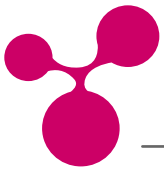
Technische Universität Dresden – Fakultät Informatik
Professur für Multimediatechnik, Privat-Dozentur für Angewandte Informatik

Prof. Dr.-Ing. Klaus Meißner
PD Dr.-Ing. habil. Martin Engelien
(Hrsg.)



GeNeMe '09

Gemeinschaften in Neuen Medien

an der
Fakultät Informatik der Technischen Universität Dresden

mit Unterstützung der

3m5. Media GmbH, Dresden
GI-Regionalgruppe, Dresden
Communardo Software GmbH, Dresden
Kontext E GmbH, Dresden
Medienzentrum der TU Dresden
nubix Software-Design GmbH, Dresden
objectFab GmbH, Dresden
SALT Solutions GmbH, Dresden
Saxonia Systems AG, Dresden
T-Systems Multimedia Solutions GmbH

am 01. und 02. Oktober 2009 in Dresden

http://www-mmt.inf.tu-dresden.de/geneme/
geneme@mail-mmt.inf.tu-dresden.de

C.7 MobilisBuddy - Integration sozialer Netzwerke in distanzabhängige Dienste auf mobilen Endgeräten

Daniel Schuster, Thomas Springer, Benjamin Söllner, Dirk Hering, Alexander Schill
Technische Universität Dresden, Fakultät Informatik, Institut für Systemarchitektur, Professur Rechnernetze

1 Einleitung

Soziale Netzwerke wie Facebook oder XING erfreuen sich einer wachsenden Beliebtheit. Sie ermöglichen es Nutzern, in Kontakt zu bleiben, persönliche Informationen in Form von Profilen zu veröffentlichen und auf diese Weise neue Kontakte zu knüpfen. Diese Basisfunktionen werden vielfach durch Möglichkeiten der Kollaboration von Nutzern ergänzt. Das Knüpfen von Kontakten ist also kein Selbstzweck, sondern dient dem Initiieren vielfältiger Interaktionen zwischen Nutzern. Die einzelnen Web-Plattformen bieten aber neben den Grundfunktionen nur wenige Möglichkeiten zur direkten Interaktion wie beispielsweise das Chatten zwischen Teilnehmern. Eine plattformübergreifende Verknüpfung von Kontakten ist bisher nicht möglich. Damit kann das Potential sozialer Netzwerke nur begrenzt benutzt werden.

Im vorliegenden Artikel wird ein Ansatz zur Integration bestehender sozialer Netzwerke in eine generische Plattform zur Kollaboration in mobilen Umgebungen vorgestellt. Ziel ist es, das Potential der sozialen Vernetzung durch das Anbieten kollaborativer Dienste verstärkt auszunutzen. In Verbindung mit Kontextinformationen, insbesondere dem Aufenthaltsort und der Entfernung zu Kontakten können auf diese Weise Szenarien zur spontanen mobilen Kollaboration gezielt unterstützt und damit ein (direkter) Mehrwert aus dem Aufbau sozialer Netzwerke gezogen werden. Wie diese Integration technisch realisiert werden kann, zeigen wir im Folgenden am Beispiel der Buddyfinder-Anwendung MobilisBuddy, die auf der Basis des Kollaborationsframeworks Mobilis [SS+08] realisiert wurde und mit Facebook interagiert.

2 Distanzabhängige Dienste

Distanzabhängige Dienste (engl.: proximity-based services) stellen eine Teilmenge von ortsabhängigen Diensten dar [SNE06, WMY08]. Ortsabhängige Informations- und Unterhaltungsdienste für mobile Endgeräte machen sich Positionsinformationen dieser Geräte (GPS, zellenbasiert, WLAN-basiert) zu Nutze, um bestehende Dienste an den Kontext des Nutzers anzupassen bzw. ganz neue Dienste zu ermöglichen.

