3	Dodgeball . . . . .	14
3.3.1	Motivation und Fokus . . . . .	14
3.3.2	Leistungen und Resultate . . . . .	14
3.3.3	Einschränkungen . . . . .	14
3.4	Bluepulse . . . . .	15
3.4.1	Motivation und Fokus . . . . .	15
3.4.2	Leistungen und Resultate . . . . .	15
3.4.3	Einschränkungen . . . . .	15
3.5	Jaiku . . . . .	15
3.5.1	Motivation und Fokus . . . . .	15
3.5.2	Leistungen und Resultate . . . . .	15
3.5.3	Einschränkungen . . . . .	16
3.6	Mologogo . . . . .	16
3.6.1	Motivation und Fokus . . . . .	16
3.6.2	Leistungen und Resultate . . . . .	16
3.6.3	Einschränkungen . . . . .	16
3.7	Mobiluck . . . . .	17
3.7.1	Motivation und Fokus . . . . .	17
3.7.2	Leistungen und Resultate . . . . .	17
3.7.3	Einschränkungen . . . . .	17
3.8	Sixsense . . . . .	17
3.8.1	Motivation und Fokus . . . . .	17
3.8.2	Leistungen und Resultate . . . . .	17
3.8.3	Einschränkungen . . . . .	18
3.9	Serendipity . . . . .	18
3.9.1	Motivation und Fokus . . . . .	18
3.9.2	Leistungen und Resultate . . . . .	18
3.9.3	Einschränkungen . . . . .	18

3.10 Lovegety . . . . .	18
3.10.1 Motivation und Fokus . . . . .	18
3.10.2 Leistungen und Resultate . . . . .	18
3.10.3 Einschränkungen . . . . .	19
3.11 Spontact . . . . .	19
3.11.1 Motivation und Fokus . . . . .	19
3.11.2 Leistungen und Resultate . . . . .	19
3.11.3 Einschränkungen . . . . .	19
3.12 BlueDating . . . . .	19
3.12.1 Motivation und Fokus . . . . .	19
3.12.2 Leistungen und Resultate . . . . .	20
3.12.3 Einschränkungen . . . . .	20
<b>4 Aufgabenstellung</b>	<b>21</b>
4.1 Attraktivität des Dienstes in der Startphase . . . . .	21
4.2 Kostengünstiger Betrieb . . . . .	21
4.3 Benutzerfreundlichkeit auf dem Mobiltelefon . . . . .	22
<b>5 Methodik- und Architekturentscheide</b>	<b>23</b>
5.1 Benutzerschnittstelle . . . . .	23
5.2 Erfassung von Standorten . . . . .	24
5.3 Serverbasierte und verteilte Dienste . . . . .	24
5.4 Fazit . . . . .	24
<b>6 Implementierung</b>	<b>26</b>
6.1 Die Main-Klasse . . . . .	27
6.2 Das Welcome Screen-Modul . . . . .	27
6.3 Das Bluetooth-Modul . . . . .	27
6.3.1 Einführung . . . . .	27
6.3.2 Funktionalität . . . . .	27
6.3.3 Implementierung . . . . .	28
6.4 Das Crypto-Modul . . . . .	28
6.4.1 Einführung . . . . .	28
6.4.2 Funktionalität . . . . .	29
6.4.3 Implementierung . . . . .	29
6.5 Friend Finding-Modul . . . . .	29
6.5.1 Einführung . . . . .	29
6.5.2 Funktionalität . . . . .	30
6.5.3 Implementierung . . . . .	30
6.6 Server . . . . .	31
6.6.1 Einführung . . . . .	31
6.6.2 Funktionalität . . . . .	31
6.6.3 Implementierung . . . . .	32
<b>7 Resultate – Evaluation</b>	<b>33</b>
7.1 Resultate . . . . .	33
7.2 Evaluation . . . . .	33
<b>8 Ansätze zur Weiterentwicklung und Problembehebung</b>	<b>35</b>
<b>A Details der Implementation</b>	<b>36</b>
A.1 BTEngine . . . . .	36
A.1.1 BTEngine.java . . . . .	36
A.1.2 BTEDeviceDisc.java . . . . .	37
A.1.3 BTEServiceDisc.java . . . . .	37
A.1.4 BTEConnection.java . . . . .	37
A.1.5 BTEMessage.java . . . . .	37
A.2 Server . . . . .	38
A.2.1 Server.java . . . . .	38

---

A.2.2	Verbindung.java	38
A.2.3	Verbindungsvector.java	38
A.2.4	SMSVerbindung.java	39
A.2.5	Nachrichtenvector.java	39
A.2.6	Nachricht.java	39
<b>B</b>	<b>Aufgabenstellung</b>	<b>42</b>
B.1	Student Thesis "Mobile Social Networking"	42
B.2	Subject	42
B.3	Time schedule in hours (Total: 2*275h = 550h)	42
B.4	The Student's Duties	43
B.5	General	43
B.6	Contacts/Advisers	43

# Abbildungsverzeichnis

6.1	Der modulare Aufbau von veNETa. . . . .	26
6.2	Die Zustandsfolge des Bluetooth-Moduls. . . . .	28
6.3	Die Zustandsfolge des Friend Finding-Moduls. . . . .	30
6.4	Der Interaktionsmechanismus des Friend Finding-Moduls. . . . .	30
A.1	Der modulare Aufbau von veNETa. . . . .	39

# Tabellenverzeichnis

7.1	Resultate der Evaluation der Akkulaufzeit . . . . .	34
7.2	Resultate der Evaluation der Zuverlässigkeit von Multihop-Übertragungen . . . .	34

# Kapitel 1

## Einleitung

### 1.1 Motivation

Im Rahmen unserer Semesterarbeit im Herbst 2007 war es unsere Aufgabe, ein Mobile Social Network<sup>1</sup> zu entwerfen, das die Verwaltung und Erweiterung des Freundeskreises auf einem mobilen Gerät ermöglicht. Die Besonderheit dieses Projekts war der Gedanke, nach potentiell interessanten Gesprächspartnern in der Umgebung nicht alleine auf klassischem Weg über Profilinformationen und -vergleiche zu suchen, sondern auch einen Mechanismus zu unterstützen, der bereits aus auf dem Gerät gespeicherten Daten Schlüsse über die Ähnlichkeit von Kontakten zu ziehen vermag.

Wir haben uns für diese herausfordernde Aufgabe entschieden, weil wir den etablierten Social Networks eine freie, fortschrittliche, mobile Alternative gegenüberstellen wollten, die ihren Benutzern auch in einer Startphase eine attraktive Dienstleistung bietet, neuartige Ansätze erforscht und verwirklicht und gleichzeitig auch einen praktischen Nutzen erbringt.

### 1.2 Dokumentstruktur

Dieser Bericht führt, wo nötig, in die relevanten Themen ein und legt Rechenschaft über die Vorgehensweise bei der Erfüllung dieser Aufgabe ab. Der verbleibende Teil dieses Dokuments gliedert sich folgendermassen: Zunächst führen wir in das Thema ein und geben einen kurzen Abriss der historischen Entwicklung der Idee des Mobile Social Networking. In Kapitel 3 gehen wir auf bedeutsame Vorleistungen anderer Entwickler ein und stellen in diesem Rahmen etablierte Ansätze und Konzepte vor. Im Anschluss beleuchten Kapitel 4 und 5 Einzelheiten der gestellten Aufgabe und die Ansätze, die wir wählten, um diese Semesterarbeit reibungslos und erfolgreich abzuschliessen. Eine Beschreibung der Implementierung von veNETa findet sich im Kapitel 6. Im nächsten Abschnitt fassen wir unsere Resultate zusammen und evaluieren die Software, und im Kapitel 8 untersuchen wir Bereiche, die noch Raum für Verbesserungen lassen.

Der Anhang enthält die Angaben, die es dem interessierten Leser ermöglichen sollen, unsere Arbeit nachzuvollziehen, zu verändern oder zu erweitern, die aber im Hauptteil keinen Platz gefunden haben. Dort gehen wir auf Details wie eingesetzte Software oder architektonische Einzelheiten bei der Implementation von veNETa ein.

### 1.3 Danksagungen

Zu besonderem Dank sind wir unserem Betreuer, Michel Kuhn, verpflichtet: Seine bedingungslose Unterstützung, die Selbständigkeit, mit der wir diese Aufgabe angehen konnten, aber auch die grossen Erwartungen an das Projekt ermöglichten erst das Resultat, wie es in diesem Bericht jetzt beschrieben wird. Herzlich danken möchten wir an dieser Stelle auch dem verantwortlichen Vorsteher der Gruppe, Prof. Roger Wattenhofer – die freundliche Stimmung, die

---

<sup>1</sup> In diesem Dokument werden der Einfachheit halber die Begriffe *Social Networking-Dienst* synonym zu *Social Network* verwendet. Die Aufgabenstellung hat nicht zum Inhalt, eine Nutzerbasis für den Dienst aufzubauen.

grosszügige Bereitstellung von Hardware und Infrastruktur sowie die unkomplizierte, eigenverantwortliche Atmosphäre sind sicherlich nicht zufällig entstanden, haben aber auch auf diese Arbeit eine sehr positive Wirkung ausgeübt.

## Kapitel 2

# Mobile Social Networking

Computerbasierte social networking-Dienste im Internet konnten sich in den vergangenen Jahren über ein grosses Wachstum freuen. Parallel dazu nahm die Verbreitung der Mobiltelefone rasant zu. Die Rechenleistung dieser Telefone ist mittlerweile beachtlich und zusätzlich verfügen viele von ihnen noch über Vorkehrungen zur drahtlosen Kommunikation über kurze Distanzen wie zum Beispiel Bluetooth. Dies macht sie zur geeigneten Plattform für Mobile Social Networking-Dienste. Dennoch blieb bis heute der grosse Erfolg dieser mobilen Dienste aus. Für die weitere Diskussion der Ursachen für dieses Ausbleiben geben wir hier einen kurzen Einblick in die Entstehung von Social Networking und die Entwicklung des Mobilfunksektors.

### 2.1 Social Networking

Die ersten elektronischen Social Networking-Dienste entstanden bereits vor mehreren Jahrzehnten. Einer der ersten Dienste im Internet war das Usenet, welches 1979 ins Leben gerufen wurde und bis heute noch existiert. Noch vor dem Usenet entstanden ausserhalb des Internets so genannte Bulletin Board Systems (BBS). Die ersten Social Networking-Dienste im World Wide Web erblickten hingegen erst viele Jahre später das Licht der Welt. Die heute noch existierende Seite Classmates.com war eine der ersten ihrer Art und wurde 1995 erschaffen. Ihr Hauptnutzen besteht darin, dass sie es vereinfacht, mit ehemaligen Klassenkameraden in Kontakt zu bleiben oder ehemalige Klassenfreunde wieder zu finden. 2005 wurde ein weiterer wichtiger Meilenstein erreicht: erstmals hatte eine Social Networking-Seite (Myspace.com) mehr Seitenaufrufe als die Suchmaschine Google<sup>1</sup>. Besonders beeindruckend dabei ist die Tatsache, dass Myspace damals nur knapp zwei Jahre alt war und somit enorm schnell gewachsen ist.

Etwa zur selben Zeit wurde Myspace von der Firma News Corporation für 580 Millionen U.S. Dollar gekauft. Spätestens seit diesem Zeitpunkt war klar, dass Social Networking-Dienste im Internet sehr gefragt sind. Und auch heute noch geht die Erfolgsgeschichte weiter: Myspace hat mittlerweile mehr als 200 Millionen Mitglieder und wuchs im vergangenen Jahr um ganze 72%. Auch andere Anbieter erfreuen sich grösster Beliebtheit. Der Konkurrent Facebook zum Beispiel konnte innerhalb eines Jahres ein Wachstum von 270% verbuchen, Tagged.com sogar 774%<sup>2</sup>. Unter den zehn meistbesuchten Seiten des Internets sind mittlerweile drei Social Networking-Dienste<sup>3</sup> und laut einer Studie von NetRatings haben bereits 45% der aktiven Internetnutzer Social Networking-Seiten besucht<sup>4</sup>.

### 2.2 Mobiltelefonie

Auf ein mindestens so beeindruckendes Wachstum wie die Social Networking-Dienste kann die Mobilfunkindustrie zurückblicken. Der erfolgreichste Mobilfunkstandard GSM erreichte im Jahre

<sup>1</sup>[http://www.businessweek.com/technology/content/jul2005/tc20050719\\_5427\\_tc119.htm](http://www.businessweek.com/technology/content/jul2005/tc20050719_5427_tc119.htm)

<sup>2</sup>[http://www.news.com/8301-13577\\_3-9752857-36.html](http://www.news.com/8301-13577_3-9752857-36.html)

<sup>3</sup>[http://www.alexa.com/site/ds/top\\_500](http://www.alexa.com/site/ds/top_500)

<sup>4</sup>[http://www.businessweek.com/technology/content/may2006/tc20060530\\_170086.htm](http://www.businessweek.com/technology/content/may2006/tc20060530_170086.htm)

2004 eine Milliarde Nutzer – zwölf Jahre nach dem Start des ersten GSM-Netzes. Nur zweieinhalb Jahre später, im zweiten Quartal 2006, waren es bereits zwei Milliarden Nutzer<sup>5</sup>. Mit der zunehmenden Verbreitung der Mobiltelefone stiegen auch die Verkaufszahlen: Im Jahr 2001 wurden knapp 400 Millionen Mobiltelefone verkauft, zwei Jahre später, 2004, waren es bereits 713 Millionen Stück<sup>6</sup> und für das Jahr 2007 sollen es bereits 1.13 Milliarden Geräte sein<sup>7</sup>. Mittlerweile verwenden bereits 2.3 Milliarden Menschen ein Mobiltelefon<sup>8</sup>.

Doch nicht nur die Verbreitung der Mobiltelefone nimmt ständig zu sondern auch ihre Leistungsfähigkeit. Viele der aktuellen Mobiltelefone sind derzeit in der Lage, Programme zu laden und auszuführen, und ähnlich viele können auch auf das Internet zugreifen. Zusätzlich sind sie oft mit Kommunikationsvorrichtungen für kurze Distanzen wie Bluetooth oder Wireless LAN ausgestattet, welche eine kostenlose Kommunikation mit anderen Geräten in der Umgebung ermöglichen. Durch die grosse Verbreitung und die gestiegene Leistungsfähigkeit sind sie bestens geeignet für Mobile Social Networking-Anwendungen. Ein weiterer Aspekt der für die Verwendung dieser Anwendungen auf Mobiltelefonen spricht, ist der Umstand, dass Mobiltelefone oft den ganzen Tag lang in der Nähe ihres Nutzers sind und so bei Bedarf direkt Kontakt mit anderen Nutzern in der Umgebung aufgenommen werden kann, im Gegensatz zu den Anwendungen, welche nur mit einem PC mit Internetverbindung genutzt werden können. Darüber hinaus verfügen Mobiltelefone über zum Teil weitreichende Informationen, welche viel über ihren Nutzer verraten können. Social Networking-Dienste sind auf solche Informationen angewiesen, um Profile ihrer Nutzer zu erstellen, welche zum Beispiel dabei helfen, interessante Gesprächspartner zu finden oder eine interessante Gesprächsgrundlage darstellen können. Das integrierte Telefonbuch zum Beispiel enthält viele Angaben zu den sozialen Kontakten des Benutzers, welche beispielsweise anhand der Anzahl der geführten Gespräche und der Gesprächsdauer klassifiziert werden könnten. Aus den gespeicherten Kurznachrichten könnten Informationen zu interessanten Gesprächsthemen gefunden werden die gespeicherten Musikdateien verraten viel über den persönlichen Musikgeschmack. Anhand des im Mobiltelefon integrierten Kalenders könnte festgestellt werden, wann der Telefonbenutzer freie Zeit zur Verfügung hat, und da die neueren Telefone zum Teil integrierte GPS-Empfänger besitzen, könnte auch die Lieblingsbar oder der Arbeitsplatz ausfindig gemacht werden.

