Position Management für ortsbezogene Community-Dienste

Dissertation

eingereicht an der

Fakultät für Mathematik, Informatik und Statistik
der
Ludwig-Maximilians-Universität München

vorgelegt von

Georg Treu

Datum der Einreichung: 19. April 2007
Datum des Rigorosums: 11. Juni 2007

1. Berichterstatter: Prof. Dr. Claudia Linnhoff-Popien, LMU München
2. Berichterstatter: Prof. Dr. Christian Becker, Universität Mannheim

Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter in den Jahren 2004 bis 2007 am Lehrstuhl für Mobile und Verteilte Systeme des Instituts für Informatik der Ludwig-Maximilians-Universität München.

Besonders dankbar bin ich meiner Doktormutter, Frau Prof. Dr. Claudia Linnhoff-Popien. Claudia stand mir stets wohlwollend und mit wertvollem Rat zur Seite und ließ mir alle Freiheiten und Möglichkeiten, die mir wichtig waren. Gleichzeitig forderte sie Leistung ein. Dies war offensichtlich genau der richtige Weg, um mich dazu zu bewegen, einen solch umfangreichen Text wie diesen zu schreiben.

Dank gebührt auch Herrn Prof. Dr. Christian Becker, der sich bereit erklärt hat, diese Arbeit als Zweitgutachter zu betreuen. Christian gab mir bereits im Jahre 2005 anlässlich eines Vortrags auf der KiVS sehr hilfreiche Anregung zur Fortführung der in der Arbeit behandelten Themen.

In der Dissertation werden so genannte ortsbezogene Dienste behandelt, die Herr Dr. Axel Küpper thematisch in den Lehrstuhl einbrachte. Sein dazu erschienenes Buch „Location–based Services — Fundamentals and Operation“ war fast fertig, als ich an den Lehrstuhl kam. Axel bot mir an, mit ihm gemeinsam auf dem Gebiet der ortsbezogenen Dienste zu forschen, wofür ich ihm sehr dankbar bin.

Weiterer Dank gilt den jüngeren Team-Mitgliedern: Johannes Martens, Peter Ruppel und Diana Weiß. Alle drei wurden von Axel und mir bei ihrer Diplomarbeit betreut und haben damit (zum Teil ohne es zu ahnen) einen großen Beitrag zu dieser Arbeit geliefert. Auch unseren externen Mitgliedern, Florian Fuchs, Caroline Funk und Dr. Markus Strassberger, bin ich sehr dankbar für eine Menge gewinnbringender fachlicher Diskussionen und schöne gemeinsame Arbeiten.

Herzlicher Dank geht natürlich auch an Herrn Prof. Dr. Heinz-Gerd Hegering und sein MNM-Team, mit dem die Zusammenarbeit stets freundschaftlich und anregend war.

Zu guter Letzt danke ich meiner Familie für den großen Rückhalt und die guten Nerven während der vergangenen drei Jahre.

Zusammenfassung

Bei *ortsbezogenen Community-Diensten (Location-based Community Services, LBCSs)* tauschen mobile Nutzer ihre Ortsinformationen miteinander aus oder setzen sie zueinander in Beziehung, zum Beispiel um zu erfahren, welche anderen Community-Mitglieder gerade in der Nähe sind. Dabei regelt das so genannte *Position Management* die Übertragung, Analyse, Aufbereitung und Zugriffsverwaltung der Ortsinformationen entlang einer entsprechenden Wertschöpfungskette. Diese erstreckt sich vom Endgerät der Zielperson, auf dem die Ortsinformation zum Beispiel mittels *GPS* abgeleitet wird, über Intermediäre wie dem Location und LBS Provider bis hin zum Nutzer. Bei Community-Diensten ergeben sich die folgenden Herausforderungen an das Position Management, die sich grob in die Bereiche *Datenschutz* und *Effizienz* einteilen lassen:

Zum einen soll die Zielperson Kontrolle darüber erhalten, wer unter welchen Umständen auf ihre Ortsinformationen zugreifen darf. Um dies zu gewährleisten, müssen Ortsdaten vor dem Location und LBS Provider anonymisiert werden können, wofür bislang keine für Community-Dienste geeignete Technik besteht. Außerdem soll die Zielperson in der Lage sein, den Zugriff anderer Nutzer auf eine möglichst einfache und sozial akzeptable Form zu steuern.

Zum anderen sollen so genannte *proaktive Mehrpersonen-LBCSs* effizient realisiert werden. Hier wird automatisch erkannt, wenn zwei oder mehr Zielpersonen eine vorher definierte räumliche Konstellation zueinander einnehmen. Ein Beispiel ist *Buddy Tracking*, bei dem automatisch festgestellt wird, ob sich zwei Personen einander angenähert haben. Das Problem hierbei ist die häufige Übertragung von Ortsinformationen über die knappe Luftschnittstelle und der hohe, damit einhergehende Energieverbrauch auf dem Enderät der Zielperson. Auch hier fehlen bislang geeignete Lösungsansätze.

Die Dissertation entwickelt daher neuartige Konzepte in beiden vorgestellten Bereichen und zeigt deren Eignung anhand zahlreicher Simulationen sowie analytischer Bewertungen. Auch wird die *TraX-Plattform* vorgestellt, welche die erarbeiteten Konzepte praktisch umsetzt.

Abstract

In *Location-based Community Services (LBCSs)* mobile users interchange and correlate their spatial positions, for example, in order to find out which other community members are currently staying nearby. The so-called *position management* is responsible for the transmission, analysis, processing and access control of position information, which is directed along a corresponding supply chain. The supply chain spans from the mobile device of the target person, where the position is derived, for example, by *GPS*, via intermediaries like the location or LBS provider, to the domain of the user. Community services pose special requirements on position management, which can be coarsely divided into the fields *privacy protection* and *efficiency*:

First, the target person must be able to control by who and under which circumstances her position information is accessed. To guarantee that, it must be possible to anonymize the position data with respect to the location and LBS provider, for which so far no technique exists that is suited for community services. Also, the target person must be able to authorize requests to access her position in an easy and socially acceptable fashion.

Second, concepts for efficiently realizing so-called proactive multi-target LBCSs are needed. These services are automatically triggered as soon as two or more target person have entered into a certain pre-defined spatial constellation. An example is *buddy tracking*, which automatically detects when two persons have approached each other below a certain proximity distance. The technical problem to solve is the frequent transmission of position information over the scarce air-interface and the associated energy consumption at the mobile terminal of the target person.

This dissertation develops new concepts in both of the sketched fields and shows their feasibility based on numerous simulations and analytical reflection. Also the *TraX-platform* is presented, which practically implements the developed concepts.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ortsbezogene Dienste (Location-based Services, LBSs) sind dank der immer höheren Verbreitung von *Global Positioning System (GPS)*-fähigen mobilen Endgeräten endlich reif, für einen breiten Markt attraktiv zu werden. Durch die von LBSs vorgesehene Integration von Ortsinformationen mobiler Zielpersonen ergeben sich völlig neue Dienstarten. Zusätzlich zur Nutzung klassischer Navigationsanwendungen, die mit GPS im Endgerät nicht mehr nur im Auto, sondern auch zu Fuß oder auf dem Fahrrad möglich werden, kann sich der Nutzer zum Beispiel innerhalb so genannter *ortsbezogener Community-Dienste* verwirklichen. Diese Dienste – Beispiele sind *Friend-Finder*- oder ortsbezogene *Instant Messaging (IM)*-Dienste – berücksichtigen die Beziehungen zwischen ihren Nutzern und verwenden zu diesem Zweck gleichzeitig deren Aufenthaltsort.

LBSs tauchen nicht das erste Mal auf dem Radarschirm der Marktanalysten auf. Bereits Anfang der 90er-Jahre wurden sie als die neue Killer-Applikation gehandelt. Ähnlich wie andere Technologien, die in ihrer ersten Generation mit hohem Aufwand vorangetrieben wurden, geschah dies damals bei LBSs jedoch mit nur sehr begrenztem Erfolg.

Vergleichbar sind sie in dieser Hinsicht mit dem *Wireless Application Protocol (WAP)* der Version 1.0. Web-Inhalte wurden dem Nutzer hier über leichtgewichtige, auf die Leistungsmerkmale mobiler Endgeräte zugeschnittene Netzwerkprotokolle und Markup-Sprachen nahegebracht. Die Anbindung an das fest verdrahtete *World Wide Web (WWW)* erfolgte über spezielle WAP-Gateways. Nicht zuletzt aufgrund der geringen Kompatibilität dieser Speziallösung mit klassischen WWW-Seiten gab es jedoch nur sehr wenige externe Dienstbetreiber, die sich den Umstand machten, ihr Web-Angebot über WAP bereitzustellen. Eine Besserung ergab erst die Einführung von WAP 2.0, das auch klassische Internet-Protokolle und Markup-Sprachen vorsieht und so auch herkömmliche WWW-Seiten auf dem mobilen Endgerät zur Anzeige bringt. Die Eintrittsschwelle für externe Betreiber, ihr Web-Angebot an das mobile Umfeld anzupassen, verkleinerte sich somit. Gleichzeitig wurde die Technik benutzbarer, da leichter zu konfigurieren, und kostengünstiger, und endlich war eine wahrnehmbare Nutzerzahl bereit, sich auf diese neue Technologie einzulassen.

Ein wichtiger Treiber dieses Prozesses war technologischer Art. Immer leistungsfähigere Modelle mobiler Endgeräte machten es möglich, auf die speziellen Optimierungen, die WAP 1.0 vorsieht, zu verzichten. Erst mit der Verbreitung dieser modernen Geräte, die in der Lage sind, einen voll funktionsfähigen WWW-Browser auszuführen, erhielt das klassische WWW Einzug in die mobile Welt.

Eine vergleichbare Situation deutet sich derzeit bei LBSs an. Wie gesagt wurden Anfang der 90er-Jahre LBSs mit sehr hohen Erwartungen besetzt, die Kassen der Mobilfunkbetreiber, die nicht zuletzt unter den hohen Investitionen in UMTS-Frequenzen und -Netztechnologien gelitten hatten, wieder aufzubessern. Grundlegender technologischer Ansatz war dabei die netzbasierte Ortung: Mittels für das *Global System for Mobile Communications (GSM)* spezifischer Signalisierung wurde die Position des Mobilfunkteilnehmers ermittelt, indem sie zum Beispiel dem Standort der aktuell bedienenden Basisstation gleichgesetzt

wurde (Cell-Id-Verfahren). Die so abgeleitete Position wurde dann entweder vom Netzbetreiber selbst verwendet, um dem Teilnehmer eigene LBSs anzubieten (ein klassisches Beispiel ist der Taxi-Finder-Dienst), oder sie wurde externen LBS-Anbietern gegen eine Gebühr zur Verfügung gestellt.

Wie im nächsten Kapitel genauer diskutiert, hat dieser Ansatz aufgrund verschiedener Einschränkungen nicht funktioniert: Zum einen waren die LBSs, die direkt vom Netzbetreiber erbracht wurden – von bestimmten Ausnahmen abgesehen – nur wenig innovativ. Zum anderen hatten es die externen LBS-Betreiber schwer, LBSs überhaupt zum Laufen zu bringen. Dies lag zum großen Teil an dem sehr rigiden Kostenmodell, was eine Bezahlung pro Ortung vorsieht und es somit erschwert, ein tragbares Geschäftsmodell zu finden. Gleichzeitig machte die nur sehr ungenaue Ortung durch das Cell-Id-Verfahren mögliche LBSs nur begrenzt attraktiv bzw. schränkte die Auswahl an LBSs, die überhaupt sinnvoll waren, stark ein. Schließlich lag eine Hauptschwierigkeit für Drittanbieter darin begründet, dass zur Erreichung einer kritischen Nutzermasse Verträge bzw. technische Schnittstellen mit mehreren Mobilfunkbetreibern gleichzeitig geschlossen werden mussten.

Im weitesten Sinn ist die netzbasierte Wertschöpfungskette, in deren Zentrum der Netzbetreiber stand und die den LBSs der ersten Generation zugrunde lag, also mit WAP 1.0 vergleichbar: Diese LBSs hatten eine unzureichende Qualität (Cell-Id), die Dienstdiversität war sehr beschränkt, und sie waren mit hohen Kosten verbunden.

Gleichermaßen wie bei WAP ergibt sich für LBSs in den letzten Jahren eine technologische Entwicklung, die diese Rahmenbedingungen grundlegend verändert. Mobiltelefone sind zunehmend mit GPS-Empfängern ausgestattet, die es möglich machen, die aktuelle Position ihres Besitzers mit einer sehr hohen Genauigkeit zu bestimmen. Bei GPS handelt es sich um ein endgerätbasiertes Ortungsverfahren, was zur Folge hat, dass der Nutzer die Kontrolle über seine eigenen Ortsdaten hat. Er ist somit in der Lage, diese mittels öffentlicher Datendienste, wie dem *General Packet Radio Service (GPRS)* oder *Universal Mobile Telecommunications System (UMTS)* im paketvermittelnden Modus, direkt an externe LBS-Betreiber zu übermitteln, wodurch sich eine bisher nicht da gewesene Wettbewerbssituation ergibt. Dem Mobilfunkbetreiber kommt in dieser neuen Wertschöpfungskette keine aktive Rolle mehr zu.

Durch die Integration von GPS in Mobiltelefone wird es jetzt also möglich, LBSs mit einer sehr hohen Qualität (10 bis 15 m Genauigkeit) anzubieten. Gleichzeitig ist mit einer Vielzahl konkurrierender LBS-Betreiber auf dem Markt zu rechnen und mit einer entsprechenden Vielfalt innovativer Dienste. Da der Netzbetreiber sein Preismonopol auf die Ortsdaten verloren hat, richten sich die entstehenden Kosten prinzipiell nur noch nach den Preisen der zur Übertragung benutzten Datendienste, welche momentan stark sinkend sind.

Eine im Festnetz besonders erfolgreiche Dienstform sind so genannte *Community-Dienste*, die die Beziehungen zwischen ihren Nutzern bei der Diensterbringung berücksichtigen. Beispiele sind Blogs, Foren, Chat-Räume und Messenger-Anwendungen sowie Dienste, die auf dem sozialen Netzwerk ihrer Nutzer aufbauen. Mit der Möglichkeit, Ortsinformationen von Nutzern zu integrieren, ergibt sich für mobile Community-Dienste eine Vielfalt neuer Anwendungen: Nutzer können sich zum Beispiel über den Aufenthaltsort der anderen Community-Mitglieder informieren und dementsprechend ihr Kommunikationsverhalten anpassen. Oder sie können sich automatisch benachrichtigen lassen, sobald andere Community-Mitglieder in der Nähe sind, um persönlich in Kontakt zu treten.

Während die Anreicherung klassischer Einzelnutzer-Dienste mit Ortsinformationen durch

die neue, GPS-basierte Wertschöpfungskette relativ leicht wird, stellen sich bei Community-Diensten jedoch besondere Anforderungen an das so genannte *Position Management*.

Eine Schwierigkeit ist die Erbringung *proaktiver Mehrpersonen-LBSs*. Dabei wird automatisch erkannt, wenn sich zwischen mehreren Zielpersonen derselben Community eine bestimmte räumliche Konstellation ergibt. Einfachstes und grundlegendstes Beispiel, dem in dieser Arbeit eine besondere Bedeutung zukommt, ist die Nahbereichserkennung. Der Nutzer wird hierbei automatisch informiert, sobald sich ihm eine Person auf eine bestimmte räumliche Distanz angenähert hat, zum Beispiel im Rahmen eines proaktiven Friend-Finder-Dienstes. Hierzu müssen die Positionen der unterschiedlichen Teilnehmer regelmäßig miteinander abgeglichen werden. In einem naiven Ansatz, bei dem die Positionen dazu ständig an einen zentralen Location Server übermittelt werden, würde dies jedoch zu großen technischen Problemen führen: Die begrenzte Luftschnittstelle würde überlastet und die Batterieressourcen des Endgeräts, die zum Übertragen der Nachrichten gebraucht werden, in einem nicht akzeptablen Maße beansprucht. Für die Realisierung dieser und verwandter Funktionen werden in dieser Arbeit neue, effizientere Lösungen entwickelt und evaluiert.

Ein zweites Problem, das entsteht, wenn so sensible Daten wie Ortsinformationen zwischen verschiedenen LBS-Akteuren ausgetauscht werden, ist der *Datenschutz*. Diesem sehr komplexen Thema widmet sich der zweite Teil dieser Arbeit, sowohl auf einer rein technischen Ebene (Anonymisierung) als auch auf einer nutzerfokussierten (soziale Akzeptanz von Autorisierungsverfahren).

Die Struktur der Arbeit ist wie folgt: Das direkt anschließende Kapitel vermittelt die im Weiteren benötigten Grundlagen von LBSs. Besprochen werden die verschiedenen Klassen vorstellbarer LBSs sowie die bei der Erbringung eines LBS involvierten Rollen und Akteure. Ferner wird detailliert auf die bereits angesprochenen LBS-Wertschöpfungsketten der ersten und der zweiten Generation eingegangen. Kapitel 3 setzt dann das Thema LBSs und Community-Dienste klarer in Zusammenhang, und es werden die in dieser Arbeit zu behandelnden Herausforderungen an das Position Management im Detail erarbeitet. Kapitel 4 entwickelt neuartige Mechanismen zur effizienten Realisierung der proaktiven Nahbereichs- und Trennungserkennung zwischen zwei mobilen Zielpersonen. Kapitel 5 erweitert diesen Mechanismus dann auf die proaktive Erkennung von Cliquen, wobei automatisch erkannt wird, wenn sich mehr als zwei Personen zueinander in räumlicher Näher befinden. Kapitel 6 geht ausführlich auf die beiden angesprochenen Datenschutz-Themen ein. In Kapitel 7 wird die praktische Realisierung der entwickelten Konzepte anhand der so genannten TraX-Plattform gezeigt. Dabei ergibt sich unter anderem eine funktionale Schichtung, die nicht nur die effiziente und einfache Erkennung von Cliquen und nahe gelegenen Paaren ermöglicht, sondern auch die Lösung weiterer wichtiger Problemstellungen, wie der Überwachung der k nächsten Nachbarn einer Zielperson, in Aussicht stellt. Kapitel 8 fasst schließlich die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf weitere Forschungsmöglichkeiten in dem behandelten Gebiet.

2 Grundlagen ortsbezogener Dienste

2.1 Akteure und Rollen

LBSs sind IT-Dienste, die die geographische Position eines oder mehrerer Individuen einbeziehen, um für ihre Nutzer Informationen zu erzeugen, zu filtern oder auszuwählen. Die Ausführung eines LBS erfolgt organisationsübergreifend. Dies bedeutet, dass zur Erbringung eines LBS verschiedene *Akteure* zusammenarbeiten. Solche Akteure können natürliche Personen, Firmen oder andere Organisationsformen sein. Jeder dieser Akteure übernimmt dabei eine oder mehrere *Rollen*. Diese fassen jeweils eine Reihe technischer Funktionen zusammen, die direkt mit der Erbringung des LBS verknüpft sind, vergleiche [98; 94]. Die folgenden Rollen sind für diese Arbeit von Bedeutung.

- **Positioning Enabler:** Als Positioning Enabler wird die Organisation bezeichnet, die die zur Ortung notwendige Infrastruktur einrichtet und betreibt.
- **Zielperson:** Das sind die Individuen, deren Ortsinformationen für den LBS herangezogen werden.
- **Location Provider:** Er tritt als Intermediär auf, der die Ortsinformationen mehrerer Zielpersonen verwaltet und aufbereitet und sie verschiedenen LBS Providern zur Verfügung stellt.
- **Content Provider:** Er unterstützt LBSs durch die Bereitstellung zusätzlicher Inhalte, zum Beispiel Kartenmaterial oder Routeninformationen.
- **LBS Provider:** Dieser erbringt letztendlich den LBS.
- **Nutzer:** Das sind die Personen, die auf ortsbezogene Inhalte zugreifen bzw. sie nutzen.

2.2 LBS-Klassifikation

Im Folgenden werden verschiedene Merkmale beschrieben, mit denen sich LBSs funktional klassifizieren lassen, vergleiche [98; 111]. Eine solche LBS-Klasse ist dabei entweder inhärent oder variabel. Im ersten Fall sind die Merkmale, die die Klasse definieren, fest an die Struktur des LBS gekoppelt. Treffen solche inhärenten Merkmale also nicht mehr auf einen LBS zu, so handelt es sich auch nicht um denselben Dienst. Eine variable LBS-Klasse lässt sich im Gegensatz dazu während der Ausführung desselben LBS mit Hilfe geeigneter

Handover-Mechanismen wechseln. Die inhärenten Klassenmerkmale eines LBS sind also wesentlich stärker als die variablen. Die drei folgenden inhärenten Klassenmerkmale finden sich auch wieder in Kapitel 3 zur Identifizierung der Herausforderungen an Community-Dienste:

- **Reaktive und proaktive LBSs:** Während bei reaktiven LBSs dem Nutzer ortsbezogene Informationen nur auf Anfrage ausgeliefert werden, lösen proaktive LBSs beim Eintreten vordefinierter räumlicher Ereignisse, wie etwa dem Annähern einer Zielperson an einen *Point of Interest (PoI)*, Dienstaktionen aus.

- **Selbst- und querverweisende LBSs:** Bei selbstverweisenden LBSs ist der Nutzer mit der Zielperson identisch, das heißt, die dem Nutzer präsentierte Information bezieht sich auf dessen eigene Position. Im Gegensatz dazu stehen querverweisende LBSs, bei denen die Nutzer bezüglich des Aufenthaltsorts anderer Personen informiert werden, also nicht mit der Zielperson gleichgesetzt sind.

- **Einzel- und Mehrpersonen-LBSs:** Einzelpersonen-LBSs verbinden die Position einer einzelnen Zielperson mit geographischen Inhalten, zum Beispiel zur Darstellung des aktuellen Aufenthaltsorts der Zielperson auf einer Karte. Bei Mehrpersonen-LBSs werden hingegen die Positionen mehrerer Zielpersonen miteinander verknüpft, zum Beispiel um herauszufinden, ob sich bestimmte Personen in geographischer Nähe befinden.

Als Beispiele für variable LBS-Klassenmerkmale seien die folgenden genannt. Verschiedene weitere Unterteilungen sind jedoch denkbar:

- **Zentralisierte und Peer-to-Peer-LBSs:** Zentralisierte LBSs werden von einem zentralen Location oder Application Server angeboten und verwaltet. Peer-to-Peer-LBSs organisieren sich hingegen selbstständig, das heißt, die Positionsdaten werden direkt zwischen den Nutzern bzw. Zielpersonen ohne die Anwesenheit von Intermediären ausgetauscht.

- **Outdoor- und Indoor-LBSs:** Outdoor-LBSs verfügen über ein großes Abdeckungsgebiet im Freien und verwenden in der Regel satellitengestützte oder zellbasierte Positionierungsverfahren. Im Gegensatz dazu unterstützen Indoor-LBSs den Nutzer innerhalb eines Gebäudes und basieren daher auf lokalen Ortungsverfahren. Bei Indoor-LBSs spielen in der Regel topologische Beziehungen, wie zum Beispiel die Erreichbarkeit verschiedener Räume anhand eines Gebäudeplans, eine größere Rolle als bei Outdoor-LBSs.

Weil der dynamische Handover eines LBS zwischen Peer-to-Peer und zentraler Ausführung bzw. zwischen Indoor und Outdoor für den Nutzer transparent verlaufen kann, werden die beiden Klassenmerkmale also als variabel eingestuft.

Abbildung 2.1 gibt einen Überblick auf die unterschiedlichen Klassen von LBSs. Typische LBSs der ersten Generation sind reaktiv, selbstverweisend, auf einzelne Nutzer beschränkt und fokussieren sich auf den Outdoor Bereich. Die Mehrheit dieser LBSs lässt sich am besten durch den Begriff Finder-Dienste charakterisieren, da sie ihren Nutzern den Weg

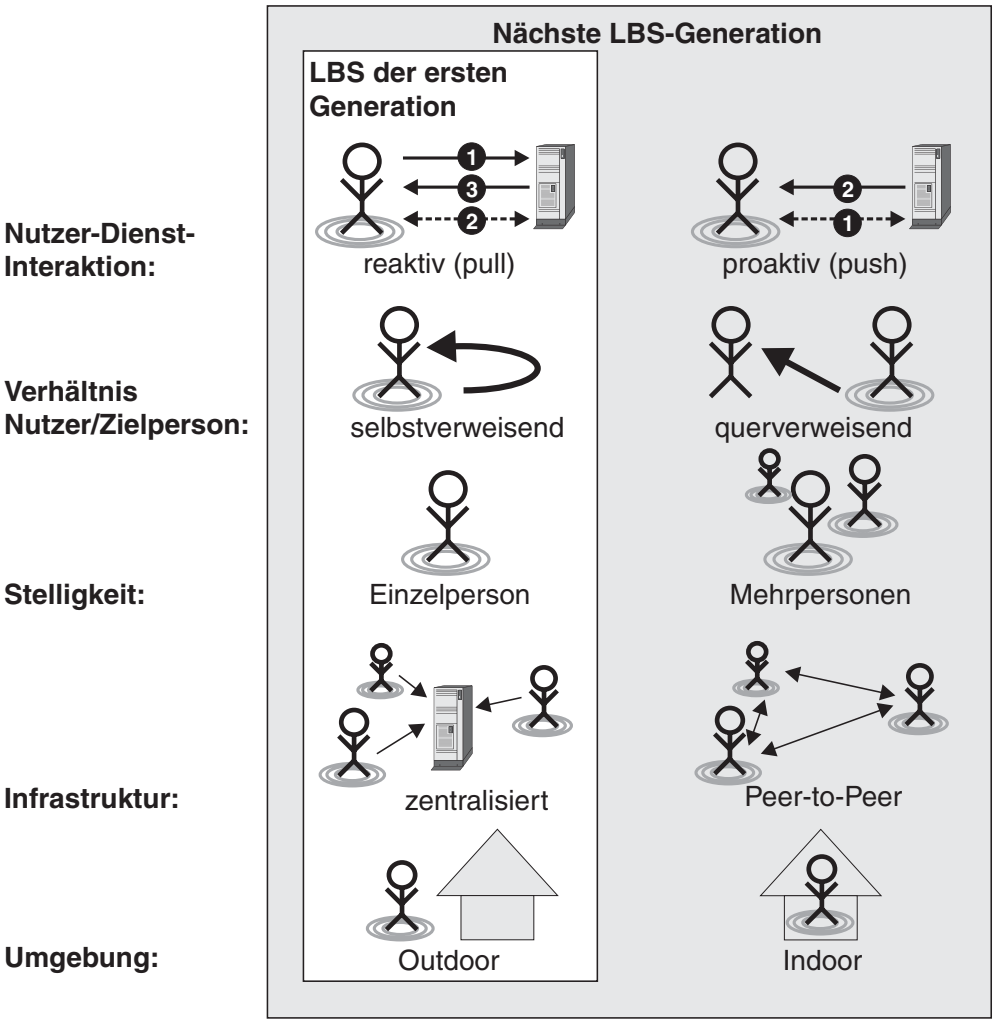

Abbildung 2.1: Funktionale Klassifikation von LBSs

zu in der Nähe gelegenen PoIs wie Restaurants oder Geldautomaten weisen. In den letzten Jahren hat sich jedoch gezeigt, dass höher entwickelte LBSs größeren Zuspruch erhalten könnten. Wie es scheint, stellen sie die nächste Generation von LBSs dar. Beispiele sind Child-Tracker-Dienste, mit deren Hilfe Eltern den Aufenthaltsort ihrer Kinder bestimmen können (reaktiv, querverweisend und Einzelperson), oder Community-Dienste, bei denen die Mitglieder einer virtuellen Community ihre geographischen Positionen auf Wunsch miteinander austauschen bzw. in Bezug zueinander setzen können (reaktiv und proaktiv, querverweisend, Einzel- und Mehrpersonen).

2.3 LBS-Wertschöpfungsketten

Bei der Erbringung eines LBS fließen die Ortsinformationen zwischen den verschiedenen LBS-Akteuren entlang einer entsprechenden LBS-Wertschöpfungskette. Abbildung 2.2 zeigt die klassische, netzwerkzentrische Wertschöpfungskette. Sie hat ihren Ursprung in der

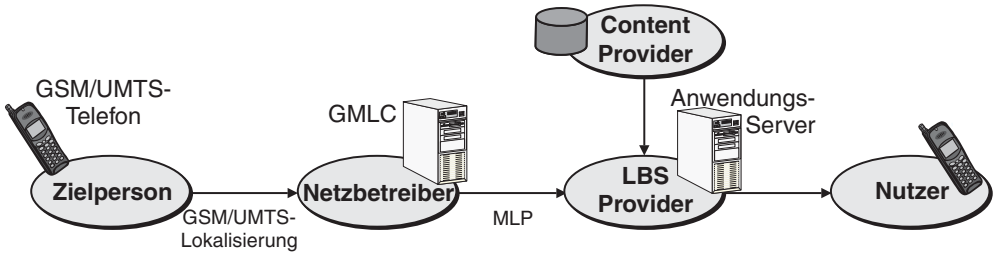

Abbildung 2.2: Klassische, netzwerkzentrische LBS-Wertschöpfungskette

Gesetzesdirektive E911 der Vereinigten Staaten, die in den späten neunziger Jahren durchgesetzt wurde und die es Netzbetreibern vorschreibt, notrufende Mobilfunkteilnehmer mit einer vorgegebenen Genauigkeit zu orten. Nachdem die vorhandenen Netze damals nicht in der Lage waren, die geforderte Genauigkeit zu erreichen, führte E911 zu enormen Anstrengungen seitens der Netzbetreiber, die Netze durch entsprechende Ortungstechnologien aufzurüsten. LBSs stellten damals eine viel versprechende Möglichkeit dar, diese Technologien auf kommerzieller Basis zu nutzen und die auferlegten Investitionen zumindest teilweise wieder hereinzuholen.

Als Ergebnis ergab sich eine netzwerkzentrische Wertschöpfungskette, in der der Netzbetreiber gleichzeitig als Location Provider auftritt. Die Positionsbestimmung der Zielperson wird innerhalb des zellulären Mobilfunknetzes kontrolliert und durch netzspezifische Technologien wie *Cell-Id* oder *Uplink Time Difference of Arrival (U-TDoA)* realisiert. Die abgeleiteten Ortsinformationen werden am *Gateway Mobile Location Center (GMLC)* des Netzbetreibers bereitgestellt und dort von einem externen LBS Provider mit Hilfe des *Mobile Location Protocol (MLP)* [131], das von der *Open Mobile Alliance (OMA)* standardisiert und gepflegt wird, gegen Gebühr abgeholt.

Leider stellte sich der netzwerkzentrische Ansatz aus mehreren Gründen nicht als Erfolg heraus. Erstens versuchen noch viele Netzbetreiber, insbesondere außerhalb der Vereinigten Staaten, die hohen Kosten, die bei der Installation hoch entwickelter Ortungstechnologien entstehen, zu vermeiden. Stattdessen wird oft nur eine Ortung mittels Cell-Id angeboten, welche die Genauigkeitsanforderungen vieler LBSs leider nicht erfüllen kann. Zweitens stellen viele Netzbetreiber die abgeleiteten Ortsinformationen ihrer Teilnehmer nur zu sehr hohen Preisen bereit.

Mit der immer breiteren Verfügbarkeit präziser endgerätbasierter Ortungsverfahren, wie GPS oder dem geplanten europäischen System Galileo, ergibt sich zur netzwerkzentrischen Wertschöpfungskette eine Alternative. Positionsdaten können nun direkt vom mobilen Endgerät der Zielperson ermittelt und an den LBS bzw. Location Provider mit Hilfe paketvermittelnder mobiler Datendienste wie GPRS oder UMTS weitergeleitet werden. Das GMLC des Netzbetreibers wird dabei einfach umgangen.

Neben satellitengestützten Systemen wie GPS und Galileo stehen noch andere endgerätbasierte Methoden zur Verfügung. So bestimmt zum Beispiel das Endgerät beim endgerätbasierten Pendant der netzwerkbasierten Cell-Id-Methode seine Position anhand der momentan bedienenden Basisstation oder eines sich in Reichweite befindlichen WLAN Access Points selbst. Eine lokal vorgehaltene Datenbank liefert dabei die Abbildung vom Identifikator der Basisstation auf dessen geographische Position. Die durch dieses Verfah-

ren erzielte Genauigkeit entspricht also dem Radius der Zelle, welcher bei GSM bis zu 35 km betragen kann. Sie lässt sich verbessern, indem gleichzeitig mehrere Basisstationen sowie gemessene Signalstärken berücksichtigt werden, wie es zum Beispiel im Rahmen des PlaceLab-Projektes [100] vorgesehen ist. In diesem Fall spricht man auch von *Location Fingerprinting* [83]. Es ermöglicht insbesondere die Ortung innerhalb von Gebäuden, wo satellitengestützte Systeme bislang nicht einsetzbar sind. Andere mögliche endgerätbasierte Verfahren, die mit Hilfe des zellulären Netzes umgesetzt werden, sind *Enhanced Observed Time Difference (E-OTD)* und *Observed Time Difference of Arrival (OTDoA)*. Einen umfassenden Überblick auf die verschiedenen Verfahren liefert [94]. Eine spezielle Behandlung für Ortungstechniken innerhalb von Gebäuden findet sich in [90].

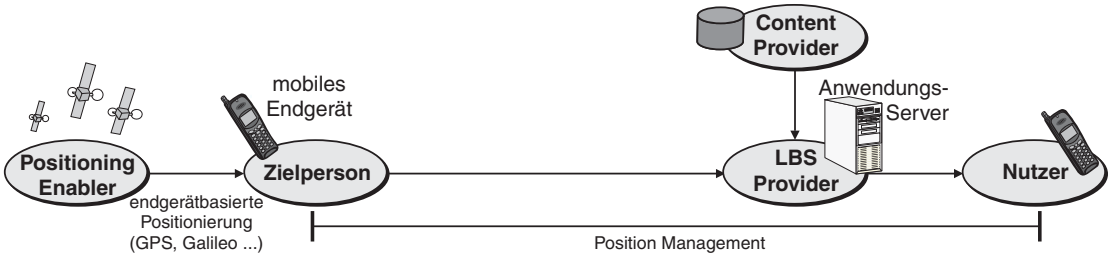

Abbildung 2.3: Direkte endgerätzentrische LBS-Wertschöpfungskette

Zwei verschiedene endgerätzentrische Wertschöpfungsketten sind möglich. In beiden Fällen verwaltet ein Positioning Enabler die zur Ortsbestimmung notwendige Infrastruktur. Dieser ist jedoch nicht – wie beim netzwerkzentrischen Ansatz – mit der Vermittlung von Ortsinformationen betraut.

Die direkte endgerätzentrische Wertschöpfungskette ist die einfachere von beiden und wird in Abbildung 2.3 dargestellt. Mit ihrer Hilfe lassen sich leicht rudimentäre LBSs realisieren, die keine kontinuierliche Verfolgung der Zielpersonen erfordern und auch nicht die Positionen mehrerer Zielpersonen korrelieren. Gemeint sind also reaktive und selbstverweisende Einzelpersonen-LBSs, wie die klassischen Finder-Dienste der ersten LBS-Generation, die auf der klassischen, netzwerkzentrischen Wertschöpfungskette beruhten.

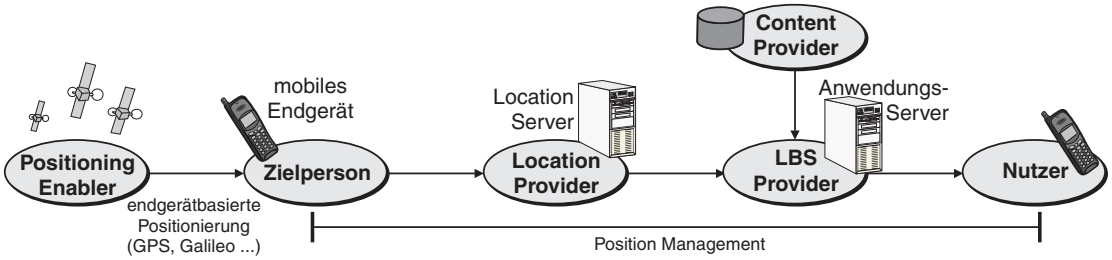

Abbildung 2.4: Indirekte endgerätzentrische LBS-Wertschöpfungskette

Zur Realisierung proaktiver, querverweisender oder Mehrpersonen-LBSs müssen beim direkten Ansatz die entsprechenden Tracking-Funktionalitäten jedoch von jedem LBS Provider separat bereitgestellt werden. Deshalb fungiert bei der indirekten endgerätzentrischen

Wertschöpfungskette, die in Abbildung 2.4 dargestellt ist, ein Location Provider als Vermittler zwischen Zielperson und LBS Provider. Der Location Provider realisiert bereits höherwertige Funktionen, zum Beispiel zur Erkennung räumlicher Konstellationen zwischen mehreren Zielpersonen, und stellt sie dem LBS Provider über wohldefinierte Schnittstellen bereit, vergleiche [162]. Er verwaltet und wertet die Daten einer Menge von Zielpersonen im Auftrag mehrerer LBSs aus, wodurch sich eine Reihe von Synergien ergibt. Das indirekte Modell bietet außerdem den Vorteil, dass der Location Provider aus Sicht der Zielperson der einzige Zugangspunkt ist. Eine Zielperson, die ihrem Location Provider vertraut, muss somit den einzelnen LBS Providern weniger Vertrauen entgegenbringen. Dies ist insbesondere bei proaktiven Mehrpersonen-LBSs entscheidend, die typischerweise sehr viele Daten über die Zielpersonen sammeln.