Bei distanzabhängigen Diensten werden dem Nutzer Informationen über seine direkte Umgebung (meist in einem Radius von ca. 100 Metern) angeboten. Mögliche

Anwendungen sind Werbung für Dienstleistungen oder Produkte in der Umgebung des Nutzers, Matching von persönlichen Profilen (z.B. für die Anbahnung von Kontakten auf Messen), automatisches Check-In an Flughäfen oder die Anzeige von bekannten Personen in der Nähe auf einer Karte. Letztere Anwendungsklasse bezeichnen wir im Folgenden als **Buddyfinder-Anwendungen.**

Diese Anwendungen sind dadurch gekennzeichnet, dass sich die Umgebung des einzelnen Nutzers ständig ändert, auch wenn dieser selbst seine Position nicht ändert. Dem Nutzer wird dann ein sogenanntes **Proximity Event** zugestellt, wenn sich ein zweiter Nutzer in dessen Nähe bewegt, zu dem er in einer Beziehung steht.

Bisher existierende Buddyfinder-Anwendungen nutzen dafür entweder eine serverbasierte Architektur, indem periodisch Positionsupdates an einen zentralen Server geschickt werden, oder sie bauen Bluetooth-Verbindungen zwischen den Nutzern auf und registrieren so Nutzer in der näheren Umgebung. Aka-aki [Aka09] ist ein Beispiel für einen Bluetooth-basierten Dienst, bei dem der Nutzer ein Proximity Event angezeigt bekommt, sobald sich ein anderer Nutzer des Dienstes in der Nähe befindet. Ein Beispiel für einen serverbasierten Dienst ist Google Latitude [Goo09]. Hier werden zellenbasierte Ortsinformationen an Google-Server übermittelt, so dass ein Nutzer andere registrierte Nutzer auf einer Karte sehen kann.

Weitere dem hier vorgestellten System ähnliche Arbeiten sind das BuddyMob-System [BM09], das ebenfalls als Android-Anwendung konzipiert ist und das Extensible Messaging and Presence Protocol (XMPP) als Basis-Protokoll benutzt, sowie das BuddyCloud-System [BC09], das die automatische Übertragung von symbolischen Positionen wie „Büro“ oder „Café“ unterstützt und zu diesem Zweck die eigens entwickelte XMPP-Erweiterung XEP-0255 [Xep0255] verwendet. Daneben gibt es noch viele weitere Systeme und Forschungsprototypen, die umgebungsbasierte Suche ermöglichen (siehe z.B. [MT+06]).

Das größte Problem dieser Systeme ist jedoch ihre Isoliertheit, da jeweils neue mobile soziale Netze entstehen, die von den Nutzern gepflegt werden müssen. Ein soziales Netz ist erst dann für den Nutzer wertvoll, wenn auch die meisten Personen, mit denen der Nutzer im realen Leben verbunden ist, dieses Netz benutzen. Im Gegensatz zu den existierenden Systemen baut unsere Lösung kein separates soziales Netzwerk auf, sondern integriert erstmals existierende soziale Netze wie Facebook, um die Buddyfinder-Funktionalität zu realisieren. Das hier vorgestellte System MobilisBuddy ist offen für die Unterstützung beliebiger Social Networks und kann damit eine breite Nutzergemeinde erreichen.

3 MobilisBuddy

Ähnlich wie BuddyMob und BuddyCloud ist MobilisBuddy als Client-Server-System konzipiert, welches im Folgenden detailliert betrachtet wird. Zunächst wird ein konzeptioneller Überblick für das Mobilis-System mit den ergänzten MobilisBuddy-Komponenten gegeben.

3.1 Architektur

Mobilis [SS+08] ist ein Framework für mobile Kollaboration, das nach dem Paradigma der Service Oriented Architecture (SOA) eine flexible Umgebung mit kollaborativen Diensten bietet, die die Entwicklung von kollaborativen Anwendungen erleichtern sollen. Als **Clients** können verschiedenste Geräte fungieren, zum Beispiel Smartphones, PDAs oder Notebooks. Wichtige Voraussetzungen für die Nutzung der Mobilis-Dienste sind der Internetzugang und eine installierte Mobilis-Client-Anwendung. Zusätzlich dazu ist ein GPS-Empfänger von Vorteil. Für MobilisBuddy ist außerdem ein Account bei einem unterstützten sozialen Netzwerk nötig.

Der Client kommuniziert über das offene XML-Protokoll **XMPP** (Extensible Messaging and Presence Protocol) mit dem Server, welches besonders geeignet für den bidirektionalen Austausch von Daten in Echtzeit ist [XSF09]. Dies geschieht auf Grundlage von XML-Streams. Zu den vom Protokoll übertragenen XML-Informationseinheiten (Stanzas) zählen Message-, Presence- und Info/Query-Pakete. Letztere dienen zur Realisierung von Request-Response-Diensten. Ihnen kommt bei MobilisBuddy eine besondere Bedeutung zu (siehe Abschnitt 3.2).