## 2.3 Die Zukunft mobiler Social Networking-Dienste

Aufgrund des grossen Erfolges der internetbasierten Social Networking-Dienste, der hohen Verbreitung der Mobiltelefone und ihrer Eignung für Mobile Social Networking-Dienste ist die Erwartungshaltung an mobile Ausführungen dieser Dienste entsprechend hoch. Laut Dennis Crowley, dem Gründer des Mobile Social Networking-Dienstes Dodgeball, könnte die mobile Variante genau so gross werden wie die grossen internetbasierten Dienste. Gemäss Jill Aldort, Analyst bei der Yankee Group, wird sogar fast jeder Social Networking-Dienst innerhalb der nächsten Jahre auch eine mobile Komponente beinhalten<sup>9</sup>. Noch einen Schritt weiter geht Chales Golvin, Analyst bei Forrester Research: Seiner Meinung nach soll der mobile Teil eines social networks in Zukunft der wichtigste sein, und alle Dienste werden über einen solchen verfügen.

<sup>5</sup><http://www.gsmworld.com/news/statistics/index.shtml>

<sup>6</sup><http://prnewswire.co.uk/cgi/news/release?id=149384>

<sup>7</sup><http://www.gartner.com/it/page.jsp?id=514407>

<sup>8</sup>[http://www.theregister.co.uk/2007/07/03/mobile\\_subscriptions\\_skyrocket/](http://www.theregister.co.uk/2007/07/03/mobile_subscriptions_skyrocket/)

<sup>9</sup>[http://www.businessweek.com/technology/content/may2006/tc20060530\\_170086.htm](http://www.businessweek.com/technology/content/may2006/tc20060530_170086.htm)

# Kapitel 3

## Related Work

Angetrieben von dem grossen Erfolg der internetbasierten Social Networking-Anwendungen versuchen mittlerweile zahlreiche Dienste, welche sich auch oder nur für mobile Plattformen eignen, die Gunst der Nutzer zu erlangen. Wir führen hier ohne Anspruch auf Vollständigkeit einige bereits existierende Mobile Social Networking-Anwendungen auf, die unter verschiedenen Gesichtspunkten interessant erscheinen.

### 3.1 Nokia Sensor

#### 3.1.1 Motivation und Fokus

Mit Hilfe dieser Software kann der Benutzer ein Profil von sich direkt auf dem Mobiltelefon erstellen. Andere Nutzer in der Umgebung können dieses Profil anschauen, und falls gewünscht Nachrichten via Bluetooth austauschen. Das Programm kann gratis von Nokia bezogen werden, funktioniert aber nach den verfügbaren Angaben nur auf einigen Telefonen dieses Herstellers.

#### 3.1.2 Leistungen und Resultate

Mit Nokia Sensor ausgerüstete Mobiltelefone erlauben dem Benutzer, benachbarte Teilnehmer zu finden und zu kontaktieren. Darüber hinaus erlaubt die Software das Austauschen von Dateien. Eine Spezialität von Nokia Sensor sind die sogenannten *Folios* – eine Kollektion von Seiten, auf denen der Benutzer mit Bildern und Text eine Art öffentlich einsehbares Profil erstellt oder Informationen für kollaborative Zusammenarbeit speichert. Dank spezifischer Versionen für die unterstützten Modelle sind Bildschirm Aufbau und Menüführung gut an das jeweilige Endgerät angepasst.

#### 3.1.3 Einschränkungen

Bedauerlicherweise werden weniger als zehn spezifische Modelle unterstützt, und natürlich läuft die Software nur auf Geräten von Nokia. Die Unterstützung beschränkt sich auf veraltete Modelle, und Software sowie Interfaces sind closed-source. Angaben von Nutzern zufolge hat das Programm Stabilitätsprobleme und ist stellenweise unintuitiv gestaltet. Nokia Sensor ist eine vollständig verteilte Applikation. Auf eine zentrale Serverinfrastruktur wurde verzichtet, ohne dass Vorkehrungen getroffen wurden, um die sich daraus ergebenden Nachteile auszugleichen.

### 3.2 Plazes

#### 3.2.1 Motivation und Fokus

Plazes ist ein internetbasiertes Social Network mit Fokus auf Örtlichkeiten und die Aktivitäten, die dort stattfinden. Der Dienst ermöglicht es den Benutzern unter der Bezeichnung *Plazes-to-go*, ihren Standort mittels Mobiltelefon mitzuteilen, und erlaubt den anderen Nutzern den Zugriff

auf diese Daten. Dadurch können Freunde sehen, wo man sich gerade aufhält und was man gerade unternimmt. Es ist auch möglich, seine geplanten, zukünftigen Aufenthaltsorte einzugeben. Plazes wird primär vom PC aus verwendet, unterstützt unterwegs aber auch einfache Abfragen via SMS (und auf modernen Mobiltelefonen via Internetbrowser).

### 3.2.2 Leistungen und Resultate

Plazes ist ein etablierter Social Networking-Dienst mit integrierten Google Maps-Karten und ausgereiftem User Interface. Die ort-zentrische Perspektive dieses Social Network macht das Suchen und Stöbern in der grossen Datenbank angenehm und übersichtlich, und dank der Möglichkeit, Texte und Bilder an einen bestimmten Ort anzuhängen, wird diese Beschreibung zu einem kollaborativen Tagebuch.

### 3.2.3 Einschränkungen

Als primär internetbasiertes Angebot ist Plazes nicht im Sinn des Wortes ein Mobile Social Network, obwohl der räumliche Aufenthaltsort im Mittelpunkt des Dienstes steht. Das Fehlen von Software für Mobiltelefone, im Idealfall mit Unterstützung von GPS-Empfängern, macht das Aktualisieren des Aufenthaltsorts unbequem, unzuverlässig und unter Umständen teurer als nötig. Die Interaktion von Benutzern von Plazes wird nur auf einer Meta-Ebene unterstützt, beispielsweise durch das Abonnieren des Aufenthaltsortes ausgewählter Personen oder Gruppen, aber nicht automatisch durch Vergleiche der Aufenthaltsorte von Nutzern, oder ihrer Beziehungsstruktur oder Interessen.

## 3.3 Dodgeball

### 3.3.1 Motivation und Fokus

Ähnlich wie bei Plazes teilen bei Dodgeball die Nutzer dem Server ihren Standort via SMS mit. Dieser informiert dann mittels Kurznachrichten über Freunde, welche sich in der Umgebung aufhalten. Zusätzlich können auch die Freunde eines Freundes in der Umgebung angezeigt werden. Seit Mai 2005 gehört die Firma zu Google.

### 3.3.2 Leistungen und Resultate

Als Internet-Kurznachricht-Hybride verbindet Dodgeball Vorteile von herkömmlichen Social Networks und mobilen Applikationen. Dodgeball legt das Hauptaugenmerk auf den Aufenthaltsort, seine primäre Funktion ist das Benachrichtigen von Teilnehmern, die sich gleichzeitig innerhalb weniger hundert Meter aufhalten. In den unterstützten Gebieten sind viele öffentliche Plätze bereits im Netzwerk gespeichert und müssen deshalb nicht mit der vollen Adresse, sondern nur dem Namen der Lokalität bezeichnet werden, was die Bedienung via SMS vereinfacht. Dodgeball integriert einige innovative Funktionen in seinen Dienst, beispielsweise das Konzept von *crushes*, also Personen, die nicht zum eigenen Freundeskreis gehören, aber auf die man ein Auge geworfen hat und benachrichtigt werden möchte, falls sie sich in der Nähe aufhalten, sowie das friends-of-friends-Konzept, das das Erforschen und Kontaktieren des erweiterten Freundesnetzwerks erlaubt.

### 3.3.3 Einschränkungen

Wie bei Plazes schränken das Fehlen von Software für Mobiltelefone den Benutzerkomfort deutlich ein. Zusätzlich funktioniert Dodgeball nur in 22 grösseren Städten in den USA bestimmungsgemäss. Ausserdem wurde der Dienst seit der Übernahme durch Google nicht mehr signifikant erweitert, und Mitte 2007 verliess das ursprüngliche Entwicklerteam das Unternehmen.

## 3.4 Bluepulse

### 3.4.1 Motivation und Fokus

Bluepulse ist eine weitere standortbasierende Mobile Social Networking-Anwendung, welche es dem Benutzer erlaubt, Freunde, welche sich gerade in der Nähe aufhalten, zu finden. Der eigene Standort muss auch hier manuell mitgeteilt werden, wobei im Unterschied zu anderen Netzwerken auch Fotos oder Videos eingebunden werden können. Dadurch können die Aufenthaltsorte von Freunden virtuell besucht werden. Zusätzlich steht noch eine Gruppen-Chat-Funktion zur Verfügung.

### 3.4.2 Leistungen und Resultate

Bluepulse kombiniert auf elegante Weise eine ausgereifte Benutzerschnittstelle für den Nachrichtenversand über unterschiedliche Kanäle (SMS, email, Bluepulse-Nachrichten) mit Standortfunktionen und einem einfachen Blog. Wie andere Social Networks erlaubt Bluepulse das Erforschen des erweiterten Freundeskreises mit einer *Friend's Friends*-Funktion. Die Verwendung des Browsers auf dem Mobiltelefon ermöglicht eine ansprechendere Interaktion des Benutzers mit dem Dienst als auf Kurznachrichten basierende Lösungen. Ein Java-Client für den Dienst ist in der öffentlichen Beta-Phase und soll nächstens produktiv eingesetzt werden. Als eines der wenigen primär mobilen Social Networks hat Bluepulse mit dem Einblenden von Werbung auf den aufgerufenen Seiten ein tragfähiges Modell für die Finanzierung der Plattform gefunden.

### 3.4.3 Einschränkungen

Trotz des Namens setzt Bluepulse keine der gängigerweise auf Mobiltelefonen verfügbaren Technologien wie Bluetooth, Lokalisierung via GPS oder Cell ID und dergleichen ein. Die Realisierung als Web-Applikation zieht relativ grosse Datentransfervolumina und damit unter Umständen hohe Kosten für den Benutzer nach sich. Es wird sich zeigen, inwieweit der Java-Client diese Einschränkungen zu entschärfen vermag.

## 3.5 Jaiku

### 3.5.1 Motivation und Fokus

Ebenfalls zur Kategorie der standortbasierten Mobile Social Networking-Anwendungen gehört Jaiku, wobei dieser Dienst auch via Internet von einem Computer aus benutzt werden kann. Der Benutzer kann ein Profil über sich erstellen, welches nähere Angaben über ihn enthält und auch Fotos können eingefügt werden. Der eigene Standort kann hier auch manuell mitgeteilt werden, jedoch soll Jaiku auch die Möglichkeit bieten, den Standort automatisch zu übermitteln. Die dazu benötigten Standortinformationen bezieht der Dienst vom Netzbetreiber, weshalb diese Funktion auch nicht in allen Netzen und nicht bei allen Mobiltelefonen verfügbar ist. Weiter kann der Kalender, welcher auf dem Mobiltelefon gespeichert ist, für andere Nutzer zugänglich gemacht werden, damit erkannt werden kann, welcher Teilnehmer im Moment beschäftigt ist und wer gerade nichts zu tun hat. Mit Hilfe von Bluetooth kann Jaiku zudem feststellen, wie viele andere Nutzer sich gerade in der unmittelbaren Umgebung befinden und natürlich kann diese Information auch allen zugänglich gemacht werden. Jaiku wurde im Oktober 2007 von Google übernommen.

### 3.5.2 Leistungen und Resultate

Eine Spezialität von Jaiku ist die Integration von auf dem Mobiltelefon verfügbaren Daten wie Aufenthaltsort, Benutzer in Bluetooth-Reichweite und Kalendereinträge ins Social Network. Die Möglichkeit, den Aufenthaltsort unterwegs in einer mobilen Anwendung für die Symbian S60-Plattform, einem Java-Client, über einen mobilen Webbrowser oder via SMS festzulegen, deckt alle Bedürfnisse ab. Jaiku ist eine technisch ausgereifte Lösung, die über das Jaiku-API auch

Erweiterungen der Funktionalität zulässt, beispielsweise die Präsentation von Präsenzdaten in einem Blog.

### 3.5.3 Einschränkungen

Gerade vor dem Hintergrund aller seiner technischen Errungenschaften, der vielseitigen Zugriffsmöglichkeiten und einer einzigartigen Lokalisierungsmethode lässt Jaiku Funktionen wie einen erweiterten Freundeskreis oder fortschrittlichen Gruppenfunktionen vermissen. Obwohl alle grundlegenden Funktionen herkömmlicher Social Networks implementiert sind, gleicht Jaiku mehr einem mobilen Blog als einem Social Network. Es bleibt abzuwarten, wie sich die Integration des Unternehmens in den Google-Konzern (und möglicherweise eine Kombination mit Funktionen von Dodgeball und Orkut, die beide bereits zu Google gehören) auf eine intensivere Nutzung der brachliegenden Möglichkeiten auswirken wird.

## 3.6 Mologogo

### 3.6.1 Motivation und Fokus

Ein weiterer Vertreter der standortbasierten Mobile Social Networking-Dienste ist Mologogo. Er unterstützt sowohl die manuelle Eingabe des momentanen Aufenthaltsortes als auch die automatische Positionsbestimmung via GPS. Die momentane Position wird auf einer Strassenkarte dargestellt. Sollte sich ein Freund in der Umgebung aufhalten, dann wird auch seine Position angezeigt und der Benutzer informiert. Mologogo verfügt zudem über ein Verzeichnis von interessanten Orten (points of interest) und stellt diese auf Wunsch direkt auf der Karte dar. Die Informationen über den aktuellen, eigenen Standort können zudem in HTML-Seiten eingebunden werden (z.B. für Myspace oder Blogs).