Auf den ersten Blick hat das indirekte Modell Ähnlichkeit mit der netzwerkzentrischen Lösung. Tatsächlich könnte der Netzbetreiber auch hier die Rolle des Location Providers übernehmen und die Ortsinformationen seiner Teilnehmer verwalten. Im Gegensatz zum netzwerkzentrischen GMLC hat der Netzbetreiber nun jedoch sein Monopol auf die Ortsinformationen verloren und steht in Konkurrenz zu anderen Location Providern. Die Bereitstellung der typischen Funktionen des Location Providers ist nun außerdem nicht mehr abhängig von systemspezifischen Positionierungsverfahren und erlaubt daher auch die Integration nicht standardisierter Verfahren und Techniken zur Indoor-Lokalisierung, womit sich die teilweise weniger flexiblen Netzbetreiber unter Umständen schwerer tun als auf die Rolle des Location Providers spezialisierte Unternehmen.

2.4 Position-Update-Methoden

Diese Arbeit geht von der indirekten endgerätzentrischen Wertschöpfungskette aus. Ein Hauptteil der Arbeit beschäftigt sich mit der Reduktion von Nachrichtenaufwand über die begrenzte Luftschnittstelle, der bei der Realisierung proaktiver Mehrpersonen-LBSs anfällt. Im Folgenden werden daher verschiedene Methoden beschrieben, mit deren Hilfe sich der zentralisierte Location Server des Location Providers über die Positionen der Zielpersonen informieren kann. Die Geräte der Zielpersonen übermitteln zu diesem Zweck so genannte *Position Updates* an den Server, was anhand der folgenden *Position-Update-Methoden* geschehen kann, vergleiche [106; 96]:

- **Polling.** Der Location Server fragt die aktuelle Position des Geräts an. Dies kann durch eine Anfrage seitens der Anwendung an den Server ausgelöst werden oder auf periodischer Basis bzw. aufgrund spezieller Caching-Strategien vom Server ausgehen.

- **Periodisches Position Update.** Das Gerät löst ein Position Update aus, nachdem – bezogen auf das letzte Update – ein vordefinierter Zeitraum, das so genannte *Update-Intervall*, verstrichen ist.

- **Distanzbasiertes Position Update.** Das mobile Endgerät sendet ein Position Update, sobald die Distanz zwischen der zuletzt übermittelten und der aktuellen Position einen vordefinierten Schwellwert übersteigt, welcher im Folgenden als *Update-Distanz* bezeichnet wird.

- **Zonenbasiertes Position Update.** Ein Position Update wird beim Betreten oder Verlassen einer vorgegebenen geographischen Zone ausgelöst, welche zum Beispiel als Kreis mit eindeutig definiertem Mittelpunkt und Radius oder als Polygon definiert sein kann. Im Folgenden wird eine solche Zone als *Update Zone* bezeichnet.

- **Koppelnavigation (Dead Reckoning).** Basierend auf der zuletzt gesendeten Position sowie zusätzlich übermittelten Parametern, wie der zu dem Zeitpunkt gemessenen Geschwindigkeit und Richtung, berechnen Endgerät und Server simultan dessen geschätzte aktuelle Position, wozu eine beiderseits bekannte Vorhersagefunktion f verwendet wird. Stellt das Gerät fest, dass die berechnete Position von seiner wahren Position um mehr als einen vorgegebenen Schwellwert abweicht, so wird ein Position Update übermittelt. Verschiedene Arten der Berechnung von f sind bislang bekannt, worauf an späterer Stelle im Detail eingegangen wird.

- **Piggybacking.** Hier wird die Position des Endgeräts dem LBS-Dienstaufruf direkt angehängt, zum Beispiel zur Erfragung umliegender PoIs. Der zur Ortung notwendige, zusätzliche Nachrichtenaustausch kann so vermieden werden. Piggybacking eignet sich vor allem für reaktive, selbstverweisende LBSs.

Um die auf der Luftschnittstelle stark begrenzte Bandbreite zu schonen sowie den Energieverbrauch mobiler Endgeräte einzuschränken, soll das durch diese Methoden erzeugte Nachrichtenaufkommen möglichst gering gehalten werden. Gleichzeitig sollen die jeweiligen Anwendungsanforderungen bezüglich Verfügbarkeit und Aktualität der ermittelten Positionen erfüllt werden. Die verschiedenen Methoden können daher dynamisch konfiguriert und entsprechend dieser Anforderungen angewendet werden. Abbildung 2.5 zeigt den typischen Verlauf eines Nachrichtenaustausches, wie er zwischen dem Location Server und dem mobilen Endgerät der Zielperson im Rahmen einer solchen Positionierungssitzung stattfinden kann.

Nach der Registrierung für die Sitzung (1) übermittelt der Location Server dem mobilen Endgerät einen so genannten *Position Update Request* (2), welcher die Konfiguration trägt, anhand derer Position Updates ausgelöst werden sollen. Dies entspricht also der Auswahl einer oder mehrerer der genannten Update-Methoden sowie der Definition entsprechender Parameter, also zum Beispiel distanzbasierte Updates parametrisiert durch die Update-Distanz. Das Gerät aktiviert dann, falls noch nicht geschehen, eines der auf dem Endgerät verfügbaren Verfahren zur Positionsbestimmung, zum Beispiel GPS, von dem es ab nun kontinuierlich Positionsinformationen erhält (3). Die so ermittelten Positionen werden stetig mit der empfangenen Update-Konfiguration verglichen. Im Falle einer Übereinstimmung sendet das Gerät ein *Position Update* an den Server (4). Unter anderem enthält diese Nachricht die letzte Position des Zielobjekts sowie einen Zeitstempel. Diese Werte werden dann vom Location Server verarbeitet, beispielsweise werden sie mit den Positionen anderer Zielpersonen verglichen oder an einen Anwendungsserver weitergeleitet. Eine Update-Konfiguration bleibt so lange gültig, bis sie vom Location Server aktualisiert (5) oder explizit beendet (8) wird. Zusätzlich kann der Location Server unabhängig von der ausgeführten Update-Methode gezielt Positionen nachfragen, indem er einen so genannten *Polling Request* an das Gerät absetzt (6). Dieser wird mittels einer *Polling Response*, welche wiederum die aktuell gemessene Position trägt, vom Gerät beantwortet (7).

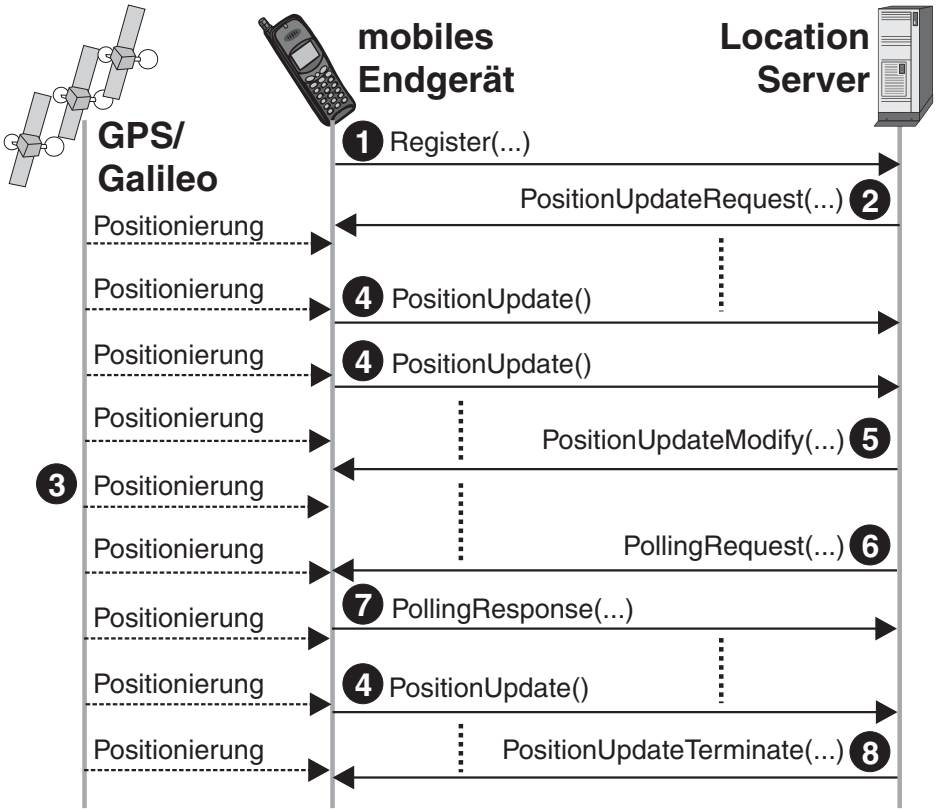

Abbildung 2.5: Nachrichtenaustausch während einer Positionierungssitzung

Allgemein gesprochen dienen die unterschiedlichen Update-Methoden dazu, eine sekundäre Kopie von Ortsinformationen auf dem Location Server aktuell zu halten, basierend auf einer primären Kopie auf dem mobilen Endgerät. Wegen Einschränkungen bezüglich der begrenzten Luftschnittstelle sowie der Batteriereserven des Geräts kann der Stand der Sekundärkopie dem der primären jedoch nicht immer exakt entsprechen. Die Ortsinformation auf dem Location Server ist also stets mit einer gewissen *Unsicherheit* verknüpft. Im Gegensatz zur Ungenauigkeit, die aus einer ungenauen Positionsbestimmung resultiert, beschreibt Unsicherheit unvollständiges Wissen des Location Servers bezüglich einer bereits gemessenen (und möglicherweise ungenauen) Position.

Zum Erreichen eines festen Unsicherheitsschwellwerts d zwischen Gerät und Server wurden die Eigenschaften von periodischem und distanzbasiertem Updating [106] wie auch verschiedener Varianten der Koppelnavigation [166] [105] in der Literatur bereits hinreichend untersucht. Hierbei soll die auf dem Endgerät gemessene Position von der auf dem Server verfügbaren nie um mehr abweichen als eine vorgegebene Luftliniendistanz d. Dies ermöglicht es dem Location Server, gewisse Lokalisierungsanfragen zu bearbeiten, ohne dafür das Endgerät explizit anfragen zu müssen, sondern nur durch Inspektion seines lokalen Caches. Nimmt man eine bestimmte Häufung solcher Anfragen an, so lässt sich durch die Methoden der insgesamt zwischen Gerät und Server entstehende Kommunikationsaufwand reduzieren. Es zeigt sich, dass distanzbasierte Updates den periodischen vorzuziehen sind, falls sich die betrachteten Zielobjekte unregelmäßig bewegen, wie es zum Beispiel

bei Personen in typischen Büroumgebungen der Fall ist. Überdies stellt sich die Koppelnavigation als effizienter heraus als distanzbasierte Updates, insbesondere wenn sie durch Kartendaten unterstützt wird.

Bestehende Untersuchungen beziehen sich auf die kontinuierliche Verfolgung der Positionen einzelner Zielpersonen bezüglich eines fest vorgegebenen Unsicherheitsschwellwerts. Die in dieser Arbeit behandelte proaktive Nahbereichs- und Trennungserkennung sowie die Erkennung von Cliquen betrachten hingegen das räumliche Verhältnis mehrerer Zielpersonen. Es wird später erklärt, wie die verschiedenen Position-Update-Methoden zu diesem Zweck dynamisch parametrisiert werden können. Zuvor klärt das nächste Kapitel jedoch erst die Begrifflichkeiten und Anforderungen im Bezug auf die behandelten Community-Dienste.

3 Problemstellung

Dieses Kapitel identifiziert Herausforderungen, die bei der Realisierung so genannter *ortsbezogener Community-Dienste (Location-based Community Services, LBCSs)* auftreten. Zuerst werden die Bedeutungen der Begriffe Community, Community-Dienst und LBCS geklärt und es wird kurz auf bestehende LBCSs eingegangen. Dann wird anhand der LBS-Klassifikation aus dem letzten Kapitel diskutiert, welche Herausforderungen die verschiedenen Typen von LBCSs an das so genannte *Position Management* stellen. Aus den Problemstellungen motivieren sich die später in dieser Arbeit dargelegten Lösungskonzepte.

3.1 Begriffsdefinitionen

Der Begriff der *Community* wird fachübergreifend verwendet, und es existiert eine Vielzahl unterschiedlicher, sich zum Teil widersprechender Definitionen und Sichtweisen, vergleiche auch [139]. So meinen beispielsweise Betriebswirte, Soziologen und Informatiker oft völlig verschiedene Dinge, wenn sie von Communities sprechen. In dieser Arbeit wird der Begriff Community mit *virtueller Community* bzw. *virtueller Gemeinschaft* synonym verwendet. Diese wurden von Howard Rheingold in seinem Buch „the virtual community“ erstmalig im Zusammenhang mit Gemeinschaften innerhalb des Internets definiert [140]:

„Virtuelle Communities sind Gemeinschaften, die aus dem Netz entstehen, wenn genug Leute öffentliche Diskussionen lange genug aufrecht erhalten, mit genügend menschlichen Gefühlen, um persönliche Bindungen im Cyberspace zu bilden.“

Eine weitere Definition von virtuellen Communities entstammt [61]. Sie ist aus Sicht des Autors klarer gefasst als die vorherige und wird dieser Arbeit zugrunde gelegt:

„Eine virtuelle Community ist ein Zusammenschluss von Menschen mit gemeinsamen Interessen, die untereinander mit gewisser Regelmäßigkeit und Verbindlichkeit auf computervermitteltem Wege Informationen austauschen und Kontakte knüpfen.“

Eng verbunden mit Communities ist der Begriff *Community-Dienst*. Darunter versteht man IT-Dienste, die gemeinsam von einer Community benutzt werden und die die Beziehungen zwischen Community-Mitgliedern miteinbeziehen. Im Gegensatz zu klassischen Einzelnutzerdiensten, bei denen nur ein Nutzer im Dienstmodell vorgesehen ist, betrachtet die Ausführung eines Community-Dienstes also stets mehrere Community-Mitglieder gleichzeitig.

Für Community-Dienste, die hauptsächlich über ein mobiles Endgerät genutzt werden, hat sich auch der Gattungsbegriff *Mobile Social Software (MoSoSo)* eingebürgert. Beispiele sind die Dienste Dodgeball [2], Sociallight [15] , AT&T’s Find Friend [154], 6th sense [14],

Freever (Chat und Foren) [18] und erweiterte IM-Dienste wie Hubbub [80] oder ConNexus-Awarenex [157].

Eine weitere Unterklasse von Community-Diensten bilden die so genannten *Groupware*-Dienste. Eine Groupware, auch *kollaborative Software* genannt, bezeichnet eine Anwendung zur Unterstützung der Zusammenarbeit innerhalb einer Gruppe von Personen, und zwar über zeitliche und/oder räumliche Distanzen hinweg. Solche Anwendungen entstammen dem Forschungsgebiet *Computer Supported Collaborative Work (CSCW)*. Beispiele für eine mobile, ortsbezogene Groupware-Implementierung sind mPower [104] und AGAPE [44].

Eine umfassende Diskussion des Community-Begriffs sowie Architekturen und Tools zur Unterstützung von Community-Diensten bietet [88].

Angelehnt an [65] verstehen sich in dieser Arbeit LBCSs als erweiterte LBSs, die nicht nur einzelne Nutzer, sondern eine Nutzer-Community durch den Ortsbezug unterstützen. Andersherum gesprochen sind LBCSs also Community-Dienste, die den Aufenthaltsort ihrer Nutzer einbeziehen.

Im Rahmen des COSMOS-Projekts [138] wurden zum Beispiel die folgenden LBCSs entwickelt und pilotiert, vergleiche [45; 72]: ein reaktiver Friend-Finder-Dienst, bei dem sich der Nutzer die aktuelle Person eines anderen Community-Mitglieds auf einer Karte anzeigen lassen kann (reaktiv, querverweisend, Einzelperson); ein Friend-Alert-Dienst, bei dem Nutzer proaktiv über die räumliche Annäherung eines anderen Community-Mitglieds hingewiesen werden (proaktiv, querverweisend, Mehrpersonen); ein Virtual-Post-it-Dienst, bei dem sich Community-Mitglieder gegenseitig Nachrichten schicken können. Im Unterschied zu herkömmlichen Messaging-Diensten wird die Nachricht dem Empfänger erst dann zugestellt, wenn er eine bestimmte räumliche Zone betritt (proaktiv, querverweisend, Einzelperson).

Bezogen auf die Klassifikation des letzten Kapitels bedeutet dies also, dass nicht nur Mehrpersonen-LBSs, welche ja die Ortsinformationen mehrerer Zielpersonen miteinander verknüpfen, LBCSs sind. Es ist vielmehr jeder querverweisende LBS, also jeder LBS, an dessen Ausführung mehr als eine Person (Nutzer oder Zielperson) beteiligt ist, ein LBCS. LBCSs können sowohl proaktiv als auch reaktiv sein.

3.2 Herausforderungen an das Position Management

Diese Arbeit vereint unter dem Begriff *Position Management* alle Funktionen eines LBS, die sich mit der Übertragung von Ortsinformationen zwischen verschiedenen Akteuren, ihrer Manipulation und Veredelung sowie der Verwaltung von Zugriffsrechten und dem Schutz vor unrechtmäßigem Zugriff Dritter beschäftigen. Kurz gefasst regelt das Position Management die Verteilung von Ortsinformationen entlang der LBS-Wertschöpfungskette. Vor dem Hintergrund der aktuellen technologischen Entwicklungen wird dabei die im letzten Kapitel beschriebene indirekte endgerätzentrische Wertschöpfungskette zugrunde gelegt.

LBCSs, die wie gesagt aus technischer Sicht mit der Klasse querverweisender LBSs identisch sind, stellen besondere Herausforderungen an das Position Management. Die Anforderungen lassen sich grob in die Bereiche *Effizienz* und *Datenschutz* einteilen. Wie im Folgenden beschrieben wird, ist die Schwierigkeit der Erfüllung dieser Anforderungen abhängig vom jeweiligen LBCS-Typ.

3.2.1 Effizienz

Bei reaktiven LBCSs erfolgt eine Ortsbestimmung der Zielperson(en) in der Regel einmalig bei jeder Anfrage des Nutzers (von speziellen Caching-Strategien, die zum Beispiel in [106] behandelt werden, einmal abgesehen). Bei reaktiven Mehrpersonen-LBCSs werden die ermittelten Ortsinformationen zusätzlich beim Location Provider bzw. LBS Provider miteinander verglichen, zum Beispiel zur Berechnung der relativen Distanzen der Personen. Technisch gestalten sich reaktive LBCSs jedoch relativ einfach.

Anders verhält es sich bei proaktiven LBCSs. Hier müssen räumliche Ereignisse in Bezug auf die Zielperson(en) automatisch erkannt werden, was eine ständige Überwachung des aktuellen Aufenthaltsorts der Zielperson(en) bedingt. Wird wie in dieser Arbeit eine endgerätbasierte Ortung zugrunde gelegt,[1] muss mit zwei begrenzten Ressourcen besonders effizient umgegangen werden:

- **Luftschnittstelle zur Datenübertragung.** Im Gegensatz zur drahtgebundenen Kommunikation, bei der sich beliebig viele neue, zum Teil exklusiv nutzbare Verbindungskanäle zwischen Kommunikationspartnern herstellen lassen, stellt die Luftschnittstelle bei der Mobilkommunikation eine einzige, von allen Teilnehmern gemeinsam genutzte Ressource dar, die natürlich begrenzt ist.

  Da es sich bei LBCSs um querverweisende LBSs handelt, wird die auf dem Endgerät der Zielperson abgeleitete Ortsinformation schlussendlich in einer bestimmten aufbereiteten Form dem Nutzer übermittelt. Ortsinformationen werden also prinzipiell zwischen dem mobilem Endgerät der Zielperson und einem Location Server im Internet ausgetauscht. Häufige Datenübertragungen zwischen Endgerät und Location Server führen jedoch zum Engpass auf der Luftschnittstelle. Ziel muss also sein, den Nachrichtenaufwand über die Luftschnittstelle so weit wie möglich zu reduzieren, was insbesondere bei der Realisierung proaktiver LBCSs eine Herausforderung darstellt:

  Proaktive Einzelpersonen-LBCSs gestalten sich relativ einfach. Es muss nur dann ein Ereignis ausgelöst werden, wenn die Zielperson eine bestimmte geographische Zone entweder betritt oder verlässt. Eine Schonung der Luftschnittstelle lässt sich bereits mit Hilfe der im letzten Kapitel beschriebenen zonenbasierten Position Updates erreichen.

  Weit schwieriger verhält es sich bei proaktiven Mehrpersonen-LBCSs. Hier kommt erschwerend hinzu, dass die Positionen mehrerer Zielpersonen kontinuierlich verglichen werden müssen, zum Beispiel um ein Ereignis zu erzeugen, sobald sich zwei oder mehr Personen auf eine bestimmte Distanz angenähert haben. Die Entwicklung von effizienten Strategien, die räumliche Mehrpersonen-Ereignisse erkennen und gleichzeitig den Nachrichtenaustausch über die Luftschnittstelle reduzieren, ist ein in der Wissenschaft noch relativ unbehandeltes Gebiet. Diesem Thema kommt daher in dieser Arbeit eine besondere Bedeutung zu.

- **Energiereserven auf dem Endgerät.** Während moderne Endgeräte immer leistungsfähiger werden und mit einer zunehmenden Vielfalt an Kommunikationstechnologien

[1] Bei netzbasierter Ortung lassen sich proaktive LBSs nur sehr schwer bis gar nicht realisieren, da hier jeder Prüfung einer Ortsinformation eine individuelle Signalisierung über die Luftschnittstelle vorausgehen muss. Netzbasierte Ortung wird daher in dieser Arbeit nicht weiter diskutiert.

ausgestattet sind, konnte die Bereitstellung entsprechender Energiereserven, zum Beispiel in Form von Akkus, mit dieser rapiden Entwicklung nicht mithalten. Beschränkte Akkulaufzeiten sind momentan eine der größten technologischen Hürden bei der Entwicklung mobiler Endgeräte und Anwendungen. Proaktive LBSs schlagen hier besonders zu Buche. Zwar bewirkt die anvisierte Reduzierung der über die Luftschnittstelle gesendeten Nachrichten eine relative Senkung des Energieverbrauchs des Endgeräts. Gleichzeitig stellt aber die kontinuierliche Positionsbestimmung, zum Beispiel durch GPS, ein gravierendes Problem für den Energiehaushalt des Endgeräts dar. Gesucht werden daher zum Beispiel Verfahren, die es ermöglichen, den GPS-Empfänger bei der Detektierung von Update-Zonen (im Fall von zonenbasierten Position Updates) die meiste Zeit abgeschaltet zu lassen. Für die grobgranulare Positionsbestimmung könnte auf anspruchslosere Technologien wie *Proximity*- oder *Motion-Sensing* zurückgegriffen werden. Während diese Arbeit die Motivation zur Entwicklung solcher Techniken liefert, müssen diese jedoch an anderer Stelle im Detail untersucht werden.

3.2.2 Datenschutz

Bei allen LBCSs spielt der Datenschutz eine kritische Rolle. Der eigene Aufenthaltsort wird bei diesen Diensten nicht lokal ausgewertet, wie bei klassischen Navigationsanwendungen, sondern an andere LBS-Akteure ausgehändigt. Ziel muss dabei stets sein, dass die Zielperson die Kontrolle darüber behält, wer genau Zugriff auf ihre Ortsinformationen erhält und mit welchem Detailgrad die Informationen sichtbar sind. Um solche Kontrollmechanismen durchzusetzen, sind natürlich geeignete Sicherheitstechniken vorauszusetzen, zum Beispiel zur Authentifizierung der LBS-Akteure untereinander und zur gesicherten Kommunikation. Diese werden in dieser Arbeit jedoch nicht vertieft. Es werden vielmehr zwei besonders für LBCSs relevante Datenschutzprobleme aufgegriffen, die wiederum unterschiedliche LBCS-Dienstklassen unterschiedlich stark betreffen:

- **Gewährleistung der Anonymität.** Blickt man auf bestehende Community-Dienste im Internet, zum Beispiel Blogs, so zeigt sich, dass die Nutzer dieser Dienste oft Pseudonyme verwenden, um ihre wahre Identität zu verbergen. Die prinzipielle Möglichkeit, bei der Dienstnutzung anonym zu bleiben, ist aus Sicht des Autors essentiell wichtig für einen offenen und freien Umgang mit Community-Diensten. Anonymität soll dabei nicht nur in Bezug auf andere Nutzer desselben Dienstes möglich sein. Die Zielperson soll insbesondere vor Intermediären wie dem Location Provider und dem LBS Provider anonym bleiben können. Ein damit verbundenes Problem, welches in Abschnitt 6.1 genauer beschrieben wird, ist das folgende: Aufgrund der zum Teil großen Menge der durch den Location Provider bzw. LBS Provider gesammelten Ortsinformationen lässt sich in Verbindung mit entsprechendem Hintergrundwissen sehr leicht vom verwendeten Pseudonym auf die wahre Identität einer Zielperson schließen. Verschärft tritt dieser Umstand bei proaktiven Mehrpersonen-LBCSs hervor, da hier potentiell die meisten Ortsinformationen über die Zielperson gesammelt werden. Ein Ziel der Arbeit ist es deshalb, Mechanismen zu entwickeln, die eine Deanonymisierung gesammelter, pseudonymisierter Ortsdaten durch den LBS Provider bzw. Location Provider verhindern oder zumindest erschweren.

- **Autorisierung von Nutzern.** Bei LBCSs soll die Zielperson kontrollieren können, welche Nutzer auf ihren Aufenthalt zugreifen dürfen. Wie auch in Abschnitt 6.2 beschrieben, wird dafür klassischerweise eine *explizite Autorisierung* angewendet. Die Zielperson entscheidet hierbei explizit, entweder durch Hinterlegung entsprechender Datenschutz-Policies beim Location Provider oder LBS Provider oder durch Ad-hoc-Autorisierung auf Anfrage, wer auf ihre privaten Daten Zugriff erhält und wer vom Zugriff ausgeschlossen ist. Die explizite Autorisierung wirft allerdings zwei Problempunkte auf: Erstens erfordern die bekannten Mechanismen bei der Zielperson einen relativ hohen Verwaltungsaufwand. Zweitens hat explizite Autorisierung ein relativ hohes Potenzial, zu negativen sozialen Konsequenzen für die Zielperson zu führen: Innerhalb einer Partnerschaft oder eines Arbeitsverhältnisses könnte sie sich zum Beispiel durch sozialen Druck dazu genötigt fühlen, einer Autorisierung bestimmter Nutzer zuzustimmen. Andererseits könnte eine negative Autorisierung vom Nutzer als Zurückweisung empfunden werden, was sozialen Sprengstoff liefert.

  Aus Sicht des Autors liegt die Hauptschwierigkeit bei der Vermittlung einer Zurückweisung („Wie sag' ich Petra, dass sie mich nicht orten soll?“) in der Tatsache begründet, dass bei expliziter Autorisierung eine Zugriffserlaubnis bei der Zielperson denselben Aufwand erzeugt wie eine Zugriffsverweigerung. Verweigerungen werden daher vom Nutzer potentiell als Zurückweisung eingestuft.

  Es werden also neue Konzepte benötigt, die dem Nutzer Raum lassen, Zugriffsverweigerungen mehrdeutig zu interpretieren. So könnte zum Beispiel eine Nicht-Ortbarkeit einer bestimmten Zielperson durch technische Schwierigkeiten oder durch knappe ökonomische Ressourcen der Zielperson bedingt sein. In diesem Fall sinkt die negative soziale Auswirkung einer Zugriffsverweigerung.

  Es zeigt sich, dass das beschriebene Problem vor allem bei reaktiven LBCSs auftritt. Hier wartet der Nutzer nämlich bei einer Anfrage an den Dienst auf eine konkrete Rückantwort. Will sich die Zielperson nicht orten lassen, so muss dies dem Nutzer entsprechend kommuniziert werden. Proaktive LBCSs sind in dieser Hinsicht vorteilhafter. Bei einer ungewünschten Ortung können tatsächlich eingetretene räumliche Ereignisse dem Nutzer verschwiegen werden, ohne dass dieser das unbedingt merkt. Das Problem der Vermittlung einer Zurückweisung an den Nutzer stellt sich bei proaktiven LBCSs somit nur bedingt.

3.3 Zusammenfassung

Nach der Klärung der für diese Arbeit wichtigen Begrifflichkeiten wurden verschiedene Problemstellungen aus den inhärenten Eigenschaften von LBCSs motiviert. Konkret handelt es sich zum einen um die effiziente Realisierung proaktiver Mehrpersonen-LBCSs, die den größten Teil dieser Arbeit ausmachen. Entsprechend sind dem Thema die folgenden Kapitel 4 und 5 gewidmet. Zum anderen werden in Kapitel 6 die beiden besprochenen Datenschutz-Themen behandelt: Anonymisierung von Ortsdaten bei proaktiven Mehrpersonen-LBCSs und Nutzerautorisierung bei reaktiven LBCSs. Einen allgemeinen Überblick zu Datenschutzmechanismen bei LBSs bietet zum Beispiel [151].

4 Proaktive Nahbereichs- und Trennungserkennung

Dieses Kapitel stellt die fortgeschrittenen LBCS-Funktionen der proaktiven *Nahbereichs-* und *Trennungserkennung* vor und entwickelt verschiedene zur Umsetzung geeignete Strategien, die sowohl argumentativ als auch mittels Simulationen verglichen werden. Nahbereichserkennung beschreibt die Fähigkeit eines LBCS, automatisch zu erkennen, wann der räumliche Abstand eines Paares innerhalb einer Gruppe mobiler Zielobjekte eine vorgegebene *Nahbereichsdistanz* unterschreitet. Analog dazu bezeichnet die Trennungserkennung die Fähigkeit, zu erkennen, dass zwischen zwei Objekten eine vorgegebene *Trennungsdistanz* überschritten wird. Die beiden Funktionen eignen sich zur Umsetzung proaktiver Mehrpersonen-LBCS, bei denen Mitglieder einer Community benachrichtigt werden, sobald andere Mitglieder in ihre Nähe gekommen sind oder sich von ihnen entfernt haben. Kombinationen daraus sind ebenfalls möglich. Typische Anwendungsgebiete umfassen Friend-Finder- und Dating-Dienste, Instant Messaging, Mobile Gaming, Flottenmanagement- und Logistiksysteme sowie Systeme zur Ortung pflegebedürftiger Menschen, von Kindern und von Haustieren. Darüber hinaus dienen die Funktionen höherwertigen Erkennungsfunktionen als Grundbaustein. In Kapitel 5 zeigt sich dies anhand der proaktiven Erkennung von Cliquen, bei der die Nahbereichs- und Trennungserkennung dynamisch auf bestimmte Paare von Zielpersonen angewendet wird, um automatisch ein Ereignis zu generieren, sobald sich mehr als zwei Personen paarweise nahe sind.

Für beide Funktionen wird die indirekte endgerätzentrische Wertschöpfungskette aus Kapitel 2 zugrunde gelegt: Die Zielpersonen verfügen über ein zelluläres Mobilfunkgerät, welches mit einem endgerätbasierten Ortungsverfahren wie GPS gekoppelt ist. Mit Hilfe der bereits geschilderten Position-Update-Methoden werden die ermittelten Ortsinformationen zwischen dem Gerät und dem Location Server des Location Providers ausgetauscht. Die benutzte Update-Methode und ihre Parameter werden dabei vom Server in Abhängigkeit der zur Nahbereichs- und Trennungserkennung verwendeten Strategie konfiguriert. Ziel der vorgestellten Strategien ist, die Anzahl von Nachrichten sowohl auf dem *Downlink* (vom Server zum Endgerät) als auch auf dem *Uplink* (vom Endgerät zum Server) so weit wie möglich zu reduzieren. So kann wertvolle Bandbreite auf der Luftschnittstelle eingespart werden, eventuelle monetäre Kosten, die bei den Zielpersonen für die Benutzung von Trägerdiensten wie GPRS oder paketvermittelndem UMTS entstehen, werden gesenkt, und der Batterieverbrauch des mobilen Endgeräts wird eingeschränkt. Außerdem wird durch die Nachrichtenreduktion auch die Last am Location Server, der für die Verarbeitung der Ortsinformationen zuständig ist, abgesenkt. Die Skalierbarkeit erhöht sich, da mehr Endgeräte gleichzeitig pro Location Server bedient werden können.

Das Kapitel ist folgendermaßen strukturiert. Der folgende Abschnitt gibt zunächst einen Überblick über verwandte Arbeiten. In Abschnitt 4.2 wird die zu untersuchende Problemstellung formalisiert. Verschiedene Strategien zur Nahbereichs- und Trennungserkennung,

die auf den Position-Update-Methoden aus Abschnitt 2.4 aufsetzen, werden in Abschnitt 4.3 entwickelt. Angefangen wird mit einem sehr einfachen Ansatz, der die Position der Zielperson beim Server immer dann aktualisiert, wenn eine fest eingestellte Distanz überwunden wird. Darauf folgen zwei Verbesserungen, die die Update-Methode dynamisch und in Abhängigkeit der nächsten Nachbarn der Zielperson konfigurieren. Der einzige bestehende Ansatz, der Streifenalgorithmus [29], wird ebenfalls vorgestellt. In Abschnitt 4.4 werden die verschiedenen Strategien per Simulation verglichen. Ferner werden zwei der eigenen Ansätze anhand eines Prototyps praktisch evaluiert. Abschnitt 4.5 stellt eine Optimierung für die Nahbereichserkennung vor, die auf der Koppelnavigation als Update-Methode basiert. Schließlich fasst Abschnitt 4.6 die gewonnenen Ergebnisse zusammen.

4.1 Verwandte Arbeiten

Nahbereichs- und Trennungserkennung zwischen mobilen Zielobjekten sowie verwandte Funktionen werden im Bereich der Datenbanktheorie schon seit mehreren Jahren erforscht. Zum Beispiel schlagen [122] Algorithmen vor, die anhand einer vorliegenden räumlichen Datenbank Bereichsabfragen, k-nächste-Nachbarn-Abfragen sowie sortierte Distanzabfragen ausführen. Während Bereichsabfragen alle Zielobjekte innerhalb eines vorgegebenen geographischen Bereichs liefern, finden k-nächste-Nachbarn-Abfragen die k nächsten Nachbarn eines Objekts. Sortierte Distanzabfragen geben Zielobjekte in aufsteigender Reihenfolge bezüglich der Distanz zu einem vorgegebenen Referenzpunkt aus. Eine weitere Funktion ist die Erkennung von Cliquen, wobei festgestellt wird, ob eine größere Menge von Zielobjekten nahe beieinander stehen. [167] bietet hierfür eine formale Spezifikation und beschreibt zwei Algorithmen, die auf die Reduktion von Rechenlast ausgelegt sind. Die Autoren orientieren sich an bestehenden Indizierungstechniken zur Verwaltung mobiler Zielobjekte, wie zum Beispiel [27; 89; 34]. Ziel ist es bei allen Ansätzen, die Arbeitslast, welche an einem räumlichen *Datenbankmanagementsystem (DBMS)* anliegt, zu reduzieren, Antwortzeiten zu verkürzen sowie eine hohe Skalierbarkeit zu erreichen. In den meisten Fällen wird davon ausgegangen, dass die Positionen der Zielobjekte stets beim Location Server verfügbar sind, oder dass diese anhand eines einfachen periodischen Protokolls übermittelt werden.

Wie bereits erwähnt, ist das Hauptziel der vorgestellten Algorithmen jedoch die Reduktion von Position Updates und anderen Nachrichten, die zwischen Server und Endgerät über die Luftschnittstelle ausgetauscht werden und die dazu dienen, die Positionen der verfolgten Zielpersonen miteinander vergleichen zu können. Leider sind bislang nur wenige ähnliche Initiativen bekannt. Während verschiedene Position-Update-Methoden von [106] identifiziert wurden und zum Teil von dem *3rd Generation Partnership Project (3GPP)* [23; 24] standardisiert werden, so gibt es nur wenige Ansätze, die diese Methoden zur Nahbereichs- und Trennungserkennung oder für die anderen erwähnten Funktionen anwenden. Beispielsweise beschreibt [120] einen Ansatz, welcher die k nächsten Nachbarn einer Zielperson in zellulären Netzen stetig überwacht und welcher auch von einem zentralisierten Location Server ausgeht. Einen Ansatz zur Nahbereichs- und Trennungserkennung, der sich für Peer-to-Peer sowie für zentrale Umgebungen anwenden lässt, stellt der in [29] vorgeschlagene Streifenalgorithmus dar. In der Peer-to-Peer-Variante werden Ortsinformationen direkt zwischen den Geräten der überwachten Zielpersonen ausgetauscht und Nahbereichs-

und Trennungsereignisse auf deren mobilen Endgeräten festgestellt. Während dieser Ansatz zwar Datenschutzvorteile birgt, weil Ortsinformationen an keinen externen Location Server gelangen, leidet er grundsätzlich unter einem hohen Verbrauch an Übertragungskapazität, weil die Ortsinformationen jeweils zwischen allen möglichen Paaren von Zielpersonen ausgetauscht werden. Eine zentralisierte Version des Streifenalgorithmus wurde den in dieser Arbeit entwickelten Ansätzen zum Vergleich entgegengesetzt. Eine weitere Arbeit, die die Verringerung übertragener Nachrichten zum Ziel hat und von einer zentralisierten Architektur ausgeht, ist [171]. Hier wird automatisch erkannt, wann sich innerhalb einer Gruppe überwachter Zielobjekte die Zusammensetzung der k zueinander nächstgelegenen Paare verändert. Weiterhin beschreiben [74] ein Rahmenwerk zur kontinuierlichen Überwachung räumlicher Anfragen auf beweglichen Objekten, welches ebenfalls die Reduktion von Nachrichten zum Ziel hat. In dem Artikel wird sowohl auf Range-Anfragen als auch auf das Problem der k nächsten Nachbarn eingegangen.