Die **Server-Seite** gliedert sich in zwei eigenständige Komponenten: den XMPP-Server und die Mobilis-Server-Anwendung. Der XMPP-Server bildet das Rückgrat für die Kommunikation zwischen Mobilis-Clients und dem Mobilis-Server. Clients und mehrere zentrale Komponenten des Mobilis-Servers sind unter eindeutigen Nutzernamen, den sogenannten Jabber-IDs (JIDs), beim XMPP-Server angemeldet. Dieser realisiert eigenständig die Zustellung von Nachrichten.

Wie in Abbildung 1 zu sehen, ist für den Mobilis-Server als zentrale Instanz ein Coordinator vorgesehen, welcher gegenüber dem XMPP-Server als XMPP-Client auftritt. Der Coordinator gestattet dem Client das Auffinden registrierter **Broker Services**. Ein Broker Service nimmt XMPP-Nachrichten für eine einzelne Anwendung der Mobilis-Plattform, wie z.B. für MobilisBuddy oder MobilisGuide (ein kollaborativer Touristenführer), entgegen und verarbeitet diese. Dazu besitzt der Broker Service eine eigene XMPP-Verbindung zum XMPP-Server. Der Broker Service kann so zur besseren Skalierung leicht auf einen weiteren Server ausgelagert werden. Aber auch eine einfache Integration in bestehende XMPP-Systeme und die Unabhängigkeit von der verwendeten XMPP-Server-Implementation wird durch diesen Ansatz gewährleistet.

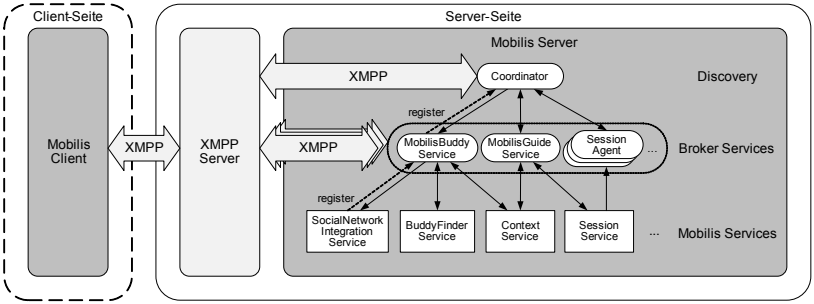


Abbildung 1: Mobilis-Architektur

Der Broker Service ist in der Regel ein kompositer Dienst, welcher maßgeblich auf untergeordnete Mobilis Services für die Erbringung seiner Funktionalität zurückgreift. Welche Mobilis Services durch den Broker Service bereitgestellt werden, kann per XMPP-Service-Discovery bei dem jeweiligen Broker Service in Erfahrung gebracht werden. Abbildung 2 zeigt die wesentlichen MobilisBuddy-Komponenten des Clients und Servers, welche nachfolgend im Detail erläutert werden.

Bei dem **Entwurf des Clients** wurde zur optimalen Trennung zwischen Benutzungsschnittstelle (GUI) und Anwendungslogik (Services) das Android-spezifische Intent-Konzept [And09] verwendet, das einen flexiblen Aufruf von verschiedenen Benutzer-Bildschirmen (Activities) und Service-Methoden ermöglicht. Folgende Dienste der Client-Seite sind von besonderer Bedeutung für MobilisBuddy:

- **Session Service:** Verwaltet weitere Client Services und ist zentraler Zugriffspunkt für die XMPP-Verbindung des Clients.
- **Context Management Service:** Verarbeitet eingehende Location Updates, bewirkt die Aktualisierung der Umgebungskarte sowie die Anzeige eines Proximity Events über den Android Notification Manager.
- **Social Network Management Service:** Initialisiert und verwaltet die jeweiligen Services zur Nutzung von sozialen Netzwerken, gibt Auskunft über alle Netzwerke, bei denen sich der Nutzer für die aktuelle Session angemeldet hat.
- **Buddy List Service:** Verwaltet die Kontaktliste und ist zuständig für das Publizieren von Kontakten aus dem lokalen Telefonbuch und dem XMPP-Roster an den Mobilis-Server.