### 3.6.2 Leistungen und Resultate

Mologogo stellt die grundlegenden Funktionen für Social Networking zur Verfügung, und erlaubt Zugriff auf das Kartenmaterial von Google und Chats zwischen angemeldeten Benutzern. Die Benachrichtigungen des Benutzers über Freunde in der Nähe seines Aufenthaltsortes und die Aktualisierung des eigenen Standortes erfolgen bevorzugt über einen Client, der auf ausgewählten Mobiltelefonen lauffähig ist. Dieser erlaubt ausserdem das Erweitern des POI-Verzeichnisses, die Darstellung zurückgelegten Wegs auf einer Strassen- oder Satellitenkarte sowie die Einblendung von Staus und Wetterinformationen. Eine Besonderheit von Mologogo ist die Verfügbarkeit von vorgefertigten Bausätzen, die sich für die Verfolgung des Aufenthaltsortes von Fahrzeugen eignen, sowie die Integration mit dem Social Network Twitter. Durch die Bereitstellung von ausgewählten GPS-Telefonen mit vorinstallierter Software (sog. Starter Kits) ist der Einstieg in die Mobile Social Networking-Welt für wenig versierte Benutzer bequemer als bei anderen Lösungen, und passende Daten-Verträge kosten laut Angaben des Herstellers monatlich knapp zehn Franken.

### 3.6.3 Einschränkungen

Die direkt dem Social Networking zuzurechnenden Funktionen von Mologogo beschränken sich auf das Erstellen einer Freundesliste und der Suche in der Umgebung nach weiteren Teilnehmern, sowie das Erstellen von und Stöbern in Listen von interessanten Orten. Auf die Implementierung weitergehender Funktionen, wie die Korrellierung von Standortdaten von Nutzern, die unterschiedlichen Freundesnetzwerken angehören, oder Friend-of-Friend-Funktionen, wurde verzichtet, ebenso wie die Bereitstellung eines plattformunabhängigen Java-Clients. Daraus ergibt sich der Nachteil, dass nur eine beschränkte Auswahl an Mobiltelefonen unterstützt wird. Hinweise für das Neustarten von GPS-Mobiltelefonen auf der Website deuten ausserdem auf technische Schwierigkeiten bei der Integration von Hard- und Software hin.

## 3.7 Mobiluck

### 3.7.1 Motivation und Fokus

Mobiluck gehört zu den Mobile Social Networking-Diensten, welche nicht zwingend auf einen Server angewiesen sind. Dies hat den grossen Vorteil, dass keine Verbindung zum Internet benötigt wird, welche bei mobilen Geräten oft recht kostspielig ist. Der Hauptnutzen von Mobiluck besteht darin, dass andere Bluetoothgeräte in der Umgebung gefunden werden können. Sobald ein Gerät gefunden wurde kann eine Nachricht oder ein Photo gesendet werden. Zudem können von anderen Mobiltelefonen mit Mobiluck-Software das Profil und ein Bild des Nutzers übertragen werden.

### 3.7.2 Leistungen und Resultate

Mobiluck verbindet instant messaging und vollständig verteilte Mobile Social Networking-Funktionen. Die Verfügbarkeit von Clients für alle gängigen Symbian-Versionen, Java und Zugriff auf die Website in mobilen Browsern schafft eine potentiell breite Nutzerbasis. Eine Spezialität von Mobiluck ist die Integration populärer Instant Messaging-Dienste, gegenwärtig MSN. Interessant ist auch die Funktion, die via Bluetooth Nachrichten an Mobiltelefone in der Nähe sendet, die die Software nicht installiert haben, um so auf die Software aufmerksam machen zu können. Die Voraussetzung dafür ist natürlich, dass die entsprechenden Geräte für andere Bluetooth-Geräte sichtbar sind.

### 3.7.3 Einschränkungen

Das grosse Gewicht, das auf den verteilten, serverlosen Einsatz von Mobiluck gelegt wurde, schlägt sich in bescheidener Funktionalität im Internetportal nieder. Mobiluck ist ein nützliches Hilfsmittel, um bestehende und neue Freunde in Bluetooth-Reichweite zu finden und mit ihnen zu chatten oder Dateien auszutauschen. Der Zugriff auf den erweiterten Freundeskreis oder weitergehende Gerätefunktionen wie die Zell-ID oder vorhandene GPS-Empfänger fehlen jedoch. Ausserdem unterstützt Mobiluck nur direkte Bluetooth-Verbindungen, was die Anwendungsmöglichkeiten wegen der geringen Reichweite von rund zehn Meter deutlich einschränkt.

## 3.8 Sixsense

### 3.8.1 Motivation und Fokus

Sixsense ist ebenfalls eine Anwendung, welche mittels Bluetooth in der Umgebung nach weiteren Anwendern sucht. Speziell bei sixsense ist, dass nicht Mobiltelefone, sondern nur Notebooks den vollen Funktionsumfang bieten. Im Gegenzug steht neben Bluetooth auch Wireless LAN für die Erkundung der Umgebung zur Verfügung. Unterstützt wird die Erstellung von Profilen und das Senden von Mitteilungen. Weiter bietet Sixsense die Möglichkeit, einen Ort mit einer Nachricht zu versehen, welche von anderen Nutzern am selben Ort gelesen werden kann. Es besteht zudem die Möglichkeit, die Position seiner Freunde anzuzeigen.

### 3.8.2 Leistungen und Resultate

Die herausragende Innovation von Sixsense ist der Einsatz von vollwertigen Computern als Plattform. Durch den auf den Campuseinsatz zugeschnittenen Funktionsumfang und die elegante plattformunabhängige Implementation als Firefox-Plugin bietet sich Sixsense als bequem zu bedienende Kommunikations- und Kollaborationsplattform im universitären Umfeld an. Für Nutzer mit Symbian S60- und Windows-Mobile-Telefonen steht ausserdem ein Client mit eingeschränktem Funktionsumfang zur Verfügung.

### 3.8.3 Einschränkungen

Die Ausrichtung auf den Einsatz auf Campuses und den Einsatz von Notebooks und Wireless beschränkt die Umgebungen, innerhalb derer sich Sixsense praktischerweise einsetzen lässt. Trotz eleganter Lösungswege in anderen Bereichen bietet Sixsense beim eigentlichen Social Networking nur die bewährte grundlegende Funktionalität, die andere Netzwerke auch bieten. Features wie das Stöbern im erweiterten Netzwerk oder ein verteilter Zugriffsmodus ohne Serververbindung fehlen jedoch.

## 3.9 Serendipity

### 3.9.1 Motivation und Fokus

Ein Projekt des Massachusetts Institute of Technology, in dessen Rahmen unter anderem eine Mobile Social Networking-Software veröffentlicht wurde. Diese verwendet Bluetooth, um andere Teilnehmer in der Umgebung zu finden und den eigenen Aufenthaltsort zu bestimmen. Durch den Vergleich von manuell eingegebenen Benutzerprofilen werden so interessante Gesprächspartner in der Umgebung gefunden. Serendipity in seiner heutigen Form wurde für den Einsatz an Tagungen konzipiert mit dem Ziel, die Teilnehmer untereinander stärker interagieren zu lassen.

### 3.9.2 Leistungen und Resultate

Serendipity ist als Framework für mobile Applikationen modular und erweiterbar ausgelegt. Mit so unterschiedlichen Anwendungsfeldern wie Mobile Social Networking, Netzwerktheorie oder empirischer Soziologie ist Serendipity mehr als ein Social Network mit dem Potential, bislang unabhängige Forschungsgebiete verbinden und innovative Funktionen hervorbringen zu können. Die Plugin-artige Architektur für die Module, die die Berechnung von Ähnlichkeitsfaktoren auf dem Server erlauben, umfasst neben einfachen profilbasierten Vergleichen auch Module für den Vergleich von Aufenthaltsorten.

### 3.9.3 Einschränkungen

Durch die Ausrichtung der Social Networking-Komponente auf den Einsatz auf dedizierter Hardware mit relativ kurzen Einsatzdauern in gut erschlossenen Umgebungen entfiel die Implementation von verteilten Betriebsmodi oder Datenaustausch über Bluetooth. In der heutigen Implementation übernehmen die Mobiltelefone nur die Suchfunktionen via Bluetooth, alle Berechnungen werden über eine Datenverbindung auf dem Server ausgeführt. Zudem ist nur ein Symbian S60-Client verfügbar, was die Verbreitungschancen von Serendipity in der breiten Bevölkerung stark reduziert.

## 3.10 Lovegety

### 3.10.1 Motivation und Fokus

Die vermutlich erste Realisierung von mobilem Social Networking stammt aus Japan, trägt den Namen Lovegety und wurde im Jahre 1998 von Tamagotchi-Erfinder *Erfolg* auf den Markt gebracht<sup>1</sup>. Es handelt sich hierbei um einen kleinen batteriebetriebenen Funksender, welcher Töne von sich gibt, sobald sich ein passender Gesprächspartner auf fünf bis zehn Meter nähert. Insgesamt wurden mehr als 600'000 solcher Geräte verkauft.

### 3.10.2 Leistungen und Resultate

Das Lovegety-Gerät besticht durch die Einfachheit der Bedienung und die Verwendung von dedizierter Hardware. Es ist praktisch zustandslos (gespeicherte Information  $\approx 2$  bit) und kommt

<sup>1</sup><http://www.wired.com/culture/lifestyle/news/1998/06/12899>

so ohne Server und Serververbindung aus, vermochte aber vielleicht gerade wegen der Einfachheit des Prinzips einen grossen asiatischen Käuferkreis anzusprechen und generierte so einen beträchtlichen Umsatz.

### 3.10.3 Einschränkungen

Bei einer erfolgreichen Pionierleistung fällt es schwer, gravierende Einschränkungen zu finden. Wäre die einfache technische Umsetzung als unbefriedigend empfunden worden, so hätten die Benutzer die Geräte nicht in dieser Zahl gekauft. Interessanterweise konnte dieser Erfolg weder mit Kopien noch anspruchsvolleren Nachahmungen in gleichem Masse wiederholt werden. Möglicherweise ist seit der Allgegenwärtigkeit von Mobiltelefonen die Bereitschaft, ständig mehrere elektronische Geräte mitzuführen, gesunken.

## 3.11 Spontact

### 3.11.1 Motivation und Fokus

Spontact ist ebenfalls eine Software für Mobiltelefone, welche mit Hilfe von Bluetooth nach interessanten Gesprächspartnern in der Umgebung sucht und gegebenenfalls den Benutzer informiert. Die Relevanz anderer Teilnehmer wird anhand eines zuvor eingegebenen Profils bewertet. Spontact wurde als Diplomarbeit im Rahmen einer Public-Private-Partnership zwischen der Hochschule für Technik in Rapperswil und der Crealogix AG entwickelt.

### 3.11.2 Leistungen und Resultate

Spontact vereint Mobile Social Networking-Funktionen mit dem populären OSCAR-Protokoll (ICQ, AOL Instant Messenger) und gleicht in dieser Hinsicht Mobiluck. Durch die Implementation als Java-Applet ist Spontact auf Mobiltelefonen vieler Hersteller ausführbar. Trotz der Ausführung in einer Virtual Machine realisieren die Autoren aber einige elegante Ideen in der Bluetooth-Kommunikation, beispielsweise den Abbruch des Device Discovery-Vorgangs nach dem Erhalt eines ersten Resultats. Ausserdem unterstützt Spontact das Importieren von ICQ-Adressbüchern und ist vorbildlich dokumentiert.

### 3.11.3 Einschränkungen

Durch die Umsetzung im Rahmen einer Doppel-Diplomarbeit ist der Funktionsumfang relativ bescheiden - Spontact unterstützt das Auffinden anderer Mitglieder des Netzwerks in Bluetooth-Reichweite sowie das Chats mit anderen ICQ/AIM-Clients, weitergehende Funktionen oder Servertechnologie ausserhalb der OSCAR-Server sind nicht enthalten. Der Einsatz der verschiedenen Datenschnittstellen ist aussergewöhnlich: Profile können nur über Bluetooth ausgetauscht werden, aber ein nachfolgender Chat setzt auf beiden Geräten eine Datenverbindung über das Mobilfunknetz voraus.

## 3.12 BlueDating

### 3.12.1 Motivation und Fokus

BlueDating ist eine weitere Entwicklung für Mobiltelefone, welche Bluetooth verwendet, um mit anderen Nutzern, welche sich in der Nähe aufhalten, zu kommunizieren. Auch bei dieser Variante werden interessante Gesprächs- und andere Partner durch den Vergleich von manuell eingegebenen Profilen gefunden. BlueDating ist Teil des Blue\*-Projekts der Communication Systems Group der ETH Zürich.

### 3.12.2 Leistungen und Resultate

Durch die Implementation im Rahmen eines grösseren Projekts mussten sich die Autoren weniger um technische Details kümmern und waren so freier in der Gestaltung der Funktionalität. Dieser Umstand findet seinen Ausdruck beispielsweise in einem umfangreichen Profilsystem, das für das Matching eingesetzt wird, oder einem ausgereiften GUI-Konzept für den Java-Client. Interessant ist auch das Augenmerk, das die Autoren auf einen kommerziellen Einsatz der Software gelegt haben, und die dafür vorgesehenen Konzepte.

### 3.12.3 Einschränkungen

Aufgrund der Realisierung von BlueDating im Rahmen einer Doppel-Semesterarbeit ist der Funktionsumfang relativ bescheiden, aber in sich abgeschlossen; viele Möglichkeiten, die die mobile Plattform bietet, werden nur unvollständig eingesetzt. Datenschutzüberlegungen finden in der zugehörigen Dokumentation einen verhältnismässig grossen Platz, die daraus abgeleiteten Massnahmen lassen sich aber mit minimalem Aufwand umgehen.

# Kapitel 4

## Aufgabenstellung

Vor dem Hintergrund der im Kapitel 2) beschriebenen Situation erscheint es überraschend, dass das Mobile Social Networking bislang ein Nischendasein führt. Der grosse Erfolg der internetbasierten Social Networking-Dienste, die geeigneten technischen Voraussetzungen, welche die Mobiltelefone für Mobile Social Networking-Anwendungen bereits mitbringen, und nicht zuletzt die guten Prognosen versprechen dem Mobile Social Networking eigentlich grosse Erfolgsaussichten. Trotz des Vorhandenseins einer relativ grossen Zahl von Social Networking-Implementationen (siehe Kapitel 3) konnte sich jedoch noch kein solcher Dienst etablieren. In diesem Kapitel diskutieren wir mögliche Ursachen und Lösungsansätze für die Probleme, die auch in der Aufgabenstellung (siehe Appendix B angesprochen werden.

### 4.1 Attraktivität des Dienstes in der Startphase

Die erste Zeit nach dem Start ist eine schwierige Zeit für Social Networking-Dienste, da diese erst eine gewisse Anzahl Nutzer erreichen müssen, um für diese überhaupt interessant zu sein und von Mund-zu-Mund-Propaganda profitieren zu können – wir nennen diese *kritische Nutzerzahl* oder *-dichte*. Damit die kritische Dichte deshalb überhaupt erreicht wird, ist es besonders in der Anfangsphase wichtig, dass die Nutzer auch von einem Dienst mit relativ geringer Teilnehmerzahl profitieren können<sup>1</sup>.

Gerade bei mobilen Diensten, welche nicht über einen zentralen Server verfügen und nur Benutzer innerhalb eines relativ kleinen räumlichen Radius' erreichen können, liegt diese besonders hoch, da sonst die Wahrscheinlichkeit, dass sich noch andere Nutzer in der Umgebung aufhalten, zu klein ist, um einen sinnvollen Betrieb zu ermöglichen. Ein zentraler Server senkt also die kritische Nutzerdichte, indem er die Teilnehmer verbindet, ohne dass diese sich am selben Ort aufhalten müssen, erfordert aber temporäre oder permanente Verbindungen zum Internet.