4.2 Problemstellung

Bevor im Folgenden verschiedene Algorithmen zur Nahbereichs- und Trennungserkennung vorgestellt werden, ist es notwendig, das gewünschte Verhalten anhand der Parameter *Trennungsdistanz* d_s, *Nahbereichsdistanz* d_p und der so genannten *Grenzlinientoleranz* b formal zu definieren. Sei $dist(t_i, t_j)$ die aktuelle räumliche Distanz zweier Zielpersonen t_i und t_j, die aus der Menge aller zu beobachtenden Zielobjekte beliebig ausgewählt wurden. Dann soll für eine gegebene Nahbereichsdistanz $d_p > 0$ und eine Grenzlinientoleranz $b > 0$ der Nahbereich bezüglich t_i und t_j anhand der folgenden Regeln überwacht werden.

- Wenn $dist(t_i, t_j) < d_p$, dann *muss* ein Nahbereichsereignis ausgelöst werden.
- Wenn $d_p \leq dist(t_i, t_j) \leq d_p + b$, dann *kann* ein Nahbereichsereignis ausgelöst werden.
- Wenn $dist(t_i, t_j) > d_p + b$, dann *darf kein* Nahbereichsereignis ausgelöst werden.

Wie sich aus diesen Bedingungen ableitet, dient die Grenzlinientoleranz b als Unschärfeintervall, welches zur Nahbereichsdistanz addiert wird. Innerhalb des Intervalls kann ein Nahbereichsereignis ausgelöst werden, dies muss aber nicht geschehen. Ziel ist dabei, übermäßigen Datenaustausch zu verhindern, falls sich $dist(t_i, t_j)$ der Nahbereichsdistanz annähert. Ohne Grenzlinientoleranz wäre es dann nämlich nötig, minimale Positionsveränderungen von t_i und t_j zu beobachten, um den exakten Zeitpunkt der Grenzüberschreitung zu erfassen. Die Konsequenz wäre eine sehr hohe Belastung der Luftschnittstelle, welche zum Nachrichtenaustausch dient, als auch des Location Servers, welcher die übermittelten Nachrichten verarbeitet und die Positionen der Teilnehmer korreliert. Indem b konfigurierbar bleibt, kann solch hohes Nachrichtenaufkommen verhindert bzw. an die Genauigkeitsanforderungen der entsprechenden Anwendung angepasst werden.

Überdies kann durch angemessene Auswahl von b sprunghaftes Systemverhalten, welches durch ungenaue Positionsbestimmungen entsteht, verhindert werden. Sei σ die horizontale Genauigkeit des verwendeten Positionierungsverfahrens bezogen auf ein Konfidenzintervall von 95%. Dann schwankt die wahre Distanz zwischen t_i und t_j um die gemessene

Distanz $dist(t_i, t_j)$ innerhalb des Intervalls $[dist(t_i, t_j) - 2\sigma; dist(t_i, t_j) + 2\sigma]$. Bei relativ exakten Verfahren wie GPS, das eine horizontale Genauigkeit von etwa 15 m aufweist, genügt also eine Grenzlinientoleranz von circa 60 m, um entsprechende Schwankungen auszugleichen. Finden jedoch zellbasierte Positionierungstechniken wie E-OTD oder U-TDOA Verwendung, die Ungenauigkeiten im Bereich von 100 m oder mehr aufweisen, so muss b entsprechend höher gewählt werden.

Die Bedingungen zur Trennungserkennung zwischen t_i und t_j werden ganz analog definiert:

- Wenn $dist(t_i, t_j) > d_s + b$, dann *muss* ein Trennungsereignis ausgelöst werden.
- Wenn $d_s \leq dist(t_i, t_j) \leq d_s + b$, dann *kann* ein Trennungsereignis ausgelöst werden.
- Wenn $dist(t_i, t_j) < d_s$, dann *darf kein* Trennungsereignis ausgelöst werden.

4.3 Strategien zur Nahbereichs- und Trennungserkennung

Im Folgenden werden aufeinander aufbauend verschiedene Strategien zur Nahbereichs- und Trennungserkennung vorgestellt und evaluiert, die allesamt auf den in Abschnitt 2.4 erläuterten Position-Update-Methoden aufsetzen. Es werden zunächst die eigenen Ansätze vorgestellt, nämlich die *periodische Strategie*, basierend auf periodischen Updates, sowie die *statische*, *dynamisch-zentrierte* und *dynamisch-verschobene Kreisstrategie*, welche auf distanz- bzw. zonenbasierte Updates setzen. Daraufhin wird der einzige bekannte Ansatz, der *Streifenalgorithmus*, vorgestellt und mittels Simulationen mit den eigenen Strategien verglichen. Es folgt eine Diskussion der Vor- und Nachteile der unterschiedlichen Strategien.

4.3.1 Periodische Strategie

Ein nahe liegender Ansatz zur Nahbereichserkennung greift auf periodische Updates zurück. Der Location Server konfiguriert hierbei die Endgeräte aller beobachteten Zielpersonen mit einem festen Zeitintervall, anhand dessen ständig alle Positionen übermittelt werden. Ein Problem des Ansatzes liegt in der Bestimmung des passenden Update-Intervalls, welches in Abhängigkeit der Fortbewegungsgeschwindigkeit v der Zielpersonen gewählt werden muss. Entscheidet man sich für einen hohen Wert von v, beispielsweise der Durchschnittsgeschwindigkeit von Fahrzeugen auf Autobahnen, so erhält man eine sehr hohe Anzahl an Position Updates, welche unnötig wären für Zielpersonen, die sich mit niedrigerer Geschwindigkeit bewegen oder die sich längere Zeit stationär verhalten. Setzt man andererseits v relativ niedrig an, so wird der Ansatz fehleranfällig, falls manche Personen die Geschwindigkeit überschreiten. Aus diesem Grund werden periodische Updates an dieser Stelle nicht weiter betrachtet; der Fokus liegt im Folgenden vielmehr auf Strategien, die distanz- und zonenbasierte Updates verwenden.

4.3.2 Statische Kreisstrategie

In einer einfachen Anwendung der distanzbasierten Methode konfiguriert der Location Server die Endgeräte aller beobachteten Zielpersonen t_n mit derselben statischen Update-Distanz, anhand der Position Updates durchgeführt werden. Folglich ist jedes der Objekte von einem virtuellen Kreis – der namensgebend für die Strategie ist – mit Radius r_n umgeben, dessen Mittelpunkt c_n auf der zuletzt berichteten Position liegt. Dies wird in Abbildung 4.1 verdeutlicht.

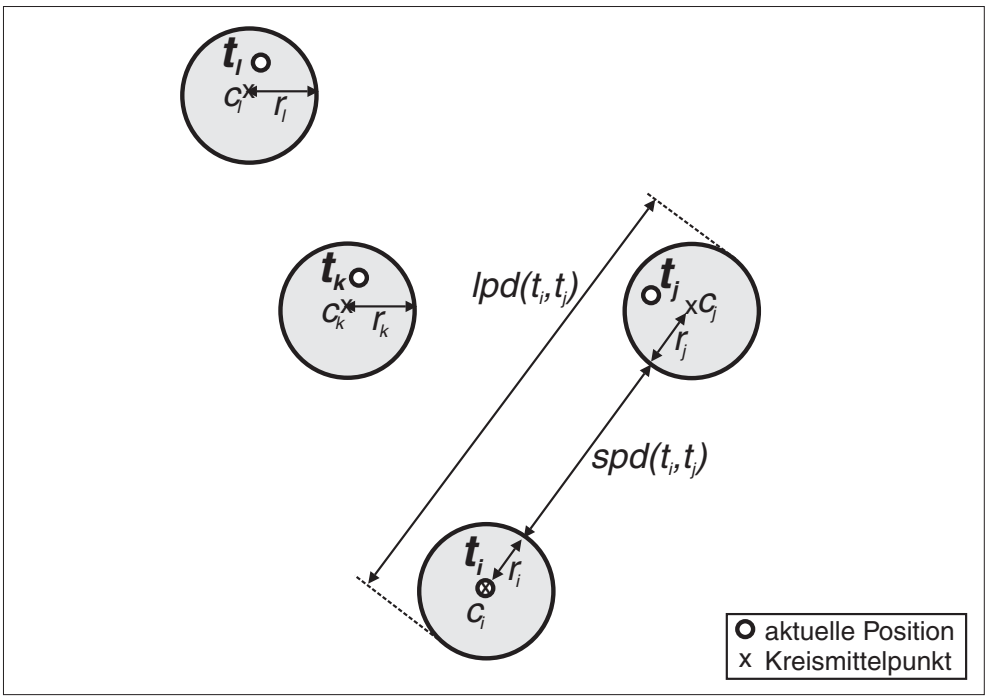

Abbildung 4.1: Statische Kreisstrategie

Verlässt eine Zielperson ihren Kreis, so sendet das mobile Endgerät ein Position Update an den Location Server. Der Kreismittelpunkt wird darauf auf die eben übermittelte Position gelegt und die Positionsbestimmung fortgesetzt. Um zu gewährleisten, dass Nahbereichs- bzw. Trennungsereignisse anhand der zuvor genannten Bedingungen korrekt erkannt werden, muss die Update-Distanz, welche r_n entspricht, auf $\frac{b}{2}$ gesetzt werden. Bei größeren Werten könnten Ereignisse zu spät erkannt werden, das heißt, wenn $dist(t_i, t_j)$ den Wert d_p bereits unterschritten bzw. den Wert $d_s + b$ überschritten hat. Kleinere Werte würden im Gegenzug unnötige Position Updates erzeugen.

Der Location Server korreliert übermittelte Positionen mit den Kreisen der jeweils anderen Zielpersonen, um mögliche Nahbereichs- und Trennungsereignisse zu erkennen. Zu diesem Zweck werden nun die Begriffe der *kleinstmöglichen* und *größtmöglichen* Distanz eingeführt. Die kleinstmögliche Distanz (shortest possible distance) zweier Zielpersonen t_i und t_j wird definiert als $spd(t_i, t_j) = dist(c_i, c_j) - r_i - r_j$ und die größtmögliche Distanz (largest possible distance) als $lpd(t_i, t_j) = dist(c_i, c_j) + r_i + r_j$. Ist die Position p_i von t_i genau bekannt, dann gilt $c_i = p_i$ und $r_i = 0$. Weiterhin bezeichnet $T_{p,i}$ die Menge aller Zielpersonen, für die bislang noch kein Nahbereichsereignis bezüglich t_i ausgelöst wurde.

$T_{s,i}$ steht für die Menge aller Zielpersonen, die bezüglich t_i noch kein Trennungsereignis erzeugt haben.

Anhand dieser Definitionen läuft die Nahbereichserkennung folgendermaßen ab: Bei Erhalt eines Position Updates von t_i überprüft der Location Server für alle $n \in T_{p,i}$, ob $spd(t_i, t_n) < d_p + \frac{b}{2}$ gilt. Trifft dies für kein t_n zu, muss auch nichts unternommen werden. Andernfalls werden die Endgeräte aller t_n nach ihrer Position gefragt, für die die Bedingung zutrifft. Somit stehen dann deren exakte Positionen innerhalb ihrer Kreise fest. Wird beispielsweise die Position von t_j auf diese Weise ermittelt, so kann der Location Server nun t_i und t_j bezüglich einer möglichen Nahbereichsunterschreitung untersuchen. Trifft die Nahbereichsbedingung zu, gilt also $d_p \leq dist(t_i, t_j) \leq d_p + b$, so wird ein Alarm abgesetzt und an die entsprechende Anwendung vermittelt. Zusätzlich verschwindet t_i aus $T_{p,j}$ und t_j aus $T_{p,i}$. Zur Trennungserkennung prüft der Server beim Eintreffen eines Position Updates von t_i auf die Bedingung $lpd(t_i, t_n) > d_s + \frac{b}{2}$. Der restliche Verlauf ist ganz analog.

4.3.3 Dynamisch-zentrierte Kreisstrategie

Ein Nachteil der statischen Kreisstrategie ist die feste Beschränkung der Update-Distanz auf $\frac{b}{2}$. Im Vergleich zur durchschnittlich erwarteten Distanz zwischen mobilen Zielpersonen ist das ein relativ kleiner Wert. Die statische Strategie erzeugt also eine hohe Menge von Position Updates, selbst wenn die betrachteten Personen noch relativ weit davon entfernt sind, Nahbereichs- bzw. Trennungsereignisse auszulösen. Zur Erzielung eines besseren Verhaltens wird daher vorgeschlagen, die Update-Distanz dynamisch und für jede Zielperson einzeln festzulegen, und zwar in Abhängigkeit von der Distanz zum nächsten bzw. am weitesten entfernten Nachbarn. Dabei ist zu erwarten, dass die erzeugten Kreise im Durchschnitt größer werden als in der statischen Strategie, was schließlich zu einer verringerten Anzahl ausgelöster Position Updates führen soll. Der Ansatz bezeichnet sich als *dynamisch-zentrierte Kreisstrategie (Dynamic Centered Circles, DCC)* und wurde ursprünglich in [160] vorgestellt.

Zuerst soll die Festlegung der dynamischen Update-Distanz anhand der Nahbereichserkennung veranschaulicht werden. Einer Zielperson t_i wird vom Location Server jeweils eine neue Distanz zugewiesen, nachdem sie sich entweder erstmalig am Server registriert hat, t_i selbst ein Position Update erzeugt hat oder – in der Folge eines Position Updates einer anderen Zielperson – vom Server nach der Position angefragt wurde. Zur Berechnung der Update-Distanz für Zielperson t_i vergleicht der Server alle kleinstmöglichen Distanzen $spd(t_i, t_n)$ für alle $t_n \in T_{p,i}$ miteinander. Die Update Distanz r_i ergibt sich dann aus $spd(t_i, t_j) - d_p$, wobei t_j diejenige Zielperson ist, deren kleinstmögliche Distanz zu t_i minimal ist oder, mit anderen Worten, deren Kreis der aktuellen Position von t_i am nächsten ist. Die Entfernung der beiden Kreise entspricht dann also der Nahbereichsdistanz d_p, was gewährleistet, dass die Distanz zwischen t_i und t_j den Wert d_p nicht unterschreiten kann, ohne vom Location Server bemerkt zu werden.

Abbildung 4.2 illustriert die DCC-Strategie anhand eines Beispiels. Die Endgeräte der Zielpersonen $t_i, ...t_l$ sind mit individuellen Update-Distanzen konfiguriert, was durch Kreise verschiedener Radien dargestellt ist, deren Mittelpunkt sich jeweils auf der zuletzt berichteten Position befindet. Die Zielpersonen $t_j, ...t_l$ haben inzwischen ihre zuletzt übermittelte Position verlassen und bewegen sich frei innerhalb der Kreise. Das Endgerät von Person t_i hat hingegen aufgrund eines vorangegangenen Position Updates gerade erst einen neu-

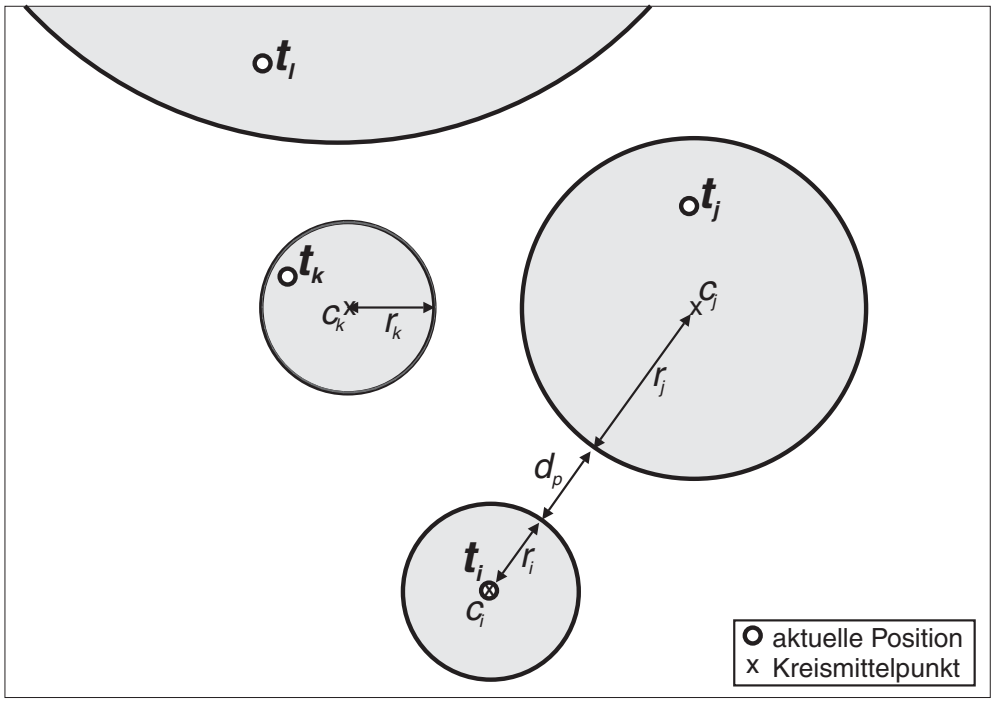

Abbildung 4.2: Dynamisch-zentrierte Kreisstrategie (DCC)

en Kreis erhalten. Die zugrunde liegende Update-Distanz bezieht sich auf t_j, da dessen kleinstmögliche Distanz zu t_i von allen betrachteten Personen minimal ist.

Die Trennungserkennung mittels DCC funktioniert analog. Um die Update-Distanz von Zielperson t_i zu erhalten, berechnet der Location Server alle $lpd(t_i, t_n)$ für alle $t_n \in T_{s,i}$ und wählt die Person t_j aus, deren größtmögliche Distanz maximal ist. Die Update-Distanz r_i wird dann bestimmt durch $d_s + b - lpd(t_i, t_j)$, wodurch feststeht, dass t_i und t_j sich nicht um mehr als $d_s + b$ voneinander wegbewegen können, ohne vom Server bemerkt zu werden.

Überprüfungen bezüglich eventueller Nahbereichs- und Trennungsereignisse geschehen beim Location Server ganz ähnlich wie bei der statischen Kreisstrategie. Ein Unterschied besteht jedoch darin, dass beim DSC-Ansatz beide Zielpersonen t_i und t_j neue Update-Distanzen zugewiesen bekommen, falls nach einem Polling von t_j und dem nachfolgenden Abprüfen der Auslösebedingungen kein Nahbereichs- bzw. Trennungsereignis erkannt werden konnte. Angenommen, es steht dabei anhand der zuletzt übermittelten Positionen fest, dass t_i und t_j sich paarweise am nächsten sind (das heißt, kein Kreis einer dritten Zielperson ist einem der beiden näher), dann bekommen beide dieselbe Update-Distanz zugeteilt. Für die Nahbereichserkennung beträgt diese $\frac{dist(t_i,t_j)-d_p}{2}$ und für die Trennungserkennung $\frac{d_s+b-dist(t_i,t_j)}{2}$.

4.3.4 Dynamisch-verschobene Kreisstrategie

Um die zugewiesenen Kreise im Durchschnitt noch mehr zu vergrößern und so die Anzahl übertragener Position Updates zu reduzieren, wurde die *dynamisch-verschobene Kreisstrategie (Dynamic Shifted Circles, DSC)* entwickelt, die auch in [97] beschrieben ist. Im Gegen-

satz zur DCC-Strategie entspricht hier der Kreismittelpunkt nicht der zuletzt übermittelten Position einer Zielperson. Vielmehr wird der Kreis von dieser Stelle aus um einen gewissen Betrag verschoben, was zur Folge hat, dass der Kreisradius vergrößert werden kann. Nachdem in diesem Ansatz der Kreismittelpunkt nicht mit dem letzten Position Update übereinstimmt, handelt es sich auch nicht um distanzbasierte Updates, sondern zonenbasierte. Die Update-Zone entspricht dabei einem Kreis, dessen Mittelpunkt durch c_i und dessen Radius durch r_i bestimmt ist. Position Updates werden jeweils bei Verlassen der Zone ausgelöst.

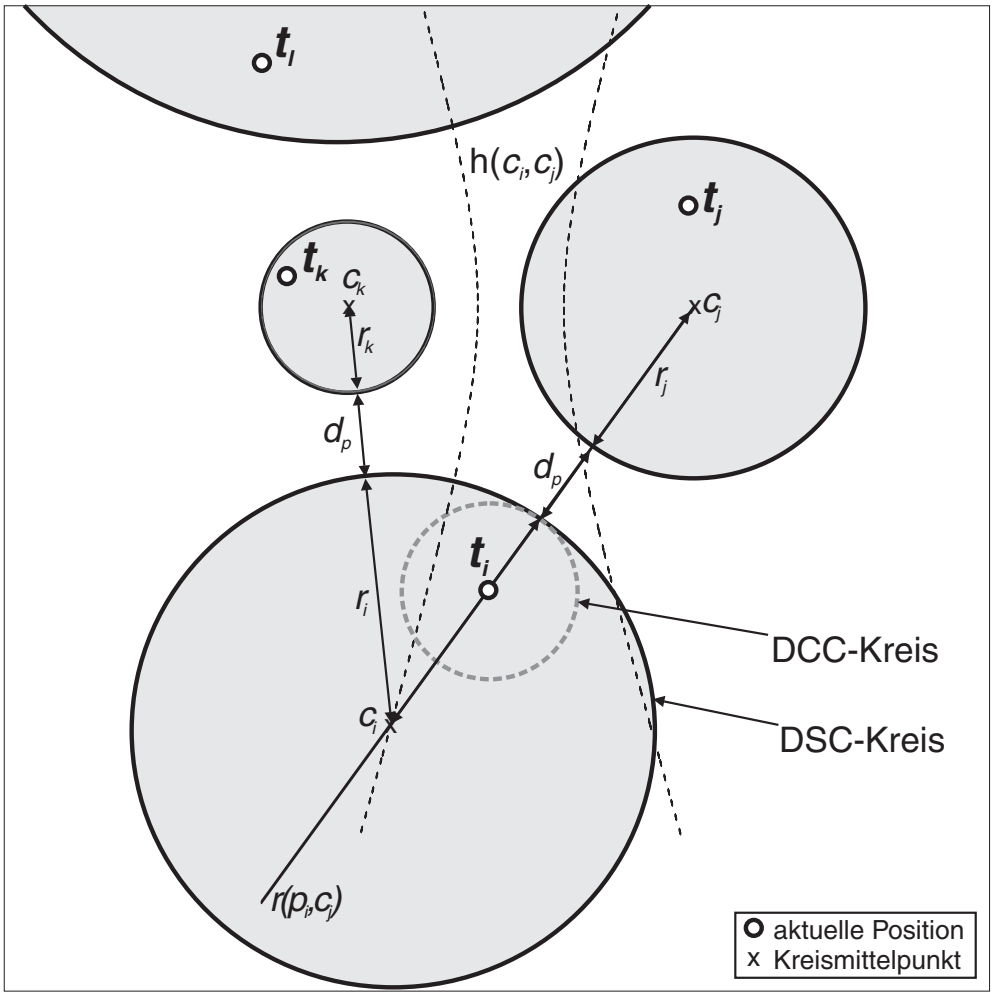

Abbildung 4.3: Dynamisch-verschobene Kreisstrategie (DSC)

Abbildung 4.3 veranschaulicht die Anwendung verschobener Kreise für die Nahbereichserkennung wieder anhand des Szenarios aus Abbildung 4.2. Man sieht, wie der gestrichelte Kreis um t_i herum, der in der DCC-Strategie zugeteilt wurde, nun durch einen größeren, mit durchgezogener Linie gezeichneten Kreis ersetzt wird, der der DSC-Strategie entspricht. Die Erzeugung eines solchen verschobenen Kreises für Zielperson t_i wird zuerst für die Nahbereichserkennung behandelt. Sie richtet sich nach den folgenden Vorgaben:

1. Der zu erzeugende Kreis (im Folgenden als neuer Kreis bezeichnet) muss den zen-

trierten Kreis, der von der DCC Strategie erstellt würde, vollkommen überdecken. So kann gewährleistet werden, dass die DSC-Strategie für beliebige Bewegungsverhalten der Zielperson nicht mehr Position Updates bewirkt, als es bei DCC der Fall wäre.

2. Analog zur DCC-Strategie muss der neue Kreis in Abhängigkeit des Kreises der Zielperson t_j ausgerichtet sein, für welche die kleinstmögliche Distanz zu t_i minimal ist. Gleichzeitig richtet sich die Ausrichtung nach dem Kreis einer zweiten Person t_k, welche noch genauer spezifiziert werden muss. Der neue Kreis muss nun von diesen beiden Kreisen exakt um die Nahbereichsdistanz d_p entfernt sein. Kleinere Abstände würden die Bedingungen der Nahbereichserkennung verletzen, während größere wiederum den Bemühungen widersprächen, den neuen Kreis möglichst groß zu machen.

3. Die Distanz zwischen dem neuen Kreis und dem aller verbleibenden Zielpersonen aus der Menge $T_{p,i}$ muss mindestens d_p betragen, um ebenfalls nicht mit den Bedingungen der Nahbereichserkennung zu kollidieren.

Im ersten Schritt wird der Mittelpunkt des neuen Kreises auf eine Gerade $r(p_i, c_j)$ angenähert, die die aktuelle Position p_i von t_i mit dem Kreismittelpunkt c_j von t_j verbindet, vergleiche Abbildung 4.3. Obwohl die Annäherung durch die Gerade alleine nicht gewährleistet, dass Bedingung (1) erfüllt ist, so stellt sie doch eine wichtige Voraussetzung dar. Man sieht anhand der Abbildung, dass bei einem Verschieben von c_i entlang von $r(p_i, c_j)$ nach links unten bei konstantem Abstand d_p der beiden Kreise (also sich vergrößerndem r_i) Bedingung (1) erhalten bleibt.

Im nächsten Schritt werden nun Mittelpunkt und Radius des neuen, verschobenen Kreises so bestimmt, dass Bedingung (2) erfüllt wird. Veranschaulicht wird dies in Abbildung 4.3, wo der Kreis von t_i von dem von t_j sowie dem einer weiteren Zielperson t_k exakt den Abstand der Nahbereichsdistanz d_p hat. Zum generellen Verständnis der Bestimmung des Kreismittelpunkts c_i ist es hilfreich, sich über den folgenden Zusammenhang bewusst zu werden:

$$\begin{aligned} & dist(c_i, c_j) - r_j = dist(c_i, c_k) - r_k \\ \Leftrightarrow \quad & dist(c_i, c_j) - dist(c_i, c_k) = r_j - r_k \end{aligned}$$

Mit anderen Worten ausgedrückt bedeutet dies, dass der Unterschied zwischen den Distanzen von c_i zu jeweils einem der beiden anderen Kreismittelpunkte genauso groß ist wie der Unterschied der Radien der beiden Kreise. c_i befindet sich also auf einer Hyperbel. Allgemein ist eine Hyperbel definiert als die Menge aller Punkte der Zeichenebene, für die die Differenz der Abstände zu zwei gegebenen Punkten, den so genannten Brennpunkten F_1 und F_2, gleich einer Konstante κ ist. Diese Eigenschaft lässt sich zur Bestimmung von c_i folgendermaßen ausnutzen: Wird F_1 gleich c_j gesetzt, F_2 gleich c_k und κ gleich $|r_j - r_k|$, dann muss c_i auf der resultierenden Hyperbel $h(c_j, c_k)$ liegen, vergleiche Abbildung 4.3. Die endgültige Position von c_i ergibt sich dann durch einen der beiden Schnittpunkte von $h(c_j, c_k)$ und $r(p_i, c_j)$. Einer dieser Schnittpunkte lässt sich einfach ausschließen, da dieser die Nahbereichsbedingung zwischen den Kreisen verletzen bzw. sich überschneidende Kreise erzeugen würde.

Die zweite Zielperson t_k, welche zur Bestimmung von c_i benötigt wird, stellt sich heraus durch die Betrachtung aller möglichen Hyperbeln $h(c_j, c_n)$ mit $j \neq n$ und $n \in T_{p,i}$. Falls sich von diesen Hyperbeln mehr als eine mit der Geraden $r(p_i, c_j)$ schneidet, wird derjenige Schnittpunkt als c_i gewählt, der am nächsten zur aktuellen Position p_i von t_i liegt (wodurch Bedingung (3) erfüllt wird). Wenn in manchen Fällen kein solcher Schnittpunkt gefunden wird, dann darf der Kreismittelpunkt entlang der Geraden so lange verschoben werden, bis ein vorher zu definierender Maximalradius erreicht ist. Dieser Fall tritt insbesondere dann ein, wenn sich nur zwei Zielobjekte auf dem Feld befinden. Die beiden entstehenden Kreise können dann beliebig groß werden, und der zwischen ihnen entstehende Freiraum nähert sich – in Abhängigkeit des Maximalradius – mehr oder weniger demselben Streifen an, der auch vom Streifenalgorithmus (vergleiche nächster Abschnitt) erzeugt würde.

Für die Trennungserkennung verbindet die Gerade $r(p_i, c_j)$ ebenfalls die aktuelle Position p_i mit dem Kreismittelpunkt c_j, wobei nun – wieder analog zum DCC-Algorithmus – t_j diejenige Zielperson ist, für die die größtmögliche Distanz maximal ist. c_i wird auf der Geraden in die zur Nahbereichserkennung entgegengesetzte Richtung verschoben, und zwar zwischen p_i und c_j. Ausgehend von der Position p_i, die bei der DCC-Strategie als Mittelpunkt gewählt würde, lässt sich also c_i im besten Fall mit der Position c_j gleichsetzen. In diesem Fall ergibt sich der Kreisradius r_i durch $d_s + b - r_j$. Für den allgemeinen Fall erhält man r_i durch $d_s + b - r_j - dist(c_i, c_j)$. Anders als bei der Nahbereichserkennung können die Kreise für die Trennungserkennung also nicht beliebig groß werden. Zur exakten Bestimmung von c_i müssen wiederum alle möglichen Hyperbeln $h(c_j, c_k)$ mit $j \neq k$ und $k \in T_{s,i}$ mit der Geraden $r(p_i, c_j)$ geschnitten werden. Der restriktivste, also am nächsten bei p_i liegende Schnittpunkt zwischen p_i und c_j wird ausgewählt. Lässt sich kein solcher Schnittpunkt finden, wird c_i auf c_j gesetzt. Bedingung (2) ändert sich dahingehend, dass die Kreise von t_j und t_k nicht um exakt d_p von dem von t_i entfernt liegen müssen, sondern dass sie jeweils bezüglich t_i eine größtmögliche Distanz von genau $d_s + b$ zulassen. Nachdem r_i unbekannt ist, basiert die Hyperbeldefinition also nun auf folgender Umformung:

$$\begin{aligned} & dist(c_i, c_j) + r_j = dist(c_i, c_k) + r_k \\ \Leftrightarrow \quad & dist(c_i, c_j) - dist(c_i, c_k) = r_j + r_k \end{aligned}$$

Der Brennpunkt F_1 wird also weiterhin gleich c_j gesetzt und F_2 gleich c_k. Die Konstante κ ergibt sich jetzt aus $|r_j + r_k|$. Bedingung (3) bezieht sich nun auf die Trennungserkennung.

4.3.5 Streifenalgorithmus

Eine existierende Strategie zur Nahbereichs- und Trennungserkennung wurde in [29] entwickelt. Die Kernidee zur Nahbereichserkennung besteht darin, jeweils paarweise zwischen allen beobachteten Zielobjekten t_i und t_j einen mittig ausgerichteten Streifen $s(t_i, t_j)$ zu bestimmen. Betritt eine der Zielpersonen den Streifen, so wird ein Position Update ausgelöst. Der Ansatz bedient sich folglich zonenbasierter Updates. Dabei definiert die im entsprechenden Position Update Request übermittelte Zone den Streifen. Dieser hat eine unendlich längliche Ausdehnung und eine Breite d_p. Letztere Eigenschaft gewährleistet, dass sich t_i und t_j nicht unter die Nahbereichsdistanz annähern können, ohne dass dies vom Location Server bemerkt wird. Wie in Abbildung 4.4 dargestellt, befindet sich der Streifen zwischen

den Positionen p_i von t_i und p_j von t_j, die zum Zeitpunkt der Streifenerstellung übermittelt wurden. Dabei ist die Mitte des Streifens identisch mit der Mittelsenkrechten der Verbindungsstrecke zwischen p_i und p_j. Wie man ebenfalls anhand der Abbildung sieht, wird für mehr als zwei Zielpersonen auf dem Feld jedes Endgerät von einem Polygon bzw. Polygonzug umschlossen, welcher sich durch die Überschneidungen der jeweils zugeteilten Streifen ergibt.

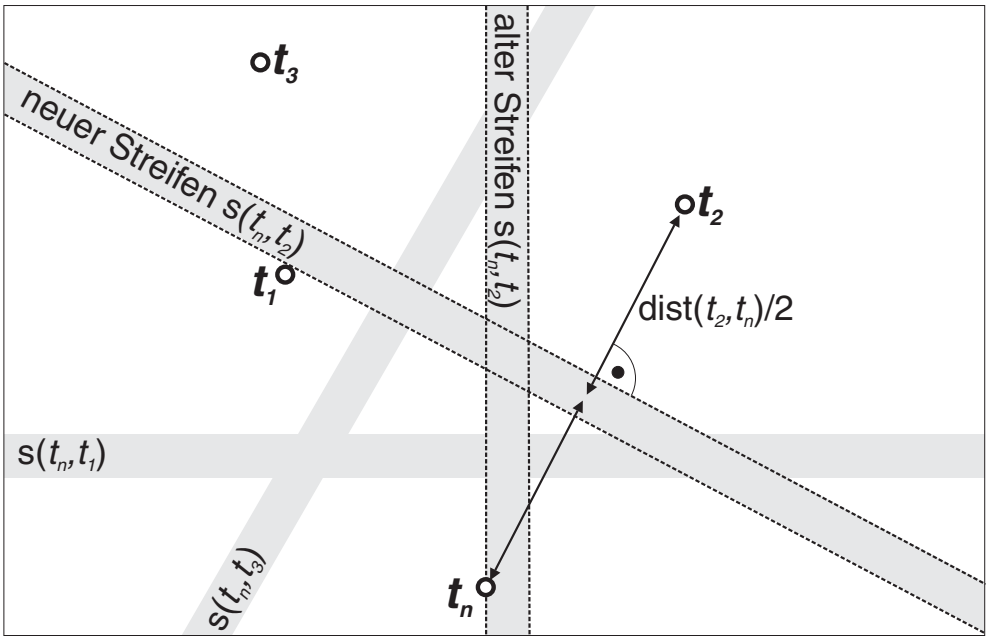

Abbildung 4.4: Streifenalgorithmus

Betritt t_i seinen Streifen bezüglich t_j und schickt ein entsprechendes Position Update an den Server, muss dieser t_j nach dessen genauer Position anfragen und die Nahbereichsbedingung überprüfen. Ist die Bedingung nicht erfüllt, so wird anhand der ermittelten Positionen ein neuer Streifen berechnet und an t_i und t_j versendet. Andernfalls wird ein Nahbereichsereignis ausgelöst. Im Gegensatz zu den vorgestellten eigenen Strategien wird eine Zielperson also immer nach der Position angefragt, sobald der entsprechende Gegenpart ein Position Update abgesetzt hat.

Die Trennungserkennung funktioniert analog. Anstelle von Streifen wird einem Paar von Zielpersonen t_i und t_j jeweils ein Kreis mit Radius $r_i = r_j = \frac{d_s+b}{2}$ zugewiesen, wobei sich der Mittelpunkt $c_i = c_j$ wieder genau in der Mitte zwischen p_i und p_j befindet. Ein Position Update wird versendet, sobald einer der beiden den Kreis verlässt, was wiederum garantiert, dass die Distanz der beiden Personen den Wert $d_s + b$ nicht überschreiten kann, ohne vom Server bemerkt zu werden.

4.3.6 Diskussion

Bevor die verschiedenen Strategien bezüglich der über die Luftschnittstelle ausgetauschten Nachrichten untersucht und gegenübergestellt werden, sind noch die folgenden grundlegenden Betrachtungen von Bedeutung.

Die Anzahl der Streifen bzw. Kreise, die beim Streifenalgorithmus pro Gerät verwaltet werden müssen, steigt linear an mit der Anzahl von Zielpersonen, da jeweils ein Streifen bzw. Kreis pro Zielperson betrachtet wird. Im Gegensatz dazu ist die Anzahl gleichzeitig zu verwaltender Kreise bei den eigenen Strategien unabhängig von der Anzahl an Zielpersonen, da die zugrunde liegenden Update-Distanzen bzw. Update-Zonen entweder statisch gewählt sind oder in Abhängigkeit der nächsten bzw. am weitesten entfernten Nachbarn bestimmt werden. In Bezug auf dieses Problem wird allerdings von [29], die den ursprünglichen Streifenalgorithmus für Peer-to-Peer-Umgebungen, also ohne Einbezug eines zentralen Location Servers, konzipiert haben, eine Optimierung ihres Ansatz vorgeschlagen.