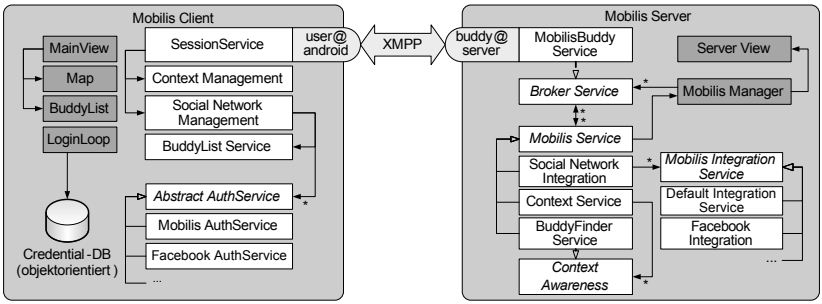


Abbildung 2: Struktur von MobilisBuddy

Zudem wird das automatische Einloggen (LoginLoop) in verschiedene Netzwerke durch auf dem Client hinterlegte Zugangsdaten (Credentials) ermöglicht. Die dafür verwendete db4o-Datenbank ist objektorientiert und auf geringen Speicherverbrauch optimiert. Sie ermöglicht eine einfache Erweiterung um neue Datenstrukturen. Die grafische Nutzerschnittstelle umfasst vorwiegend eine Google-Map mit angezeigten Positionen von Freunden in der Umgebung sowie eine Liste aller eingeloggten persönlichen Kontakte (BuddyList).

Der **Entwurf des Servers** verdeutlicht die service-orientierte Architektur. Eine besondere Rolle kommt hierbei den folgenden **Mobilis-Services** zu – sie liefern die eigentliche Funktionalität des Mobilis-Systems:

- **Social Network Integration Service**: Führt Informationen aus allen eingebundenen sozialen Netzwerken zusammen – genauer beschrieben in Abschnitt 3.3.
- **Context Service:** Neue Kontextinformationen gehen bei diesem Service ein. Der Context Service nimmt u.a. Location Updates von Clients entgegen und verwaltet ihre Positionen pro JID. Andere Mobilis-Services, welche an eingehenden Kontext-Informationen interessiert sind und das Context Awareness Interface implementieren, können sich bei diesem Dienst registrieren. So wird beispielsweise der BuddyFinder Service über Location Updates informiert.

- **BuddyFinder Service:** Bei einem eingehenden Location Update eines Nutzers fordert er vom Social Network Integration Service eine Liste aller Freunde des Nutzers an, die aus den verbundenen sozialen Netzwerken stammen und zurzeit ebenfalls im Mobilis-System eingeloggt sind. Darauf führt er mit Hilfe des Context Service einen beidseitigen **Proximity Check** zwischen dem Nutzer und jedem seiner Freunde durch. Ein Proximity Event tritt ein, sobald sich ein Teilnehmer innerhalb des vom anderen Teilnehmer festgelegten Radius befindet. In diesem Fall informiert der BuddyFinder Service den Teilnehmer per Proximity Event über den Freund in der Nähe.

3.2 Realisierung der Kollaboration

Die Kollaboration zwischen verschiedenen Clients wird durch die bereits aufgezeigten Services und durch den strukturierten Datenaustausch ermöglicht. Das MobilisBuddy-System nutzt zur Ermöglichung der Request-Response-Mechanismen spezielle XMPP-Datenpakete (Info/Query-Stanzas). Anhand eines Beispielablaufs soll die Realisierung des **umgebungsbasierten Buddyfinder-Dienstes** verdeutlicht werden (Abbildung 3).

Das Beispiel zeigt, wie nach einer Positionsveränderung des Client1 automatisch ein Location Update (LocationIQ SET) an den Server gesendet wird. Unter Zuhilfenahme des Context Service und des Social Network Integration Service ermittelt der BuddyFinder Service, dass sich Client1 innerhalb des von Client2 festgelegten Radius befindet und sendet deshalb das Location Update weiter an Client2, auf dessen Bildschirm daraufhin das Proximity Event angezeigt wird. Alle erfolgreich verschickten Location Updates werden per LocationIQ RESULT bestätigt.