### 4.2 Kostengünstiger Betrieb

Wie im Kapitel 3 beschrieben, sind viele bereits bestehende Mobile Social Networking-Dienste für den bestimmungsgemässen Betrieb auf eine ständige Internetverbindung mit einem Server angewiesen. Diese Verbindung ist in den meisten Fällen relativ teuer, und da diese Dienste normalerweise häufig genutzt werden – Myspace-Nutzer greifen im Durchschnitt 215 Minuten pro Monat auf den Dienst zu<sup>2</sup> – sind die dabei anfallenden Kosten für viele Anwender zu hoch.

<sup>1</sup>Die verlinkte Grafik verdeutlicht, wie wichtig für den nachhaltigen Erfolg eine grosse Benutzergruppe nur schon für serverbasierte Dienste im Internet ist:

[http://www.alexam.com/data/details/traffic\\_details?site0=facebook.com&y=p&z=0&h=400&w=700&range=3y&size=Large&url=myspace.com](http://www.alexam.com/data/details/traffic_details?site0=facebook.com&y=p&z=0&h=400&w=700&range=3y&size=Large&url=myspace.com)  
Die Abbildung zeigt die Anzahl der Seitenaufrufe der Seite [www.facebook.com](http://www.facebook.com) während der letzten drei Jahre. Es ist gut zu erkennen, wie schnell die Nutzerzahl angestiegen ist, nachdem eine gewisse Anzahl Nutzern erreicht wurde. Facebook wurde bereits 2004 gegründet.

Ein weiteres Beispiel ist [www.orkut.com](http://www.orkut.com), der Dienst von Google:

[http://www.alexam.com/data/details/traffic\\_details?site0=orkut.com&y=p&z=3&h=400&w=700&range=3y&size=Large&url=myspace.com](http://www.alexam.com/data/details/traffic_details?site0=orkut.com&y=p&z=3&h=400&w=700&range=3y&size=Large&url=myspace.com)

<sup>2</sup>[http://www.businessweek.com/technology/content/may2006/tc20060530\\_170086.htm](http://www.businessweek.com/technology/content/may2006/tc20060530_170086.htm)

Einen eleganten Lösungsansatz für dieses Problem fand Myspace: Dieser Betreiber ist mit Cingular Wireless, dem grössten Mobilfunknetzbetreiber in den USA, eine Partnerschaft eingegangen. Das ist für beide Seiten attraktiv, denn für Netzbetreiber könnte Mobile Social Networking zu einer attraktiven zusätzlichen Einnahmequelle werden. Bereits in wenigen Jahren sollen mindestens fünf Prozent aller gesendeten Kurznachrichten (SMS) auf mobile Social networking-Dienste zurückgehen, schätzt Tole Hart, Analyst bei Gartner<sup>3</sup>. Zudem würden bei serverbasierten Diensten Einnahmen durch den zusätzlichen Datentransfer entstehen, für welche die Netzbetreiber Abonnemente verkaufen könnten.

Falls die Möglichkeit einer solchen Kooperation aber nicht gegeben ist und nicht gänzlich auf einen zentralen Server verzichtet werden kann oder soll, liegen die Möglichkeiten für die Reduktion der Zugriffsgebühren im Einsatz von PC-Software für die Übertragung der Daten über den Internetanschluss eines Desktop-Computers oder in der Reduktion der Verbindungszeit und des übertragenen Datenvolumens. Dabei kann der Datenverkehr reduziert werden durch den Einsatz effizienter Protokolle, Kompression, oder den Aufbau eines Mesh-Netzwerkes, das die einzelnen Teilnehmer über mehrere Verbindungen zusammenschliesst und einen Teil der Nachrichten ohne Umweg über das Internet an seinen Bestimmungsort bringen kann.

### 4.3 Benutzerfreundlichkeit auf dem Mobiltelefon

Ein grosses Problem einiger Mobile Social Networking-Dienste ist die schlechte Adaptierung an die mobile Umgebung, welche oft daraus resultiert, dass die mobilen Dienste die Konzepte der grossen internetbasierten Pendanten im wesentlichen unverändert auf ein kleines Display kopieren. Um dieses Problem zu vermeiden, ist es wichtig, dass die Anwendungen gut an die jeweilige mobile Plattform angepasst sind – komplizierte Funktionen sollen vermieden, vereinfacht, erklärt oder in Menüs verborgen werden, und die dem Nutzer präsentierte Informationsmenge soll sich an der bescheidenen Displaygrösse orientieren.

Andererseits bietet die mobile Plattform nicht nur Einschränkungen. Im Vergleich zum Internet sind die Mobilität und die beinahe permanente Erreichbarkeit des Nutzers bestechende Eigenschaften. Gleichermassen erfordert der Einbezug von im Kalender, Telefonbuch und Fotoalbum gespeicherten Informationen oder der aktuellen geographischen Position zwar die Abwendung von auf dem Desktop-Computer vertrauten Paradigmen, erlaubt jedoch die komfortable Bedienung oder gar Automatisierung bestimmter Funktionen. Diese müssen dem Benutzer jedoch auf einfache Weise präsentiert und zugänglich gemacht werden.

---

<sup>3</sup>[http://www.businessweek.com/technology/content/may2006/tc20060530\\_170086.htm](http://www.businessweek.com/technology/content/may2006/tc20060530_170086.htm)

# Kapitel 5

## Methodik- und Architekturentscheide

In Kapitel 3 wurde ein weit gefächertes Spektrum verwandter Mobile Social Networking-Anwendungen vorgestellt. Während einige Merkmale dieser Geräte und Programme auf jedes Produkt zutreffen (beispielsweise das Ziel, die Interaktion zwischen einander unbekanntenen Personen in öffentlichen Umgebungen zu erleichtern), unterscheiden sie sich in Bezug auf andere Eigenschaften stark. Viele dieser Ansätze betreffen direkt oder indirekt die Themen der Implementation, der Kosten für den Benutzer beim Gebrauch des Dienstes, den Benutzerkomfort und die unterstützten Plattformen bzw. Geräte.

### 5.1 Benutzerschnittstelle

Einige der angebotenen Dienste setzen ganz oder zu einem wesentlichen Teil auf den Einsatz von mobilen Internetbrowsern, die die Funktion der Benutzerschnittstelle übernehmen. Die Implementation einer mobilen Anwendung als Webdienst ist bewährt, relativ einfach und im Sinne der Plattformunabhängigkeit attraktiv, kann für den Benutzer aber hohe Verbindungskosten verursachen, die durch den Einsatz eines spezialisierten Programmes reduziert werden können. Diese Dienste werden oftmals mit einem Kurznachrichten-Gateway kombiniert, um dem Nutzer eine schnelle Möglichkeit zu bieten, seinen Aufenthaltsort zu aktualisieren.

Im Gegensatz dazu verfügen die Programme, die als Java- oder Symbian-Applikation[?, 7] auf einem Mobiltelefon laufen, über die Möglichkeit, grössere Mengen an Daten auf dem Gerät zwischenspeichern, Berechnungen lokal auszuführen oder ein an das jeweilige Gerät angepasstes Darstellungsprofil zu wählen (beispielsweise an dessen Displaybreite, um horizontales Scrolling zu vermeiden). Diese Kombination von Server und mobilem Client ermöglicht Dienste, die auch ohne permanente Verbindung zu einem Server ihre Funktion erfüllen oder auf zentrale Server völlig verzichten. Der erhöhte Implementationsaufwand für solche Programme rechtfertigt sich durch reduzierten Datenverkehr dank binärem Protokoll und die flexiblere Gestaltung der Benutzerschnittstelle.

Dank der weiten Verbreitung von Java auf Mobiltelefonen sind vorcompilierte Programmarchive, sogenannte Midlets, auf praktisch allen modernen Mobiltelefonen ausführbar. Der J2ME-Standard unterstützt inzwischen auch den Datenaustausch über Bluetooth und das Internet oder die standardisierte Steuerung von integrierten Lautsprechern, Videokameras oder GPS-Empfängern.

Um den Aufwand für die Implementation eines auf möglichst vielen Mobiltelefonen unterschiedlicher Marken funktionierenden Dienstes im Rahmen einer Semesterarbeit zu halten und nicht auf den bequemen Zugriff zu Bluetooth[5]-, GPS[3]- und Telefonbuchschnittstelle[6] verzichten zu müssen, wählten wir für die Realisierung eines Proof-of-Concept-Midlets die *Java Platform, Mobile Edition*.

## 5.2 Erfassung von Standorten

Zur Erfassung des gegenwärtigen (und ggf. zukünftigen) Aufenthaltsortes der Benutzer bieten sich verschiedene Methoden an: die manuelle Eingabe des Namens oder der Adresse eines Ortes, die Auswahl aus einer vorgefertigten Liste (bei den Campusapplikationen), oder die automatische Aquisition von Koordinaten via GPS, Netzfunktionen von GSM, oder spezielle Bluetoothsender, deren Standort bekannt ist. Da in mobilen Browsern nach dem heutigen Stand der Technik nicht auf integrierte Gerätefunktionen zugegriffen werden kann, setzt die automatische Erfassungsmethode mittels GPS die Ausführung von Programmcode voraus.

Für Applikationen, die nicht nur innerhalb einer begrenzten Umgebung wie einem Campus oder auf einer Konferenz eingesetzt werden sollen, ist das abschliessende Aufzählen aller in Frage kommender Standorte allerdings nicht möglich. In diesem Fall bleiben nur Adressen und Koordinaten für die eindeutige Bestimmung einer Position, allenfalls in Kombination mit einer Liste der populärsten Aufenthaltsorte. Dabei müssen die Vorteile von automatischer Erfassung und Übermittlung des Standortes beim Einsatz von GPS oder vergleichbaren Technologien mit den Vorteilen der interaktiven Methoden verglichen werden, die Unabhängigkeit von der Hardware des Endgeräts und der lokalen Abdeckung durch den Dienst, intuitiv nachvollziehbare Funktionsweise und zusätzliche Kontrolle des Benutzers über seine Privatsphäre einschliessen.

Da wir allerdings einen Dienst realisieren wollten, der hohen Nutzerkomfort bietet und die räumliche Position unabhängig von vorgängiger Kartographierung nutzt, entschieden wir uns für die Aquisition von Koordinaten auf GPS-fähigen Mobiltelefonen. Dazu steht auf kompatiblen Geräten das JSR 179 zur Verfügung, das eine schlanke Schnittstelle für die Bestimmung der Position des Gerätes bereitstellt.

## 5.3 Serverbasierte und verteilte Dienste

Während einige Mobile Social Networks gänzlich auf den Einsatz von zentraler Serverinfrastruktur verzichten, sind für weitergehende Funktionen wie Postfächer, auf Karten eingezeichnete Routen verschiedener Benutzer oder virtuelle Pinwände persistente Datenspeicher nötig, die von allen Nutzern möglichst jederzeit kontaktiert werden können. Nur wenige dieser Konzepte vermögen allerdings diesen Nachteil durch ein raffiniertes Benutzungskonzept aufzuwiegen.

Die verteilte Natur von Mobile Social Networks und die geringe Reichweite der heutigen Funktechnologien schliessen eine vollständig verteilte Implementation solcher Funktionen aus und machen zentrale Server für solche Funktionen notwendig. Da diese Funktionen aber nur gelegentlich eingesetzt werden, besteht die Möglichkeit, zu diesen Servern nur bei Bedarf zu verbinden, da so die für die Benutzer anfallenden Kosten gesenkt werden können.

## 5.4 Fazit

Für das Gelingen dieser Semesterarbeit war die Auswahl einer geeigneten Architektur von entscheidender Bedeutung. Dabei bot sich nicht nur die Gelegenheit, einigen der im vorhergehenden Kapitel angesprochenen Schwierigkeiten auszuweichen, sondern auch andere wichtige Eigenschaften der Applikation wie unterstützte Plattformen oder Stromverbrauch zu beeinflussen. In diesem Abschnitt erörtern wir den geplanten Aufbau von veNETa und begründen diese Entscheide.

Das Ziel dieser Arbeit war die Realisierung eines Mobile Social Networking-Dienstes, der für die Benutzer neuartig und attraktiv, einfach zu bedienen und kostengünstig im Einsatz ist. Von besonderer Bedeutung war dabei, dass das Finden interessanter Kontakte in der Umgebung nicht nur durch den Vergleich von manuell eingegebenen Profilen, sondern auch auf automatischem Weg durch den Einsatz von auf dem Gerät gespeicherten Daten erfolgt. Um auch in der Anfangsphase möglichst viele Benutzer verbinden zu können, entschieden wir uns ausserdem für den Einsatz eines zentralen Servers und die Übertragung dieser Daten via Internet.

Um aber den Datenverkehr über das Internet und damit die Kosten für die Nutzer möglichst gering zu halten, sollten die Daten nicht auf den Server übertragen und dort verarbeitet, sondern möglichst direkt zwischen den Nutzern ausgetauscht werden, auch über mehrere Ver-

bindungen. So können insbesondere in Umgebungen mit hoher Nutzerdichte viele Teilnehmer miteinander verbunden werden, ohne dass dabei Gebühren der Netzbetreiber anfallen. Die einzige auf Mobiltelefonen verbreitete Technologie, die sich dafür eignet, ist Bluetooth.

Damit die Nutzung des Dienstes sich nicht nur auf das Auffinden von Gesprächspartnern beschränkt, sollte veNETa auch den Austausch von verschlüsselten Textnachrichten unterstützen. Diese sollten zunächst über Bluetooth gesendet und nur beim Ausbleiben einer Antwort aus der Umgebung über das Internet verschickt werden.

Um veNETa zu einem echten Mobile Social Network zu machen, wollten wir ausserdem den Standort des Benutzers in den Aufbau des Dienstes integrieren. Die Erfassung solcher Standortdaten vereinfacht das JSR 179, das die Lokalisierung des Mobiltelefons via GPS (und ggf. andere unterstützte Methoden) erlaubt. Zum Auffinden anderer Benutzer in einem grösseren Bereich sollten diese auf dem Server verglichen werden können.

Aufbauend auf diesen Überlegungen entschieden wir uns zur Implementation eines Clientprogramms für die Java Platform Mobile Edition. Die einzige realistische Alternative zu J2ME, die Implementation eines Client-Programmes für das Symbian OS, verwarfen wir wegen geringerer Verbreitung und aufgrund von Vorwissen in der Java- und J2ME-Programmierung. Nach diesem Entscheid lag es nahe, die *Java Platform, Standard Edition* für die Serverprogrammierung zu verwenden, weil sie den Einsatz von identischen Codefragmenten im Client- und Server-Teil ermöglichte, beispielsweise für Verschlüsselungsfunktionen.