Ein Vorteil der distanzbasierten Strategien, also der statischen Kreisstrategie sowie der dynamisch-zentrierten Kreisstrategie, ist, dass sie ohne weitere Anpassung auch auf nichtkartesische Koordinatensysteme anwendbar sind. Beide Strategien benötigen lediglich eine Funktion zur Berechnung der räumlichen Distanz zweier Punkte, sind also beispielsweise auch leicht für geographische Koordinatensysteme wie WGS84 benutzbar. Ist auf dem Location Server und dem Endgerät entsprechendes Kartenmaterial vorhanden, so können die distanzbasierten Strategien auch für topologische Raumbeschreibungen angewendet werden: Die Update-Distanz, welche auf dem Endgerät mit der bereits zurückgelegten Distanz verglichen wird, wie auch die auf dem Server berechneten kleinst- und größtmöglichen Distanzen der einzelnen Zielpersonen entsprechen dabei topologischen Distanzen, beziehen sich also auf Pfadlängen innerhalb des durch die Karte vorgegebenen Graphen.

Die dynamisch-verschobene Strategie und der Streifenalgorithmus beruhen, wie oben beschrieben, auf Eigenschaften, die für kartesische Koordinatensysteme spezifisch sind. Für eine allgemeine Anpassung an geographische Koordinatensysteme bzw. zur Erkennung topologischer Distanzen wären weitere Betrachtungen erforderlich, die an dieser Stelle jedoch nicht ausgeführt werden. Wirklich korrekt (bezüglich eines vorgegebenen maximalen Fehlers) funktionieren diese Strategien also in der existierenden Form im Zusammenhang mit echt gemessenen geographischen Positionsdaten nur jeweils innerhalb eines vorher bestimmten Ausschnittes einer auf die geographischen Koordinaten angewandten (kartesischen) Kartenprojektion, zum Beispiel in das *Universal Transverse Mercator (UTM)*-System. Die Größe dieses Ausschnitts hätte im Falle der DSC-Strategie auch Auswirkungen auf den Maximalradius eines Kreises. Denn dieser könnte ja theoretisch unendlich groß werden und würde ohne Beschränkung die Grenzen des Ausschnitts überschreiten.

4.4 Evaluierung

Die praktische Anwendbarkeit der präsentierten Strategien zur Nahbereichs- und Trennungserkennung soll anhand eines ortsbezogenen Community-Dienstes belegt werden, der seine Benutzer proaktiv benachrichtigt, sobald andere Community-Mitglieder in die Nähe kommen. Im Fokus des Dienstes liegt also die Erkennung von Nahbereichsereignissen, um entsprechende Alarme auszulösen. Gleichwohl ist die Trennungserkennung vonnöten, nämlich um Paare von Zielpersonen, für die bereits ein Nahbereichsereignis erkannt wurde und die sich nun wieder voneinander entfernen, beizeiten für die Nahbereichserkennung neu zu initialisieren. Zuerst wurden dem Dienstszenario entsprechende Simulationen durchgeführt, die die Leistungsmerkmale der Strategien bezüglich der über die Luftschnittstelle ausgetauschten Nachrichten ermitteln sollen. Daraufhin wurde ein Prototyp, der auch in Abschnitt

7.3.5 beschrieben wird, umgesetzt und getestet. Beides wird in den nächsten zwei Abschnitten dargestellt.

4.4.1 Simulation

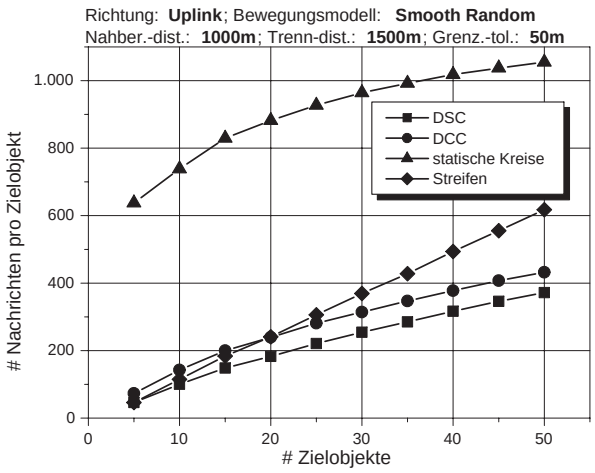

Abbildung 4.5: Uplink-Nachrichten pro Zielobjekt bei Smooth Random

Die Simulationen wurden mit Hilfe eines selbst entwickelten Rahmenwerks umgesetzt, welches in Abschnitt 7.3.3 genauer beschrieben ist. Für das anvisierte Szenario werden Zielpersonen auf einem Spielfeld der Größe 10 km $\times$ 10 km bewegt und die vorgeschlagenen Strategien ausgeführt. Die Bewegungen richten sich sowohl nach künstlichen als auch realen Bewegungsmustern. Für Erstere wurden unterschiedliche, in der Literatur beschriebene Modelle untersucht und schließlich das *Smooth-Random*-Bewegungsmuster [38] ausgewählt. Zum Erhalt echter Bewegungsdaten wurden mehrere Studenten mit GPS-Empfängern und *Personal Digital Assistants (PDAs)* ausgestattet und dazu angehalten, sich in München und Umgebung frei zu bewegen. Daraus ergaben sich 69 so genannte *Traces*, die sowohl fußgängerisches Verhalten als auch Fahrtwege von Autos dokumentieren. Die zeitliche Dauer der Traces variiert zwischen mehreren Minuten und einigen Stunden, wobei gesammelte GPS-Positionsdaten im Sekundentakt aufkommen. Vor der Einspeisung in die Simulation bzw. vor der dauerhaften Abspeicherung wurden die Traces selbstverständlich anonymisiert. Alle Traces wurden zum Zweck der Simulation mit demselben Startzeitpunkt versehen und bis zum Ende des Vorgangs abwechselnd vorwärts und rückwärts abgespielt.

Wie in Abschnitt 2.4 beschrieben, werden drei Operationen zur Realisierung der Strategien benötigt: Position Update Requests zur Konfiguration der jeweiligen Position-Update-Methode, Position Updates und Pollings. In Abhängigkeit der verwendeten Strategie verursacht jede der Operationen eine oder mehrere Nachrichten, die über die Luftschnittstelle übertragen werden, wofür zelluläre Trägerdienste wie GPRS oder UMTS in Frage kommen. Dabei lässt sich zwischen Nachrichten unterscheiden, die im Uplink, also vom Endgerät zum Location Server, oder im Downlink, also vom Server zum Endgerät, übertragen wer-

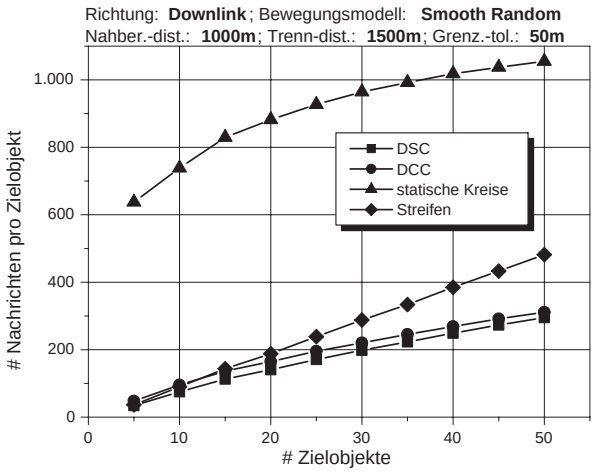

Abbildung 4.6: Downlink-Nachrichten pro Zielobjekt bei Smooth Random

den. Die beim mobilen Nutzer für die Datenübertragung entstehenden Kosten richten sich typischerweise nur nach dem übertragenen Datenvolumen und werden nicht nach Uplink und Downlink unterschieden. Von einer technischen Warte aus betrachtet, ist es jedoch hilfreich, die Übertragungsrichtungen getrennt zu untersuchen. Für den Transfer von Downlink-Nachrichten ist es beispielsweise nötig, das Zielgerät innerhalb seiner aktuellen Location bzw. Routing Area durch Paging auszurufen, was wiederum erhöhten Signalisierungsaufwand bedeutet, der im Uplink nicht notwendig ist. Das Senden von Nachrichten im Uplink belastet wiederum die Batterie des Endgeräts mehr als das Empfangen im Downlink. Abbildungen 4.5 bis 4.8 stellen daher die Anzahl übertragener Nachrichten im Uplink und Downlink in jeweils getrennten Diagrammen dar.

Tabelle 4.1 bietet einen Überblick auf die Anzahl an Nachrichten, die mit jeder Operation verbunden sind. Abgesehen von der statischen Kreisstrategie wird angenommen, dass Position Updates, die ja im Uplink übertragen werden, jeweils durch die Übertragung eines neuen Position Update Requests im Downlink bestätigt werden (Position Updates erzeugen also keine Nachrichten im Downlink). Der Erhalt neuer Position Update Requests muss im Gegenzug vom Endgerät bestätigt werden. Wird die statische Kreisstrategie benutzt, so müssen keine Position Update Requests an das Gerät versendet werden (weil die Update-Distanz statisch festgelegt ist). Folglich müssen hierbei Position Updates vom Server explizit bestätigt werden. Bei allen Strategien besteht die Polling Operation aus einem Polling Request im Downlink und einer Polling Response im Uplink.

Abbildungen 4.5 bis 4.8 stellen die Ergebnisse der Simulationen in Abhängigkeit der Anzahl am Community-Dienst teilnehmender Zielpersonen dar. Die Simulationen wurden mit einer Nahbereichsdistanz von 1000 m, einer Trennungsdistanz von 1500 m und einer Grenzlinientoleranz von 50 m konfiguriert. Die simulierte Zeit betrug zwei Stunden, während der

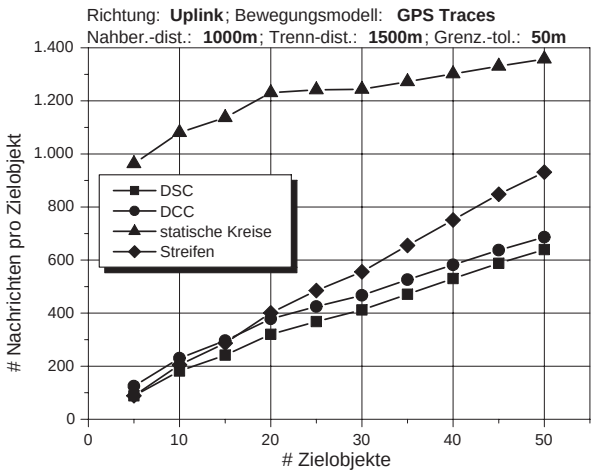

Abbildung 4.7: Uplink-Nachrichten pro Zielobjekt bei den GPS Traces

Operation	**Uplink**	**Downlink**
Position Update	1	0 (1)*
Position Update Request	1	1
Polling	1	1

*statische Kreisstrategie

Tabelle 4.1: Anzahl von Uplink- und Downlink-Nachrichten pro Operation

alle Zielpersonen kontinuierlich bewegt wurden. Ruhezeiten wurden also explizit vermieden. Aus diesem Grund erscheint die Gesamtanzahl von Nachrichten pro Zielperson relativ hoch. Realistisch ist jedoch, dass Personen oft für längere Zeitabschnitte ihren Aufenthaltsort nicht verändern, beispielsweise am Arbeitsplatz oder zu Hause. Dabei ist dann insgesamt von deutlich weniger übertragenen Nachrichten auszugehen.

Zunächst fällt auf, dass die Ergebnisse, die auf dem künstlichen Bewegungsmuster basieren (Abbildungen 4.5 und 4.6), denjenigen, die mit Hilfe von GPS Traces erzeugt wurden (Abbildungen 4.7 und 4.8), sehr ähnlich sehen. Obwohl sich beide Ansätze in der Gesamtanzahl ausgetauschter Nachrichten unterscheiden (was darauf zurückführbar ist, dass sich die durch GPS simulierten Objekte schneller bewegten als die künstlich gesteuerten), bleibt das Verhältnis zwischen den jeweils entstehenden Nachrichtenmengen relativ konstant. Dies bestätigt nicht nur die gute Eignung des Smooth-Random-Bewegungsmusters, sondern betont auch die Relevanz der gezeigten Ergebnisse für realistische Szenarien.

Nicht überraschend ist, dass für alle getesteten Strategien die Anzahl ausgetauschter Nachrichten pro Zielperson mit steigender Anzahl überwachter Personen zunimmt. Dies lässt sich dadurch begründen, dass die Dichte der Zielpersonen zunimmt. Die Spielfeldfläche bleibt gleich groß, also muss die Anzahl von Nahbereichs- und Trennungsereignissen zwangsläufig zunehmen.

Alle der Diagramme 4.5 bis 4.8 weisen klar auf das schlechte Abschneiden der statischen Kreisstrategie hin. Die Zahl übermittelter Nachrichten beträgt sowohl im Uplink als auch

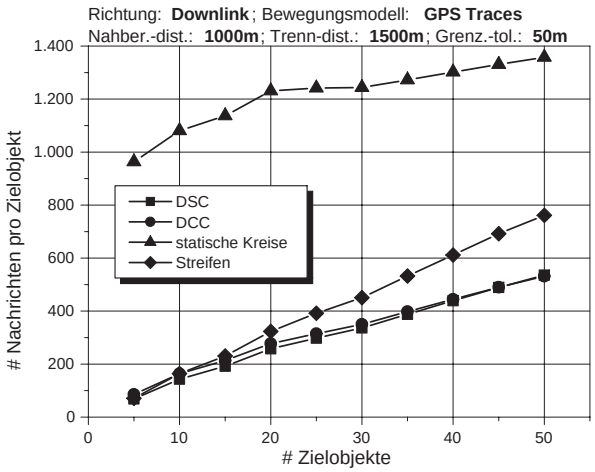

Abbildung 4.8: Downlink-Nachrichten pro Zielobjekt bei den GPS Traces

im Downlink ein Vielfaches von dem, was die anderen Strategien ausmachen. Wie sich aus den Abbildungen 4.9 bis 4.11 ableiten lässt, entstammt der Hauptteil der Nachrichten aus Position-Update-Operationen. Diese treten sehr häufig auf, da die Update-Distanz gleich $\frac{b}{2}$, also relativ klein ist. Die statische Kreisstrategie benutzt als einzige keine Position Update Requests. Es lässt sich also der Schluss ziehen, dass sich der zusätzliche Aufwand, der durch die dynamische Konfiguration von Update-Distanzen und Update-Zonen entsteht, wie es die anderen Strategien vorsehen, durchaus lohnt. Der Kommunikationsaufwand scheint sich dadurch insgesamt zu verringern.

Einzuräumen ist allerdings, dass die statische Kreisstrategie die anderen, dynamischen Strategien schlägt, wenn die Grenzlinientoleranz im Verhältnis zur Dichte der Zielpersonen relativ groß gewählt wird. Dieser Zusammenhang zeigt sich in Abbildung 4.12, welche die Gesamtanzahl ausgetauschter Nachrichten für ein Szenario mit 25 sich auf dem Spielfeld befindlicher Zielpersonen, einer Nahbereichsdistanz von 1000 m und einer Trennungsdistanz von 1500 m darstellt. Während die dynamischen Strategien relativ unabhängig von der Grenzlinientoleranz sind – die Nachrichtenmenge bleibt beinahe konstant –, nimmt der Nachrichtenaustausch bei der statischen Kreisstrategie mit zunehmender Grenzlinientoleranz ab und unterbietet die anderen Strategien für Werte zwischen 200 m und 300 m.

Dies liegt darin begründet, dass ab dieser Schwelle die durchschnittliche Häufigkeit von Position Updates gering genug ist, so dass sich der zusätzliche Aufwand zum Übertragen von Position Update Requests, wie es die anderen Strategien vorsehen, nicht mehr lohnt. Der Schwellwert der Grenzlinientoleranz, ab dem dieser Wechsel auftritt, hängt von der Dichte der Zielpersonen ab. Allgemein lässt sich aussagen, dass er umso kleiner ist, je höher die Dichte der Zielpersonen ist.

Vergleicht man die unterschiedlichen dynamischen Strategien miteinander, so zeigt sich für eine geringe Anzahl an Zielpersonen ein relativ ähnlicher Nachrichtenaufwand, vergleiche auch Abbildungen 4.5 bis 4.8. Erhöht man die Anzahl jedoch, so schneiden die DCC- und DSC-Strategien deutlich besser ab als der Streifenalgorithmus, für den die Kurve

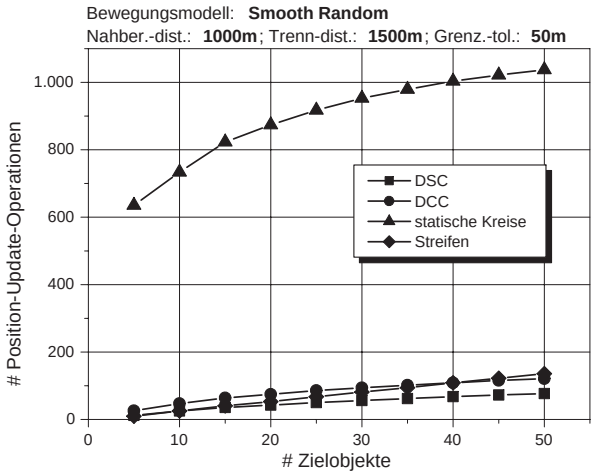

Abbildung 4.9: Position Updates pro Zielobjekt bei Smooth Random

schneller ansteigt als bei den anderen Strategien. Abbildungen 4.9 bis 4.11 lassen Rückschlüsse auf die Ursache zu: Während sich alle dynamischen Strategien nur leicht in der Anzahl durchgeführter Position Updates unterscheiden, erzeugt der Streifenalgorithmus wesentlich mehr Pollings und Position Update Requests. Dieser Umstand lässt sich dadurch erklären, dass bei den Streifen das Auslösen eines Position Updates durch eine Zielperson jeweils das Anfragen (Polling) des entsprechenden Gegenparts zur Folge hat. Schließlich muss (falls kein Nahbereichsereignis erkannt wurde) genau in der Mitte zwischen den beiden Objekten ein neuer Streifen definiert werden, wozu beide exakten Positionen benötigt werden. Auf der anderen Seite werden bei den DSC- und DCC-Strategien Zielpersonen nur dann angefragt, wenn sich die Zielperson, welche ihren Kreis verlassen hat, an den Kreis einer anderen Person hinreichend angenähert bzw. von ihm entfernt hat. Dieser Umstand führt also dazu, dass die eigenen Strategien mit steigender Anzahl an Zielpersonen besser abschneiden als der existierende Streifenalgorithmus.

Abbildungen 4.9 bis 4.11 zeigen außerdem, warum die verschobenen Kreise der DSC-Strategie im Vergleich zur DCC-Strategie zu einer weiteren Verringerung des Nachrichtenaufkommens führen. Dynamisch-verschobene Kreise sind im Durchschnitt größer als die Kreise der dynamisch-zentrierten Strategie, und so werden weniger Position Updates und Position Update Requests erzeugt. Gleichzeitig fällt jedoch auf, dass die DCC-Strategie mit weniger Pollings auskommt. Nachdem die DCC-Kreise im Mittel kleiner sind als die der DSC-Strategie, nehmen zu ihnen im Mittel auch weniger häufig andere Zielpersonen kleinst- bzw. größtmögliche Abstände ein, die Pollings zur Folge haben.

Schließlich zeigt Abbildung 4.13, wie sich die Anzahl ausgetauschter Nachrichten bei allen Strategien in Abhängigkeit der Nahbereichs- und Trennungsdistanz verhält. Vorausgesetzt werden hier 25 beobachtete Zielpersonen sowie eine Grenzlinientoleranz von 25 m. Die Trennungsdistanz wird jeweils als $1\frac{1}{2}$-faches der Nahbereichsdistanz angesetzt. Bei allen Strategien steigt die Anzahl ausgetauschter Nachrichten mit größer werdender Nahbereichs- und Trennungsdistanz an, was darauf zurückgeführt werden kann, dass sich die Anzahl von

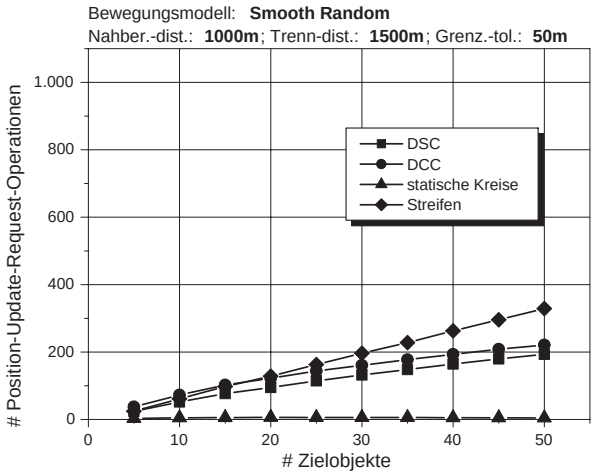

Abbildung 4.10: Position Update Requests pro Zielobjekt bei Smooth Random

Nahbereichs- und Trennungsereignissen bei größeren Distanzen erhöht. Wie vorher liegen die Ergebnisse der dynamischen Strategien sehr nahe beisammen, wobei die DSC-Strategie wiederum am besten abschneidet.

4.4.2 Prototyp

Client-seitig läuft die prototypische Implementierung des Community-Dienstes auf allen Endgeräten, die den in Abschnitt 7.3.1 beschriebenen TraX-Client ausführen können. Dieser ist unter anderem als so genanntes *Midlet* für die *Java 2 Platform, Micro Edition (J2ME)* und als *.NET-Anwendung* umgesetzt worden. Systemvoraussetzungen sind also eine mögliche Selbstortung des Endgeräts, was beispielsweise durch Kopplung an einen externen GPS-Empfänger mittels Bluetooth erreicht werden kann, sowie die Unterstützung der vom TraX-Client benötigten J2ME- bzw. .NET-Umgebung. Die im Folgenden beschriebenen Tests wurden basierend auf J2ME auf einem Mobiltelefon des Typs Siemens/BenQ SXG-75 ausgeführt, welches bereits über ein integriertes GPS-Modul verfügt. Die Funktionen der so genannten *Positionsübermittlungsschicht* (vergleiche Abschnitt 7.2), die die Position-Update-Methoden realisiert, werden vom TraX-Client bereitgestellt. Dies betrifft also die Verarbeitung und das Empfangen von Position Update Requests, die Übermittlung von Position Updates sowie die Beantwortung von Pollings. Der TraX-Server (vergleiche Abschnitt 7.3.2), welcher als Anwendung für die *Java Platform, Standard Edition (J2SE)* umgesetzt wurde, übernimmt die Rolle des Location Servers. Entsprechend den vorgestellten Strategien nimmt er also Position Updates entgegen und konfiguriert die Position-Update-Methoden der verbundenen Geräte dynamisch durch Position Update Requests. Die Verbindung zwischen TraX-Client und TraX-Server wurde im Versuch mittels GPRS über ein öffentliches Mobilfunknetz hergestellt.

Die prototypische Evaluierung umfasste einen Feldtest mit sechs Zielpersonen, die sich zu Fuß durch den Englischen Garten in München bewegten. Nach dem Ausprobieren ver-

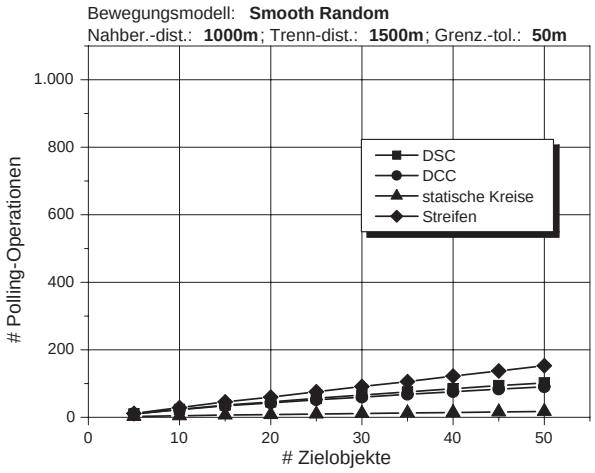

Abbildung 4.11: Pollings pro Zielobjekt bei Smooth Random

schiedener Konfigurationen wurden für die Nahbereichsdistanz 50 m und für die Trennungsdistanz 100 m als für den Test am sinnvollsten erachtet. Die Grenzlinientoleranz wurde auf 25 m angesetzt.

Der Feldversuch ergab, dass die DCC- und die DSC-Strategie korrekt funktionieren und somit geeignet sind, die gewünschten Nahbereichs- und Trennungsereignisse zwischen mobilen Zielpersonen zu erkennen. Es zeigte sich auch, dass die DSC-Strategie die dynamisch-zentrierten Kreise bezüglich zwischen Server und Client transferierter Nachrichten leicht unterbietet. Wie bereits angesprochen, erforderte die DSC-Strategie jedoch einen höheren Implementierungs- und Rechenaufwand: Für die Berechnung der dynamisch-zentrierten Kreise werden die als (geographische) WGS84-Koordinaten übermittelten Positionen erst im Server in das (kartesische) UTM-System überführt, was wiederum an die für München geltende UTM-Zone gebunden ist. Der Maximalradius der DSC-Kreise wird, um den durch die UTM-Kartenprojektion entstandenen Fehler nicht zu vergrößern, auf 100 km gesetzt. Im Gegensatz dazu sind für die Umsetzung der DCC-Strategie keinerlei Koordinatentransformationen notwendig. Der Server kann die den dynamisch-zentrierten Kreisen zugrunde liegenden Update-Distanzen direkt anhand der übermittelten WGS84-Koordinaten berechnen.

Während des Feldversuchs traten jedoch auch Probleme auf, welche sich vor allem auf die unzureichende Abdeckung durch GPS und GPRS zurückführen ließen. Während GPS wie erwartet in der Nähe von Gebäuden und Bäumen schlecht verfügbar bzw. mit hohen Ungenauigkeiten verbunden war, so erwies sich auch GPRS als relativ unzuverlässig. Teilweise war gar kein GPRS-Signal empfangbar, und oft wurden Netzverzögerungen in der Größenordnung mehrerer Sekunden zum Austausch eines Datenpakets zwischen Endgerät und Server verzeichnet. Die resultierenden Fehler traten in der Form falscher oder ausbleibender Nahbereichs- und Trennungsereignisse zutage. Wie anhand der auf dem Server geführten Protokolldateien sichtbar wurde, liegt der Ursprung dieser Fehler jedoch nicht in den vorgestellten Strategien. Grundsätzlich hat wohl jeder ortsbezogene Dienst, der sich auf

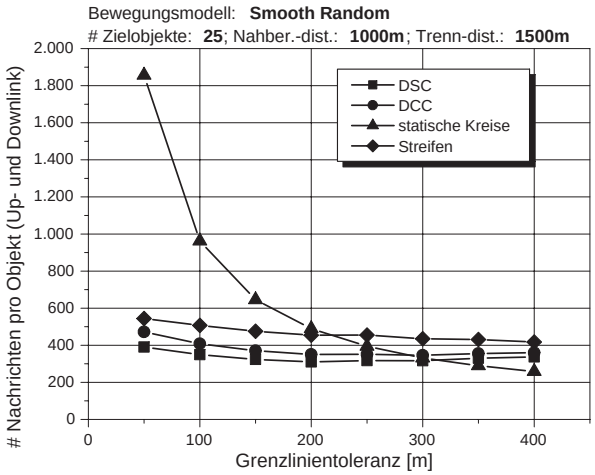

Abbildung 4.12: Nachrichten pro Zielobjekt abhängig von b bei Smooth Random

GPRS und GPS stützt, unter ähnlichen Problemen zu leiden.

Um das Problem wenigstens zu lindern, ist der Prototyp mit den folgenden Methoden zur Fehlertoleranz ausgestattet: Falls die GPRS-Verbindung kurzzeitig unterbrochen ist und der Server ein Gerät erfolglos nach dessen Position anfragt, wird zur Nahbereichs- und Trennungserkennung die zuletzt übermittelte Position verwendet. Wird basierend auf der so ermittelten, möglicherweise veralteten Ortsinformation einer Zielperson ein Alarm generiert, so wird dieser Umstand der zu benachrichtigenden Anwendung zusätzlich mitgeteilt, so dass diese die Information als unzuverlässig einstufen kann. Sobald GPRS wieder verfügbar ist, meldet sich das Gerät erneut beim Location Server an und erhält, falls notwendig, neue Update-Konfigurationen. Auf ganz ähnliche Art und Weise wird verfahren, wenn die GPS-Abdeckung plötzlich verloren geht.

Überdies übermittelt das mobile Endgerät regelmäßig *Keep-alive-Nachrichten*, deren Periodizität vom Server dynamisch angepasst werden kann. Ziel der Nachrichten ist es, die Verbindung zum Server offen zu halten, welche andernfalls, also bei länger ausbleibendem Nachrichtenverkehr, von der Firewall des Netzbetreibers, die *Network Address Translation (NAT)* anwendet, automatisch zurückgesetzt würde. Die Übermittlung von Keep-alive-Nachrichten widerspricht in gewissem Maße den Bestrebungen der Strategien, den Nachrichtenverkehr möglichst gering zu halten. Die Nachrichten könnten jedoch durch entsprechende Vorkehrung auf der Seite des Netzbetreibers vermieden werden, was in dem durchgeführten Testfall natürlich nicht möglich war. Die Umgehung der NAT-Firewall stellt also kein grundsätzliches technisches Problem dar, sondern eher ein organisatorisch-vertragliches, in dem Fall zwischen Location Provider und Netzbetreiber.

Allgemein gesprochen können Anwendungen, die die beschriebenen Strategien zur Nahbereichs- und Trennungserkennung nutzen, von neuen Technologien sicherlich nur profitieren. Es ist zu erwarten, dass sich die Stabilität von Netzverbindungen sowie die erreichbare Verzögerung durch 3G-Systeme wie UMTS oder die sich ankündigenden 4G-Netze bedeutend verbessern. Dem mobilen Nutzer stehen bei 4G mehrere Trägerdienste gleichzei-

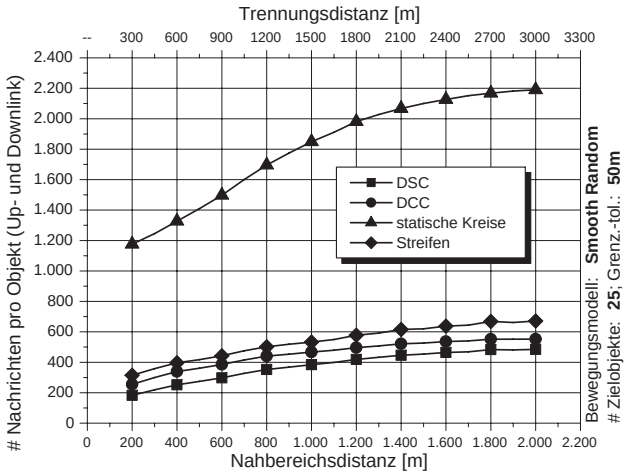

Abbildung 4.13: Nachrichten pro Zielobjekt abhängig von d_p und d_s bei Smooth Random

tig zur Verfügung, wie zum Beispiel GPRS, paketvermittelndes UMTS oder WLAN. Fällt ein Träger plötzlich aus oder zeichnet sich eine Qualitätsverschlechterung ab, können Endgeräte zwischen den Technologien nahtlos hin- und her wechseln (vertikales Handover). Auf ähnliche Weise wird sich die Zuverlässigkeit und Verfügbarkeit von Ortungstechnologien verbessern. Das europäische Satellitennavigationssystem Galileo, das mit dem amerikanischen GPS konkurrieren soll, wird voraussichtlich im Jahr 2011 in Betrieb gehen. Durch Verwendung entsprechender Hardware können beide Systeme in Kombination verwendet werden. Insbesondere können dann vom Endgerät gleichzeitig Satelliten beider Systeme zur Positionsberechnung genutzt werden. Dadurch erhöht sich die Wahrscheinlichkeit, dass sich auch in der Nähe großer Gebäude genügend Satelliten in Sichtlinie befinden.

4.5 Koppelnavigation als Optimierung für die Nahbereichserkennung

Anhand der durchgeführten Simulationen zeigt sich sehr schön, dass das Bewegungsverhalten von Zielpersonen zwar großen Einfluss auf die Gesamtanzahl ausgetauschter Nachrichten hat, das Leistungsverhältnis der einzelnen Strategien untereinander jedoch relativ konstant bleibt. Somit lässt sich unabhängig vom zugrunde liegenden Bewegungsverhalten entscheiden, welche der Strategien zu gegebenen Anforderungen am besten passen. Beispielsweise eignet sich die DCC-Strategie vor allem dann, wenn nicht nur die Nachrichtenanzahl gering gehalten werden soll, sondern gleichzeitig eine sehr einfache Implementierung gewünscht ist. Die dynamisch-verschobenen Kreise sind vorzuziehen, wenn die Reduktion ausgetauschter Nachrichten absoluten Vorrang hat.

Für letzteren Fall lassen sich jedoch noch weitere Optimierungen vorstellen, nämlich insbesondere solche, die gewisse Annahmen über das Bewegungsverhalten von Zielpersonen treffen. Eine solche Optimierung ist der im Folgenden entwickelte, auf Koppelnavigation

basierende Ansatz, der die aktuelle Bewegungsgeschwindigkeit und -richtung der Zielpersonen in seine Berechnungen mit einbezieht und der auch in [164] erläutert wird. Der Fokus liegt dabei rein auf der Nahbereichserkennung, die ein größeres Optimierungspotential verspricht als die Trennungserkennung. Typischerweise bewegen sich die Zielobjekte bei der Trennungserkennung ja nur innerhalb relativ eng gefasster Bereiche, weswegen in diesem Fall die Betrachtung von Bewegungsrichtung und -geschwindigkeit als weniger zielführend erachtet wird.

Die Koppelnavigation war schon unter Seefahrern des 16. Jahrhunderts eine weit verbreitete Methode zur Positionsbestimmung. Unter Einbeziehung eines vorgegebenen Bezugspunkts, wie des Abgangshafens, konnte die aktuelle Position eines Schiffes damals mit Hilfe stetig fortgeführter Richtungs- und Geschwindigkeitsmessungen ermittelt werden. Ein Kompass, der seit dem 14. Jahrhundert zur nautischen Standardausrüstung zählte [25], gab die Richtung vor. Außerdem wurde die aktuelle Geschwindigkeit ermittelt, indem man die Zeit maß, die ein schwimmendes Stück Treibholz brauchte, um eine vorgegebene Länge entlang des Bootsrands zurückzulegen.

Die damals rein zur Positionsbestimmung verwendete Koppelnavigation hat inzwischen eine Art Renaissance durchlaufen. [93] zufolge eignet sie sich beispielsweise, um innerhalb mobiler Ad-hoc-Netze auf effiziente Weise Routingtabellen aktuell zu halten. Weitere Anwendungsfelder ergeben sich bei verteilten interaktiven Simulationen [46] sowie beim Mobility Management für persönliche Kommunikationssysteme [107]. Die Koppelnavigation ist außerdem bekannt als effiziente Position-Update-Methode, also zur Aktualisierung der vom mobilen Endgerät ermittelten Position beim Location Server. Sie dient der im Folgenden entwickelten Strategie zur Nahbereichserkennung.

Zuerst werden die gemeinsamen Merkmale der bisher entwickelten Strategien anhand einer ihnen gemeinsamen anhaftenden Prozedur analysiert und darauf aufbauend Anforderungen an den zu entwickelnden Ansatz formuliert. Danach wird die Koppelnavigation als Position-Update-Methode, wie sie in der Literatur bisher vorgesehen ist, erörtert und anschließend eine für die Problemstellung möglicherweise vorteilhaftere Variante präsentiert. Es wird ferner gezeigt, wie die Koppelnavigation mit der bereits bekannten Prozedur zur Nahbereichserkennung zusammenspielt. Schließlich werden Simulationsergebnisse präsentiert, die durch den neuen Ansatz erzielt wurden.

4.5.1 Analyse der bisherigen Strategien

Alle drei vorgestellten dynamischen Strategien, die dynamisch-zentrierte und dynamisch-verschobene Kreisstrategie sowie der Streifenalgorithmus, machen sich die dynamische Konfiguration von Position-Update-Methoden (vergleiche Abschnitt 2.4) zunutze und unterscheiden sich prinzipiell nur darin, welche geometrische Form die entsprechenden Unsicherheitsbereiche haben und wie diese zwischen den Zielpersonen aufgeteilt werden. Den Ansätzen liegt jedoch eine gemeinsame Prozedur zugrunde, die in Abbildung 4.14 für die Nahbereichserkennung veranschaulicht ist: Erst teilt der Server jedem Zielobjekt t_i einen initialen Unsicherheitsbereich A_i zu (1). Dann positionieren sich die Geräte kontinuierlich selbst, beispielsweise mit GPS, und überprüfen, ob ihre aktuelle Position noch im zugeteilten Bereich liegt (2). In dem Fall, dass eine Person t_i ihren Bereich verlassen hat, meldet sie ihre aktuelle Position p_i an den Server. Hier wird p_i mit den Unsicherheitsbereichen A_j aller Personen t_j verglichen, die bezüglich t_i unter Beobachtung stehen (3). Abhängig von

der Strategie wird eine bestimmte Untermenge dieser Personen nach ihrer genauen Position befragt (4). Für diejenigen t_j daraus, für die der Positionsabgleich mit t_i (5) die Nahbereichsbedingung erfüllt, wird ein entsprechendes Ereignis ausgelöst (6). Schließlich werden alle Zielpersonen, die durch die Prozedur inspiziert wurden (entweder durch Übermittlung eines Position Updates oder durch Polling), mit neuen Unsicherheitsbereichen ausgestattet (1).