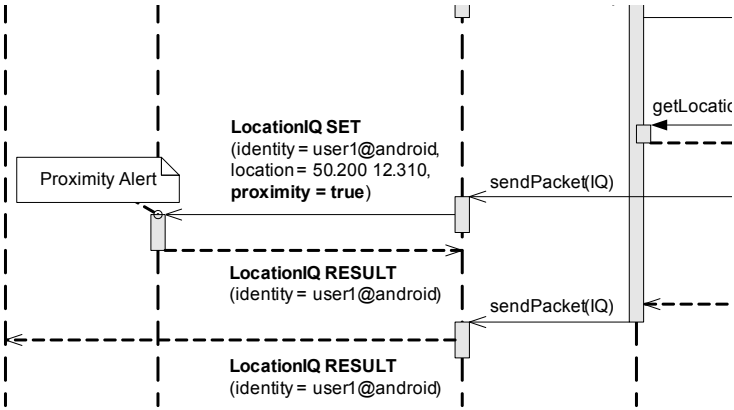


Abbildung 3: Location Updates und Proximity Check

3.3 Einbindung von sozialen Netzwerken

Die Integration von Informationen aus einem einzelnen sozialen Netzwerk geschieht auf Server-Seite durch einen **Mobilis Integration Service**. Dieser verfügt über ein Verzeichnis von Nutzer-IDs (spezifisch für das jeweilige soziale Netzwerk), die zu jeder Nutzer-ID gehörige Jabber-ID im Mobilis-System und deren Freundeslisten. Für jedes zu integrierende soziale Netzwerk muss damit ein spezifischer Mobilis Integration Service entwickelt werden. Der Social **Network Integration Service** verwaltet alle Mobilis Integration Services und kann von diesen nach Jabber-IDs aufgelöste Freundeslisten abfragen und als eine kombinierte Liste zur Verfügung stellen.

Als Beispiel für die Integration eines sozialen Netzwerkes in MobilisBuddy wird die Einbindung von Facebook über den **Facebook Integration Service** beschrieben. Dieser fragt die Facebook-Freundesliste des Nutzers ab. Der Zugriff wird durch eine bei Facebook registrierte Desktop-Application erlangt. Für die Gewährleistung der Sicherheit werden alle Nutzerdaten, die für ein Login bei Facebook benötigt werden, vom Client direkt an Facebook gesendet. Das von uns unter diesem Gesichtspunkt entwickelte Konzept zur Herstellung einer serverseitigen Verbindung mit Facebook ist in Abbildung 4 verdeutlicht.

Nach einem Verbindungswunsch des Clients fordert der Mobilis-Server bei Facebook einen Auth-Token an. Jegliche Anfragen an Facebook benötigen den auf dem Mobilis-Server hinterlegten API-Key und das API-Secret. Das private API-Secret bleibt nur dem Mobilis-Server bekannt, wohingegen Auth-Token und API-Key an den Client übertragen werden. Der Client loggt sich darauf mit Auth-Token, API-Key und seinen Account-Daten bei Facebook ein und meldet eine erfolgreiche Verbindung dem Mobilis-Server.

Jede weitere Kommunikation mit Facebook erfolgt aus Effizienzgründen ausgehend vom Mobilis-Server. Der Server besorgt sich von Facebook mit dem Auth-Token und API-Secret einen Session-Key und hat damit eingeschränkte Leserechte auf dem Account des Nutzers. Der mit Hilfe des Auth-Tokens eingeloggte Nutzer hat bei Facebook eine spezielle ID, welche nun vom Mobilis Server ausgelesen werden kann. Der Server kennt somit diese Facebook-ID und die zugehörige Jabber-ID eines jeden eingeloggten Nutzers. Er kann nun auch die Freundesliste aus Facebook abrufen. Nur die Freunde, die zurzeit ebenfalls im Mobilis-System eingewählt sind, werden in die für den jeweiligen Nutzer aktuelle Freundesliste übernommen. Alle Freundeslisten werden einheitlich auf Jabber-IDs abgebildet.