# Kapitel 6

## Implementierung

veNETa ist eine auf J2ME basierende Mobile Social Networking-Software, welche geschrieben wurde, um einen Lösungsansatz zu dem genannten Problemen zu bieten. Sie besteht aus einer Art "Friend Finder", der andere Teilnehmer in der Umgebung sucht, welche die gleichen Telefonnummern im Telefonbuch gespeichert haben. Die Idee dahinter ist exakt dieselbe, die den populären Friend-of-Friend-Funktionen etablierter Dienste zugrundeliegt: Dass ein solcher gemeinsamer Bekannter oft mehr aussagt als ein übereinstimmendes Profil. Auf diese Weise ist veNETa in der Lage, Angehörige des erweiterten Freundesnetzwerks ohne zentrale Infrastruktur automatisch aufzufinden.

veNETa verfügt über einen in Java Platform, Standard Edition programmierten, zentralen Server, über welchen verschlüsselte Mitteilungen an andere Nutzer versendet werden können. Die dazu benötigten Schlüssel können via Bluetooth verteilt oder direkt vom Server bezogen werden. Über Bluetooth ist es ebenfalls möglich, verschlüsselte Nachrichten zu übertragen. Wenn ein Mobiltelefon seine Position entweder mittels eines netzbasierten Verfahrens oder mit Hilfe eines GPS-Empfängers bestimmen kann, dann können zudem andere Teilnehmer in der Umgebung angezeigt werden.

Um die Wiederverwendbarkeit des Programmcodes zu gewährleisten und gleichzeitig die Aufteilung der Implementationsarbeit zu vereinfachen, wurden die Funktionen von veNETa nicht monolithisch, sondern modular realisiert. Dieses Vorgehen verringerte den Testaufwand, erleichterte die Aufteilung der Aufgaben dieser Doppelsemesterarbeit und vereinfacht zukünftige Erweiterungen von veNETa. Der Aufbau ist in Abbildung 6.1 abgebildet.

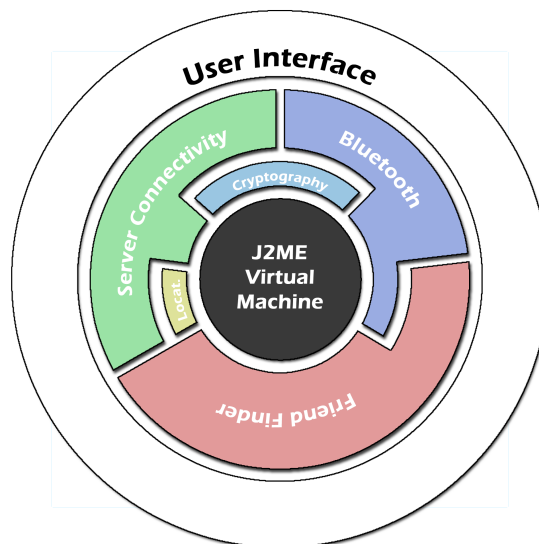


Abbildung 6.1: Der modulare Aufbau von veNETa. Die kleinen Sektoren stehen für die Location- und Krypto-Module.

Auf dem Mobiltelefon startet die Main-Klasse die benötigten Subsysteme und verwaltet einen

Teil der Benutzerschnittstelle. Die anderen Module werden entweder bei Bedarf zur Laufzeit gestartet (Friend Finder- und TCP-Module) oder als Singletons von der Main-Klasse geladen (Einstellungs-, Crypto- und Bluetooth-Modul). In diesem Kapitel werden die relevanten Einzelheiten der implementierten Module beschrieben.

## 6.1 Die Main-Klasse

Die Main-Klasse enthält Funktionen für den Start und die Verknüpfung der verschiedenen Module von veNETa und stellt das Hauptmenu dar. Die Main-Klasse prüft nach jedem Start, ob das Programm bereits vollständig konfiguriert wurde, indem sie das Einstellungs-Modul anweist, die Benutzer- und Konfigurationsdaten zu lesen. Abhängig vom Resultat wird entweder der Hauptbildschirm angezeigt, oder das Welcome Screen-Modul gestartet, das es dem Benutzer erlaubt, die vor dem ersten Start notwendigen Einstellungen vorzunehmen. Gleichermassen wird beim Beenden von veNETa das Beenden der Server- und Bluetoothverbindungen sowie das Speichern der Einstellungen veranlasst.

## 6.2 Das Welcome Screen-Modul

Nach dem Installieren des veNETa-Midlets auf dem Mobiltelefon ist die Applikation noch unkonfiguriert und nicht einsatzbereit. Damit der Benutzer von veNETa eindeutig identifiziert werden kann, muss das RSA-Schlüsselpaar erzeugt werden. Für einwandfreie Bedienung ist es ausserdem erforderlich, dass die Benutzerdaten im Profil abgelegt, der gewünschte Suchmodus ausgewählt und Telefonbucheinträge für das Friend Finder-Modul gespeichert werden, die für den Vergleich von Telefonbüchern herangezogen werden können. Das Welcome Screen-Modul erfüllt diese Aufgaben, indem es dem Benutzer auf verschiedenen Bildschirmen die Bedeutung der Einstellungen erklärt und die notwendigen Angaben entgegennimmt. Ausserdem zeigt das Welcome Screen-Modul die für dieses Programm massgebliche Version der GNU GPL [11] an, da diese dem Benutzer in interaktiven Programmen zwingend angezeigt werden muss.

## 6.3 Das Bluetooth-Modul

### 6.3.1 Einführung

Der Bluetooth-Standard beschreibt eine Funktechnologie mit bescheidenem Stromverbrauch, relativ geringer Datenrate und einigen Metern Reichweite. Da die meisten modernen Mobiltelefone über die für Bluetooth-Kommunikation nötige Hardware verfügen und der Zugriff auf diese Funktionalität innerhalb der Java Virtual Machine im JSR-82 [5] standardisiert ist, ist es üblicherweise unnötig, innerhalb von Java-Applikationen einen kompletten Bluetooth-Stack zu implementieren. Stattdessen wird innerhalb der Virtual Machine das im JSR-82 definierte High-Level-Interface angesprochen, das die implementationsspezifischen Details vor der Software verbirgt. Dieser Ansatz hat den Vorteil, dass er die Software portabel macht und ihre Komplexität niedrig hält. In der Praxis werden diese Vorteile allerdings durch unvollständige und instabile Implementierungen teilweise wieder aufgehoben.

### 6.3.2 Funktionalität

Das Bluetooth-Modul stellt den anderen Softwarekomponenten eine einfache, auf die Bedürfnisse von Social Networking-Software ausgerichtete Schnittstelle zur Verfügung. Diese umfasst eine globale, regelmässig aktualisierte Liste aller Geräte in Reichweite, die ebenfalls die veNETa-Software ausführen, sowie eine Liste aller gegenwärtig hergestellten Verbindungen mit Geräten in der Umgebung. Falls Verbindungen neu erstellt werden oder abbrechen, werden Callback-Funktionen aufgerufen. Ausserdem unterhält das Bluetooth-Modul eine Liste von bereits gefundenen Mobiltelefonen, die die veNETa-Software nicht ausführen, damit im Interesse

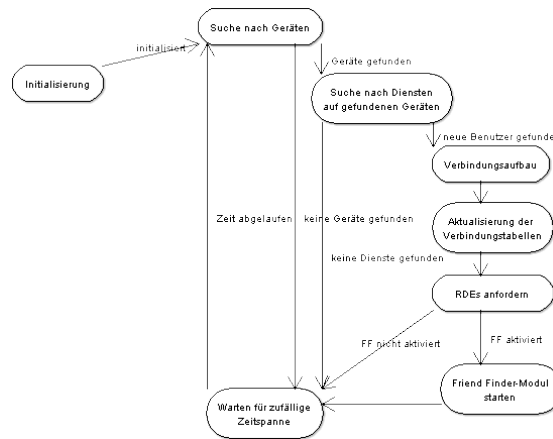


Abbildung 6.2: Die Zustandsfolge des Bluetooth-Moduls.

der Energieeffizienz diese nicht bei jeder Suche nach Geräten für die Suche nach dem veNETa-Dienst kontaktiert werden müssen.

Zusätzlich ist im Bluetooth-Modul ein modifizierter Floodingmechanismus enthalten, der Nachrichten an entfernte Geräte weiterleitet. Beim Empfang einer Nachricht von einem benachbarten Gerät oder beim Versenden einer neuen Nachricht prüft das System zunächst, ob der Empfänger direkt erreichbar ist, und leitet gegebenenfalls die Nachricht mit dekrementiertem Time-to-live-Feld direkt weiter. Anderenfalls wird die Nachricht auf allen Verbindung ausser derjenigen, über die die Nachricht empfangen wurde, gefloodet. Dieser Ansatz erlaubt es, mit einer TTL von zwei ein drei Verbindungen entferntes Gerät zu erreichen, da die letzte Verbindung nicht mit Flooding, sondern gezielt überbrückt wird. Weitere Angaben zu diesem Thema finden sich in Appendix A.1.

### 6.3.3 Implementierung

Diese Funktionen werden im Bluetooth-Modul in einem endlichen Zustandsautomaten (siehe Abbildung 6.2) realisiert. Nach der Initialisierung und dem Start des Bluetooth-Moduls wird über das Java-Bluetooth-Interface zuerst nach Geräten, und auf diesen danach nach dem veNETa-Dienst gesucht. Falls das Programm auf den gefundenen Geräten ausgeführt wird, erstellt das Bluetooth-Modul eine Verbindung, anderenfalls wird die Adresse des Gerätes in die Liste inaktiver Geräte aufgenommen. Daraufhin schläft die Suchfunktion für eine zufällige Zeitspanne und wiederholt darauf den Vorgang. So wird vermieden, dass gleichzeitig eingeschaltete Geräte sich nicht finden können, weil sie immer gleichzeitig nach Geräten in der Umgebung suchen. Sobald eine Verbindung aufgebaut wurde, sendet das Bluetooth-Modul eine spezielle Nachricht, die Remote Device Entry-Nachricht, an das neu gefundene Gerät, die unter anderem den Namen des Benutzers und dessen öffentliche Schlüssel enthält. Dann wird mittels Callback die Kontrolle an das Friend Finding-Modul abgegeben, sofern veNETa entsprechend konfiguriert ist.

Parallel dazu nimmt das Bluetooth-Modul Nachrichten von Verbindungen zu anderen Geräten und von anderen Modulen an, die gemäss dem modifizierten Flooding-Algorithmus weitergeleitet werden.

## 6.4 Das Crypto-Modul

### 6.4.1 Einführung

Um die Authentizität und Integrität von über Funknetzwerke und das Internet gesendete Nachrichten zu garantieren und gegebenenfalls die Vertraulichkeit der verschickten Daten sicherzustellen, ist es heutzutage Stand der Technik und für viele Anwendungen für den Schutz der

Privatsphäre unabdingbar, Daten kryptographisch gesichert zu übertragen. Da sichere Verschlüsselungsverfahren der erforderlichen Stärke, insbesondere die verbreitete asymmetrische Kryptographie, einen für Mobiltelefonprozessoren beträchtlichen Rechenaufwand nach sich ziehen, kommt der Wahl des Verschlüsselungsverfahrens und ihrer Parameter eine entscheidende Bedeutung zu. Diese Entscheide werden allerdings dadurch erleichtert, dass Angaben über diese sorgfältig studierte mathematische Disziplin verfügbar und dank freier Bibliotheken auch einfach zugänglich sind.

### 6.4.2 Funktionalität

Im Crypto-Modul stellt in einer zustandslosen Klasse das zentrale Interface für Ver- und Entschlüsselung sowie die Überprüfung von Signaturen bereit. Diese Funktionen wurden mittels des RSA-Algorithmus [9] implementiert, der bei der verwendeten Schlüsselstärke im Vergleich zu anderen Verfahren ein vorteilhaftes Verhältnis von Sicherheit der Datenübertragung zu Rechenaufwand besitzt.

### 6.4.3 Implementierung

Während es eine spannende und herausfordernde Aufgabe wäre, einen auf die J2ME-Umgebung zugeschnittenen, bezüglich Stromverbrauch und Ausführungszeit optimierten Ver- und Entschlüsselungsalgorithmus oder eine entsprechende Implementation zu schreiben, entschieden wir uns im Interesse von Termintreue und der Vermeidung von Fehlern in sicherheitsrelevanten Bereichen für die Verwendung der als Freie Software lizenzierten Bouncy Castle-Library [10]. Das Crypto-Modul stellt also nur eine Schnittstelle zwischen den von veNETa benötigten Verschlüsselungsfunktionen und der praxiserprobten Bibliothek bereit. Dabei wurde als Kompromiss zwischen Prozessorbelastung und Verschlüsselungsstärke eine Schlüssellänge von 768 bit gewählt. Für den Hashing-Algorithmus zur Signierung von Nachrichten wählten wir aus denselben Gründen den 128 bit langen MD5-Algorithmus.

Falls sich in der Zukunft die Wahl der Parameter als technisch überholt erweisen sollte, können durch diesen Aufbau an einer zentralen Stelle die Parameter der gesamten Verschlüsselungsfunktionalität angepasst oder, falls gewünscht, auch durch ein anderes Verfahren ersetzt werden.

## 6.5 Friend Finding-Modul

### 6.5.1 Einführung

Die Hauptaufgabe von veNETa ist die Suche nach "interessanten" Personen in der Umgebung des Benutzers. Während die naheliegendste Lösung dieser Aufgabe die Erfassung von Alter, Geschlecht, Hobbies etc. in einem Profil und der Vergleich von Profilen über das Funknetzwerk darstellt, waren wir der Überzeugung, dass diese Aufgabe dank der auf modernen Mobiltelefonen gespeicherten Informationen eleganter und für den Benutzer komfortabler gelöst werden können muss. Die zentrale Idee hinter dem für das Friend Finding-Modul gewählten Ansatz besteht darin, dass Personen, die sich nicht direkt, aber über einen gemeinsamen Bekannten kennen, häufig auch ähnliche Interessen haben könnten. So bringt das Friend-Finding-Modul das populäre Konzept des erweiterten Freundeskreises, ein wichtiges Element vieler etablierter Social Networks im Internet, auch auf verteilte mobile Plattformen.

Bei solchen Anwendungen, die auf Teile des Telefonbuches auf dem Mobiltelefon zugreifen und diese über Funknetzwerke vergleichen, ist allerdings der Datenschutz ausserordentlich wichtig. Die Voraussetzung für eine akzeptable Lösung war, dass ein Angreifer nur mit einem sehr hohen Aufwand Informationen über die im Telefonbuch gespeicherten Nummern anderer Benutzer erlangen kann, im Idealfall nur durch die Suche über den gesamten Raum der Mobiltelefonnummern. Gleichzeitig sollte der Vergleich aber zuverlässig und möglichst schnell ablaufen, damit Personen sich nicht unrealistisch lange in Reichweite ihrer Bluetoothsender aufhalten müssen, bis der Telefonbuchvergleich abgeschlossen ist, auch wenn sie über eine grosse Anzahl von Kontakten verfügen. Für diesen Vergleich werden kryptographische Verfahren der Klasse der Two Party Computation Set Intersection verwendet.

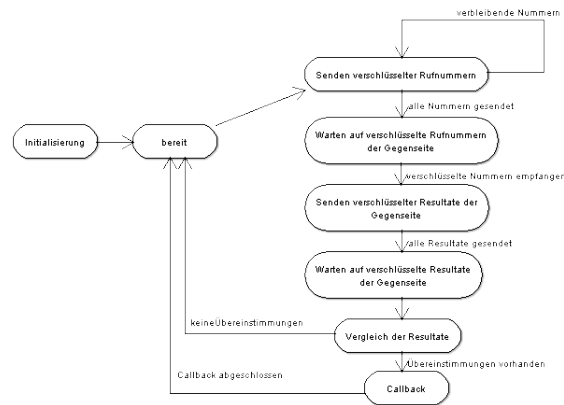


Abbildung 6.3: Die Zustandsfolge des Friend Finding-Moduls.