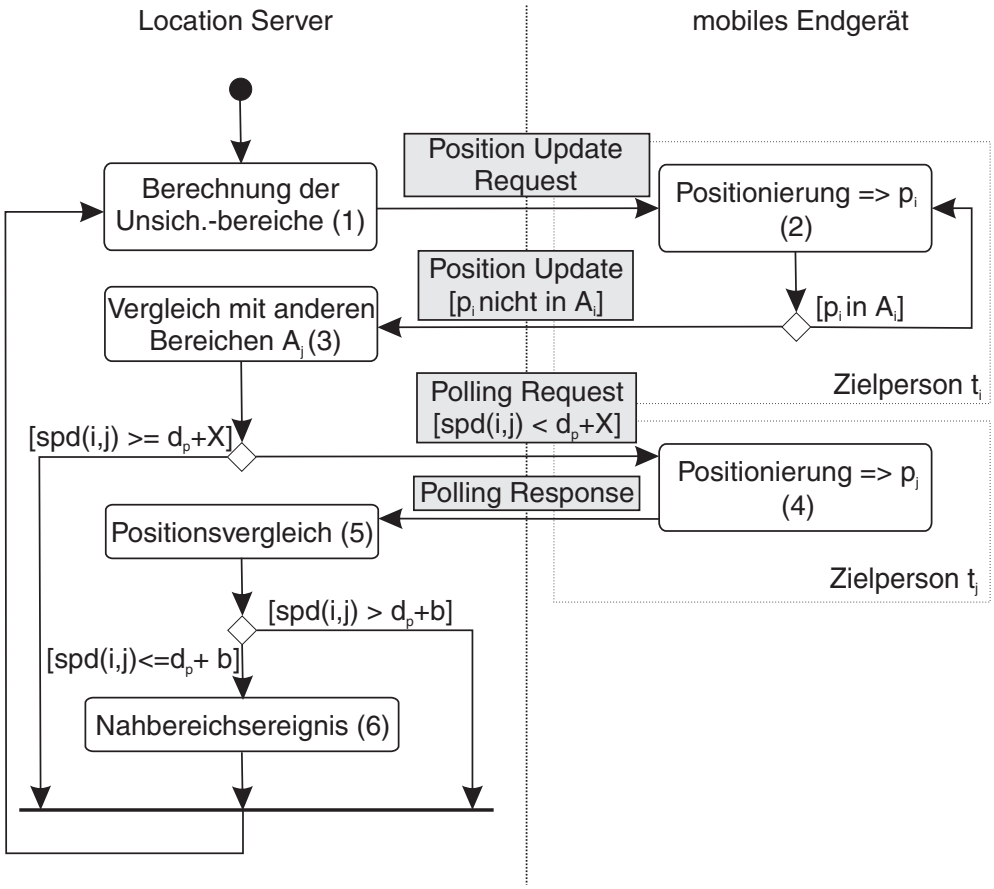

Abbildung 4.14: Basisprozedur zur Nahbereichserkennung

Dabei lassen sich insbesondere bezüglich Größe und Form der Unsicherheitsbereiche folgende Aussagen formulieren, die sich auch durch die bereits beschriebenen Simulationen bestätigt haben.

- **Bereichsgröße als Kompromiss.** Große Unsicherheitsbereiche führen im Mittel zu weniger Position Updates, was wünschenswert ist. Wird den Zielpersonen innerhalb ihrer Bereiche viel Bewegungsspielraum gelassen, so verlassen sie diese offensichtlich weniger häufig.

  Andererseits führen große Bereiche zu potentiell mehr Polling-Operationen. Durch größere Unsicherheit wird es wahrscheinlicher, dass die kleinst- bzw. größtmöglichen Distanzen zwischen Paaren von Zielpersonen nach Position Updates in den kritischen Bereich kommen, der ein Polling erforderlich macht. Beispielsweise wird beim Streifenalgorithmus der ganze verfügbare Raum zwischen den Objekten aufgeteilt, was im

Mittel zu relativ großen Unsicherheitsbereichen führt.[1] Während dadurch die Anzahl von Position Updates reduziert wird, folgt gleichzeitig auf jedes Update mindestens eine Polling-Operation, was wiederum mit dem Austausch von Nachrichten verbunden ist. Ein Gegenbeispiel ist die DCC-Strategie mit ihren relativ kleinen Kreisen, bei der im Mittel mehr Position Updates erzeugt werden. Hier wird nicht durch jedes Update in der Folge ein Polling verursacht. Dadurch reduziert sich der Gesamtaufwand an Nachrichten.

- **Form und Ausrichtung.** Unabhängig von der Größe der Unsicherheitsbereiche hat deren Form wie auch die Ausrichtung der zuletzt übermittelten Position innerhalb des Bereichs Auswirkungen auf die Menge übertragener Nachrichten. Wird die letzte Position einer Person beispielsweise näher am Bereichsrand angesetzt, so ist bei zufällig gewähltem Bewegungsverhalten im Mittel eine höhere Anzahl von Position Updates zu erwarten. Offensichtlich ist die minimale Distanz, die eine Person bewältigen muss, um bis zum Rand zu stoßen, bei der Definition der Bereiche von Belang. Deswegen werden in der DSC-Strategie die Kreise auch so gewählt, dass sie die entsprechenden DCC-Kreise jeweils einschließen. Nur so ist gewährleistet, dass die minimale Distanz zum Rand erhalten bleibt. Allerdings ist es grundsätzlich schwer, Zusammenhänge zwischen Bereichsform sowie -ausrichtung und der Anzahl ausgetauschter Nachrichten auf analytischem Wege herzuleiten, da diese stark vom zugrunde liegenden Bewegungsmuster abhängen.

Als Schlussfolgerung aus diesen Betrachtungen stellt sich die Anforderung, zwar die Anzahl von Position Updates so weit wie möglich zu senken, dabei aber den entstehenden Unsicherheitsbereich der verfolgten Zielpersonen möglichst wenig zu vergrößern. Gleichzeitig sollte die Zielperson innerhalb des Bereichs mittig ausgerichtet sein, und der Bereich sollte eine relativ einfache Form aufweisen.

4.5.2 Strategie basierend auf Koppelnavigation

Klassisch funktioniert Koppelnavigation folgendermaßen: Sei p_i die letzte, von Zielperson t_i zur Zeit z_1 übermittelte Position. Ferner sei $\vec{v_i}$ der ermittelte Bewegungsvektor von t_i, der also gemessene Geschwindigkeit und Richtung definiert und der dem Server gleichfalls zum Zeitpunkt z_1 übermittelt wurde. Dann kann die geschätzte Position E_i von t_i für jeden Zeitpunkt z_2 mit $\Delta z := z_2 - z_1$ anhand von Formel 4.1 berechnet werden. Optional kann auch die zum Zeitpunkt z_1 gemessene Beschleunigung von t_i in die Formel einfließen, was jedoch an dieser Stelle nicht weiter betrachtet wird.

$$E_i \quad := \quad p_i + \vec{v_i}\Delta z \tag{4.1}$$

Ein fortgeschrittener Ansatz wird von [105] vorgeschlagen. Additiv dazu, die Schätzfunktion durch Messwerte wie Geschwindigkeit und Richtung zu parametrisieren, wird das Verfahren durch Kartendaten unterstützt, mit deren Hilfe Aussagen über die Route einer

[1] Die Tatsache, dass Streifen jeweils in der Mitte zwischen zwei Zielpersonen angebracht werden, führt für eine steigende Anzahl überwachter Personen zu immer erheblicheren räumlichen Einschränkungen, wovon jetzt allerdings abgesehen wird.

Zielperson möglich sind. Es zeigt sich, dass der vorgeschlagene kartenbasierte Ansatz in den meisten Fällen besser abschneidet als der klassische, also weniger Nachrichten austauscht.

Wie bereits in Abschnitt 2.4 erwähnt, wurden beide Methoden, klassisch und kartenbasiert, mit dem Ziel entwickelt, einzelne Zielpersonen bezüglich eines festgelegten Unsicherheitsschwellwerts zu überwachen. Das Endgerät von t_i überträgt also genau dann seine Position (zusammen mit aktuellen Bewegungsinformationen), wenn die wahre Position p_i vom Schätzwert E_i um mehr als eine vorgegebene Distanz d abweicht. Nachdem E_i auf Server und Endgerät simultan berechnet wird, kann der Server die Position der Zielperson für jeden Zeitpunkt z_2 auf einen Unsicherheitsbereich A_i annähern, wobei es sich bei A_i in beiden Fällen um einen sich bewegenden Kreis mit Mittelpunkt E_i und Radius d handelt.

Wiederum unterscheidet sich in diesem Punkt die Nahbereichserkennung von der steten Verfolgung einzelner Objekte, da sie nur die Distanzen zwischen Zielobjekten betrachtet. Ein festes Maß an Unsicherheit muss also nicht gewährleistet werden. Der im Folgenden entwickelte Koppelnavigationsalgorithmus weicht daher die Einschränkung des festen Unsicherheitsschwellwerts auf. Stattdessen darf A_i über die Zeit anwachsen. Während bei festem Schwellwert Position Updates mit verstreichender Zeit Δz immer wahrscheinlicher werden, ist es das Ziel anwachsender Bereiche, diesen Effekt zu vermeiden.

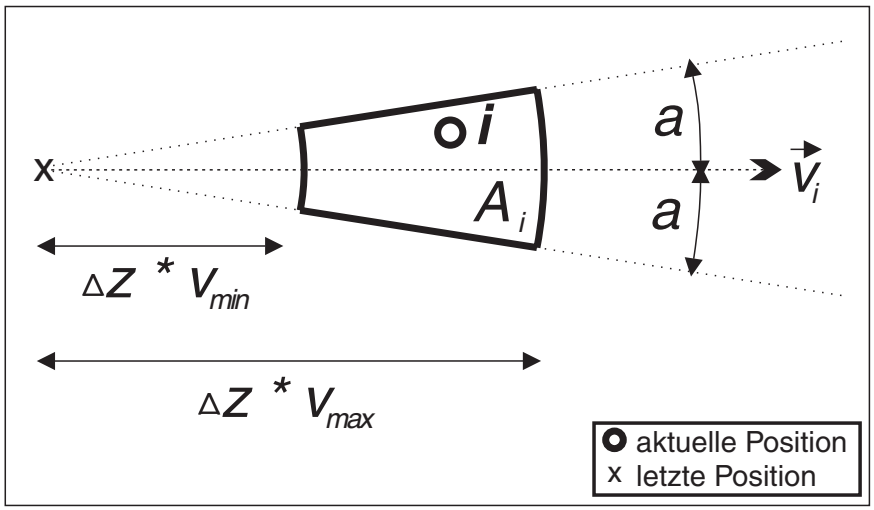

Abbildung 4.15: Aufgespannter Unsicherheitsbereich bei der Koppelnavigation

Anstelle eines distanzbasierten Schwellwerts werden Abweichungsgrenzen im vorgeschlagenen Algorithmus bezüglich Geschwindigkeit und Richtung eines Zielobjekts definiert. Wie in Abbildung 4.15 dargestellt, ergibt sich daraus für den Unsicherheitsbereich A_i einer Zielperson t_i die Form eines Ringsegments, dessen Mittelpunkt an der letzten berichteten Position liegt. Das Segment öffnet sich entlang des Bewegungsvektors $\vec{v_i}$, wobei zu beiden Seiten ein noch zu definierender Öffnungswinkel a angetragenen wird. Der innere und äußere Ringradius werden bestimmt durch die bereits verstrichene Zeit und die gemessene Absolutgeschwindigkeit $|\vec{v_i}|$, jeweils in Kombination mit einem Schwellwert zum Erhalt der minimal bzw. maximal erlaubten Geschwindigkeit, $v_{min}(|\vec{v_i}|)$ und $v_{max}(|\vec{v_i}|)$. Sei also p_i die zuletzt übermittelte Position und v_i der zu dem Zeitpunkt bestimmte Bewegungsvektor. Die aktuell auf dem Endgerät gemessene Position m_i wird genau dann nicht an den

Server gesendet, wenn gilt $\Delta z\ v_{min}(|\vec{v_i}|) \leq dist(p_i, m_i) \leq \Delta z\ v_{max}(|\vec{v_i}|)$ und gleichzeitig $ang(\vec{v_i}) - a \leq ang(p_i, m_i) \leq ang(\vec{v_i}) + a$. Mit verstreichender Zeit Δz wird der durch den Algorithmus aufgespannte Bereich offensichtlich größer. Die Schwellwerte bezüglich Richtung und Geschwindigkeit werden vom Server also idealerweise so vorkonfiguriert, dass relativ wenige Position Updates entstehen (was durch große Flächen bewirkt wird), die Flächen aber dennoch nicht so groß werden, dass sie übermäßig viele Pollings verursachen.

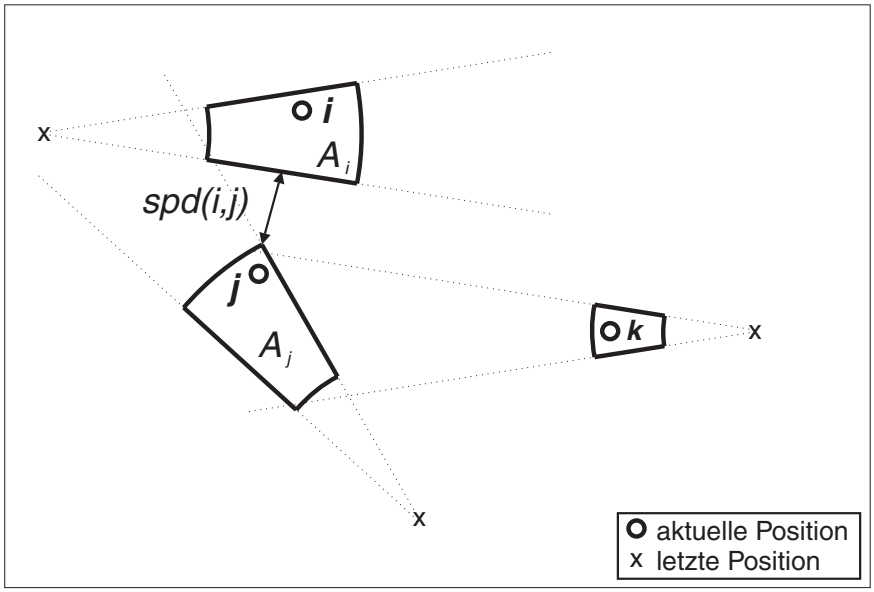

Abbildung 4.16: Berechnung der kleinstmöglichen Distanz bei der Koppelnavigation

Ähnlich wie bei den nicht bewegungssensitiven Strategien kann anhand der Ringsegmente A_i und A_j zweier Zielpersonen t_i und t_j deren kleinstmögliche Distanz $spd(t_i, t_j)$ berechnet werden, um Personen beizeiten nach ihrer exakten Position anzufragen. Zu diesem Zweck sind verschiedene geometrische Betrachtungen notwendig, die an dieser Stelle jedoch nicht ausgeführt werden. Der interessierte Leser wird dafür auf [165] verwiesen. Abbildung 4.16 veranschaulicht die Berechnung anhand einer möglichen Ausführungssituation.

Nachdem der grundlegende Ansatz, Koppelnavigation zur Nahbereichserkennung anzuwenden, dargelegt wurde, beschreibt der nächste Abschnitt genauer, wie er sich in die in Abbildung 4.14 gezeigte Basisprozedur integriert.

Erweiterung der Basisprozedur

Ganz ähnlich wie bei zonen- und distanzbasierten Position Updates kann der Location Server bei der Koppelnavigation die aktuelle Position einer Zielperson t_i zu jeder Zeit auf einen bestimmten Unsicherheitsbereich A_i eingrenzen. Im Gegensatz zu den DCC- und DSC-Strategien, die auf zonen- und distanzbasierten Updates beruhen, müssen bei der Koppelnavigation Unsicherheitsbereiche behandelt werden, die sich durchgehend bewegen und möglicherweise die Größe ändern. Folglich ist es möglich, dass zwei Zielpersonen die Nahbereichsdistanz unterschreiten, ohne dabei ihre Bereiche zu verlassen. Der Server muss also stets gewahr sein, wenn sich zwei Bereiche auf Kollisionskurs befinden, und die exakte Position der Objekte rechtzeitig vergleichen.

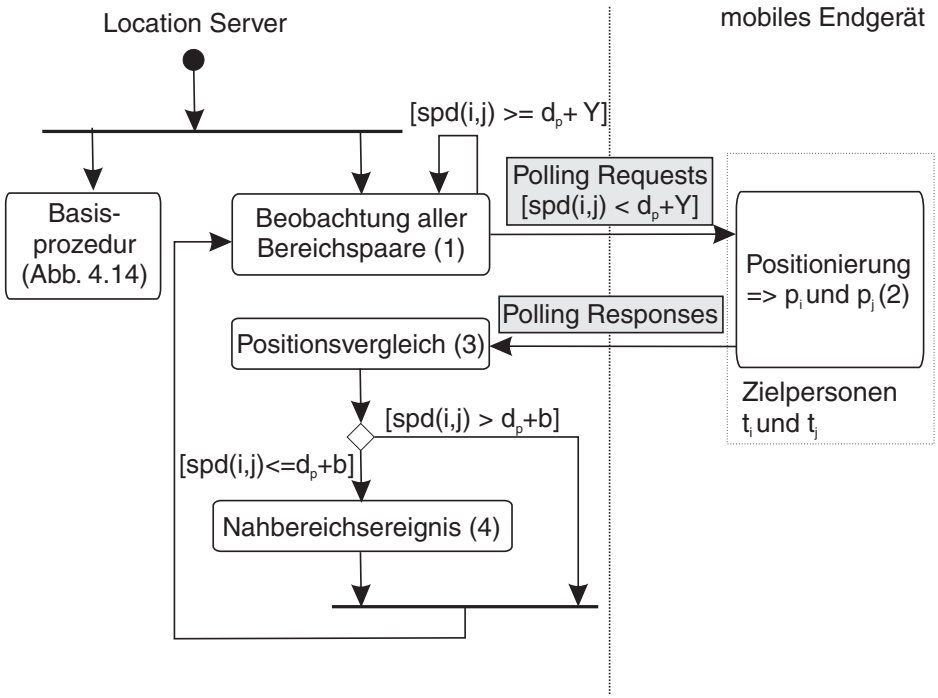

Abbildung 4.17: Erweiterte Basisprozedur zur Nahbereichserkennung

Abbildung 4.17 beschreibt die damit verbundene Prozedur, welche nun eine neue serverseitige Aktivität enthält: Zusätzlich zur Ausführung der bekannten Prozedur überwacht der Server die Unsicherheitsbereiche aller Paare von Personen (i, j) (1) und fragt beide nach der exakten Position, sobald $spd(i, j) < d_p + Y$, wobei $Y >= 0$ ein einstellbarer Wert ist (2). Ergibt der Vergleich der ermittelten Positionen (3), dass $spd(i, j) < d_p + b$, so wird ein Nahbereichsalarm bezüglich i und j ausgelöst (4).

4.5.3 Ergebnisse

Der auf der Koppelnavigation basierende Ansatz wurde in mehreren Simulationen innerhalb des in Abschnitt 7.3.3 beschriebenen TraX-Simulators getestet und mit den anderen dynamischen Strategien bezüglich des Nachrichtenaufkommens verglichen. Der Algorithmus wurde mit den folgenden Parametern, die sich durch einfaches Ausprobieren ergaben, konfiguriert, vergleiche Abbildung 4.15. Der beidseitige Öffnungswinkel a wurde auf 15 Grad gesetzt, die minimale Geschwindigkeit v_{min} auf $\frac{1}{2}|\vec{v_i}|$ und die maximale Geschwindigkeit v_{max} auf $\frac{3}{2}|\vec{v_i}|$.

Die Simulationen sollten vor allem Aufschluss darüber geben, welchen Einfluss unterschiedliche Bewegungsmuster auf die Leistung des Ansatzes haben. Nachdem es mit den vorhandenen GPS Traces leider nicht möglich war, diese sinnvoll und mit einer jeweils ausreichenden Zahl an Vertretern in verschiedene Bewegungsklassen zu unterteilen – zu viele Traces beschreiben relativ ungerichtete Bewegungen, und nur wenige längere und flüssige Autofahrten sind dokumentiert – wurden zum Testen des Ansatzes künstliche, entsprechend konfigurierbare Muster gewählt, nämlich das schon eingeführte Smooth-Random-Bewegungsmuster und das Random-Waypoint-Modell [75].

Wie bei der Evaluierung in Abschnitt 4.4.1 werden drei Operationen zur Umsetzung der Strategie betrachtet: Position Update Requests zur Konfiguration der Bereiche, Position Updates und Pollings. Die Anzahl der jeweils von den Operationen erzeugten Nachrichten ergibt sich genau wie bei den anderen dynamischen Strategien anhand von Tabelle 4.1.

Die simulierten Bewegungsszenarien wurden so gewählt, dass sie für drei verschiedene Transportmittel jeweils möglichst realistisches Nutzerverhalten nachbildeten. Dabei handelte es sich um ein Fußgänger-Szenario, ein Radfahrer-Szenario und ein Autofahrer-Szenario. Jedes Modell wurde durch eine charakteristische Kombination aus Werten für typische und maximale Geschwindigkeiten beschrieben. Dabei werden die typischen Geschwindigkeiten höher gewichtet als die maximale. Sie werden also mit einer höheren Wahrscheinlichkeit ausgewählt.

Die simulierte Fläche wurde auf 5 km $\times$ 5 km festgesetzt und die Personenzahl zwischen 5 und 60 variiert. Es wurde durchgehend eine Nahbereichsdistanz von 500 m angenommen.

Szenario 1: Fußgänger

Das Fußgänger-Szenario wurde mit einer Maximalgeschwindigkeit von 10 km/h belegt, was etwa leichtem Joggen entspricht. Die bevorzugten Geschwindigkeiten lagen bei 3 km/h und 5 km/h.

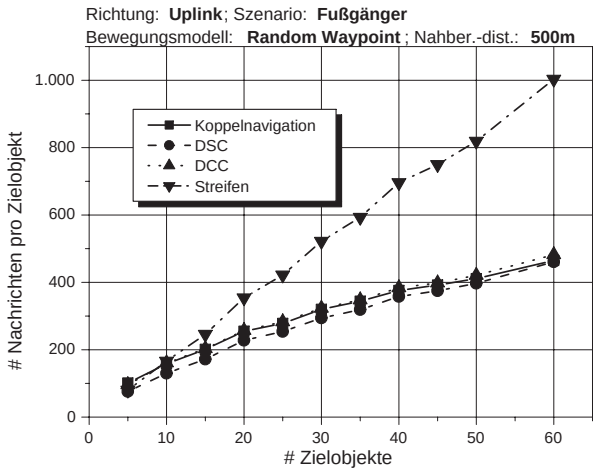

Abbildung 4.18: Uplink-Nachrichten pro Zielobjekt beim Fußgänger-Szenario

Liegt hier das Random-Waypoint-Modell zugrunde, welches nur Bewegungen auf gerader Linie vorsieht, so ist der Ansatz mit Koppelnavigation dem Streifenalgorithmus ab einer Zahl von 10 Zielpersonen überlegen, sowohl was Uplink als auch was Downlink betrifft. Im Vergleich zu DCC und DSC verhalten sich Uplink und Downlink unterschiedlich. Im Uplink liegen alle drei Strategien eng zusammen, und zwar innerhalb eines Bereichs von ungefähr 20 Nachrichten Unterschied von insgesamt beinahe 500 (vergleiche Abbildung 4.18). Im Downlink fällt die Koppelnavigation jedoch hinter der Leistung von DCC/DSC zurück (vergleiche Abbildung 4.19).

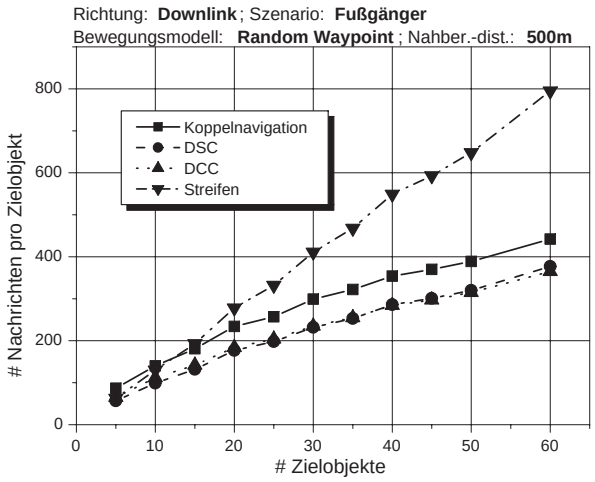

Abbildung 4.19: Downlink-Nachrichten pro Zielobjekt beim Fußgänger-Szenario

Mit dem Smooth-Random-Bewegungsmuster, welches komplexere Bewegungen vorsieht, insbesondere Beschleunigungen, Bremsvorgänge und Kurven, vergrößern sich die Unterschiede zwischen den Kreisstrategien und der Koppelnavigation. Letztere schneidet sowohl im Up- als auch im Downlink deutlich schlechter ab. Mehr Details hierzu finden sich in [165].

Szenario 2: Radfahrer

Im Radfahrer-Szenario wurden höhere Geschwindigkeitswerte eingesetzt als für den Fußgänger, nämlich maximal 30 km/h und typisch 17 km/h bzw. 20 km/h. Wird hier Random Waypoint als Bewegungsmodell ausgewählt, so lässt sich ein klarer Leistungsvorsprung des Ansatzes mit Koppelnavigation erkennen. Bei 60 Zielobjekten werden im Uplink circa 42 % weniger Nachrichten ausgetauscht als bei der zweitplatzierten DSC-Strategie und immerhin 70 % weniger als beim Streifenalgorithmus. Die Unterschiede sind im Downlink etwas kleiner, wobei die Koppelnavigation in Führung bleibt mit 30 % weniger als DCC/DSC und 65 % weniger als die Streifen.

Verglichen mit Random Waypoint führt Smooth Random zu einer leichten Verringerung des Vorsprungs der Koppelnavigation. Die Unterschiede zu den anderen Strategien sind jedoch immer noch klar erkennbar. Es werden im Uplink 29 % weniger Nachrichten als bei DSC und 60 % weniger als bei den Streifen ausgetauscht (vergleiche Abbildung 4.20). Im Downlink übersteigt die Nachrichtenzahl der DSC-Strategie den der Koppelnavigation um mehr als 200 von etwa 1000 Nachrichten (vergleiche Abbildung 4.21).

Szenario 3: Autofahrer

Im dritten Szenario werden Höchstgeschwindigkeiten von 55 km/h angenommen, was städtischen Autoverkehr widerspiegeln soll. Die konfigurierten typischen Geschwindigkeiten

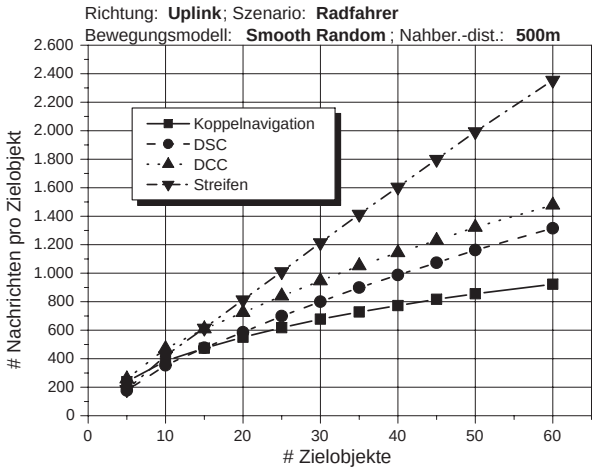

Abbildung 4.20: Uplink-Nachrichten pro Zielobjekt beim Radfahrer-Szenario

sind zum einen der Höchstwert selbst und zum anderen ein relativ niedriger Wert von 8 km/h, welcher Situationen der Hauptverkehrszeit modelliert.

Das Anheben der Geschwindigkeiten wirkt sich wiederum in einem Leistungsanstieg der Koppelnavigation aus. Für die Tests, die auf dem Random-Waypoint-Modell basieren, werden im Vergleich zum Streifenalgorithmus bei 60 Nutzern 74 % weniger Nachrichten im Uplink verursacht und 54 % weniger als beim zweitbesten Algorithmus, DCC. Im Downlink unterbietet die Koppelnavigation die von der DCC-Strategie benötigten 1800 Nachrichten um beinahe 800. Auffällig ist, dass bei einer Anzahl von circa 30 Objekten die DSC-Strategie von den dynamisch-zentrierten Kreisen überholt wird, sowohl im Uplink als auch im Downlink. Hier schlägt sich also die Einfachheit von DCC zum ersten Mal auch in der Effizienz nieder (die Spielfläche hat in dem Fall mit 5 km $\times$ 5 km nur ein Viertel der Größe wie bei den Simulationen aus Abschnitt 4.4.1, entsprechend ist die Dichte der Zielobjekte viermal so hoch).

Legt man für die Durchläufe das Smooth-Random-Bewegungsmuster zugrunde, so zeigt die Koppelnavigation im Uplink einen um 46 % geringeren Nachrichtenaufwand als DSC und 68 % weniger Nachrichten als beim Streifenalgorithmus (vergleiche Abbildung 4.22). Im Downlink liegt der Unterschied zur nächstplatzierten Kreisstragie bei circa 800 Nachrichten (vergleiche Abbildung 4.23).

4.5.4 Bewertung

In allen simulierten Szenarien und jeweils unter Verwendung beider beschriebener Bewegungsmuster zeigte sich, dass mit steigender Anzahl betrachteter Zielpersonen die Koppelnavigation bezüglich ausgetauschter Nachrichten besser skaliert als die nicht bewegungssensitiven Strategien: Streifenalgorithmus, DCC und DSC. Der Effekt fällt umso deutlicher aus, je höher die Bewegungsgeschwindigkeit angesetzt wird. Einzuräumen ist jedoch, dass für die betrachteten Szenarien Bewegungsmodelle ausgewählt wurden, die relativ gerichte-

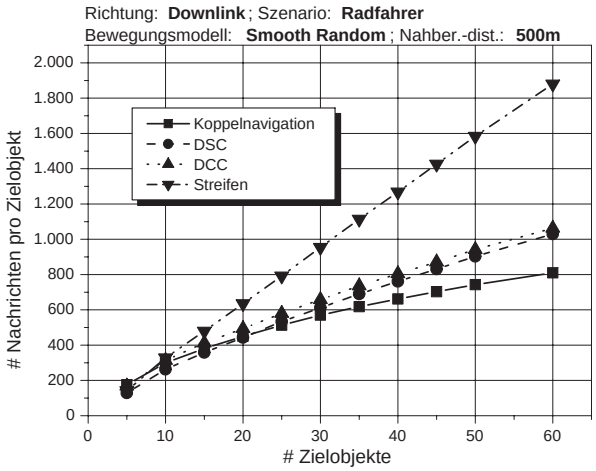

Abbildung 4.21: Downlink-Nachrichten pro Zielobjekt beim Radfahrer-Szenario

te und gleichmäßige Bewegungen produzieren, was in verschiedenen denkbaren Situationen der realen Welt nicht unbedingt gegeben ist. Wie in [165] beschrieben, ergibt sich für diffusere Bewegungen ein vergleichsweise schlechteres Abschneiden der Koppelnavigation. Auffällig ist, dass die relative Verbesserung durch die Koppelnavigation im Downlink weniger groß ist als im Uplink. Dies könnte daran liegen, dass im Falle einer möglichen Nahbereichssituation, deren Bestimmung den Vergleich beider exakter Positionen erfordert, die anderen dynamischen Strategien nur ein Polling vornehmen und sich die jeweils andere Position aus dem übermittelten Update ergibt. Bei dem Ansatz mit Koppelnavigation werden hingegen oft zwei Pollings benötigt, was zu einer zusätzlichen Nachricht im Downlink führt.

Der neue Ansatz hat bezüglich ausgetauschter Nachrichten im Vergleich zu den anderen Strategien also eindeutige Vorteile, solange die ausgeführten Bewegungen gleichmäßig und zielgerichtet sind. Ist dies nicht der Fall, so kann jedoch auch ein gegenteiliger Effekt erzielt werden. Dies macht den praktischen Einsatz der Koppelnavigation nur dann sinnvoll, wenn dem System Informationen über die ausgeführten Bewegungen bereits im Vorhinein vorliegen. Denkbar wäre ein Einsatz im Automobilbereich, beispielsweise um das im Auto fest installierte Navigationssystem mit Community-Funktionen wie der Nahbereichserkennung zu versehen. Andererseits ist vom Einsatz der Koppelnavigation abzuraten, wenn derartige Informationen nicht vorliegen, da das Verfahren neben der beschriebenen Effizienzsteigerung auf der Luftschnittstelle auch zusätzliche Schwierigkeiten und Kosten generiert.

So ist beispielsweise die korrekte Zeitsynchronisation zwischen Endgerät und Server, die für die Berechnung der Schätzfunktion notwendig ist, ein potentielles Problem. Zwar steht mit der Nutzung von GPS gleichzeitig ein zuverlässiger Zeitgeber für die Endgeräte zur Verfügung, dieser ist jedoch dem Server nicht automatisch zugänglich. Außerdem wird der Ansatz dadurch abhängig von einem speziellen Ortungsverfahren, was nicht unbedingt erwünscht ist. Ferner ist zwar die Berechnung der Schätzfunktion auf dem Endgerät relativ einfach, auf dem Server generiert jedoch die in Abschnitt 4.5.2 beschriebene ständige

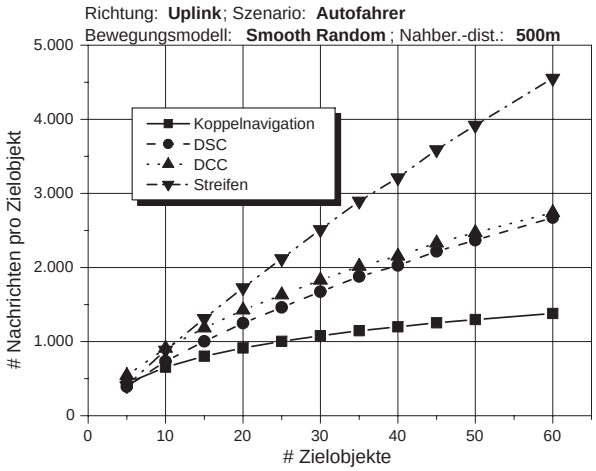

Abbildung 4.22: Uplink-Nachrichten pro Zielobjekt beim Autofahrer-Szenario

Überwachung von möglicherweise kollidierenden Zielpersonen einen nicht unerheblichen Zusatzaufwand, welcher mit einer größer werdenden Anzahl von Zielpersonen quadratisch ansteigt. Der Ansatz mit Koppelnavigation wird daher als Speziallösung betrachtet, deren Einsatz gut überlegt sein sollte.

4.6 Zusammenfassung

In diesem Kapitel wurden verschiedene Strategien zur Nahbereichs- und Trennungserkennung mobiler Zielpersonen entwickelt und bezüglich über die Luftschnittstelle ausgetauschter Nachrichten mittels Simulationen verglichen. Dabei stellte sich zunächst heraus, dass die dynamische Konfiguration von Position-Update-Methoden eindeutige Vorteile gegenüber statischen Ansätzen wie der statischen Kreisstrategie hat. Ferner zeigte sich, dass die eigenen dynamischen Kreisstrategien, DCC und DSC, die diese Methoden in Abhängigkeit der nächsten Nachbarobjekte konfigurieren, mit der Anzahl überwachter Personen besser skalieren als der bestehende Streifenalgorithmus. Ein weiterer eigener Ansatz wurde vorgestellt, der auf der Koppelnavigation basiert und – in Abhängigkeit der Bewegungsparameter der Objekte – die Anzahl ausgetauschter Nachrichten bei der Nahbereichserkennung weiter optimiert. Ein Java Applet zur Visualisierung und Simulation der Strategien ist verfügbar unter http://www.mobile.ifi.lmu.de/TraX/.

Die verschiedenen Ansätze wurden auch bezüglich ihrer praktischen Anwendbarkeit evaluiert und diskutiert. Während sich der Streifenalgorithmus gut für Peer-to-Peer-Umgebungen eignet, ist die statische Kreisstrategie empfehlenswert, falls ein hoher Wert für die Grenzlinientoleranz akzeptabel ist. Die DSC-Strategie verhält sich am besten bezüglich ausgetauschter Nachrichten, falls keine Informationen über das voraussichtliche Bewegungsverhalten der Objekte vorliegen. Andernfalls lässt sich bei der Nahbereichserkennung durch die Koppelnavigation weiterer Nachrichtenaufwand einsparen. Soll nicht nur hinsichtlich der Nachrichtenanzahl optimiert werden, sondern ist gleichzeitig eine besonders einfache

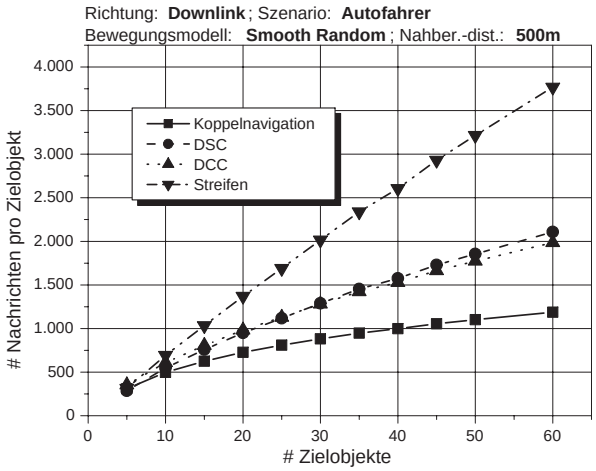

Abbildung 4.23: Downlink-Nachrichten pro Zielobjekt beim Autofahrer-Szenario

und robuste Implementierung gewünscht, sind die dynamisch-zentrierten Kreise vorzuziehen. Diese lassen sich sowohl für geographische Koordinaten als auch für topologische Distanzbeziehungen anwenden.