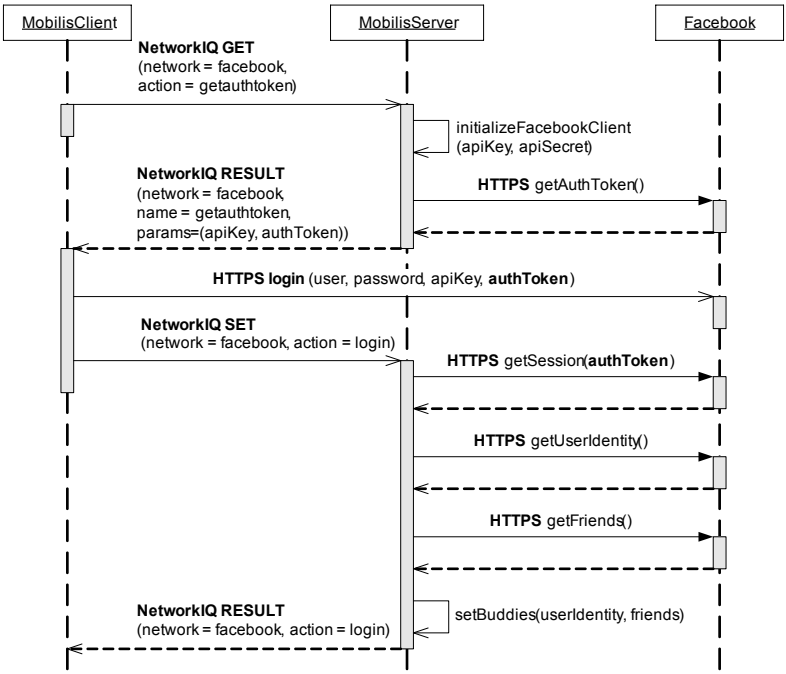


Abbildung 4: Kommunikation mit Facebook

4 Implementierung und Validierung

4.1 Implementierung der XMPP-Schnittstelle

Die in Abschnitt 3.1 beschriebene Architektur wurde innerhalb des Mobilis-Frameworks implementiert. Als Grundlage diente ein zuvor entwickelter Android-Client [Kor08], welcher schon über eine Benutzungsoberfläche mit eingeblendeter Umgebungskarte und Multi-User-Chat-Funktionalität verfügte. Zur Implementierung von Android-Applikationen bietet sich das von Google bereitgestellte Android-Eclipse-Plugin an, welches eine komfortable Unterstützung für die Java-basierte Entwicklung von Anwendungen für Android-Geräte ermöglicht.

Im Rahmen früherer Arbeiten wurde bereits ein Java-basierter Mobilis-Server entwickelt. Mobilis-Server und Android-Client kommunizieren per XMPP über einen XMPP-Server.

Als XMPP-Server-Implementierung wurde Openfire verwendet. Die Anbindung zu XMPP auf Client-, wie auch auf Server-Seite geschieht durch die Java-basierten Smack-Bibliotheken. Die Integration der Smack-Bibliotheken in die Android-Umgebung erwies sich als unproblematisch. Im Vorfeld zur Entwicklung von MobilisBuddy musste hierfür jedoch die Smack-Library geringfügig angepasst werden: Um XMPP-Nachrichten in entsprechende IQ-Objekte abzubilden, wird von Smack standardmäßig Code-Introspection verwendet. Auf Android-Geräten führte dieser Weg jedoch zu Fehlern, weshalb explizite IQ-Provider implementiert werden mussten. Zudem erwies sich die su-smack-Implementierung [SuS09] des XEP-0060 (Publish-Subscribe) [Xep0060] für Smack als nicht ausgereift genug. Insgesamt konnte aber für die Implementierung der Kommunikation per XMPP auf Standardimplementierungen zurückgegriffen werden, die mit wenig Aufwand auch an Android anpassbar sind.

4.2 Implementierung der Facebook-Schnittstelle

Auf dem Server wurde die Facebook Java API integriert, um die Kommunikation des Mobilis-Servers mit Facebook zu realisieren. Diese API wird von Facebook für den externen Zugriff bereitgestellt, auch wenn diese mittlerweile nicht mehr weiterentwickelt wird. Die Quellen der API wurden offen gelegt, die Wartung übernimmt jetzt eine unabhängige OpenSource-Community [FB09]. Vor der Integration mussten einige Änderungen an der API getätigt werden, da sich der Authentifikationsprozess für Facebook-Applikationen inzwischen geändert hat.

Die in Abschnitt 3.3 aufgezeigte Kommunikation mit Facebook wird durch die API-Klasse FacebookXmlRestClient erreicht. Sie bietet u.a. Methoden für die Anforderung eines Auth-Tokens (auth_createToken), das Erhalten eines Session-Keys (auth_getSession) und die Abfrage der Freundesliste des angemeldeten Nutzers (friends_get).