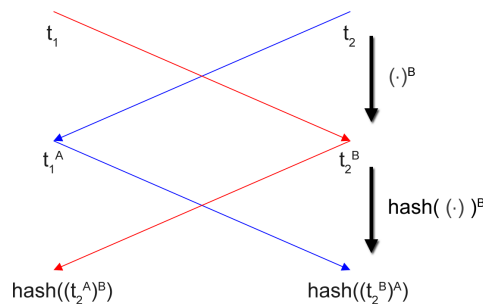


Abbildung 6.4: Der Interaktionsmechanismus des Friend Finding-Moduls.

### 6.5.2 Funktionalität

Das Friend Finding-Modul stellt für die Hauptfunktion des Friend Finding einen einzelnen Funktionsaufruf bereit, der die benötigten Nachrichten an den Empfänger schickt, dessen Antworten auswertet und allfällige Übereinstimmungen der beiden Telefonbücher an das User Interface weiterleitet und so die Hauptaufgabe eines verteilten social networking-Systems wahrnimmt. Um aber bereits in der Anfangsphase des Netzwerks, wo die Nutzerdichte bescheiden und die Wahrscheinlichkeit von Übereinstimmungen zwischen verschiedenen Telefonbüchern entsprechend gering ist, eine möglichst attraktive Dienstleistung anbieten zu können, unterstützt veNETa ausserdem das Suchen nach Alter und Geschlecht und bietet die Möglichkeit, ein eigenes Profil mit Bild anzulegen, das von anderen Benutzern eingesehen und als Basis für die Kontaktaufnahme verwendet werden kann.

### 6.5.3 Implementierung

Auch das Friend Finding-Modul ist als endlicher Zustandsautomat implementiert. Nach dem Aufruf durch die Callback-Funktionalität des Bluetooth-Moduls werden die auf die relevanten Stellen gekürzten und mit dem Friend Finding-Schlüssel chiffrierten Telefonnummern über Bluetooth versendet, und nach Erhalt der Nachrichten von der Gegenseite ebenfalls verschlüsselt und zurückgeschickt. Der genaue Ablauf ist in Abbildung 6.5.3 dargestellt, Abbildung ?? zeigt das Interaktionsschema des Algorithmus, wobei der Übersichtlichkeit halber die  $\text{mod } p$  der Gruppenoperationen weggelassen wurden. Bei diesen Operationen wird der Umstand ausgenutzt, dass in der Gruppe  $G$  die Gleichung  $g^{AB} = g^{BA}$  erfüllt ist und somit gleiche Eingabewerte unabhängig von der Reihenfolge der Operationen auch gleiche Resultate ergeben.

Die Sicherheit dieses Algorithmus' beruht direkt auf dem diskreten Logarithmusproblem und

damit auf der Schwierigkeit, bei einem bekannten  $p$  aus dem Resultat von  $g^A \bmod p$  auf  $g$  oder  $A$  zu schliessen.

TODO: Erklärung MPC wie auf S. 27 im WK-Script — Hinweise auf Fehler in Beerliova — Cite Beerliova, Mentalpoker, Efficient Secure Multi-Party Computation

## 6.6 Server

### 6.6.1 Einführung

Da der Friend Finder von veNETa wegen der beschränkten Reichweite von Bluetooth nur innerhalb einer begrenzten Umgebung funktioniert, ist er auf eine grosse Nutzerzahl angewiesen, um für die Anwender erst interessant zu werden. Denn mit nur einer geringen Anzahl von Teilnehmern ist die Wahrscheinlichkeit, dass sich andere Nutzer in Reichweite aufhalten, sehr klein. Damit veNETa aber auch mit nur wenig vorhandenen Anwendern zu Beginn der Startphase einen interessanten und attraktiven Dienst darstellt, ist zusätzlich die Verwendung eines zentralen Servers möglich, welcher den Nutzer untereinander verbindet. Für die Verwendung des Servers ist eine Internetverbindung notwendig. Die meisten neu verkauften Telefone in Westeuropa sind jedoch fähig, eine solche Verbindung herzustellen. Gemäss einer Studie der Universität Cambridge verwendeten im Jahr 2005 in Westeuropa 45 Prozent der Besitzer eines internetfähigen Mobiltelefons mindestens einmal im Monat den mobilen Internetzugang<sup>1</sup>. Deshalb ist durchaus wahrscheinlich, dass die Nutzer auch den Serverbasierten Teil des Dienstes akzeptieren werden. Da die Kosten für Datenverbindungen von Mobiltelefonen aus sehr teuer sein können, wurde darauf geachtet, dass möglichst wenig Daten übertragen werden müssen.

### 6.6.2 Funktionalität

Der Server hat mehrere Aufgaben. Er ermöglicht den Benutzern das Versenden von verschlüsselten Nachrichten an andere Teilnehmer von veNETa. Für die Empfängeradresse der Nachricht wird der MD5-Hash des Schlüssels des Empfängers verwendet. Die Nachrichten können auch an Nutzer versendet werden, welche nicht gerade mit dem Server verbunden sind. In dem Fall speichert der Server die Mitteilung und liefert sie aus, sobald sich der entsprechende Teilnehmer meldet. Wenn eine Nachricht erfolgreich zugestellt wurde, sendet der Server eine Empfangsbestätigung an den Absender.

#### Schlüsselaustausch

Der Server ist zudem zuständig für die Registrierung der Schlüssel. Jeder Anwender hat die Möglichkeit, seinen privaten Schlüssel beim Server auf seine Mobiltelefonnummer registrieren zu lassen. Dieser überprüft die Nummer, indem er eine Kurznachricht mit einer Zufallszahl an diesen Teilnehmer sendet. Dieser gibt die Zahl auf seinem Mobiltelefon ins Programm ein, welches die Nummer zurück zum Server sendet. Stimmt die empfangene Nummer mit der gesendeten überein, dann ist die Registrierung erfolgreich. So wird sichergestellt, dass keine fremden Nummern registriert werden kann. Die Verknüpfung der Schlüssel mit Telefonnummern ermöglicht es, den Server als Schlüsselverzeichnis zu verwenden, von welchem unter Angabe der Telefonnummer der entsprechende öffentliche Schlüssel bezogen werden kann. Dies ist vor allem für neue Nutzer interessant, da diese dann nicht zuerst alle ihre Freunde treffen müssen, um die öffentlichen Schlüssel auszutauschen, sondern sofort mit ihnen über den Server kommunizieren können. Dies erleichtert den Einstieg und macht den Dienst vor allem in der Startphase benutzerfreundlicher, in welcher noch nicht so viele Nutzer vorhanden sind. Eine weitere Aufgabe des Servers ist es, die Position der mobilen Teilnehmer, falls diese übermittelt wird, aufzuzeichnen, und den Nutzer zu informieren, falls sich andere Mitglieder von veNETa in der Umgebung aufhalten. Aus Datenschutzgründen werden die Standortdaten nur eine kurze Zeit gespeichert. Da diese Daten sowieso nur eine kurze Zeit gültig sind, führt dies zu keiner funktionellen Einschränkung des Dienstes.

<sup>1</sup><http://www.emarketer.com/Article.aspx?id=1003934>

## Standortermittlung

Mit der Verfügbarkeit von Standortinformationen eröffnen sich weitere interessante Anwendungsmöglichkeiten. Zum Beispiel können Freunde, welche sich in der gleichen Gegend aufhalten, angezeigt werden. Orte können mit ortsspezifischen Informationen wie Bildern oder Hinweisen versehen werden oder die Position der Freunde kann auf einer Karte graphisch dargestellt werden. Eine weitere Möglichkeit wäre, gemeinsame Urlaubsorte zu finden oder die Lieblingsbar des Nutzers zu erkennen. Es gibt unzählige Anwendungsmöglichkeiten und es existieren bereits mobile social Networking-Dienste, welche auf Standortdaten angewiesen sind wie zum Beispiel Dodgeball, Plazes oder Mologogo, wobei die bei den ersten beiden der Standort vom Benutzer manuell eingegeben werden muss und nicht automatisch erkannt werden kann. Dies ist sicher keine optimale Lösung, aber vermutlich werden diese Dienste bald auch die automatische Positionsbestimmung erlauben. Denn auf dem Markt sind bereits einige Mobiltelefone verfügbar, welche über einen integrierten GPS-Empfänger verfügen. Und die Anzahl der Telefone mit GPS wird in den nächsten Jahren noch zunehmen. Die Marktforschungsgesellschaft ABI Research geht zum Beispiel davon aus, dass Ende 2008 bereits ein Viertel aller Telefone mit GPS ausgestattet sein werden. Für die Positionsbestimmung gibt es noch weitere Möglichkeiten wie zum Beispiel die Bestimmung mittels des Standortes der Zelle, in welchem das Mobilgerät eingebucht ist. Die Netzbetreiber in der Schweiz bieten derzeit diese Methode derzeit nicht an, aber in einigen anderen Ländern ist dies bereits möglich, zum Beispiel in Deutschland, wo die Firma Qiro<sup>2</sup> diese Dienstleistung in allen Netzen anbietet. veNETa verwendet für die Standortbestimmung das von der Java 2 Mobile Edition zur Verfügung gestellte Location API. Dieses ermöglicht die Bestimmung der Koordinaten des aktuellen Aufenthaltsortes unter Verwendung von GPS, netzbasierten Methoden oder eine Kombination davon. JSR 179 ist relativ neu und wird deshalb auch erst von neueren Mobiltelefonen wie dem N73 oder N95 von Nokia unterstützt. Auf Telefonen anderer Hersteller steht es momentan nicht zur Verfügung.

### 6.6.3 Implementierung

Die Verwendung des Location API ist mit einigen Unannehmlichkeiten verbunden. So ist es zum Beispiel nicht möglich, die gewünschte Ortungsmethode (GPS, netzbasiert, etc.) festzulegen. Dies macht das API automatisch. Dies ist vor allem dann problematisch, wenn eine automatisch ausgewählte Ortungsmethode momentan nicht zur Verfügung steht, und keine andere Ortungsmethode ausgewählt werden kann. Zudem blockiert das API auf unserem Testgerät für mehrere Minuten den Thread, falls auf eine Ortungsmethode zur Zeit gerade nicht verfügbar ist und versucht wird, Koordinaten zu erhalten.

---

<sup>2</sup><https://www.myqiro.de/web/>

# Kapitel 7

## Resultate – Evaluation

Dieses Kapitel präsentiert die Resultate unserer Arbeit und beschreibt die Ansätze und Verfahren, die wir verwendeten, um veNETa in einer praxisnahen Situation zu evaluieren.

### 7.1 Resultate

Im Rahmen unserer Semesterarbeit haben wir ein neuartiges mobile social network entworfen und implementiert, das durch geeignete architektonische Massnahmen Probleme früherer Ansätze vermeidet. veNETa erlaubt einen vollständig verteiltesn sowie einen hybriden Betriebsmodus, wobei Nachrichten zuerst an verbundene Geräte übertragen werden und der Server erst beim Ausbleiben einer Antwort kontaktiert wird. Eine Besonderheit von veNETa ist die Multi-Hop-Übertragung, die die Übertragung von Nachrichten über bis zu drei aufeinanderfolgende Verbindungen zulässt.

Zur Vereinfachung der Inbetriebnahme auf dem Mobiltelefon enthält die Software ausserdem eine menügeführte Einführung in das Programm, die dem Benutzer gleichzeitig erlaubt, veNETa nach seinen Wünschen vorzukonfigurieren. Diese Einführung stellt auch sicher, dass der Benutzer die Lizenz vor dem ersten Gebrauch des Programmes akzeptiert.

Ebenfalls abgeschlossen wurde die Implementation der Friend-Finder-Funktionalität, die auf Geräten in der Umgebung über ein kryptographisches Protokoll Telefonbuchvergleiche durchführt, um gemeinsame Kontakte von einander unbekanntem Passanten zu finden, sowie die Chatfunktionalität, die den Austausch von Textnachrichten erlaubt.

Um auch in einer Einführungsphase von veNETa für eine kleine Nutzergruppe attraktiv zu sein, wurde ausserdem eine Profilsuche realisiert, die mögliche Treffer anhand von Alter und Geschlecht findet und die Möglichkeit bietet, alle Profile in der Umgebung anzusehen. Aus dem gleichen Grund wurde auch die Möglichkeit zur Alarmierung beim Kontakt zu einem beliebigen anderen Nutzer vorgesehen.

veNETa ist bereits in seiner heutigen Form ein einfaches, aber in sich abgeschlossenes mobile social network. Der Client und die Serversoftware liefen im Dauereinsatz tagelang störungsfrei und stabil. Dank des modularen Aufbaus und der Lizenzierung unter der GPL eignet sich veNETa nicht nur zum alltäglichen Auffinden neuer Freunde und Gesprächspartner, sondern gleichermaßen als spezifische Plattform für weiterführende Experimente zu soziologischen oder netzwerktechnischen Themen.

### 7.2 Evaluation

Für mobile Anwendungen ist die Akkulaufzeit eine wichtige Grösse. Zur Evaluation von veNETa massen wir deshalb die durchschnittliche Akkulaufzeit für vier verbreitete Gerätetypen mit einer modifizierten Version von veNETa, die nach Geräten in der Umgebung suchte, aber keine Verbindungen aufbaut. Um die Laufzeiten vergleichen zu können, massen wir nicht nur im Betrieb, sondern auch die Stand-By-Laufzeiten der verwendeten Geräte. Die Resultate dieser Messungen finden sich in Tabelle 7.1.

Die Reduktion der Akkulaufzeit um rund zwei Drittel ist beträchtlich, wir gehen jedoch davon

aus, dass für den praktischen Einsatz knapp 24h Autonomie in den meisten Fällen akzeptabel sind. Mit neuwertigen Batterien dürfte die Laufzeit sogar eine Grössenordnung von drei Tagen erreichen, was einen Dauereinsatz ohne Verhaltensänderungen des Benutzers erlaubt. Im Falle des Nokia N95 besteht ausserdem der Verdacht, dass die Gerätesoftware fehlerhaft ist und mit einer neuen Version eine deutlich höhere Laufzeit erreicht werden könnte.

Gerätetyp (Netzbetreiber)	spezifiziert	stand-by	veNETa	Reduktion
Nokia N95 (Swisscom)	225h	55h	22h	60%
Nokia N73 (Sunrise)	350h	>250h	52h	>79%
Nokia 6680 (Sunrise)	“up to 144-264h”	62h	15h	66%
SonyEricsson W810i (orange)	350h	103h	32h	69%

Tabelle 7.1: Resultate der Evaluation der Akkulaufzeit

Ausserdem untersuchten wir die Zuverlässigkeit der Übertragung von Textnachrichten über mehrere Verbindungen. Dafür setzten wir ebenfalls eine modifizierte Version der Software ein, die die Beeinflussung der Netzwerktopologie vereinfacht und im Unterschied zur produktiven Version über beliebig viele Verbindungen weiterleiten konnte.