In Abschnitt 7.2 wird genauer erklärt, wie sich die Funktionen zur Nahbereichs- und Trennungserkennung in die später vorgestellte TraX-Plattform eingliedern. Im folgenden Kapitel wird jedoch erst anhand der Erkennung von Cliquen gezeigt, dass die beiden Funktionen wiederum von anderen, höherwertigen Methoden genutzt werden, was bereits einen Ausblick auf die Funktionsweise der geschichteten TraX-Architektur liefert.

5 Proaktive Erkennung von Cliquen

Neben der Nahbereichs- und Trennungserkennung ist die automatische Erkennung von *Cliquen* ein weiterer Basismechanismus zur Realisierung proaktiver Mehrpersonen-LBCSs: In der Graphentheorie bezeichnet eine Clique in einem ungerichteten Graph G eine Menge von Knoten V, bei der alle Elemente paarweise durch eine Kante verbunden sind. V ist also ein voll vermaschter Teilgraph von G. Die Größe einer Clique entspricht der Anzahl enthaltener Knoten. Angelehnt an die graphentheoretische Definition versteht sich in dieser Arbeit eine Clique $C_n = \{t_1, ..., t_n\}$ als Menge von n mobilen Zielpersonen (=Knoten), bei der sich alle möglichen Paare $t_i, t_j \in C_n, i \neq j$ zueinander im Nahbereich befinden (=Kante). Das Problem der Erkennung von Cliquen ist folgendermaßen definiert: Gegeben sei eine Menge S der Größe s von mobilen Zielpersonen. Dann soll automatisch erkannt werden, wenn sich eine Clique C_n der Größe $2 \leq n \leq s$ gebildet hat, und ihre Zusammensetzung soll bestimmt werden. Die Erkennung von Cliquen ist also eine Verallgemeinerung der Nahbereichserkennung, die das Problem für $n = 2$ behandelt.

Für die Erkennung von Cliquen lassen sich verschiedene Anwendungsfelder vorstellen, zum Beispiel proaktive Friend-Alert-Dienste, die eine Gruppe von Freunden (umgangssprachlich auch „Clique“ genannt) auf ihre räumliche Nähe hinweisen. Eine weitere Möglichkeit sind mobile Mehrbenutzerspiele, die die Erkennung von Cliquen als Teil des Spielflusses vorsehen. Ein weiteres Feld, das sich nicht nur auf LBCSs bezieht, ist die Logistik. Hier muss automatisch festgestellt werden, wann sich eine bestimmte Menge von Gütern am selben Ort befindet. Ein anderes Anwendungsgebiet ist CSCW, wo es zum Beispiel nützlich wäre, eine Präsentation zu starten, sobald sich genügend der gewünschten Personen zu einem Meeting versammelt haben. Auch könnte das so genannte Web 2.0 allgemein von Mechanismen zur Untersuchung der räumlichen Beziehungen von Community-Teilnehmern profitieren. Die automatische Cliquenerkennung würde grundsätzlich bestehende MoSoSo-Produkte aufwerten, die bislang nur reaktiv arbeiten.

In einem naiven Ansatz zur Cliquenerkennung könnten die Endgeräte der Zielpersonen die lokal gemessenen Ortsinformationen dem Location Server entweder periodisch oder anhand einer festen Update-Distanz übermitteln. Server-seitig würden die berichteten Positionen ständig verglichen und auf Cliquenbildung hin überprüft. Das Problem bei diesem Ansatz ist jedoch der übermäßige Nachrichtenaustausch, der umso höher ist, desto größer die gewünschte räumliche bzw. zeitliche Genauigkeit der Cliquenerkennung sein soll. Gleichzeitig entsteht eine sehr hohe Rechenlast beim Location Server.

In diesem Kapitel wird daher eine effizientere Methode vorgestellt, die versucht, die Anzahl der zwischen Endgerät und Server ausgetauschten Nachrichten zu minimieren. Der Ansatz ist auch in [126] beschrieben. Genau wie bei der Nahbereichs- und Trennungserkennung hat die Reduzierung ausgetauschter Nachrichten die folgenden Vorteile: Die Luftschnittstelle wird entlastet, mögliche monetäre Kosten der Zielpersonen für die Benutzung mobiler Trägerdienste wie GPRS oder UMTS werden reduziert, und der durch den Versand der Nachrichten entstehende Batterieverbrauch beim Endgeräts wird gedrosselt. Außerdem

wird die Rechenlast beim Location Server gesenkt, was dessen Skalierbarkeit in Bezug auf gleichzeitig bedienbare Endgeräte erhöht.

Im Gegensatz zu einem konstruktiven Ansatz ist die grundlegende hier vorgestellte Idee, die Nichtexistenz einer Clique so lange zu beweisen, wie die Clique nicht besteht. Aufbauend auf Erkenntnissen aus der Graphentheorie, die später dargelegt werden, wird ein solcher Beweis erreicht, indem die Zielobjekte aus S in so genannte *Independent Sets* eingeteilt werden. Ein Independent Set ist eine Teilmenge aus S, deren Elemente sich paarweise *nicht* im Nahbereich befinden. Um die Gültigkeit der Independent Sets zu überwachen und um erkannte Cliquen hinsichtlich ihres Zerfalls zu überprüfen, wird die Nahbereichs- und Trennungserkennung verwendet. Die vorgeschlagene Strategie zur Erkennung von Cliquen zeichnet sich also dadurch aus, dass kein völlig neues, spezialisiertes Protokoll zur Verfolgung der Zielobjekte erdacht wird. Stattdessen wird die Nahbereichs- und Trennungserkennung dynamisch auf Paare von Zielobjekten angewendet. Wie sich zeigt, müssen nur einige wenige der möglichen Paare aus S gleichzeitig betrachtet werden, was den erforderlichen Nachrichtenaufwand stark reduziert.

Dieses Kapitel gliedert sich wie folgt. Der nächste Abschnitt erklärt den neuen Ansatz zur Erkennung von Cliquen, bei dem die Nahbereichs- und Trennungserkennung selektiv auf Paare von Zielobjekten angewendet wird. In Abschnitt 5.2 wird der Ansatz bezüglich des erzeugten Nachrichtenaufwands sowie anderer Metriken anhand von drei verschiedenartigen Bewegungsmustern evaluiert. Der Ansatz wird auch mit einer rudimentären Referenzstrategie verglichen. Abschnitt 5.3 fasst verwandte Arbeiten zusammen und zeigt auf, wie der Ansatz mit einer existierenden Lösung für ein ähnliches Problem kombiniert werden kann. Abschnitt 5.4 fasst das Kapitel zusammen.

5.1 Strategie zur Cliquenerkennung

Die im Folgenden vorgestellte Strategie zur Cliquenerkennung basiert vollkommen auf der im letzten Kapitel entwickelten Nahbereichs- und Trennungserkennung. Es kommt also prinzipiell jede der behandelten Strategien zur Nahbereichs- und Trennungserkennung auch für die Cliquenerkennung in Frage. Dabei wird ein graphenbasierter Ansatz gewählt, der die Zielpersonen als Knoten modelliert. Eine Kante zwischen zwei Knoten bedeutet, dass die räumliche Distanz $dist(t_i, t_j)$ zwischen den entsprechenden Zielpersonen t_i und t_j kleiner ist als eine vordefinierte *Cliquendistanz* $d_c > 0$. Wie schon erwähnt, ist eine Clique C_n eine Menge von n paarweise durch eine Kante verbundener Knoten. Durch die gleichen Umstände motiviert wie bei der Nahbereichs- und Trennungserkennung (Abschnitt 4.2), wird eine mit der Cliquenerkennung assoziierte *Grenzlinientoleranz* $b_c > 0$ eingeführt: Wenn $d_c \leq dist(t_i, t_j) \leq d_c + b_c$, dann können t_i und t_j als verbunden angesehen werden, müssen es aber nicht. Die beiden Zielpersonen können dann also als potentielle Kandidaten für eine Clique zugelassen werden, dies muss aber nicht der Fall sein.

Wird eine Clique erkannt, so wird sie darauf hinsichtlich ihres Zerfalls beobachtet, welcher wiederum ein entsprechendes Ereignis auslösen soll. Der Einfachheit halber wird, so lange eine Clique besteht, keines ihrer Elemente in andere Cliquen, die im Zuge derselben gestellten Anfrage detektiert werden, einbezogen. Alle Cliquenbildungen unter den verbleibenden Knoten aus S müssen jedoch trotzdem noch erkannt werden.

Der vorgeschlagene Ansatz unterscheidet sich von existierenden Graphenalgorithmen

darin, dass die Kanten des Graphen nicht vollständig bekannt sind. Stattdessen ist zum Existenzbeweis einer Kante eine sich in Ausführung befindliche Trennungserkennung zwischen den Zielpersonen notwendig. Soll die Nichtexistenz einer Kante gezeigt werden, so muss eine entsprechende Nahbereichserkennung im Gange sein. Beide Operationen erzeugen natürlich Kosten, die sich in der Anzahl zwischen Endgerät und Location Server ausgetauschter Nachrichten niederschlagen. Sie sollten also so selten wie möglich angewendet werden. Die Idee des vorgestellten Ansatzes ist folglich, beweisbar alle möglichen Cliquen zu erkennen, dabei jedoch die Anzahl beobachteter Knotenpaare so weit wie möglich zu reduzieren. Wie sich zeigt, muss dazu oft nur eine geringe Anzahl von Kanten durch die Nahbereichserkennung überwacht werden, was den Gesamtnachrichtenaufwand stark verringert.

Der vorliegende Abschnitt gliedert sich wie folgt. Erst werden die verschiedenen Beobachtungszustände, die zwischen Paaren von Zielpersonen möglich sind, formal definiert. Auch werden die Übergänge zwischen diesen Zuständen beschrieben. Dann werden verschiedene Folgerungen aus der Graphentheorie, die die Grundlage des Ansatzes bilden, erläutert. Schließlich wird der ereignisgesteuerte Algorithmus zur Erkennung von Cliquen im Detail dargelegt.

5.1.1 Beobachtungszustände

Für jeweils ein Paar von Zielpersonen lassen sich grundsätzlich vier Beobachtungszustände unterscheiden, zwischen denen der vorgestellte Algorithmus dynamisch wechselt:

- Der *lange* Zustand bezeichnet eine ablaufende Nahbereichserkennung zwischen einem Paar von Zielpersonen. Die Nahbereichsdistanz d_p wird dabei der Cliquendistanz d_c gleichgesetzt. Die Grenzlinientoleranz der Nahbereichserkennung b_p kann auf jeden Wert zwischen 0 und der mit der Cliquenerkennung verbundenen Grenzlinientoleranz b_c gesetzt werden. Laut der Definition der Nahbereichserkennung ist somit die Distanz des Paares sicher größer gleich d_c. Dies sichert das Nichtvorhandensein einer Kante zwischen den entsprechenden Knoten im Graphen.

- Der *kurze* Zustand bedeutet, dass zwei Zielpersonen gerade auf Trennung hin überwacht werden. Die Trennungsdistanz d_s wird dabei mit $d_c + b_p$ gleichgesetzt. Die Grenzlinientoleranz der Nahbereichserkennung b_s wird auf $b_c - b_p$ gesetzt. Auf diese Weise sind die Unschärfeintervalle der Nahbereichs- und Trennungserkennung disjunkt. „Ping-Pong-Effekte“, die auftreten würden, wenn Objekte gleichzeitig als nah und getrennt betrachtet werden können, werden dadurch vermieden. Die Distanz zweier Objekte, die sich zueinander im kurzen Beobachtungszustand befinden, ist somit sicher kleiner gleich $d_s + b_s = d_c + b_p + b_c - b_p = d_c + b_c$. Von der Existenz einer entsprechenden Kante kann daher ausgegangen werden.

- Ist keinerlei Information über das räumliche Verhältnis zweier Zielpersonen bekannt, so befindet sich das Paar im Zustand *unbekannt*. Der Einfachheit halber wird dieser Zustand in den Abbildungen unten nicht graphisch dargestellt.

- Ein Paar von Zielpersonen, das zwar nicht aktiv beobachtet wird, welches aber vor kurzem ein Nahbereichsereignis ausgelöst hat, befindet sich im Zustand *schwebend*.

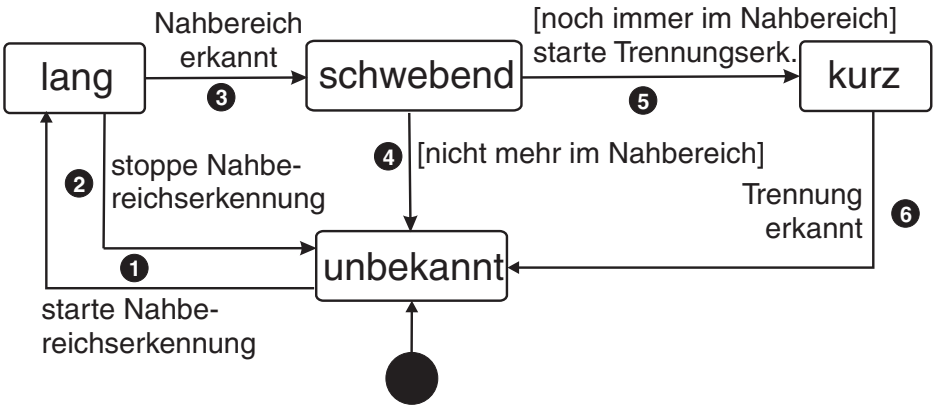

Abbildung 5.1: Übergänge zwischen den Beobachtungszuständen

> Der Zweck dieses Zustands ist es, die Paare, die sich vermutlich im Nahbereich befinden, von den anderen zu unterscheiden, ohne eine aktive Trennungserkennung einzusetzen. Diese soll nämlich so lange wie möglich hinausgezögert werden, um überflüssige Nachrichten einzusparen.

Hervorzuheben ist, dass nur die Zustände *lang* und *kurz* aktive Beobachtungen sind, bei denen wirklich Nachrichten ausgetauscht werden.

Das assoziierte Zustandsübergangsdiagramm wird in Abbildung 5.1 gezeigt. Demnach beginnt eine Beobachtung eines Paares immer im Zustand *unbekannt*. Von dort aus kann nur der Zustand *lang* erreicht werden, was durch eine Initiierung der Nahbereichserkennung erreicht wird (1). Der lange Zustand wird entweder verlassen, wenn die Nahbereichserkennung explizit angehalten wird (2), was eine Rückkehr zu *unbekannt* bewirkt, oder wenn erkannt wird, dass sich das Paar im Nahbereich befindet (3). In diesem Fall wird zum Zustand *schwebend* übergegangen. Der schwebende Zustand wird verlassen, wenn die Positionen der Zielpersonen zu einem späteren Zeitpunkt explizit angefragt und erneut verglichen werden. Befinden sich die beiden dann immer noch im Nahbereich (5), so wird die Trennungserkennung gestartet, und der Zustand geht in *kurz* über. Andernfalls (4) wird zu *unbekannt* zurückgekehrt. Die einzige Weise, den kurzen Zustand zu verlassen, ist ein entsprechendes Trennungsereignis (6). Es führt wiederum zum Neustart im Zustand *unbekannt*.

5.1.2 Graphentheoretische Grundlagen

Die grundlegende Idee des Ansatzes ist es, die Nichtexistenz einer Clique zu beweisen, solange sich diese nicht gebildet hat. Gemäß den folgenden Erkenntnissen aus der Graphentheorie ist ein solcher Beweis möglich, auch wenn nur ein kleiner Teil aller möglichen Paarbeziehungen aktiv beobachtet wird.

Zum Zeigen der Nichtexistenz einer Clique wird das Konzept so genannter *Independent Sets* verwendet. Ein Independent Set ist eine Teilmenge $I \subseteq S$ der Größe $i, 1 \leq i \leq s$ von Knoten, die paarweise *nicht* mit einer Kante verbunden sind. Es ist also bekannt, dass sich die Zielpersonen aus I paarweise nicht innerhalb der Nahbereichsdistanz befinden. Folglich sind alle möglichen Paare eines Independent Sets im langen Beobachtungszustand, das heißt, die Nahbereichserkennung ist aktiviert.

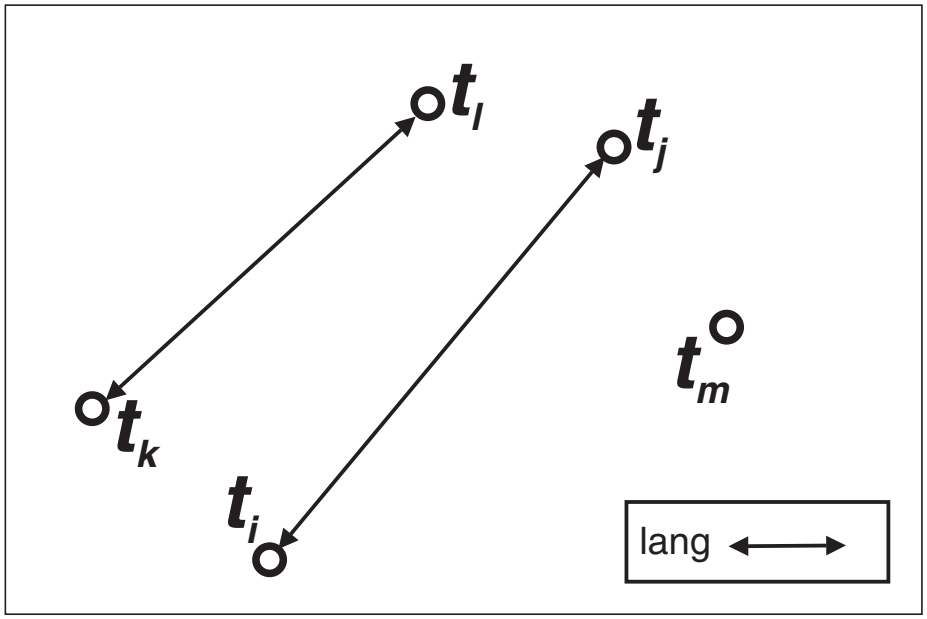

Abbildung 5.2: Beweis der Nichtexistenz einer Clique ($n = 4$, $s = 5$) mit drei Independent Sets

Die *chromatische Zahl* oder auch *Knotenfärbungszahl* $\gamma(G)$ eines Graphen G entspricht der kleinsten Anzahl von Farben, die benötigt werden, um die Knoten von G so einzufärben, dass keine zwei durch eine Kante verbundenen Knoten dieselbe Farbe haben. $\gamma(G)$ gleicht außerdem der minimalen Zahl an Independent Sets, in die sich die Knoten von G aufteilen lassen. Letzteres ist für den gewünschten Beweis entscheidend. Die Aussage lässt sich leicht einsehen, indem man genau diejenigen Knoten in ein Independent Set gruppiert, die dieselbe Färbung aufweisen.

Die *Cliquenzahl* $\omega(G)$ eines Graphen G bestimmt die Größe der größten Clique in G. Es gilt, dass die chromatische Zahl eines Graphen stets größer oder gleich der Cliquenzahl ist, $\gamma(G) \geq \omega(G)$, was sich leicht nachvollziehen lässt: Offensichtlich können keine zwei Knoten desselben Independent Sets in der gleichen Clique enthalten sein. Demzufolge kann selbst die größte Clique eines Graphen nur aus höchstens einem Knoten pro Independent Set bestehen. Nachdem die minimale Anzahl von Independent Sets gleich der chromatischen Zahl $\gamma(G)$ ist, muss also $\gamma(G) \geq \omega(G)$ gelten.

Der vorgeschlagene Algorithmus beruht auf diesen Erkenntnissen. Um die Nichtexistenz einer Clique C_n der Größe n zu zeigen, wird die Menge alle Zielobjekte S in $n-1$ Independent Sets aufgeteilt. Ist dies möglich, so muss gelten, dass $n - 1 \geq \gamma(G) \geq \omega(G)$, was Cliquen der Größe n ausschließt. Die Bedingungen aller Independent Sets werden kontinuierlich überwacht: Besteht ein Set nur aus einer Zielperson, so müssen keine Beobachtungen durchgeführt werden. Andernfalls muss für alle Paare von Zielpersonen innerhalb des Sets die Nahbereichserkennung aktiv sein (langer Zustand).

In dem Beispiel in Abbildung 5.2 wird eine Clique C_n der Größe $n = 4$ in einer Menge $S = \{t_i, t_j, t_k, t_l, t_m\}$ von $s = 5$ mobilen Zielobjekten gesucht. Gemäß den vorigen Erklärungen kann die Existenz einer solchen Clique ausgeschlossen werden, wenn sich die fünf Zielobjekte in drei Independent Sets aufteilen lassen, was in der Abbildung geschieht: Es gibt ein einzelnes, unbeobachtetes Zielobjekt (t_m), welches ein triviales Independent Set

darstellt. Außerdem existieren zwei Sets, die jeweils aus einem Paar von Objekten bestehen, (t_i, t_j) und (t_k,t_l), und die sich folglich im langen Beobachtungszustand befinden. Hervorzuheben ist an dieser Stelle, dass die Cliquenerkennung für $s = n$ mit lediglich einer sich in Ausführung befindlichen Nahbereichserkennung auskommt.

Nach [85] ist das Entscheidungsproblem der Bestimmung, ob die chromatische Zahl eines Graphen G unter einem gegebenen Wert k liegt, NP-hart. Aus diesem Grund versucht der unten beschriebene Algorithmus gar nicht erst, immer die beste Kombination von Independent Sets zu finden. Es wird vielmehr eine Approximation mit polynomieller Berechnungszeit benutzt, die zum einen entscheidet, ob eine Aufteilung in $n - 1$ Independent Sets möglich ist, und die zum anderen diese Aufteilung liefert. Wie sich bei den durchgeführten Simulation herausstellt, sind die Situationen, in denen der angenäherte Algorithmus scheitert, sehr selten[1]. Tritt eine solche Situation dennoch auf, werden einfach alle möglichen Paarbeziehungen aus S überwacht (entweder im langen oder kurzen Beobachtungszustand). Dies ist zwar aus Sicht des Nachrichtenaufwands nicht optimal, scheint aber im Hinblick auf die Seltenheit dieser Situation annehmbar.

5.1.3 Basisalgorithmus zur Cliquenerkennung

Der vorgeschlagene Algorithmus zur Cliquenerkennung verwaltet vier Mengen von Paarbeobachtungen: S_c enthält die Beobachtungen im kurzen Zustand, L_c die im langen Zustand, U_c diejenigen, die unbekannt sind, und P_c enthält die schwebenden Beobachtungen. Im Wesentlichen schiebt der Algorithmus die Beobachtungen zwischen diesen Mengen anhand der Zustandsübergänge aus Abbildung 5.1 hin und her. Es wird außerdem eine Menge X_c, die alle aktuellen Independent Sets enthält, verwaltet. Die Menge Y_c enthält alle aktuellen Cliquen, die kleiner als n sind. Wie bereits beschrieben, werden die Elemente der in X_c enthaltenen Independent Sets paarweise im langen Zustand beobachtet. Die Paarbeobachtungen innerhalb der Cliquen aus Y_c können entweder kurz oder schwebend sein, wobei die Verwendung des schwebenden Zustands zum Ziel hat, die nachrichtenintensive Trennungserkennung so gut wie möglich zu vermeiden.

Jedes Mal, wenn eine neue kurze oder schwebende Beobachtung hinzukommt, werden die Cliquen in Y_c neu organisiert. Nur bei einem solchen Ereignis kann die gesuchte Clique der Größe n erkannt werden. Eine Neuorganisation der Cliquen in Y_c wird auch durchgeführt, wenn eine kurze oder schwebende Beobachtung entfernt wird. In diesem Fall werden eine oder mehrere Cliquen jeweils in zwei kleinere Cliquen aufgeteilt. Der Algorithmus sowie die Datenstruktur zur Verwaltung der Cliquen in Y_c wurden exakt aus [92] übernommen und sollen daher an dieser Stelle nicht im Detail beschrieben werden. Um ein Ereignis bezüglich des Hinzufügens oder Löschens einer kurzen bzw. schwebenden Kante zu verarbeiten, ist eine rechnerische Komplexität von $O((n-1)\binom{s}{(n-1)})$ notwendig. Der Term ergibt sich aus der maximalen Größe von Y_c, die auf $\binom{s}{(n-1)}$ limitiert ist, da nur Cliquen enthalten sein können, die kleiner als n sind, multipliziert mit der Maximalgröße der Cliquen, $n - 1$.

X_c wird immer dann umorganisiert, wenn ein neuer Beweis basierend auf Independent Sets berechnet wird. Dies ist normalerweise dann der Fall, wenn eine Beobachtung im langen Zustand aufgrund eines Nahbereichsereignisses beendet wird. Als Folge der Reorgani-

[1] Die Ergebnisse beziehen sich auf die Verfolgung mobiler Zielobjekte und haben für allgemeine graphentheoretische Probleme keine Aussagekraft.

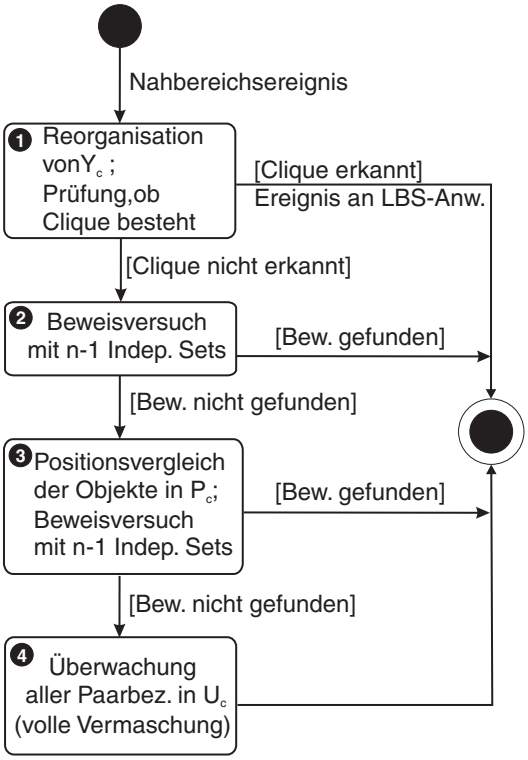

Abbildung 5.3: Prozedur zur Behandlung von Nahbereichsereignissen

sation der Independent Sets entstehen neue lange Beobachtungen, die von U_c nach L_c verschoben werden. Außerdem werden bestehende lange Beobachtungen, die von keinem der neuen Independent Sets mehr gebraucht werden, abgeschaltet, also von L_c nach U_c umgebucht. Der detaillierte Algorithmus zur Verwaltung der Independent Sets wird in Abschnitt 5.1.4 beschrieben.

Zu Beginn sind die Mengen S_c, L_c und P_c leer, und U_c beinhaltet alle möglichen Paare aus S. Y_c enthält s Cliquen, die jeweils aus einem einzelnen Zielobjekt bestehen. Entsprechend enthält X_c s Independent Sets, die jeweils ein Zielobjekt enthalten. Ein initialer Beweis der Nichtexistenz einer Clique C_n basierend auf Independent Sets wird berechnet. Als Ergebnis wird für ein oder mehrere Paare von Zielobjekten die Nahbereichserkennung aktiviert, was die entsprechenden Zustandsbeobachtungen von U_c nach L_c transferiert. X_c enthält jetzt $n-1$ Independent Sets. Die Nichtexistenz der gesuchten Clique ist dadurch zumindest so lange gesichert, bis eine der neuen langen Beobachtungen ein Nahbereichsereignis auslöst.

Der Rest des Algorithmus verläuft rein ereignisgesteuert. Zur Einfachheit werden alle Ereignisse sequentialisiert, das heißt, es wird davon ausgegangen, dass nur jeweils ein Ereignis zur selben Zeit abgehandelt wird. Zwei Ereignistypen werden unterschieden: erkannte Nahbereichsbetretungen sowie Trennungen zweier Zielobjekte. Die Verarbeitung eines Trennungsereignisses ist relativ einfach, da es nicht dazu führen kann, dass ein laufender Nichtexistenz-Beweis der gesuchten Clique ungültig wird. Die einzigen Konsequenzen eines Trennungsereignisses sind, dass die entsprechende Beobachtung von S_c nach U_c überführt wird und dass eine oder mehrere Cliquen aus Y_c aufgespaltet werden.

Die Behandlung von Nahbereichsereignissen ist hingegen aufwändiger: Wenn nach der Feststellung des Nahbereichs zwischen den Zielobjekten t_i und t_j ein Independent Set I

der Größe i existiert, welches bislang das Paar enthielt, so wird I in zwei Independent Sets aufgeteilt: ein triviales Set, welches entweder aus t_i oder t_j besteht – welches Zielobjekt ausgewählt wird, wird anhand einer Zufallsfunktion bestimmt –, sowie ein Independent Set der Größe $i-1$, welches die restlichen Objekte aus I enthält. Das Ergebnis ist, dass sich die Gesamtanzahl von Independent Sets in X_c um eins erhöht und dass folglich die Nichtexistenz einer Clique der Größe n nicht mehr gewährleistet ist. Die Beobachtung zwischen t_i und t_j wird von L_c nach P_c überführt, und alle anderen Beobachtungen, die bislang das abgespaltene Objekt mit den Elementen aus I verbunden haben, werden nach U_c verschoben. Dann werden die folgenden Schritte, die auch in Abbildung 5.3 zusammengefasst werden, nacheinander ausgeführt, wobei jeder Schritt vom Scheitern seines Vorgängers abhängt:

1. Die Cliquen in Y_c werden neu organisiert, und es wird geprüft, ob die durch das Nahbereichsereignis erkannte Kante zu einer Clique der Größe n führt. Ist dies der Fall, so wird geprüft, ob die Clique Paarbeobachtungen in P_c hat. Gegebenenfalls wird durch Pollings kontrolliert, ob die entsprechenden Paare die Nahbereichsbedingung noch immer erfüllen. Ist die Clique dann noch erhalten, so kann sie der anfragenden Anwendung gemeldet werden. Die Cliquenmitglieder werden dann von der Liste der beobachteten Zielobjekte gestrichen, und die Prozedur kehrt zurück.

2. Kann im ersten Schritt keine Clique der Größe n erkannt werden, so wird überprüft, ob ein Nichtexistenz-Beweis durch Erzeugung von $n-1$ Independent Sets (vergleiche Abschnitt 5.1.4) möglich ist, indem nur diejenigen Beobachtungen, die sich momentan in $U_c \cup L_c$ befinden, benutzt werden. Es sollen also in diesem Schritt keine expliziten Nahbereichsprüfungen stattfinden, welche potentiell Beobachtungen von P_c nach U_c freigeben könnten. Entsprechende Pollings werden also an dieser Stelle vermieden, da sie relativ teuer sind. Ist also die Erzeugung von $n-1$ Independent Sets an dieser Stelle schon möglich, werden die entsprechenden langen Beobachtungen initialisiert, und die Ereignisbehandlungsroutine kann zurückkehren.

3. Andernfalls werden die Positionen aller Zielobjekte, die Teil mindestens einer schwebenden Paarbeobachtung sind, erfragt (Polling) und miteinander verglichen. Gemäß den Zustandsübergängen aus Abbildung 5.1 werden alle Paare aus P_c, die sich dann immer noch im Nahbereich befinden, nach S_c überführt und aktiv bezüglich Trennung observiert. Alle getesteten Paare aus P_c, die sich mittlerweile voneinander entfernt haben, werden hingegen nach U_c umgebucht. Als Ergebnis enthält U_c nun möglicherweise mehr freie Paare als zuvor. Es wird daher erneut versucht, mit Hilfe der in $U_c \cup L_c$ enthaltenen Paare, $n-1$ Independent Sets zu erstellen. Funktioniert dies, wird die Routine beendet.

4. Wie bereits erwähnt, gelingt der Beweis basierend auf $n-1$ Independent Sets in den allermeisten Fällen, was bedeutet, dass dieser Schritt normalerweise nicht ausgeführt werden muss. In seltenen Fällen ist der Beweis jedoch nicht möglich, weil sich selbst nach dem Erfragen und Vergleichen der Positionen der Paare aus P_c zu wenige Kandidaten in U_c befinden, aus denen für die benötigten langen Beobachtungen ausgewählt werden kann.

   Abbildung 5.4 zeigt ein Beispiel, bei dem eine Clique der Größe $n = 3$ in einer Menge von $s = 5$ verfolgten Zielobjekten erkannt werden soll. Die Nichtexistenz

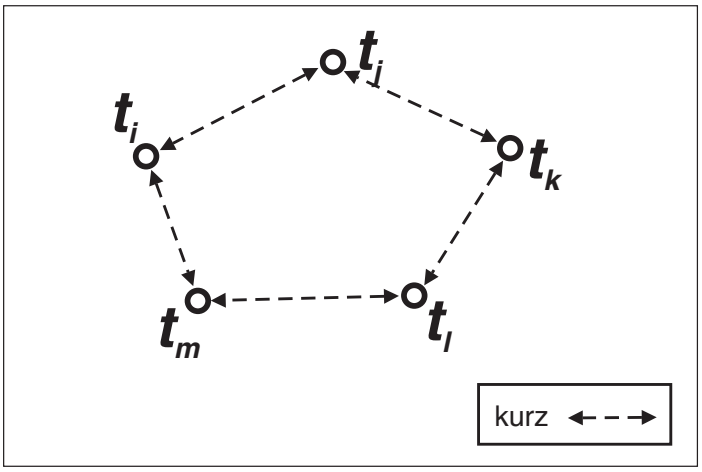

Abbildung 5.4: Beweis der Nichtexistenz einer Clique ($n = 3$, $s = 5$) mit Independent Sets unmöglich

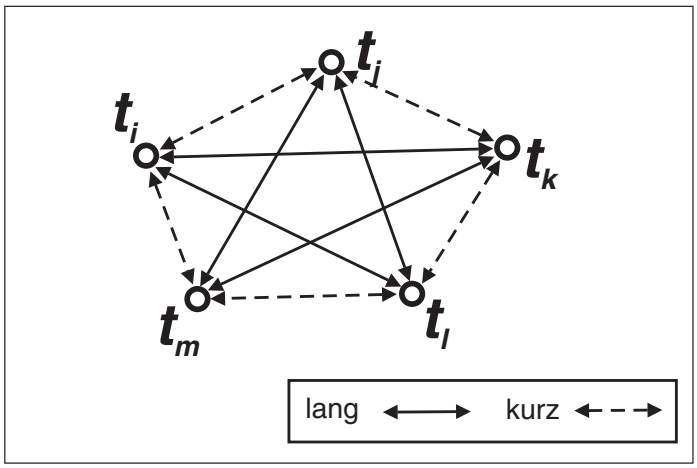

Abbildung 5.5: Beweis der Nichtexistenz einer Clique ($n = 3$, $s = 5$) durch volle Vermaschung

der Clique ließe sich beweisen, wenn die Objekte in zwei Independent Sets aufgeteilt werden könnten, was einer 2-Färbung des Graphen entspricht. Nachdem der Graph jedoch einen ungeraden Zyklus enthält, wie sich anhand der eingezeichneten kurzen Beobachtungen erkennen lässt, ist eine solche Färbung unmöglich.

Wie in Abbildung 5.5 gezeigt wird, lässt sich die Nichtexistenz der Clique dennoch zeigen – allerdings nur, indem alle in U_c verbleibenden Paare für die Nahbereichserkennung aktiv geschaltet werden, was sie nach L_c verschiebt. Alle möglichen Kanten des Graphen werden nun beobachtet, und die Prozedur kann zurückkehren. Weitere Nahbereichs- und Trennungsereignisse werden so behandelt, als würde der Beweis auf Independent Sets basieren. Für jedes weitere Nahbereichsereignis wird die gesamte Prozedur wiederholt, und es wird wiederum versucht, einen Beweis basierend auf Independent Sets herzustellen.

5.1.4 Erzeugung von $n-1$ Independent Sets

In den Schritten (2) und (3) der oben vorgestellten Prozedur wird jeweils versucht, die Anzahl von Independent Sets, die die Knoten aus S abdecken, auf $n-1$ zu reduzieren. Zu diesem Zweck werden im Folgenden zwei Funktionen vorgestellt, die sich wiederholt auf die in X_c enthaltenen Sets anwenden lassen: *Verschmelzungen* und *Verteilungen*. Beide Funktionen haben zum Ziel, die Fluktuation zwischen den Mengen U_c und L_c so gering wie möglich zu halten. Es sollen also so wenig wie möglich sich in Ausführung befindliche Beobachtungen verändert werden, um den mit der Aktion verbundenen Nachrichtenaufwand und die rechnerische Komplexität gering zu halten. Verschmelzungen sind dabei effizienter als Verteilungen, was der Grund dafür ist, dass sie immer zuerst ausprobiert werden. Eine Verteilung wird nur dann versucht, wenn keine mögliche Verschmelzung gefunden werden kann. Erst wenn in Schritt (3) der Basisprozedur nicht einmal Verteilungen möglich sind, wird die volle Vermaschung (4) angewendet.

Verschmelzung

Eine Verschmelzung erhält alle bestehenden langen Beobachtungen der Independent Sets aus X_c und ist daher sehr effizient. Zwei Independent Sets I_a und I_b, für die gilt, dass alle Paare $(I_a \times I_b)$ in $U_c \cup L_c$ enthalten sind, werden dabei aus X_c ausgewählt. Diese Paare werden dann in den langen Beobachtungszustand L_c transferiert. Dies bewirkt, dass I_a und I_b zu einem einzigen Independent Set verschmelzen, was die Gesamtzahl an Independent Sets um eins reduziert.