Auf Clientseite wurde auf die Integration der vollständigen API verzichtet, um die Rechenlast auf dem Client und den Datenverkehr im mobilen Netz zu minimieren. Lediglich die Übertragung der Login-Credentials zum Facebook-Server wird aus Sicherheitsgründen vom Client initiiert (siehe Abschnitt 3.3). Dies wird realisiert durch den sogenannten „FacebookMockLoginBrowser“ [AFCM09], welcher sich über einen simulierten HTTPS-Browser bei Facebook einloggt.

4.3 Validierung

Am Prototyp wurde ein Funktionstest durchgeführt. Dabei wurden verschiedene XMPP-Pakete (IQs) an das zu testende MobilisBuddy-System verschickt und entsprechend der zurückgesendeten Antworten das Verhalten der implementierten Schnittstelle gegenüber ihrer Spezifikation getestet. Dieser Black-Box-Test geschah in drei Etappen. Zunächst wurden die Server- und Client-Schnittstellen einzeln

getestet: Mit Hilfe der XML-Konsole des Miranda Instant Messengers wurden manuell eingegebene XMPP-IQs (zur Anmeldung bei sozialen Netzwerken, zur Positionsaktualisierung usw.) an die JID des MobilisBuddy-BrokerService bzw. die JID der Client-Instanz gesendet.
Im nächsten Schritt wurde ein klassischer Systemtest durchgeführt, indem mehrere Instanzen der Android-Client-Anwendung, sowie der Mobilis- und XMPP-Server auf verschiedenen Rechnern im Netzwerk gestartet wurden. Bewegungen der einzelnen Clients wurden über die in Eclipse eingebaute Android-Emulator-Kontrolle simuliert. Die Reaktionen des Servers und die Weiterleitung der Proximity-Events erfolgten korrekt. Allerdings stellte sich hier heraus, dass bei einer entsprechend hohen Anzahl angemeldeter, befreundeter Teilnehmer die Anzahl der durchgeführten Matchings unverhältnismäßig anstieg. Eine mögliche Lösung stellt die Weiterentwicklung des Systems, basierend auf dem XEP-0060 (Publish-Subscribe) [Xep0060], dar. Da es für diesen Teil des XMPP-Standards aber derzeit noch keine ausgereifte API-Unterstützung gibt, bleibt dies Gegenstand weiterer Arbeiten.
Schließlich wurde, um den Test zu automatisieren, ein sogenannter KML-Routing-Simulator entworfen und implementiert, der eine Google-KML-Datei [KML09] einlesen kann, welche einen Pfad entlang von geografischen Koordinaten beschreibt. Dieser Simulator ist ein Java-Programm, welches sich, genau wie der Android-Client, mit dem Mobilis-Server verbindet und dann automatisch in regelmäßigen Abständen Location Updates entsprechend dem definierten Pfad zum Server sendet (siehe Abbildung 5). Der KML-Routing-Simulator kann auch auf einem anderen Rechner ausgeführt werden. Auf diese Weise konnte eine schnelle, reproduzierbare Validierung der Schnittstellen ermöglicht werden.

5 Zusammenfassung und Ausblick

Anhand des Systemkonzeptes und der Beschreibung von Implementierungs- und Validierungserfahrungen konnte gezeigt werden, dass es möglich ist, bestehende soziale Netzwerke für mobile Kollaboration nutzbar zu machen und damit eine Vielzahl an neuen interessanten Anwendungen zu ermöglichen. Neben der hier detailliert vorgestellten Buddyfinder-Anwendung MobilisBuddy arbeiten wir derzeit noch an weiteren distanzabhängigen Anwendungen auf der Basis des Mobilis-Frameworks. So ist es mit MobilisTrader möglich, Produkte an Personen in der Umgebung des Nutzers zu verkaufen. Mit MobilisRide wird es möglich sein, spontan Mitfahrgelegenheiten innerhalb einer Stadt zu finden. Auch diese Anwendungen profitieren von einer Integration sozialer Netzwerke, da sie optional den Nutzerkreis auf bekannte Personen einschränken und somit Missbrauch der Dienste besser ausschließen können.