Eine erfolgreiche Übertragung wurde dabei definiert als das Übertragen der verschlüsselten Nachricht zum Empfänger sowie das Eintreffen der signierten Rückmeldung beim Sender innerhalb der dafür vorgesehenen Zeit von 30 Sekunden. Diese Versuche wurden mit denselben Geräten durchgeführt, die auch zur Bestimmung der Batterielaufzeit verwendet wurden. Dabei wurden nur linienförmige Anordnungen untersucht. Es ist denkbar, dass eine geflechtartige Struktur des Netzwerkes sich günstig auf die Übertragungszuverlässigkeit auswirkt und diese Zahlen deshalb im praktischen Betrieb übertroffen werden können. Die Resultate dieser Evaluation finden sich in Tabelle 7.2.

Versuchsaufbau	übertragen	verloren	Zuverlässigkeit
direkte Übertragung	50	0	100%
Übertragung via 2 Hops	47	3	94%
Übertragung via 3 Hops	42	11	82%
Übertragung via 4 Hops <sup>a</sup>	27	23	54%

<sup>a</sup>im produktiven Betrieb nicht unterstützt. Die abfallende Zuverlässigkeit ist hauptsächlich auf Retransmit-Timeouts zurückzuführen, die überschritten werden, wenn eines der Geräte nach anderen Geräten sucht und deshalb vorübergehend nicht erreichbar ist. Die Annahme ist realistisch, dass von den 46% der Nachrichten, die nicht zugestellt werden konnten, rund die Hälfte ihr Ziel erreicht hat, aber die Rückmeldung verloren ging.

Tabelle 7.2: Resultate der Evaluation der Zuverlässigkeit von Multihop-Übertragungen

Zusammenfassend ist festzuhalten, dass die Validierung eines verteilten Systems eine schwierige Aufgabe ist. Bedingt durch die hohe Komplexität der Interaktion fällt es schwer, dessen Funktionsfähigkeit in jeder Konfiguration zu garantieren. In diesem Fall kommt noch erschwerend dazu, dass grosse Teile der Entwicklungsarbeit auf spezifischen Geräten vorgenommen wurden und die Zuverlässigkeit der Resultate stark vom übereinstimmenden Verhalten der Entwicklungsgeräte mit den Zielgeräten abhängt.

Da wir allerdings aufgrund der modularen Entwicklungsmethodik einen grossen Teil der Validierung während der Entwicklungsarbeit erbracht haben, für kryptographische Funktionen eine populäre Open-Source-Library verwendeten und eine standardisierte Virtuelle Maschine programmierten, gehen wir davon aus, dass veNETa zu einer breiten Hardwarebasis kompatibel ist und die Zuverlässigkeits- und Sicherheitsansprüche an im Rahmen einer Semesterarbeit geschriebene Software weitestgehend erfüllt.

## Kapitel 8

# Ansätze zur Weiterentwicklung und Problembehebung

Obschon in das veNETa-Projekt viele neuartige Ansätze und viele Stunden konzentrierter Arbeit eingeflossen sind, bleiben viele Konzepte, die weiter verbessert oder erweitert werden könnten. In diesem Kapitel geben wir deshalb einige Vorschläge zur Weiterentwicklung, die uns besonders interessant erscheinen.

Sehr attraktiv erscheint beispielsweise die Nutzung der Koordinaten auf zum JSR 179 kompatiblen Geräten, um eine Karte der Umgebung des Aufenthaltsortes anzuzeigen. Diese einfache Erweiterung trägt nicht an sich zum social network bei, kann im Alltag aber grossen Nutzen bringen und könnte den Ausschlag dafür geben, veNETa permanent auf dem Mobiltelefon auszuführen. Eine andere einfach zu realisierende Erweiterung stellt die Aufzeichnung der Route und die Ausgabe in einem Koordinatensystem dar, wobei diese Darstellung durch eine Satellitenkarte oder die Aufenthaltsorte anderer Nutzer aufgewertet werden kann.

Von zentraler Bedeutung für die Freundessuche und den Datenaustausch zwischen den Geräten ist die Bluetooth-Technologie, deren Möglichkeiten veNETa bereits weitgehend ausschöpft. Obwohl heutzutage keine Alternative über eine ähnlich hohe Verbreitung auf Mobiltelefonen verfügt, könnten insbesondere Wireless LAN-Sender auf kompatiblen Geräten unterstützt werden, um auf diesen Geräten von der grösseren Reichweite und der schnelleren Suche nach Geräten in der Umgebung profitieren zu können.

Ebenfalls verbessert werden kann der Routing-Algorithmus, der bis jetzt auf einfachem Flooding basiert. Um die Zahl der gefloodeten Nachrichten niedrig zu halten, wurde die Time-To-Live der Nachrichten künstlich auf drei Hops beschränkt. Mit einem echten Routing, das beispielsweise auch die Lage verschiedener Geräte zueinander einbezieht, liesse sich diese Zahl und damit die Reichweite des einzelnen Gerätes deutlich erhöhen.

Ein ganzes Forschungsgebiet erschliesst sich in der Auswertung der Bewegungsdaten, die bei der Lokalisierung anfallen. In diesem Rahmen könnten auf dem Server beispielsweise Ähnlichkeitsprofile anhand der meistbesuchten Plätze oder des bevorzugten Fortbewegungsmittels erstellt werden. Falls die Nutzer einwilligen, könnte diese Information auch mit den Telefonbuchdaten kombiniert und auf dem Server gemeinsam verarbeitet werden.

Um auch Geräte, die keinen GPS-Empfänger enthalten, lokalisierbar zu machen, könnten ausserdem Bluetooth-Sender an häufig frequentierten Plätzen montiert werden, die spezielle Nachrichten mit den Koordinaten des Aufenthaltsortes versenden. Und falls auch in Zukunft nicht alle Geräte über einen GPS-Empfänger verfügen sollten, wäre es ein leichtes, die ungefähre Position eines GPS-fähigen Gerätes via Bluetooth auch an seine Umgebung weiterzugeben.

Falls veNETa jemals im grösseren Massstab eingesetzt werden soll, erscheint die Integration eines Web-Interfaces naheliegend, das beispielsweise auf dem Server gespeicherte Nachrichten anzeigen oder Aufenthaltsorte suchen kann.

# Anhang A

## Details der Implementation

### A.1 BTEngine

Das BTEngine-Package erfüllt im veNETa-Client die zentrale Aufgaben der Suche nach neuen Geräten, des Routings und dem Verbindungsmanagement. Um Veränderungen oder Erweiterungen einfacher zu gestalten, geben die folgenden Abschnitte einen Einblick in den inneren Aufbau dieses Packages.

#### A.1.1 BTEngine.java

Die Hauptklasse des BTEngine-Packages ist `BTEngine`. Diese Klasse wird von der Main-Klasse instanziiert und initialisiert (`init()`), daraufhin läuft sie selbstständig und gibt empfangene Nachrichten mittels Callbacks weiter an die Main-Klasse. Ausserdem stellt sie eine Schnittstelle zum vorübergehenden Unterdrücken von Suchvorgängen zur Verfügung, damit Friend-Finder-Verbindungen nicht unnötig unterbrochen werden. Die wichtigsten Funktionen und Objekte von `BTEngine` sind:

- `void init()` initialisiert die BTEngine und bereitet die JSR-179-Klassen vor.
- `void start(boolean fullMode)` Nach der Initialisierung nimmt die BTEngine mit diesem Aufruf ihre eigentliche Tätigkeit auf. Gemäss der Abbildung 6.2 werden periodisch neue Geräte gesucht, kontaktiert und die Verbindung hergestellt. Ausserdem wird der Routing-Mechanismus aktiviert.  
Der Parameter `fullMode` bezeichnet, ob die Applikation alle auf dem verfügbaren Verbindungen verwenden soll, oder nur eine. Dieser Parameter wurde hinzugefügt, als sich herausstellte, dass einige Geräte weniger Verbindungen unterstützen, als die entsprechenden JSR-179-Funktionen angeben. Im Welcome-Screen-Modul und den Einstellungen sind die korrespondierenden Bezeichnungen "Vollmodus" bzw. "limitierter Modus".
- `void send(BTEMessage btem)` versendet die Nachricht `btem` an die im Objekt gespeicherte Adresse. Für Nachrichten, die an alle Geräte in der Umgebung gesendet werden sollen, wird die Methode `void floodsend(BTEMessage btem)` verwendet oder die Empfängeradresse der Nachricht vor dem Senden auf `BROADCAST_BT_ADDRESS` gesetzt
- `void stop()` beendet den von `start()` gestarteten Ablauf und schliesst alle Verbindungen. Diese Funktion wird nicht nur beim Beenden von veNETa aufgerufen, sondern muss auch bei einer Änderung der `fullmode`-Variable aufgerufen werden.
- `int MIN_INTERVAL` bezeichnet die kürzeste Zeit in ms, die die BTEngine nach der Ausführung eines Suchlaufs wartet. Die Standardeinstellung von 60s hat sich dafür bewährt, da sie einen guten Kompromiss zwischen Erreichbarkeit des Geräts und raschem Finden anderer Geräte in der Umgebung darstellt.
- `int MAX_INTERVAL` bezeichnet die längste Zeit in ms, die die BTEngine nach der Ausführung eines Suchlaufs wartet. Die Standardeinstellung dieses Wertes ist 180s.

- `int MAX_MSG_DELIVERY_TIME` bezeichnet die Zeit in ms, nach welcher eine Bluetooth-Nachricht als verloren angesehen wird, sofern keine Rückmeldung des Empfängers eingetroffen ist. Der Standardwert beträgt 30s und sollte für Experimente am Routingalgorithmus erhöht werden. Für den produktiven Betrieb mit bis zu drei Hops reicht er aus (siehe auch Abschnitt 7.2).
- `RemoteDeviceVector rdv` ist ein von der `Vector`-Klasse abgeleitetes Objekt, das alle zur Zeit direkt und über benachbarte Geräte erreichbare Nodes enthält. `rdv` wird für das Profilbrowsing verwendet.

### A.1.2 BTEDeviceDisc.java

Diese Klasse implementiert das JSR 82-Interface `DiscoveryListener` und speichert alle `RemoteDevice`-Objekte, die der Suchlauf retourniert, ins `remoteDevices`-Object der BTEngine-Instanz.

### A.1.3 BTEServiceDisc.java

Diese Klasse implementiert ebenfalls das Interface `DiscoveryListener`, dient aber der Suche nach dem veNETa-Bluetooth-Dienst auf Geräten, die von `BTEDeviceDisc` gefunden wurden. Im Prinzip wäre die Implementation derselben Funktionalität in einer einzelnen Klasse möglich, in der Praxis traten dabei aber erratische Fehler, unerklärliche Deadlocks und Abstürze der VM oder des gesamten Mobiltelefons auf. Es ist ein wirklich gutgemeinter Vorschlag, diese beiden Klassen nicht ohne Not zu verändern.

### A.1.4 BTEConnection.java

Diese Klasse implementiert ein abstraktes Verbindungsobjekt. `BTEConnections` werden von der BTEngine verwaltet und haben keine Parameter, die ausserhalb gesetzt werden sollten.

### A.1.5 BTEMessage.java

Das `BTEMessage`-Objekt speichert eine komplette Nachricht mitsamt Empfänger, Sender und den passenden Nachrichtentypen. Diese Nachrichtentypen bestimmen das weitere Verfahren mit dem Nachrichteninhalt, wenn eine Nachricht empfangen wird. Die wichtigsten Felder einer `BTEMessage` umfassen:

- `byte msgtype` nimmt den Major-Nachrichtentyp auf. In der heutigen Form von veNETa werden hauptsächlich `SYSTEM_OPS`-, `MSG_FF`<sup>1</sup>- und `MSG_MESSAGE`-Nachrichten verwendet. Diese werden durch den Einsatz von Minor-Nachrichtentypen weiter unterschieden, beispielsweise in den eigentlichen Text einer Nachricht und deren Bestätigung.
- `byte msgsubtype` nimmt den Minor-Nachrichtentyp auf. Der gegenwärtige Aufbau von `BTEMessages` lässt also 65536 unterschiedliche Typen von Nachrichten zu.
- `byte[] content` nimmt den eigentlichen Payload der Nachricht auf.
- `int ttl` enthält die verbleibende Anzahl Hops, die die Nachricht noch nehmen darf. Diese Variable wird bei der Instanzierung gesetzt, und beim Weiterleiten durch die BTEngine automatisch dekrementiert.
- `byte MESSAGE_MAX_TTL` bezeichnet die Anzahl Hops, die eine Nachricht nehmen kann, bevor sie nur noch direkt weitergeleitet wird. Dieser Wert ist standardmässig auf zwei gesetzt, um so in zwei Flooding-Schritten und einem Routing-Schritt drei Verbindungen überbrücken zu können.
- `String senderBTAddr` nimmt die Adresse des ursprünglichen Senders auf. Wird sie nicht gesetzt, so schreibt die `send`-Methode der BTEngine diese automatisch.

---

<sup>1</sup>für Friend-Finder-Nachrichten

- `String receiverBTAddr` enthält die Adresse des Empfängers der Nachricht, oder `BROADCAST_BT_ADDR` für eine Verteilung an alle Clients in Reichweite.
- `String receivedFromBTAddr` enthält die Adresse des Gerätes, das die Nachricht versendet hat. Bei direkten Nachrichten stimmt diese mit `senderBTAddr` überein.

## A.2 Server

Der folgende Abschnitt gibt einen Einblick in die Struktur des Servers.

### A.2.1 Server.java

Die Hauptklasse des Servers ist `Server.java`. Diese öffnet ein `ServerSocket`, um eingehende Verbindungen empfangen zu können, und ist auch für das Sichern der noch nicht gesendeten Nachrichten und der registrierten Schlüssel zuständig. Sie erstellt zudem einen Thread, welcher neue Befehle einliest und, falls gewünscht, auch den Server beenden kann. In `Server.java` sind folgende wichtige Objekte gespeichert:

- `RSAPrivateCrtKeyParameters rsaPrivateKey` Der private Schlüssel des Servers, welcher für das signieren von Paketen verwendet wird.
- `RSAKeyParameters rsaPublicKey` Der öffentliche Schlüssel des Servers.
- `Verbindungsvector verbindungen` Ein Vector, welcher alle aktiven Verbindungen, die jeweils in einem Objekt vom Typ `Verbindung` gespeichert sind, enthält.
- `Keyvector keyvector` Hier sind alle Schlüssel von registrierten (aber nicht notwendigerweise mit dem Server verbundenen) Nutzern gespeichert.
- `Keyvector revokedKeys` Enthält alle revozierten Schlüssel.

### A.2.2 Verbindung.java

Diese Klasse wird für jede neue Verbindung erzeugt und ist zuständig für den Verbindungsaufbau, das Bereitstellen der Streams und für das Senden und Empfangen von Nachrichten. Die verschiedenen Verbindungen werden anhand des Hashs des öffentlichen Schlüssels des verbundenen Nutzers identifiziert und im `Verbindungsvector verbindungen` gespeichert. Ebenfalls in dieser Klasse wird der aktuelle Standort des Nutzers gespeichert - sofern er bekannt ist - und der Zeitpunkt, zu welchem die Position empfangen wurde.