Ein Beispiel findet sich in Abbildung 5.6. Zuerst gibt es zwei Independent Sets, (t_i, t_j, t_m) und (t_k, t_l), die die Nichtexistenz einer Clique der Größe $n := 3$ garantieren. Wenn jedoch ein Nahbereichsereignis zwischen t_i und t_j auftritt (1), wird (t_i, t_j, t_m) in (t_i) und (t_j, t_m) aufgeteilt, und es verbleiben drei Independent Sets auf dem Spielfeld (2). Wie auch anhand des Beispiels nachvollziehbar ist, stehen oft mehrere Kandidaten für eine Verschmelzung zur Verfügung (im Beispiel könnten sowohl $(t_j, t_m) - (t_k, t_l)$ als auch $(t_i) - (t_k, t_l)$ verschmolzen werden). In der Abbildung wird $(t_i) - (t_k, t_l)$ ausgewählt (3), und es existieren wieder nur zwei Independent Sets: (t_i, t_k, t_l) und (t_j, t_m). Die Nichtexistenz einer Clique ist somit bis zum nächsten Nahbereichsereignis wieder gesichert.

Verteilung

Die Verteilung wird versucht, wenn sich keine zwei zur Verschmelzung geeigneten Independent Sets in X_c finden lassen. In diesem Fall befindet sich also zwischen allen möglichen Kombinationen aus Independent Sets mindestens eine kurze bzw. schwebende Beobachtung. Verteilungen verursachen einen größeren Nachrichtenaufwand als Verschmelzungen, da hierbei lange Beobachtungen bewusst abgebrochen werden. Die Beobachtungen werden also explizit in den unbekannten Zustand versetzt. Ein Independent Set I_a kann dann verteilt werden, wenn es für jedes seiner Elemente t_a ein anderes Independent Set I_b aus X_c gibt, so dass sich kein Element $t_b \in I_b$ in einer kurzen oder schwebenden Beobachtung zu t_a befindet. Trifft diese Bedingung zu, so kann jedes Element von I_a von einem anderen Set absorbiert werden, und I_a verschwindet. Dies führt wiederum dazu, dass sich die Gesamtzahl an Sets um eins reduziert.

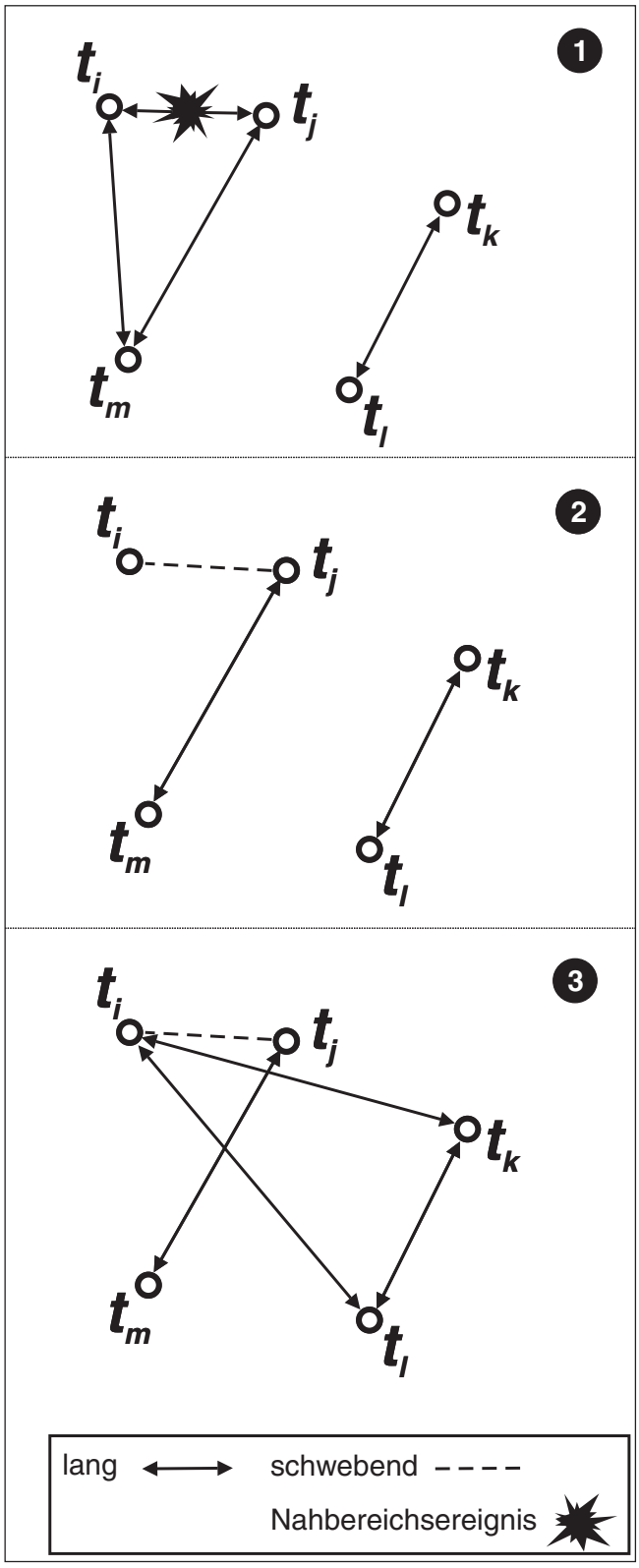

Abbildung 5.6: Verschmelzung in Konsequenz eines Nahbereichsereignisses ($n = 3$, $s = 5$)

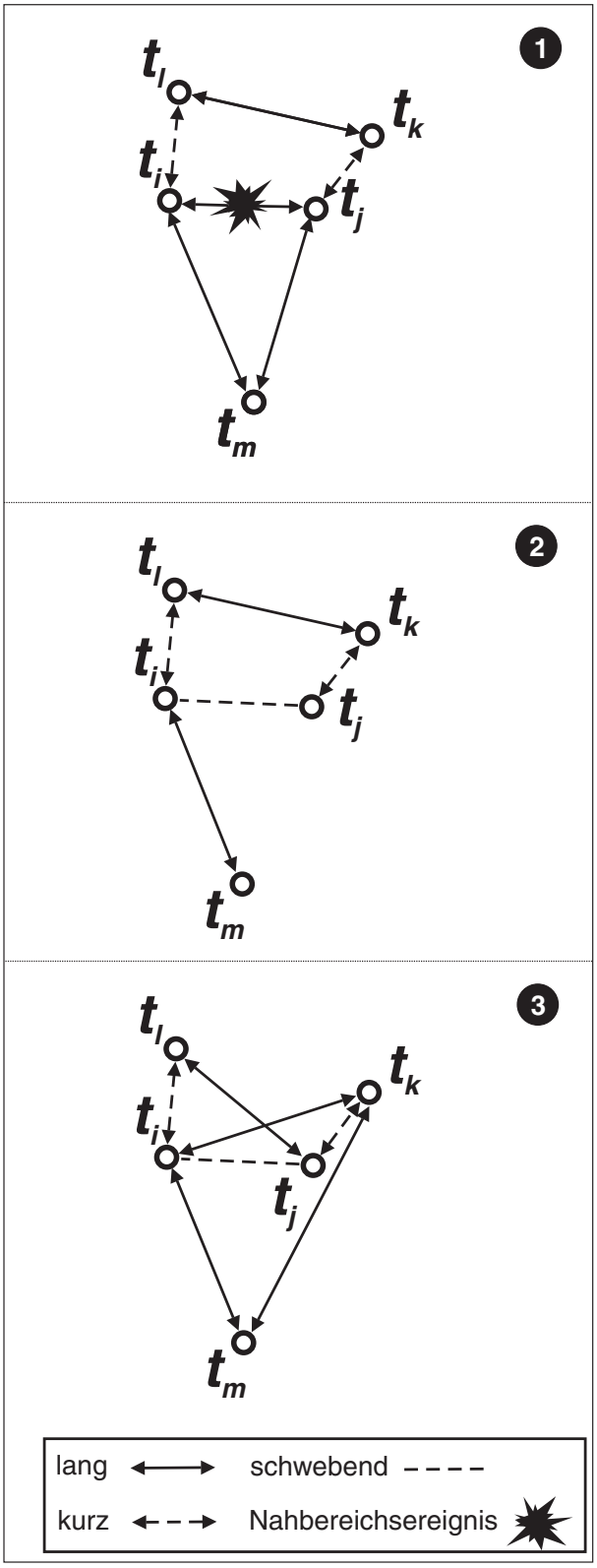

Abbildung 5.7: Verteilung in Konsequenz eines Nahbereichsereignisses ($n = 3$, $s = 5$)

Ein Beispiel für eine Verteilung findet sich in Abbildung 5.7. Wieder zerfällt zu Anfang das Independent Set (t_i, t_j, t_m) in zwei Teile, (t_j) und (t_i, t_m), aufgrund eines Nahbereichsereignisses zwischen t_i und t_j (1). Diesmal ist jedoch keine Verschmelzung möglich, da zwischen den verbleibenden drei Sets jeweils kurze bzw. schwebende Beobachtungen bestehen (2). Das einzige mögliche Kandidaten-Set, (t_l, t_k), wird daher verteilt: t_l wird von (t_j) aufgenommen und t_k von (t_i, t_m) (3). Die Nichtexistenz einer 3-Clique ist somit wieder gesichert.

Substrategien

Bei der Evaluierung, deren Ergebnisse im nächsten Kapitel vorgestellt werden, wurden zwei verschiedene Substrategien verglichen. Sie beziehen sich sowohl auf Verschmelzungen als auch auf Verteilungen. Wie bereits erklärt, entstehen oft Situationen, bei denen sich mehr als ein Paar von Independent Sets für eine Verschmelzung bzw. mehr als ein Set für eine Verteilung eignet. Bei der Entscheidung, welches Paar oder Set ausgewählt wird, sind zwei, zum Teil widerstrebende Ziele möglich:

Eine mögliche Zielvorgabe wäre, die Gesamtanzahl langer Beobachtungen, die zum Zeigen von $n - 1$ Independent Sets verwendet werden, möglichst gering zu halten. Die damit verbundene Substrategie, welche im Folgenden mit *minimale Beobachtungen* bezeichnet wird, generiert also vorzugsweise kleine Independent Sets, da die Anzahl langer Beobachtungen quadratisch mit der Größe eines Sets ansteigt. Ist in einer bestimmten Situation zum Beispiel eine Verschmelzung zweier einelementiger Sets als auch zweier zweielementiger Sets möglich, so wird die erste Kombination ausgewählt, da sie nur eine zusätzliche lange Beobachtung erfordert, im Gegensatz zu vier neuen Beobachtungen im zweiten Fall. Zur Verteilung wählt die Substrategie mit minimalen Beobachtungen immer das größte Independent Set unter den möglichen Kandidaten aus, da hierdurch die größte Anzahl langer Beobachtungen aufgelöst wird.

Die alternative Substrategie, welche mit *minimale Knoten* bezeichnet wird, versucht hingegen die Gesamtzahl beobachteter Zielpersonen zu reduzieren. Dabei wird von der Annahme ausgegangen, dass es vorteilhaft ist, wenn eine Zielperson gar nicht beobachtet wird (was bei einelementigen Independent Sets der Fall ist) bzw. wenn einige Zielpersonen in eine geringere Zahl langer Beobachtungen verwickelt werden. Es soll so der Nachteil aufgewogen werden, dass andere Zielpersonen im Gegenzug intensiver verfolgt werden müssen. Die Substrategie versucht daher, so viele einelementige Independent Sets wie möglich zu generieren. Verschmelzungen, die solche Sets einbeziehen, werden daher möglichst vermieden. Bei Verteilungen werden vorzugsweise große absorbierende Sets ausgesucht und kleine Sets aufgeteilt.

5.2 Evaluierung

Der vorgestellte Ansatz wurde mittels zahlreicher Simulationen, die Zielobjekte auf einem Spielfeld von 10 km x 10 km bewegen und die die Cliquenerkennung für verschiedene Cliquengrößen durchführen, evaluiert. Die simulierte Zeit betrug für jeden Simulationslauf 20 h. Die Cliquendistanz d_c wurde auf 500 m und die Grenzlinientoleranz b_c auf 100 m gesetzt. Jeder Durchlauf wurde 100-mal wiederholt. Zum Erreichen einer hohen Aussagekraft wur-

den drei verschiedene Bewegungsmodelle verwendet, die im Folgenden beschrieben werden.

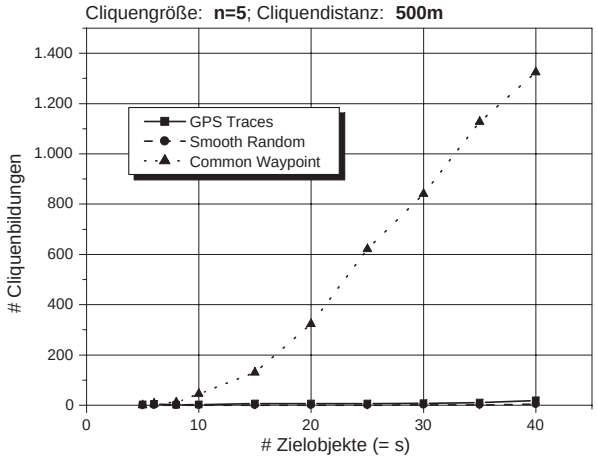

Abbildung 5.8: Anzahl gebildeter Cliquen

Zwei der Modelle gehen davon aus, dass die Bewegungen der Zielpersonen unabhängig voneinander sind. Cliquenbildungen sind somit rein zufälliger Natur und kommen relativ selten vor. Ein passendes Anwendungsszenario wäre zum Beispiel ein proaktiver Friend-Finder-Dienst. Die für dieses Szenario verwendeten Modelle sind, wie im letzten Kapitel erwähnt, das künstliche *Smooth-Random-Modell* [38] sowie die von Studenten aufgenommenen *GPS Traces*. Das Smooth-Random-Modell wurde auf eher langsame Geschwindigkeiten eingestellt (bis zu 20 km/h), jedoch mit nur sehr kurzen Bewegungspausen. Die GPS Traces, die ebenfalls bei relativ niedrigen Geschwindigkeiten aufgenommen wurden, weisen hingegen stärkere Schwankungen zwischen stationärem und mobilem Verhalten auf.

Das dritte Bewegungsmodell, im Folgenden als *Common Waypoint* bezeichnet, wurde selbständig entwickelt und motiviert sich durch Anwendungen wie Logistik, CSCW oder Mobile Gaming. Cliquenbildungen zwischen den Zielobjekten sind hier eher absichtlich und kommen dementsprechend häufiger vor. Die simulierten Bewegungen der Objekte sind also nicht unabhängig voneinander, was durch eine Anpassung des *Random-Waypoint-Modells* erreicht wurde. Bei diesem Modell wandern die simulierten Objekte jeweils zu einem zufällig gewählten Punkt (Waypoint) auf dem Spielfeld. Nachdem dieser erreicht wurde, wird eine zufällig gewählte Zeitdauer lang pausiert und daraufhin der nächste Waypoint bestimmt. Klassisches Random Waypoint bewirkt eine Ungleichverteilung der Objekte auf dem Spielfeld (vergleiche [39]) und sorgt somit bereits für eine erhöhte Anzahl von Cliquenbildungen. Dieser Effekt wurde noch künstlich verstärkt, indem die Menge der möglichen Waypoints auf einige wenige begrenzt wurde. Auf diese Weise teilen sich oft mehrere Zielobjekte zur selben Zeit denselben Waypoint, und die Wahrscheinlichkeit eines Aufeinandertreffens der Objekte nimmt zu. Die Geschwindigkeit und Pausenzeiten der Objekte wurden wie beim klassischen Waypoint-Modell unabhängig und zufällig gewählt, wobei relativ schnelle Bewegungen (bis zu 50 km/h) und kurze Pausen angenommen wurden.

Abbildung 5.8 zeigt die Anzahl gebildeter Cliquen an, die sich bei den Simulationen für die drei Bewegungsmuster ergab. Gesucht wurden jeweils Cliquen der Größe $n = 5$ innerhalb unterschiedlich großer Mengen von Zielobjekten , $s \in [5; 40]$. Wie beabsichtigt führen die jeweils 20 Stunden langen Bewegungen, die vom Smooth-Random-Modell bzw. durch die GPS Traces beschrieben werden, nur zu sehr wenigen Cliquenbildungen. Das Common-Waypoint-Modell generiert hingegen wesentlich mehr Cliquen, und zwar umso mehr, desto größer der Unterschied zwischen s und n.

Um die Leistungsfähigkeit des vorgeschlagenen Ansatzes zu untersuchen, wurde eine einfache Referenzstrategie zugrunde gelegt. Allen Zielobjekten wird dabei die gleiche statische Update-Distanz von $\frac{b_c}{4}$ zugewiesen. So ist garantiert, dass der Location Server zu jedem Zeitpunkt und für alle möglichen Paare von Objekten eindeutig entscheiden kann, ob die Nahbereichsbedingung erfüllt ist oder nicht. Dies ermöglicht es wiederum, alle vorkommenden Cliquenformationen zu erkennen.

Bezüglich des erzeugten Nachrichtenaufwands wird die gleiche Nachrichtenbelegung der Operationen Position Update, Position Update Request und Polling angenommen wie im vorigen Kapitel, vergleiche Tabelle 4.1. Die Referenzstrategie erhält dabei die gleiche Belegung wie die statische Kreisstrategie. Das bedeutet also, dass hier, im Gegensatz zu DCC und DSC, Position Updates im Downlink bestätigt werden.

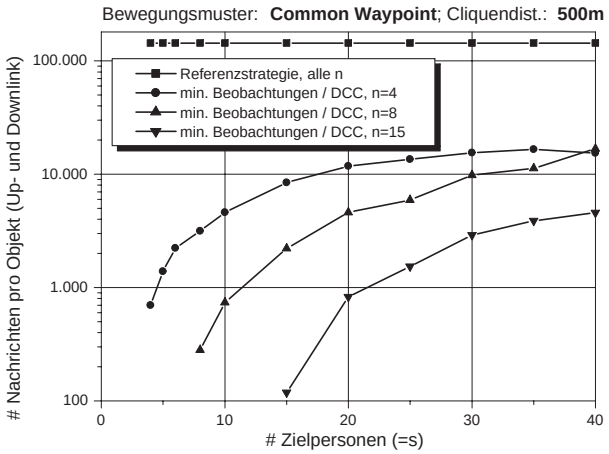

Abbildung 5.9: Nachrichten pro Zielobjekt bei Common Waypoint

Abbildungen 5.9 bis 5.11 vergleichen den neuen Ansatz, basierend auf DCC zur Nahbereichs- und Trennungserkennung und mit minimalen Beobachtungen als Substrategie, mit der Referenzstrategie. Für jedes der drei Bewegungsmodelle werden drei verschiedene Cliquengrößen ausprobiert, $n \in \{4, 8, 15\}$. Die Größe der Menge von Zielobjekten wird im Intervall $s \in [4; 40]$ variiert.

Kaum überraschend ist, dass die Nachrichtenanzahl pro Objekt, die durch die Referenzstrategie verursacht wird, unabhängig von s ist. Jedem Objekt wird schließlich dieselbe statische Update-Distanz zugewiesen. Auch wird die Nachrichtenanzahl bei der Referenzstrategie nicht von n beeinflusst, da, basierend auf den ständig übermittelten Positionen, alle

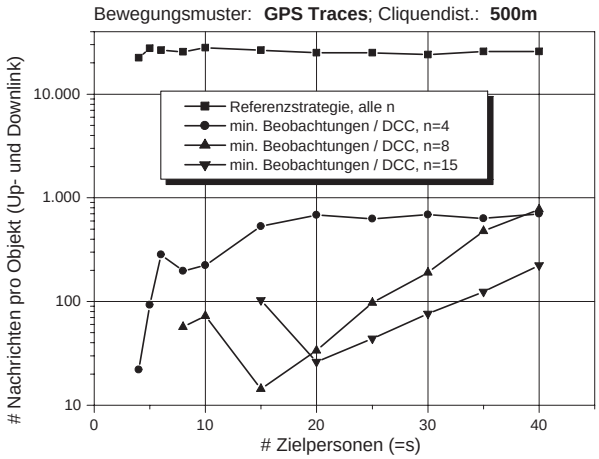

Abbildung 5.10: Nachrichten pro Zielobjekt bei den GPS Traces

möglichen Cliquengrößen erkannt werden können. Anhand der von der Referenzstrategie verwendeten statischen Update-Distanzen lässt sich das Verhalten der Bewegungsmodelle schön überblicken: Die durchschnittlich langsameren Bewegungen der GPS Traces und des Smooth-Random-Modells führen zu einer vergleichbaren Anzahl an Nachrichten (etwas weniger als 30.000), wobei die weniger regulären GPS Traces zu einer höheren Schwankung des Graphen führen. Die schnelleren und mehr geradeaus gerichteten Bewegungen von Common Waypoint bewirken ein allgemein erhöhtes Nachrichtenaufkommen (etwa 4,5-mal so hoch).

In den Experimenten zeigte sich, dass die vorgestellte Strategie zur Cliquenerkennung die Referenzstrategie in Bezug auf Nachrichteneffizienz klar schlägt, und zwar für alle Bewegungsszenarien und für alle getesteten Kombinationen aus s und n. Für $s = n = 4$ und basierend auf Smooth Random wird zum Beispiel nur 1,57 Prozent der Nachrichtenmenge der Referenzstrategie benötigt. Für das Common-Waypoint-Modell, welches mit höheren Geschwindigkeiten konfiguriert wurde und gleichzeitig mehr Zusammenstöße provoziert, reduziert sich dieser Wert sogar auf 0,49 Prozent, was circa 36 Nachrichten pro Objekt und Stunde entspricht. Für $s = n = 15$ liegt Smooth Random bei 0,42 Prozent und Common Waypoint bei 0,08 Prozent, was in beiden Fällen etwa 6 Nachrichten pro Objekt und Stunde gleichkommt und somit 6 mal weniger ist als im $s = n = 4$-Szenario. Offensichtlich passt sich die vorgestellte Strategie der Seltenheit von Cliquenformationen an (15-Cliquen innerhalb einer Menge von 15 Zielpersonen sind sogar bei Common Waypoint extrem selten) und überträgt Nachrichten wirklich nur dann, wenn sie notwendig sind.

Mit einem größer werdenden Unterschied zwischen s und n steigt die Nachrichtenzahl der neuen Strategie jedoch an, was klar wird, wenn man sich Abschnitt 5.1.2 in Erinnerung ruft: s Objekte auf $n - 1$ Independent Sets zu verteilen, erfordert mehr lange Beobachtungen (Nahbereichsüberwachungen), wenn die Differenz von s und n ansteigt. Zum Beispiel erfordert die Detektierung von 4-Cliquen in einer Menge von 20 Zielobjekten mit Common Waypoint als Bewegungsmodell 8,23 Prozent der Nachrichtenmenge der Referenzstrategie,

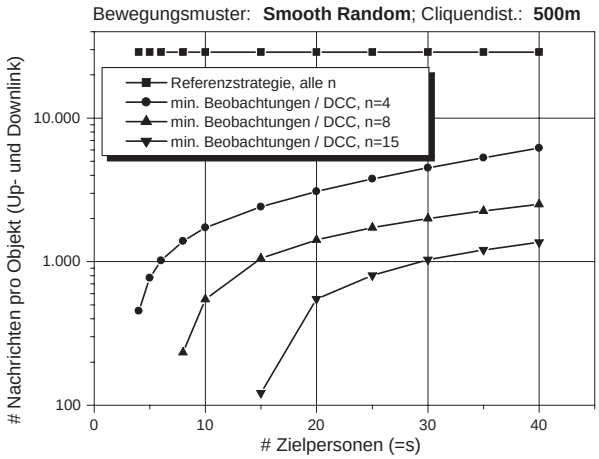

Abbildung 5.11: Nachrichten pro Zielobjekt bei Smooth Random

was etwa 590 Nachrichten pro Objekt und Stunde entspricht. Für dasselbe Szenario führt Smooth Random zu 10,69 Prozent, was 154 stündlichen Nachrichten gleichkommt.

Trotz des Nachrichtenzuwachses scheint die vorgeschlagene Strategie für viele Kombinationen aus n und s praktikabel zu sein. Sogar bei Common Waypoint, das zu mehr Cliquenformationen führt, explodiert der Nachrichtenaufwand nicht für die getesteten Parameter. Sollte jedoch s im Vergleich zu n viel größer sein, so könnte die Anzahl der durch die Independent Sets erzeugten langen Beobachtungen tatsächlich nicht mehr akzeptabel sein. In diesem Fall wird der hybride Ansatz aus dieser Strategie und der Strategie von Xu and Jacobsen, der in Abschnitt 5.3 skizziert wird, empfohlen.

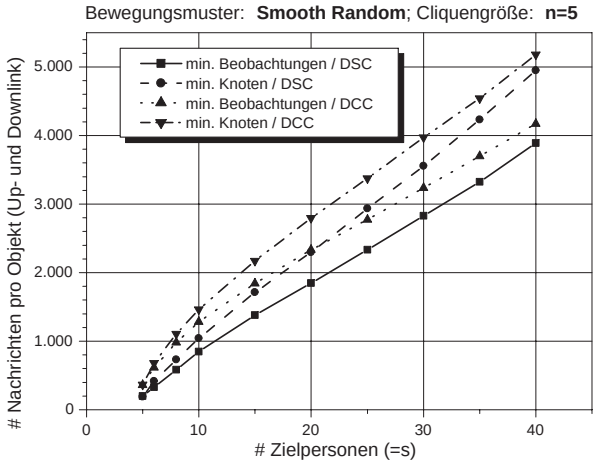

Abbildung 5.12: Nachrichten pro Zielobjekt bei den verschiedenen Substrategien

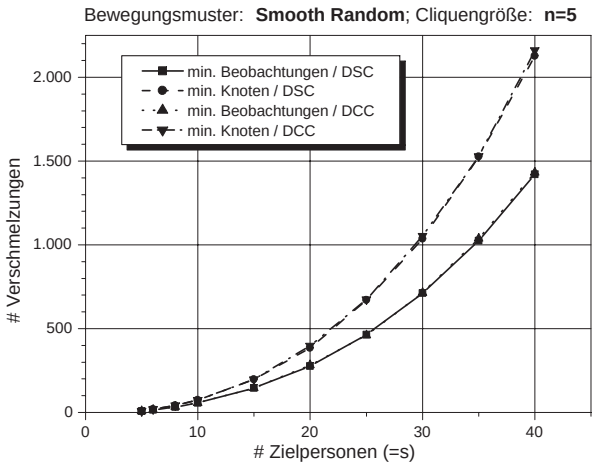

Abbildung 5.13: Anzahl von Verschmelzungen bei den verschiedenen Substrategien

Nachdem die prinzipielle Eignung der vorgeschlagenen Strategie gezeigt wurde, werden im Folgenden die Eigenschaften der zwei Substrategien, minimale Beobachtungen und minimale Knoten, jeweils in Kombination mit DCC und DSC zur Nahbereichs- und Trennungserkennung, verglichen.

Das Abschneiden dieser vier Kombinationen wird jeweils in den Abbildungen 5.12 (Nachrichten pro Objekt), 5.13 (Verschmelzungen insgesamt), 5.14 (Verteilungen insgesamt) und 5.15 (volle Vermaschungen insgesamt) angezeigt. Wie bereits erwähnt, werden während einer vollen Vermaschung alle möglichen Paare von Zielobjekten aus S aktiv überwacht, entweder auf Nahbereichs- (langer Zustand) oder Trennungsereignisse (kurzer Zustand) hin. Wegen des erhöhten Nachrichtenaufwands sind volle Vermaschungen möglichst zu vermeiden.

Zunächst sollte anhand von Abschnitt 5.1.3 klar sein, dass bei festem n die Anzahl von Paaren aus S, die bezüglich Nahbereich überwacht werden, in etwa quadratisch mit steigendem s anwachsen sollte. Wie das Diagramm zeigt, führt dies zu einem in etwa linearen Anstieg der Nachrichtenmenge pro Zielobjekt, und zwar für alle vier getesteten Kombinationen. Das erwartete quadratische Anwachsen der Anzahl von erkannten Nahbereichsereignissen mit steigendem Wert für $s - n$ wird auch durch die Diagramme der Verschmelzungen und der Verteilungen bestätigt: Diese Operationen kommen deshalb ebenfalls quadratisch häufiger vor, weil sie immer als Folge eines Nahbereichsereignisses ausgeführt werden.

Wie man anhand von Diagramm 5.15 sieht, kommen volle Vermaschungen bei allen Substrategien sehr selten vor, und aufgrund dieser Seltenheit scheint der Unterschied zwischen den vier Kurven eher an der hohen statistischen Varianz des Ereignisses zu liegen, anstatt eine methodische Ursache zu haben. Außer der Bestätigung eines allgemein seltenen Auftretens voller Vermaschungen kann also keine fundierte Aussage bezüglich der Unterschiede zwischen den vier Kombinationen gemacht werden.

Für Verschmelzungen und Verteilungen ergibt sich hingegen eine klarere Situation. Wie in Abschnitt 5.1.4 beschrieben wird, sollte die Substrategie der minimalen Beobachtun-

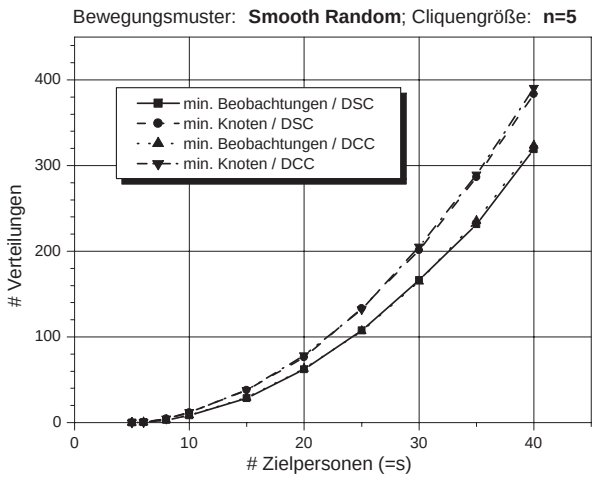

Abbildung 5.14: Anzahl von Verteilungen bei den verschiedenen Substrategien

gen zu weniger durchgeführten Nahbereichsüberwachungen als die der minimalen Knoten führen und somit entsprechend weniger Nahbereichsereignisse bewirken. Diese Vermutung wird durch die Diagramme 5.13 und 5.14 bestätigt, die zeigen, dass beide Ereignistypen, Verschmelzungen und Verteilungen, bei den minimalen Knoten häufiger auftreten als bei den minimalen Beobachtungen. Auch wird die korrekte Arbeitsweise von DSC und DCC bestätigt. Bei beiden Substrategien, der minimalen Knoten und der minimalen Beobachtungen, führt die Anwendung von DCC und DSC zur jeweils gleichen Anzahl von Verschmelzungen und Verteilungen. Aus der Sicht der vorgestellten Strategie zur Cliquenerkennung sind DCC und DSC also voll gegeneinander austauschbar (Transparenz der Strategie zur Nahbereichs- und Trennungserkennung).

Diagramm 5.12, das die Anzahl ausgetauschter Nachrichten anzeigt, dokumentiert eine interessante Entwicklung. Zuallererst lassen sich ein klarer Gewinner und ein klarer Verlierer feststellen: Die Kombination aus DSC und den minimalen Beobachtungen zeigt stets die besten Leistungseigenschaften, während DCC in Kombination mit den minimalen Knoten immer am schlechtesten abschneidet. Die Zuweisung von Platz zwei und drei hängt hingegen vom Wert von s ab: DSC zusammen mit den minimalen Knoten schneidet besser im Intervall $s \in [5; 15]$ ab, während DCC mit den minimalen Beobachtungen im Intervall $s \in [15; 40]$ besser ist. Allgemein weisen die zwei Kurven, die jeweils mit den minimalen Beobachtungen und den minimalen Knoten verknüpft sind, eine ähnliche Steigung auf. Das Kurvenpaar der minimalen Knoten weist jedoch eine höhere Steigung auf als das der minimalen Beobachtungen, was sich folgendermaßen erklären lässt. Für ein kleines s bewirken minimale Knoten eine höhere Anzahl einelementiger Independent Sets, die wünschenswert sind, da hierbei die Zielobjekte überhaupt nicht verfolgt werden müssen. Jedoch verschwinden diese einelementigen Sets mit ansteigender Differenz von s und n zunehmend. Gleichzeitig gewinnt die negative Eigenschaft der minimalen Knoten an Gewicht, dass mehr Nahbereichserkennungen als bei den minimalen Beobachtungen nötig werden.

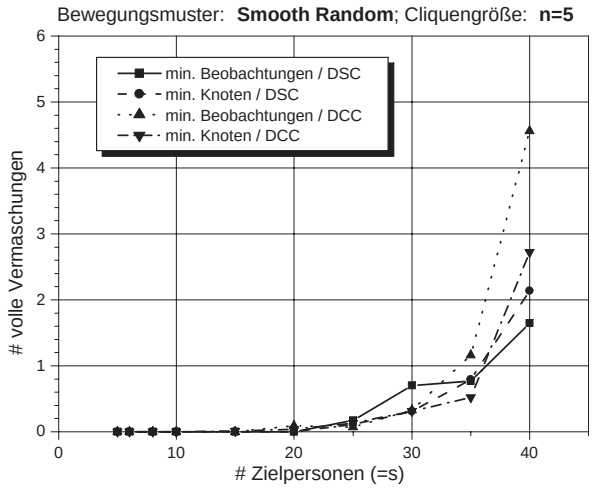

Abbildung 5.15: Anzahl voller Vermaschungen bei den verschiedenen Substrategien

5.3 Verwandte Arbeiten

Als verwandte Arbeiten sind selbstverständlich die zu nennen, die bereits im letzten Kapitel diskutiert wurden. Von diesen wird hier speziell auf den Ansatz von Xu und Jacobsen [167] eingegangen, der ein der Cliquenerkennung sehr ähnliches Problem aufgreift. Das dort definierte so genannte *n-Body Constraint* sieht vor, dass sich n bewegliche Objekte gleichzeitig in einem Kreis von Durchmesser d aufhalten. Wird $d = d_c$ angenommen, so erfüllt eine Menge von Objekten der Größe n, die das *n-Body Constraint* erfüllt, auch immer die Cliquenbedingung. Die Gegenrichtung trifft jedoch im Allgemeinen nicht zu. Als Beispiel nehme man an, dass drei Zielobjekte ein gleichseitiges Dreieck der Seitenlänge d_c bilden. Die Cliquenbedingung ($n = 3$) ist erfüllt, das n-Body Constraint jedoch nicht, da kein Kreis mit Durchmesser d das Dreieck einschließen kann. Wie bei Xu und Jacobsen motiviert sich diese Arbeit dadurch, Zielobjekte zu identifizieren, die räumlich nahe zueinander stehen. Aus Sicht des Autors scheint zu diesem Zweck die Cliquendefinition besser geeignet als das n-Body Constraint, da sie direkt die Eigenschaft der gegenseitigen Erreichbarkeit modelliert. Der Unterschied der beiden Definitionen ist praktisch jedoch von nicht allzu großer Bedeutung.

Das Problem, welches in diesem Kapitel definiert und gelöst wurde, ist jedoch allgemeiner formuliert als bei Xu und Jacobsen, die beim n-Body Constraint alle n Zielobjekte im Vorhinein kennen müssen. Die Anzahl der beobachteten Objekte ist dort also immer gleich der Größe der gesuchten Clique, $n = s$. In dieser Arbeit wird hingegen $n \leq s$ behandelt.

Abgesehen von diesen formalen Unterschieden stehen Xu und Jacobsens Ansatz, welcher sich vor allem mit der Reduzierung von Rechenlast beschäftigt, und die vorgestellte Strategie jedoch nicht in Konkurrenz zueinander, sondern die Ansätze ergänzen sich gegenseitig. In einer der Indizierungsmethoden, die der Artikel beschreibt, wird die gesamte verfügbare Spielfläche in Gitterzellen gleicher Seitenlänge aufgeteilt (2,5 km^2 erzielt die

besten Ergebnisse). Die aktuelle Zelle eines Zielobjekts ist dem Location Server stets bekannt, bei jedem Zellwechsel erfolgt also ein Position Update. Anhand dieser am Server verfügbaren Information ist es möglich, jede der laufenden Anfragen zur Cliquenerkennung (n-Body Constraints bei Xu und Jacobsen) in eine von drei Klassen einzuteilen: Die Klasse A bezeichnet Anfragen, die mit Sicherheit erfüllt sind, bei denen also eine Clique der gesuchten Größe besteht. Ist die Cliquendistanz d_c zum Beispiel größer als die Seitenlänge der Gitterzelle und befinden sich mehr als n Objekte in derselben Zelle, dann muss eine Clique der Größe n bestehen. Im Gegensatz dazu können Anfragen der Klasse B mit Sicherheit als nicht erfüllt betrachtet werden, wenn nämlich anhand der aktuellen Verteilung der Objekte auf die Zellen die Existenz einer Clique der Größe n ausgeschlossen werden kann. Dies ist der Fall, wenn die einzelnen Objekte durch genügend dazwischen liegende Zellen getrennt sind. Schließlich werden Anfragen, die anhand der aktuellen Verteilung auf die Zellen nicht eindeutig beantwortet werden können, in die Klasse C eingeordnet. Ein Beispiel ist, wenn n Objekte auf zwei benachbarte Zellen verteilt sind und die Seitenlänge der Zellen größer als $\frac{d_c}{2}$ ist. Anfragen der Klasse C werden bei Xu und Jacobsen mittels kontinuierlicher Positionsübermittlung der betroffenen Zielobjekte behandelt, wozu ein einfaches (ineffizientes) periodisches Protokoll angewendet wird.