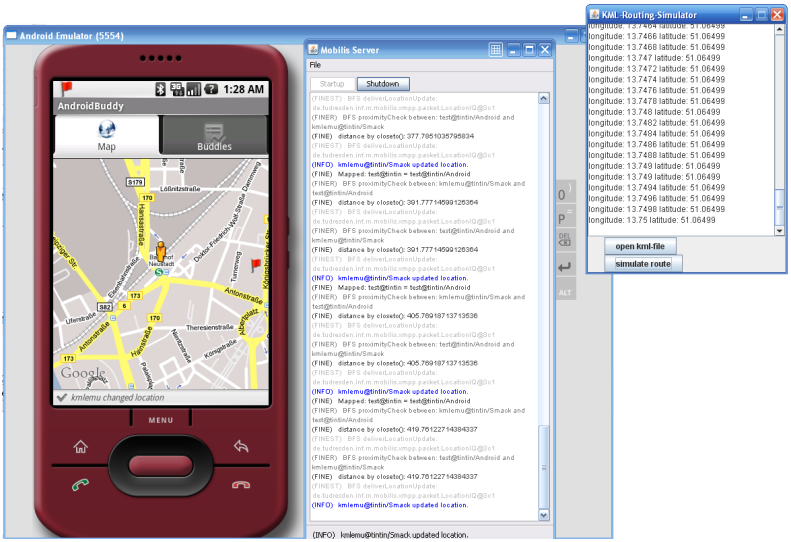


Abbildung 5: MobilisBuddy-Client, Mobilis-Server, KML-Routing-Simulator

6 Danksagung

Das Mobilis-Framework wurde gemeinsam mit den brasilianischen Partnern der PUC Rio de Janeiro und UFMG Belo Horizonte im vom BMBF geförderten Projekt Mobilis entwickelt. Maßgeblich zur Entwicklung der hier vorgestellten Prototypen haben neben den Autoren auch István Koren, Christopher Friedrich, Lukas Vierhaus und Martin Werner beigetragen.

Literatur

[AFCM09] Android Facebook Contact Manager – Google Code, 2009, http://code.google.com/p/android-facebook-contact-manager/

[Aka09] Aka-aki Networks GmbH, aka-aki, 2009, http://www.aka-aki.com/

[And09] Android Developers: Intents and Intent Filters, 2009, http://developer.android.com/guide/topics/intents/intents-filters.html

[BC09] BuddyCloud, 2009, http://www.buddycloud.com

[BM09] BuddyMob, 2009, http://www.buddymob.com

[FB09] Java – Facebook Developers Wiki, 2009, http://wiki.developers.facebook.com/index.php/Java

[Goo09] Google, Inc., 2009, Google Latitude, http://www.google.com/latitude/intro.html

[Ig09] Ignite Realtime: Openfire Server, 2009, http://www.igniterealtime.org/projects/openfire/

[KML09] KML Reference – KML – Google Code, 2009, http://code.google.com/intl/de-DE/apis/kml/documentation/kmlreference.html

[Kor08] Koren, I., Conceptual Design of a Mobile Collaborative Platform based on Android and XMPP, 2008, Bachelorarbeit, TU Dresden, Fakultät Informatik.

[MT+06] Martens, J., Treu, G., Ruppel, P., Weiss, D., Küpper, A., Linnhoff-Popien, C., Eine Plattform zur Unterstützung von proaktiven ortsbezogenen Mehrbenutzer–Anwendungen, 2006, 3. GI/ITG KuVS Fachgespräch „Ortsbezogene Anwendungen und Dienste“, Berlin, Germany.

[SNE06] Steiniger, S., Neun, M., Edwardes, A., Lecture Notes: Foundations of Location Based Services, 2006, Department of Geography, University of Zürich.

[SS+08] Springer, T., Schuster, D., Braun, I., Janeiro, J., Endler, M., and Loureiro, A. A., A flexible architecture for mobile collaboration services, 2008, ACM Middleware ‚08 Conference Companion, Leuven, Belgium.

[SuS09] su-smack, 2009, http://static.devel.it.su.se/su-smack/

[WMY08] Wang, S., Min, J., Yi, B. K., Location Based Services for Mobiles: Technologies and Standards, 2008, IEEE International Conference on Communication (ICC), Beijing, China.

[Xep0060] Millard, P., Saint-Andre, P., Meijer, R., XEP-0060: Publish-Subscribe, 2008, XMPP Standards Foundation.

[Xep0255] Timenes, H., Tennant, S., Savage, R., XEP-0255: Location Query, 2009, XMPP Standards Foundation.

[XSF09] XMPP Standards Foundation, 2009, http://xmpp.org/