### A.2.3 Verbindungsvector.java

Diese Klasse hat folgende Aufgaben:

- *Registrierung neuer Schlüssel* Um einen Schlüssel verwenden zu können, muss er zuerst auf eine Telefonnummer registriert werden. Um die Nummer zu überprüfen, wird eine Nummer per Kurznachrichte an den Anwender gesendet, welcher diese dann in `veNETa` eingeben muss.
- *Anmeldung bereits registrierter Schlüssel* Wenn sich ein Nutzer mit einem öffentlichen Schlüssel beim Server anmeldet, wird überprüft, ob er wirklich im Besitz des entsprechenden privaten Schlüssels ist.
- *Die Revozierung von ungültigen oder kompromittierten Schlüsseln*
- *Empfangen von Nachrichten* Die einzelnen Verbindung-Objekte werden nacheinander abgefragt, ob neue Daten empfangen wurden. Falls eine Verbindung geschlossen wurde, wird das Objekt gelöscht. Wenn eine Nachricht empfangen wurde, dann wird diese verarbeitet und falls notwendig, an die dafür zuständige Klasse weitergeleitet. Wird eine Nachricht empfangen, welche für einen Anwender bestimmt ist, der im Moment nicht mit dem Server verbunden ist, dann wird sie im Nachrichtenvektor `zuSenden` gespeichert.

- *Senden von Nachrichten* Falls eine Nachricht gesendet werden muss, dann wird diese an das entsprechende Verbindung-Objekt weitergeleitet, sofern die Verbindung besteht. Ansonsten wird sie beim nächsten Verbindungsaufbau gesendet.
- *Empfangen der aktuellen Position von Teilnehmern* Diese wird im entsprechenden Verbindung-Objekt gespeichert und nach einer bestimmten Zeit wieder gelöscht, um sicherzustellen, dass keine veralteten Informationen ausgeliefert werden. Sollten sich zwei Teilnehmer in der gleichen Umgebung aufhalten, dann werden diese darüber informiert.

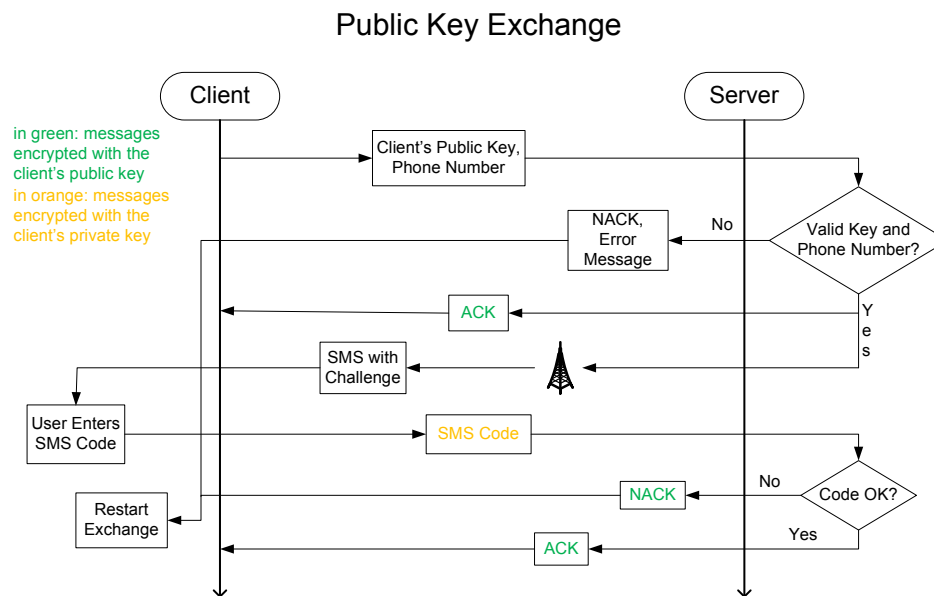


Abbildung A.1: Der modulare Aufbau von veNETa. Die kleinen Sektoren stehen für die Location- und Krypto-Module.

Abbildung A.1 zeigt den Ablauf der Registrierung eines neuen Schlüssels und die Überprüfung der angegebenen Telefonnummer mittels einer Kurznachricht.

#### A.2.4 SMSVerbindung.java

Diese Klasse ist für das Senden der Kurznachrichten, welche für die Prüfung der Telefonnummer benötigt werden, zuständig. Um eine Nachricht zu senden erstellt sie eine SMTP-Verbindung zum Server mail.ee.ethz.ch (dieser kann nur innerhalb des ETH-Netzes ohne Benutzerkonto verwendet werden) und sendet eine E-Mail mit der Prüfnummer an [Telefonnummer]@sms.switch.ch.

#### A.2.5 Nachrichtenvector.java

Diese Klasse ist von Vector.java abgeleitet und kann die auf dem Vector vorhandenen Nachrichten in eine Datei speichern oder gespeicherte Nachrichten einlesen.

#### A.2.6 Nachricht.java

Diese Klasse definiert die verschiedenen, verwendeten Nachrichtenformate. Jedes dieser Formate wird durch einen Integer definiert. Die Pakete werden durch die Methode `toByte()` erzeugt, welche das Paket als Byte-Array zurück gibt. Um ein empfangenes Paket zu verarbeiten

wird `empfangen(byte[] b, int laenge)` aufgerufen, wobei `b` der Inhalt des Paketes ist und der Integer `laenge` die Grösse des Arrays in Bytes darstellt. Die empfangenen Daten werden dann in den Objektvariablen gespeichert.

### Nachrichtenformat

Die Nachrichten bestehen aus einem Byte-Array, wobei die ersten vier Bytes als Integer die Länge des Arrays darstellen. Danach kommt ebenfalls als Integer dargestellt der Nachrichtentyp. Für die Adressierung der Nachricht wird der Hashwert des öffentlichen Schlüssels des Empfängers verwendet, welcher als nächstes im Byte-Array folgt. Der Absender wird durch den Hashwert seines öffentlichen Schlüssels identifiziert. Absender und Sender werden bei Paketen, welche an den Server gesendet werden, nicht mitgeteilt, da diese bereits durch die bestehende Verbindung bekannt sind.

Überblick über die Nachrichtenformate, die in `Nachricht.java` definiert sind.

- `textnachricht` Dies stellt eine normale, unverschlüsselte Textnachricht dar.
- `zugestellt` Dies ist die Bestätigung, dass die entsprechende Nachricht erfolgreich zugestellt wurde. Die Nachricht wird dabei anhand des Hashwertes ihres Inhalts identifiziert.
- `verschluesselformat` Der Inhalt dieser Textnachricht ist verschlüsselt.
- `gesendeteNachricht` Diese Nachricht wurde bereits gesendet und bleibt als Kopie gespeichert, damit der Anwender einen Überblick über die bereits gesendeten Nachrichten hat und damit Empfangsbestätigungen einer Nachricht zugewiesen werden können.
- `locationUpdate` Diese Nachricht enthält eine aktuelle Position des Mobiltelefons.
- `locationNotification` Wird vom Server gesendet, wenn sich mehrere Nutzer in der Umgebung aufhalten.
- `keyAnmelden` Wird vom Mobiltelefon gesendet, um sich beim Server mit dem angegebenen öffentlichen Schlüssel anzumelden.
- `anmeldungErfolgreich` Der Server hat den Schlüssel erfolgreich überprüft und sichergestellt, dass das Mobiltelefon über den entsprechenden privaten Schlüssel verfügt.
- `falscherHash` Fehlermeldung, welche vom Server gesendet wird, wenn das Mobiltelefon einen Schlüssel verwendet, welcher nicht zur angegebenen Nummer registriert ist.
- `challengeGesendet` Der Server sendet diese Nachricht, wenn ein neuer Schlüssel registriert werden soll und er eine Kurznachricht an die angegebene Mobiltelefonnummer gesendet hat, um diese zu überprüfen.
- `challenge` Die Antwort des Mobiltelefons, welche die Prüfnummer, welcher der Server per Kurznachricht gesendet hat, enthält.
- `challengeFehlerhaft` Die angegebene Prüfnummer entspricht nicht der per Kurznachricht gesendeten.
- `nummerFehlerhaft` Wird vom Server gesendet, falls die angegebene Mobiltelefonnummer fehlerhaft ist.
- `hashBereitsVerwendet` Diese Fehlermeldung wird vom Server versendet, wenn ein Mobiltelefon einen Schlüssel registrieren möchte, welcher bereits von einem anderen Nutzer verwendet wird.
- `hashFehlerhaft` Der angegebene Hashwert des öffentlichen Schlüssels passt nicht zum übermittelten öffentlichen Schlüssel.
- `keyRegistrieren` Diese Nachricht ist eine Aufforderung vom Mobiltelefon, um einen neue generierten Schlüssel zu registrieren.
- `keyRegistriert` Meldung vom Server, falls die Registrierung erfolgreich beendet wurde.

- `schluesselUnbekannt` Wird vom Server versendet, um mitzuteilen, dass die vom Mobiltelefon gesendete Nachricht an einen unbekanntem Empfänger adressiert wurde.
- `authentifikation` Diese Nachricht wird vom Server verwendet, um zu überprüfen, ob das Mobiltelefon den zum öffentlichen Schlüssel passenden privaten Schlüssel besitzt. Er sendet darin ein Byte-Array, welches mit dem entsprechenden Schlüssel verschlüsselt werden soll.
- `antwortAuthentifikation` Dies ist die Antwort des Mobiltelefons mit dem verschlüsselten Byte-Array.
- `authentifikationFehlgeschlagen` Die Antwort des Mobiltelefons ist nicht korrekt. Der Server kann deshalb den Schlüssel nicht anmelden und sendet diese Nachricht.
- `schluesselRevozieren` Diese Aufforderung wird vom Mobiltelefon gesendet, um den aktuellen Schlüssel zu revozieren.
- `schluesselRevozierenBestaetigen` Der Server verlangt stellt mit dieser Nachricht sicher, dass das Mobiltelefon auch über den zu revozierenden Schlüssel verfügt.
- `schluesselErfolgreichRevoziert` Der Server teilt mit dieser Nachricht mit, dass der Schlüssel erfolgreich revoziert wurde.
- `schluesselRevozierenFehlgeschlagen` Diese Fehlermeldung wird versendet, wenn der Schlüssel nicht revoziert werden konnte.
- `schluesselWurdeRevoziert` Wenn ein Mobiltelefon eine Nachricht an einen Empfänger senden will, der seinen Schlüssel revozieren liess, dann versendet der Server diese Fehlermeldung. Das Mobiltelefon kann dann den neuen Schlüssel anfordern, sofern einer registriert wurde.
- `keyAnfordern` Mit dieser Nachricht kann ein Mobiltelefon den zu einer angegebenen Telefonnummer passenden öffentlichen Schlüssel verlangen.
- `schluesselNichtVorhanden` Der Server hat unter der angegebenen Telefonnummer keinen Schlüssel registriert. Dieser Nachrichtentyp darf nicht mit `schluesselUnbekannt` verwechselt werden, welcher gesendet wird, falls eine Nachricht an einen unbekanntem Empfänger gesendet wurde.
- `angeforderterKey` Die Antwort des Servers mit dem Angeforderten Schlüssel.
- `neuerKey` Der verwendete Key wurde revoziert. Da aber bereits ein neuer registriert wurde, sendet der Server diesen dem Mobiltelefon.

# Anhang B

## Aufgabenstellung

### B.1 Student Thesis “Mobile Social Networking”

This document describes the subject and the general time schedule of the student thesis of Matthias Bader and Marco Von Arb in the summer term 2007. Adaptations or changes can be agreed upon by the adviser.

### B.2 Subject

Online social networking has become extremely popular over the past years. This popularity has motivated the introduction of similar services in the mobile world. Even though several systems, such as MobiLuck, Jambo, Sixth Sense, Nokia Sensor, etc. have been proposed, these systems have not found widespread acceptance. We believe that there are different reasons for the missing popularity of these systems:

- Many of them are server based and require permanent access to the internet, which is still expensive.
- In non-server-based environments the applications lack a sufficient amount of initial users to become interesting (this is particularly true for friend-finding applications)
- Often these systems are mainly copies of their internet pendants and do not take advantage of the special characteristics of a mobile environment.

The goal of this thesis is to implement a social service for mobile phones that overcomes above mentioned problems. In a first step, the exact scope of the service has to be defined. A smart design around the contact comparison idea is one possibility. However, the students can also come up with their own ideas. The service should be designed such, that it is already attractive when only a small numbers of users participate - this seems to be one of the major requirements for success.

### B.3 Time schedule in hours (Total: 2\*275h = 550h)

- Studying related work [50h]
- Defining the scope of the application and its subtasks [50h]
- Implementation of a first prototype [200h]
- Refinement of the prototype and implementation of minor features [200h]
- Writing of the final report [50h]

## B.4 The Student's Duties

- One meeting per week with the adviser (Michael).
- A final presentation of the work (15 minutes).
- Report (approximately 20 pages, English), presenting the work and results. This report should also include a critical review of the work.

## B.5 General

- Independent working is expected
- A possibility to work in the ETZ building is provided. It is also possible to work at home.

## B.6 Contacts/Advisers

1. Kuhn Michael: kuhnmi@tik.ee.ethz.ch, ETZ G61.4, phone 044 632 77 30
2. Roger Wattenhofer: wattenhofer@tik.ee.ethz.ch, ETZ G63, phone 044 632 63 12

# Literaturverzeichnis

- [1] Sun Inc., *J2ME Technology API Documentation*,  
<http://java.sun.com/j2me/docs/intex.html>.
- [2] James Gosling et al, *The Java Language Specification*,  
<http://citeseer.ist.psu.edu/gosling96java.html>, 1996.
- [3] Sun Inc., *JSR 179: Location API for J2ME*,  
<http://jcp.org/en/jsr/detail?id=179>.
- [4] Sun Inc., *JSR 139: Connected Limited Device Configuration 1.1*,  
<http://jcp.org/en/jsr/detail?id=139>.
- [5] Sun Inc., *JSR 82: Java APIs for Bluetooth*,  
<http://jcp.org/en/jsr/detail?id=82>.
- [6] Sun Inc., *JSR 185: Java Technology for the Wireless Industry*,  
<http://jcp.org/en/jsr/detail?id=185>.
- [7] Symbian, Inc., *S60 3rd Edition FP1 C++ API Reference Guide*,  
[http://www.forum.nokia.com/document/Cpp\\_Developers\\_Library/](http://www.forum.nokia.com/document/Cpp_Developers_Library/).
- [8] A. Shamir, R. Rivest, L. Adleman, *Mental Poker*, 1979  
Massachusetts Institute of Technology.
- [9] A. Shamir, R. Rivest, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*,  
Massachusetts Institute of Technology, 1978.
- [10] The Legion of the Bouncy Castle, *The Bouncy Castle Crypto APIs for Java*,  
<http://www.bouncycastle.org/java.html>.
- [11] GNU, *GNU General Public License*,  
<http://www.gnu.org>, 2005.