Eine Kombination von Xu und Jacobsens und dem hier vorgestellten Ansatz ist viel versprechend: Anhand der beschriebenen Gitterstruktur lässt sich ein großer Anteil von Cliquenüberwachungen einfach verarbeiten, indem man die Anfragen entweder in die Klasse A oder B einteilt. Basierend auf einem eher grobmaschigen Gitter scheint der durch die Zellwechsel entstehende Nachrichtenaufwand akzeptabel. Anhand des Gitters lassen sich ferner komplexe Anfragen mit $n \ll s$, bei denen die eigene Strategie schlecht abschneidet, in mehrere einfache aufteilen: $n \leq s'$, wobei $S' \subset S$ (S' ist eine „Klasse C Untermenge“ von S). Auf diese Weise wird die absolute Anzahl von Paarbeobachtungen, die ja quadratisch mit $s - n$ ansteigt, stark reduziert. Der Beitrag dieser Arbeit zur Hybrid-Strategie ist die effiziente Abarbeitung von Anfragen der Klasse C, für die bislang keine geeignete Lösung bestand.

5.4 Zusammenfassung

Die Erkennung von Cliquen ist ein Basismechanismus proaktiver Mehrpersonen-LBCSs und unterstützt soziale Communities im gleichen Maße wie Logistikdienste. Es wurde ein neuartiger Lösungsansatz vorgestellt und evaluiert, der das Problem nicht nur effizient löst, sondern gleichzeitig sehr flexibel ist. Indem der Ansatz vollständig auf der Nahbereichs- und Trennungserkennung aufsetzt, lassen sich die entsprechenden Strategien (DCC, DSC, Streifen, ...) transparent und dynamisch austauschen, so dass diese stets der aktuellen Situation der Zielperson und den Anforderungen des Nutzers gerecht werden können. Wird zum Beispiel zu einer Strategie gewechselt, die Nahbereich und Trennung basierend auf topologischen statt euklidischen Distanzen feststellt, werden automatisch Cliquen basierend auf topologischen Distanzen erkannt. Der vorgestellte Algorithmus zur Cliquenerkennung müsste dafür nicht verändert werden.

Wie diese funktionale Aufteilung zwischen paarweiser (Nahbereich und Trennung) und allgemeiner (Cliquen-) Erkennung von Mehrpersonenereignissen genau funktioniert und wie sich auf ähnliche Weise weitere allgemeine Funktionen (k nächste Nachbarn, geometri-

sche Figuren, usw.) einfach realisieren lassen, wird ausführlich in Abschnitt 7.2 diskutiert.

6 Schutz der Privatsphäre

Während sich die vorangegangenen beiden Kapitel mit der effizienten Realisierung proaktiver Mehrpersonen-LBCSs beschäftigt haben, widmet sich dieses Kapitel dem zweiten großen Themenblock dieser Arbeit, dem Datenschutz. In Kapitel 3 wurden diesbezüglich zwei für LBCSs besonders wichtige Problemstellungen identifiziert. Bei der ersten soll gewährleistet werden, dass Zielpersonen von proaktiven Mehrpersonen-LBS, die ja besonders datenintensiv sind, vor intermediären Akteuren wie dem LBS Provider und dem Location Provider anonym bleiben können. Abschnitt 6.1 untersucht dazu erst die Eignung bestehender Anonymisierungstechniken und stellt dann einen eigenen, speziell auf die proaktive Nahbereichs- und Trennungserkennung zugeschnittenen Ansatz vor. Die zweite Problemstellung betrifft die geeignete Autorisierung von Nutzerzugriffen bei querverweisenden LBCSs. Insbesondere reaktive Dienste, bei denen der Nutzer nach seiner Anfrage direkt auf eine entsprechende Rückantwort des Dienstes wartet, werfen das Problem auf, dass die Zielperson sich bei einer Zugriffsverweigerung unter Umständen vor dem Nutzer rechtfertigen muss und somit erhöhtem sozialen Druck ausgesetzt ist. Abschnitt 6.2 analysiert diese Problematik genauer, bespricht bestehende Arbeiten zur Autorisierung von Nutzern und skizziert schließlich einen neuen Ansatz, der zum Ziel hat, solche negativen sozialen Situationen abzuschwächen sowie den Verwaltungsaufwand der Zielperson bei der Autorisierung zu reduzieren. Abschnitt 6.3 fasst das Kapitel zusammen.

6.1 Anonymisierung der Nahbereichs- und Trennungserkennung

In diesem Abschnitt wird ein speziell für die Nahbereichs- und Trennungserkennung konzipierter Mechanismus zur Anonymisierung gesammelter Ortsdaten vorgestellt. Grundlegend für den Ansatz, der in [161] erstmals skizziert und dann in [146] im Detail dargestellt wurde, ist die Verwendung von Koordinatentransformationen, mit deren Hilfe benutzte Pseudonyme vor Aufdeckung durch statistische Angriffe geschützt werden sollen. Eine ausführliche Behandlung des Ansatzes findet sich in [145].

Der Aufbau ist folgendermaßen. In Abschnitt 6.1.1 werden bestehende Arbeiten zur Anonymisierung und Verschleierung bei LBSs hinsichtlich ihrer Eignung für die Nahbereichs- und Trennungserkennung untersucht. In Abschnitt 6.1.2 werden dann die zum Verständnis der Funktionsweise des eigenen Ansatzes wichtigen Eigenschaften der Nahbereichs- und Trennungserkennung diskutiert. Der vorgeschlagene Mechanismus selbst wird in Abschnitt 6.1.3 erklärt, wobei zwei mögliche Konfigurationen, die jeweils einem anderen Vertrauensmodell entsprechen, möglich sind. Eine Evaluierung des Ansatzes folgt in Abschnitt 6.1.4.

6.1.1 Bestehende Ansätze der Anonymisierung und Verschleierung

Im Folgenden wird ein Überblick auf bestehende Arbeiten zur Anonymisierung und Verschleierung gegeben, die alle das Ziel haben, die im Rahmen von LBSs gesammelten Ortsdaten von Zielpersonen vor Missbrauch durch Dritte zu schützen.

Anonymisierungstechniken lassen sich unterteilen in solche, die Dateninhalte verändern, und solche, die sich mit der Verwaltung von Pseudonymen beschäftigen. Gleichzeitig existieren Hybridformen der Techniken. Alle Formen werden im Folgenden vorgestellt.

Anonymisierung basierend auf Manipulation von Dateninhalten lässt sich durch eine *Verhüllung (engl. Cloaking)* der übertragenen Ortsinformationen erzielen, bei der man die zeitliche bzw. räumliche Auflösung der Information vergröbert. Ziel ist dabei, dass die Ortsinformationen verschiedener Individuen nicht mehr voneinander unterscheidbar sind. [67] legen ihrem Ansatz das formale Modell der k-Anonymität [155] zugrunde. k-Anonymität ist dann gegeben, wenn sich die über eine Person gesammelte Ortsinformation nicht von den Daten mindestens $k-1$ anderer Individuen unterscheiden lässt. Konkret wird dabei von einem zentralen Location Server ausgegangen, der die Ortsinformationen einer großen Zahl an Zielpersonen verwaltet. Dieser passt die räumliche und zeitliche Verhüllung so an, dass die an den anfragenden LBS herausgegebenen Ortsinformationen stets k-anonym sind.

Ein Nachteil liegt klar in der künstlich herbeigeführten Verschlechterung der Genauigkeit, welche umso größer ist, je geringer die Nutzerdichte im betrachteten Gebiet ist. Weiterhin führt die zeitliche Verhüllung durch die eingeführte Verzögerung zu einer signifikanten Erhöhung der Dienstantwortzeit. Beide Aspekte laufen den hohen Qualitätsanforderungen vieler LBSs zuwider, so dass der Ansatz wohl nur für relativ wenige Anwendungen sinnvoll einsetzbar ist. Ein grundsätzlicher Hinderungsgrund für die Anwendung bei der Nahbereichs- und Trennungserkennung ist jedoch, dass sich das Modell nur auf einzelne Positionsdaten bezieht. Bei querverweisenden sowie proaktiven LBSs sind hingegen mehrfache Ortsbestimmungen bezogen auf denselben Identifikator der Zielperson nötig, zu deren Anonymisierung das Modell nicht geeignet ist.

Sind mehrfache Ortsbestimmungen gewünscht, so werden in der Regel Pseudonyme mit den gesammelten Ortsinformationen verknüpft, wodurch die wahre Identität der Zielperson verborgen bleiben soll. Alleine die Verwendung von Pseudonymen garantiert jedoch noch keinen ausreichenden Schutz, da sie sich mittels statistischer Angriffe aufdecken lassen. Beispielsweise können gesammelte, pseudonymisierte Ortsinformationen nach Orten durchsucht werden, die eine gegebene Person bekanntermaßen häufig besucht, wie zum Beispiel die Wohnung oder den Arbeitsplatz. Sehr einfach lässt sich so eine kleine Teilmenge von Pseudonymen bestimmen, die für die gesuchte Person in Frage kommen. Im schlimmsten Fall ist eine eindeutige Abbildung zwischen wahrer Identität der Person und dem benutzten Pseudonym möglich. Zur Vermeidung solcher Angriffe schlagen [37; 36] vor, die Benutzung von LBSs auf so genannte *Anwendungszonen* zu beschränken, die per Definition typische Aufenthaltsorte einer Zielperson ausschließen. Zusätzlich wechseln Zielpersonen in speziell vorgesehenen *Mix-Zonen*, die zu den Anwendungszonen disjunkt sind, dynamisch ihr Pseudonym. In den Mix-Zonen dürfen keine Ortsinformationen ausgetauscht werden. Andernfalls ließen sich die Pseudonymwechsel sehr leicht nachvollziehen, was den Wechsel wiederum wirkungslos machen würde. Weiterhin müssen Mix-Zonen stets genügend viele verschiedene Zielpersonen beinhalten, so dass keine Verbindung zwischen einem in die

Zone eintretenden und einem aus ihr heraustretenden Pseudonym ersichtlich ist.

Unabhängig von Mix- und Anwendungszonen analysieren [66], wie oft bei mehrfacher Ortung einer Zielperson Pseudonymwechsel erfolgen müssen und wie groß die Pausen zwischen den Wechseln sein müssen, um die Wahrscheinlichkeit einer Aufdeckung gering zu halten. Im vorgeschlagenen Ansatz werden Pfade einer Zielperson in mehrere Teilstrecken zerlegt, so dass dieselbe Zielperson aus der Sicht eines LBS in jedem Teil des Pfades unter einem jeweils anderem Pseudonym erscheint. Die Anonymität von Pfaden (Path Privacy) wird insbesondere verstärkt, wenn sich die Pfade mehrerer Zielpersonen zwischen Pseudonymwechseln überschneiden. Vorgesehen ist ebenfalls, dass Ortsinformationen nicht übertragen werden, falls sich die Zielperson in einem für sie typischen Gebiet befindet.

Beide Ansätze basierend auf Pseudonymwechseln weisen zwei Nachteile auf. Erstens wird durch den Ausschluss typischer Orte auch die Benutzung eines LBS für die überwiegende Zeit ausgeschlossen. Schließlich hält sich eine Person per Definition die meiste Zeit an typischen Orten auf. Zweitens steht der häufige Wechsel von Pseudonymen speziell der Nutzung von querverweisenden LBSs entgegen, bei denen ja konsistente Identifikatoren zur Referenzierung der Zielperson durch den Nutzer notwendig sind. Allgemein sind Pseudonymwechsel problematisch für alle Arten von LBSs, die persistente Profilinformationen von Zielpersonen verwalten. Werden nämlich Nutzerprofile zwischen verschiedenen Pseudonymen migriert, so lässt sich einfach ein Zusammenhang zwischen den Pseudonymen herstellen, was wiederum den Wechsel wirkungslos macht.

In Abschnitt 6.1.4 wird noch einmal genauer darauf eingegangen, wie [37] and [67] mit einem offensichtlichen Angriffszenario umgehen und warum die beiden Ansätze nicht für die Nahbereichs- und Trennungserkennung geeignet sind. Zusammenfassend lässt sich bereits jetzt sagen, dass die Eignung eines Anonymisierungsmechanismums doch stark vom Typ des zugrunde liegenden LBSs abhängt. So ist zum Beispiel die Verhüllung von Ortsinformationen nach [67] ausreichend für reaktive, selbstverweisende LBSs, die nur geringe Genauigkeitsanforderungen haben. Darunter fallen zum Beispiel typische Finder-Dienste. Der Ansatz von Mix- und Anwendungszonen nach [37] scheint hingegen geeignet für proaktive selbstverweisende LBSs, die in für die Zielperson untypischen Zonen benutzt werden. Ein Beispiel wäre ein Touristenführer, der Besucher einer fremden Stadt automatisch darauf hinweist, wenn sie sich bestimmten Sehenswürdigkeiten annähern.

Ein Ansatz, der nicht das Ziel der Anonymisierung verfolgt, ist die Verschleierung von Ortsinformationen. Wie bei der Verhüllung wird hier die Genauigkeit absichtlich verschlechtert, die wahre Identität der Zielperson darf jedoch bekannt sein. Die Policy-basierte Definition entsprechender Genauigkeitsvorgaben ist zum Beispiel als Erweiterung für Geopriv angedacht [149]. [60] liefert ein formales Modell zur Verschleierung von Ortsinformationen mit dem Ziel, die Genauigkeit so weit zu reduzieren, dass vorher zu definierende Anwendungsanforderungen gerade noch erfüllt werden können.

Grundsätzlich erhöht die Verschleierung den Schutz vor Missbrauch gesammelter Nutzerdaten. Leider ist der erreichte Schutz für viele Situationen nicht ausreichend. Vorstellbar ist zum Beispiel, dass ein Arbeitgeber überprüfen möchte, ob sich ein bestimmter Arbeitnehmer zu einem gewissen Zeitpunkt am Arbeitsplatz befunden hat oder nicht. In diesem Fall spielt die Genauigkeit der über den Arbeitnehmer gespeicherten Ortsinformationen eine untergeordnete Rolle. Hat er sich zu der Zeit in hinreichend großem Abstand zum Arbeitsplatz befunden, so erfährt es der Arbeitgeber.

Eine weitere verwandte Arbeit ist [63]. Hier wird ein Mechanismus vorgestellt, der es

einer Instanz X erlaubt zu berechnen, ob zwei gegebene Bewegungsrouten zueinander eine gewisse Nahbereichsdistanz unterschreiten, und zwar so, dass die Ortsinformationen, die die Route beschreiben, vor X geheim gehalten werden können. Eine anonyme Nahbereichs- und Trennungserkennung könnte also realisiert werden, indem der entsprechende Algorithmus vom Location Server ausgeführt wird. Leider lässt sich der Ansatz nicht zum Verfolgen der Zielpersonen anwenden, wenn die beschrittenen Routen nicht im Voraus bekannt sind. Er ist somit für die Nahbereichs- und Trennungserkennung ungeeignet.

6.1.2 Abtastdistanz der Nahbereichs- und Trennungserkennung

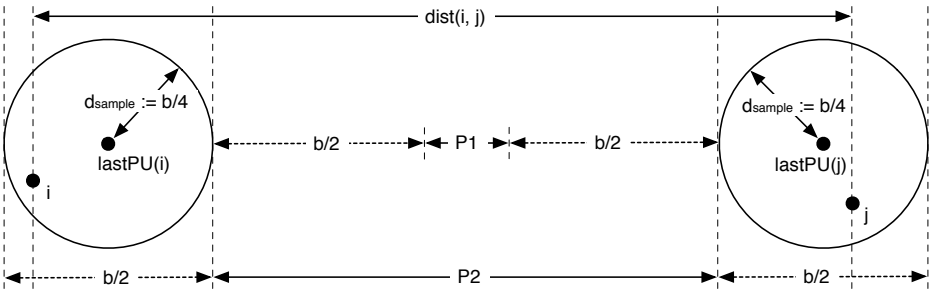

Abbildung 6.1: Minimale Abtastdistanz

Der im Folgenden vorgeschlagene Anonymisierungsmechanismus wurde für die Nahbereichs- und Trennungserkennung maßgeschneidert. Der Ansatz soll von den Zielpersonen verwendete Pseudonyme vor einer möglichen Aufdeckung durch den Location bzw. LBS Provider schützen. Grundsätzlich ist dabei zu sagen, dass die in Kapitel 4 beschriebene Nachrichtenreduktion, welche ja ursprünglich mit einer Erhöhung der Effizienz motiviert wurde, bereits zu einem erhöhten Datenschutz führt. Schließlich werden so von einer gegebenen Zielperson weniger Daten beim Location Server gesammelt.

Um eine „harte“ Anonymisierung zu gewährleisten, wird jedoch ein konkreter Schwellwert benötigt, den die so genannte minimale Abtastdistanz d_{sample} nie unterschreitet. d_{sample} entspricht dem räumlichen Mindestabstand zweier direkt aufeinander folgenden Ortsinformationen, die über eine gegebene Zielperson an den Location Server übermittelt werden. Diese Übermittlung schließt Position Updates wie auch Pollings mit ein. Die technische Realisierung der Grenze ist einfach: Sollte ein auf dem Endgerät platzierter Position Update Request schon vor Erreichen von d_{sample} ein mögliches Position Update anstoßen, so wird dieses so lange unterdrückt, bis d_{sample} erreicht ist. Ferner wird bei mehreren aufeinander folgenden Pollings vom Endgerät jeweils der letzte übermittelte und nicht der aktuelle Wert zurückgeliefert, falls die Distanz der beiden Messpunkte d_{sample} nicht überschreitet. Die Abtastdistanz ist somit ein härteres Maß als beispielsweise die der DCC-Strategie unterliegenden Update-Distanzen, welche ja bereits auf $\frac{b}{2}$ limitiert sind. d_{sample} muss unabhängig von der verwendeten Erkennungsstrategie definiert sein.

Wie sich später herausstellt, ist der durch den Ansatz gegebenen Schutz umso größer, je höher d_{sample} ist. Ziel der folgenden Argumentationskette ist daher, die bei der Nahbereichs- und Trennungserkennung erreichbare Untergrenze für d_{sample} abzuleiten. Abbildung 6.1 illustriert die Sicht des Location Servers bezüglich der Positionen zweier Zielpersonen i und

j. $lastPU(i)$ und $lastPU(j)$ sind die jeweils zuletzt gemeldeten Position Updates. Legt man eine minimale Abtastdistanz von $d_{sample} := \frac{b}{4}$ zugrunde, so können sich beide Ziele seit ihrer letzten Meldung innerhalb eines Kreises mit Radius d_{sample} bewegt haben, ohne dass dies der Location Server bemerken könnte. Für jeden möglichen Wert der Nahbereichsdistanz d_p trifft also einer der folgenden Fälle zu. Ist $d_p < P1$, dann kann, ohne die Bedingungen aus Abschnitt 4.2 zu verletzen, beruhigt *kein* Nahbereichsereignis generiert werden, da gelten muss, dass $dist(i,j) > d_p + b$. Dasselbe gilt natürlich auch für $P1 \leq d_p < P2$, da $dist(i,j) > d_p$. Schließlich kann bei $d_p >= P2$ ein Nahbereichsereignis ausgelöst werden, da gilt, dass $dist(i,j) <= d_p + b$. Für die Trennungserkennung verläuft die Argumentation ganz analog.

Der potentiell durch $d_{sample} := \frac{b}{4}$ eingeführte Fehler ist also gerade klein genug, um über Nahbereichsereignisse noch in allen möglichen Situationen entscheiden zu können. Es lässt sich sogar eine (etwas ineffizientere) Strategie zur Nahbereichs- und Trennungserkennung entwerfen, die noch bei $d_{sample} := \frac{b}{3}$ korrekt arbeitet, allerdings nur unter den folgenden Bedingungen: Erstens müssen Nahbereichs- und Trennungsereignisse zwischen zwei Zielen stets alternierend erkannt werden. Falls also i und j einmal als zueinander im Nahbereich befindlich erkannt wurden, so sind weitere Nahbereichserkennungen zwischen den beiden erst möglich, nachdem eine Trennung festgestellt wurde. Die Gegenrichtung gilt auch. Nach der Trennungserkennung muss stets der Nahbereich erkannt werden. Zweitens muss gelten, dass $d_s > d_p + b$. Die Trennungsdistanz muss also hinreichend größer sein als die Nahbereichsdistanz. Aus Platzgründen wird der Beweis für $d_{sample} = \frac{b}{3}$ an dieser Stelle nicht ausgeführt.

Im Folgenden wird grundsätzlich von einer Abtastdistanz von $\frac{b}{3}$ ausgegangen, wobei dies bereits den ungünstigsten Fall kennzeichnet. Wie besprochen werden typischerweise ja weitaus weniger Ortsinformationen von einer Zielperson gesammelt. Die durchschnittlichen Abstände der übermittelten Orte hängen dabei von der verwendeten Strategie zur Nahbereichs- und Trennungserkennung, der Dichte der Zielpersonen sowie der Anzahl von Erkennungsvorgängen ab.

6.1.3 Ansatz basierend auf Koordinatentransformationen

Im Folgenden wird ein speziell für die Nahbereichs- und Trennungserkennung entwickelter Ansatz vorgestellt, welcher sich nach [33] auch als Anonymisierungstechnik basierend auf Datenverschleierung einordnen lässt. Anonymität bezeichnet dabei den Zustand, wenn eine Zielperson innerhalb einer Menge von Subjekten, der so genannten Anonymitätsmenge, nicht identifizierbar ist, vergleiche [134]. Abhängig vom zugrunde liegenden Vertrauensmodell hat der Ansatz zum Ziel, Identitäten verfolgter Zielpersonen vor Aufdeckung seitens des LBS bzw. des Location Providers zu schützen. Der Ansatz setzt die Verwendung von Pseudonymen voraus, dynamische Pseudonymwechsel sind jedoch explizit kein integraler Bestandteil, um die Eignung für querverweisende LBSs zu gewährleisten (vergleiche Abschnitt 6.1.1).

Vertrauensmodell

Im Folgenden bezeichnet der Begriff *vertrauenswürdige Entität* einen LBS-Akteur, dem es gestattet ist, die echte Position und Identität einer Zielperson zu kennen. Dabei wird

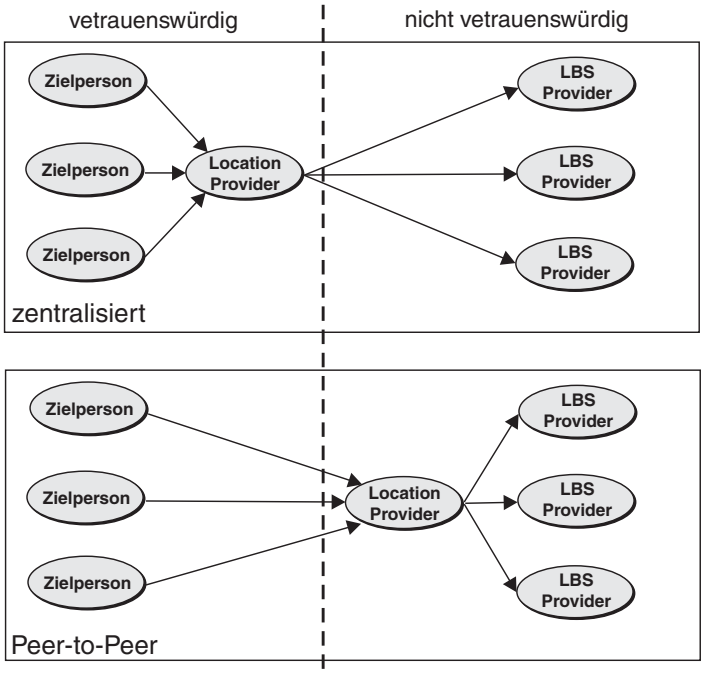

Abbildung 6.2: Vertrauensmodelle: zentralisiertes (oben) und Peer-to-Peer-Modell (unten)

von einer sicheren Kommunikation zwischen Zielpersonen und vertrauenswürdigen Entitäten ausgegangen. Ausgehend von der in Abschnitt 2.3 beschriebenen indirekten endgerätzentrischen LBS-Wertschöpfungskette unterstützt der Ansatz zwei grundlegende Vertrauensmodelle, vergleiche Abbildung 6.2 (die Pfeile kennzeichnen die Übertragung von Ortsinformationen).

Im Peer-to-Peer-Modell sind weder der LBS Provider noch der Location Provider vertrauenswürdige Entitäten. Vertrauenswürdig in Bezug auf eine bestimmte Zielperson sind lediglich diejenigen anderen Zielpersonen, für die Nahbereichs- und Trennungsereignisse erkannt werden sollen. Dabei werden Funktionen zur gegenseitigen Authentifizierung, Schlüsselinitialisierung und -verteilung, zur Verwaltung der verwendeten Pseudonyme sowie zur Verschleierung der Ortsinformationen von den Zielpersonen kooperativ erbracht. Eine detaillierte Beschreibung dafür benötigter, existierender Techniken zur sicheren Gruppenkommunikation sowie entsprechender Protokolle zur Generierung von Gruppenschlüsseln findet sich in [172].

Im zentralisierten Modell tritt der Location Provider als vertrauenswürdige Entität auf, die somit alle gerade genannten Funktionen übernehmen kann. Der Location Provider bedient eine Reihe verschiedener nicht vertrauenswürdiger LBS Provider mit den Ortsinformationen der Zielpersonen. Dieses Modell wird auch von den bestehenden, bereits besprochenen Ansätzen zur Anonymisierung vorausgesetzt. Andernfalls, also Peer-to-Peer, würde zum Beispiel die Berechnung einer Verhüllungsfunktion, die genau k-Anonymität erzeugt, erheblichen Kommunikationsaufwand zwischen den Zielpersonen verursachen. Ähnliches gilt für die Berechnung von Mix-Zonen.

Während in beiden Vertrauensmodellen die Rollen des Location Providers und des LBS Providers von jeweils verschiedenen Akteuren übernommen werden können, lässt es das

Peer-to-Peer Modell auch zu, dass ein Akteur beide Rollen übernehmen kann. Das Peer-to-Peer-Modell wirkt zunächst einmal viel versprechend, da es nur den verfolgten Zielpersonen den Zugriff auf sensible Informationen gestattet. Werden jedoch die Positionen vieler Zielpersonen miteinander korreliert, so wird hier die Schlüsselverwaltung zum Problem.

Idee zur Anonymisierung

Wie bereits gesagt, sind Ortsdaten grundsätzlich hochgradig identifizierend, da sie in der Regel Informationen über die Wohnung oder den Arbeitsplatz einer Zielperson enthalten und auch Aufschluss über Freizeitaktivitäten oder andere charakteristische Gewohnheiten geben, zum Beispiel die von der Person präferierten Verkehrsmittel. Bei querverweisenden sowie proaktiven LBSs fällt dieser Umstand besonders ins Gewicht, da hier typischerweise viele gesammelte Ortsdaten mit demselben Pseudonym verknüpft werden, um eine konsistente Referenzierung der Zielperson zu gewährleisten. Im Folgenden wird eine solche zeitliche Ansammlung von Ortsdaten bezogen auf ein Pseudonym als *Trace* bezeichnet. Vereinfacht gesagt gilt: Je mehr Ortsdaten ein Trace enthält, desto identifizierender ist er für die Zielperson, vergleiche auch [67].

Ziel des vorgestellten Ansatzes ist es, Traces weniger identifizierend zu machen. Dabei wird der Umstand ausgenutzt, dass zur Nahbereichs- und Trennungserkennung nur relative Distanzen zwischen den Zielpersonen gemessen werden. Die Idee besteht darin, distanzerhaltende Koordinatentransformationen gleichermaßen auf die Traces der miteinander in Beziehung gebrachten Zielpersonen anzuwenden. So können Nahbereichs- und Trennungsereignisse weiterhin korrekt erkannt werden, während den Traces möglicherweise identifizierende Merkmale genommen werden.

Zweistufige Verschleierung

Der Ansatz basiert auf den folgenden Definitionen, vergleiche auch Abbildung 6.3.

- $E = \{e_1, e_2, ..., e_n\}, 1 < i \leq n$ ist eine Menge von Zielpersonen, deren Positionen miteinander korreliert werden sollen,
- $p(e,t) : E \times \mathbb{R} \rightarrow \mathbb{R}^2$ liefert die wahre Position einer Person $e \in E$ zum Zeitpunkt t,
- $s_G \in \mathbb{N}$ ist ein geheimer, für E spezifischer Schlüssel,
- $p^*(e,t,s_G) : E \times \mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}^2$ berechnet die verschleierte Position von e zum Zeitpunkt t.

Ein zweistufiger Verschleierungsalgorithmus wird auf die Koordinaten aller Personen in E gleichermaßen angewendet. Als Erstes ist eine zeitunabhängige, globale Transformation vorgesehen, bestehend aus einer Rotation mit Winkel α um den Punkt $(j,k) \in \mathbb{R}^2$ in Kombination mit einer Translation $(x_{global}, y_{global}) \in \mathbb{R}^2$. Die zweite Stufe wendet eine zeitabhängige Translation $\vec{v} := (x_{local}, y_{local}) \in \mathbb{R}^2$ an.

Dabei soll die erste Stufe den Koordinaten ihren globalen Bezugspunkt nehmen, so dass Angriffe basierend auf typischen Aufenthaltsorten vermieden werden. Die zweite Stufe *verwischt* lokale Bewegungen der Zielpersonen und soll Angriffe basierend auf wohlbekannten

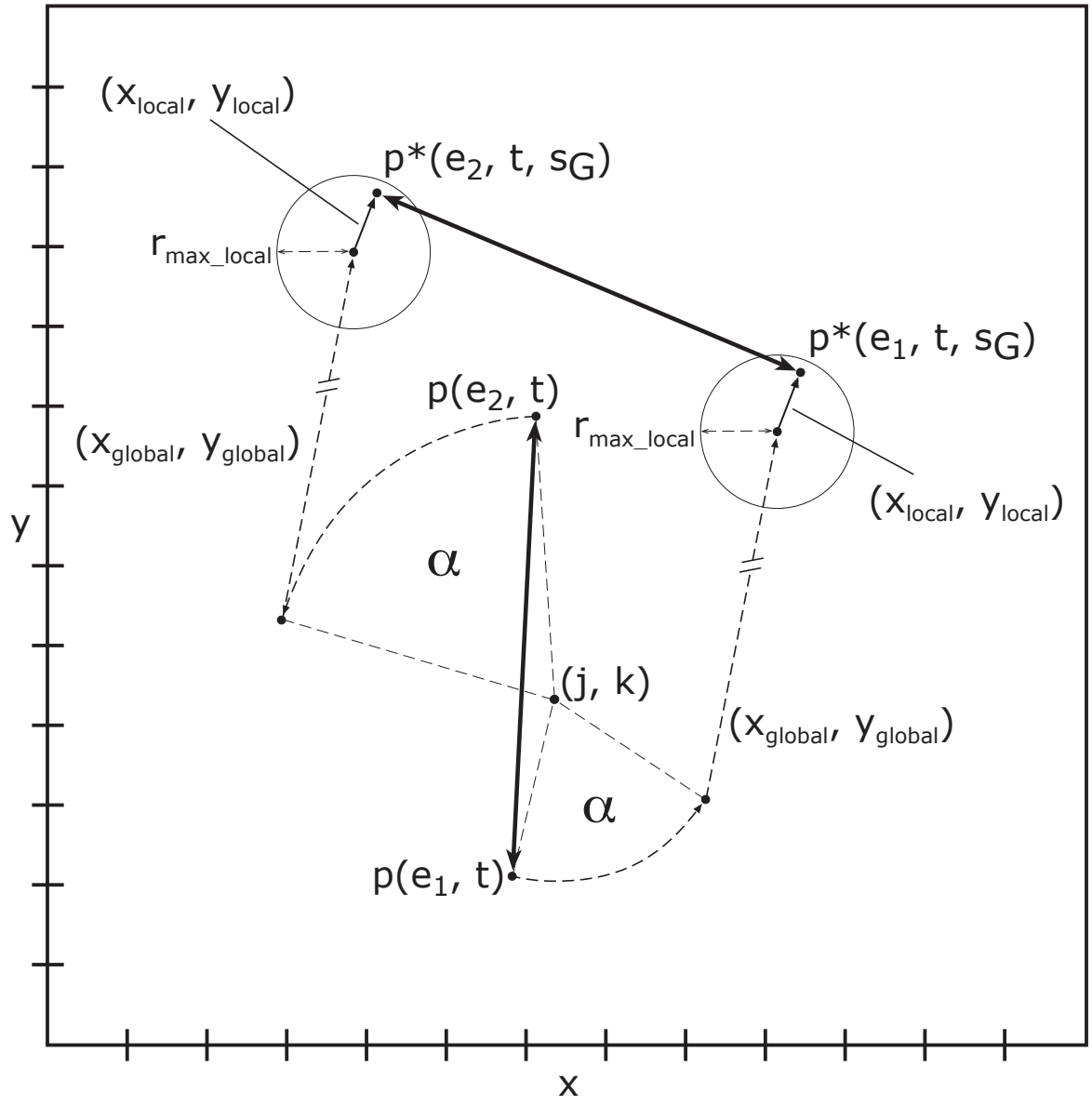

Abbildung 6.3: Zweistufige Verschleierung

Bewegungsmustern einer Zielperson sowie Straßenmustern verhindern. Abschnitt 6.1.4 behandelt mögliche Angriffsszenarien im Detail.

Die Parameter α, j, k, x_{global} und y_{global} (erste Stufe) hängen von s_G ab. Der lokale Translationsvektor $\vec{v} := (x_{local}, y_{local})$ (zweite Stufe) ist begrenzt auf eine Länge von $|v| \leq r_{max\_local}$ und hängt von s_G sowie von der aktuellen Uhrzeit ab. Wie später diskutiert, ist eine Voraussetzung für die Robustheit des Ansatzes, dass die lokale, zeitabhängige Translation nicht von selbst, also ohne Bewegung der Zielperson, Position Updates provoziert. Ausgehend von den Schlussfolgerungen aus Abschnitt 6.1.2 muss daher gelten, dass $r_{max\_local} \leq d_{sample}$. Überschreitet nämlich der Wert der lokalen Translation den der minimalen Abtastdistanz nicht, wird ein automatisches Auslösen von Position Updates vermieden.

Im zentralisierten Modell kann die aktuelle Uhrzeit vom Location Provider ermittelt werden. Selbstverständlich können hier auch logische Uhrzeiten verwendet werden. Im Peer-to-Peer-Modell hingegen müssen die Uhrzeiten der Endgeräte synchronisiert sein, was eine prinzipielle Schwierigkeit darstellt, die sich aber, ausgehend von endgerätbasierter Ortung wie GPS oder Galileo, elegant lösen lässt. GPS-Satelliten sind mit Atomuhren ausgestattet und übertragen die gemessenen Zeiten zusammen mit den Signalen, die zur Positionsbestimmung verwendet werden. Auch lassen sich die Uhrzeiten standardmäßig, zum Beispiel mit Hilfe des Protokolls NMEA-0183 der *National Marine Electronics Association* [21], aus GPS-Empfängern auslesen. Eine skalierbare Methode zur Uhrensynchronisation der Endgeräte ist somit gegeben.

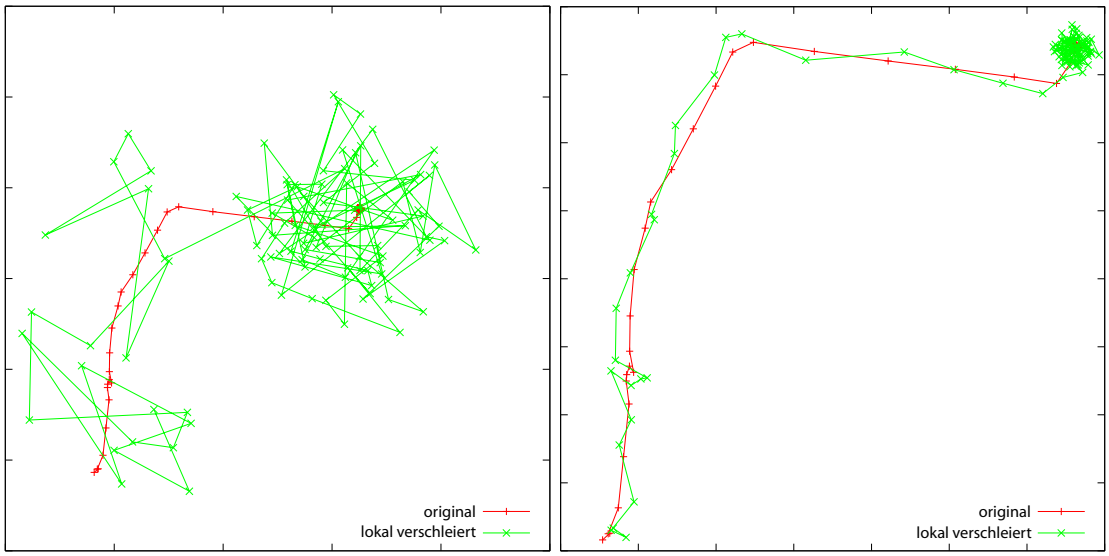

Abbildung 6.4: Ein Trace im Original (+) und lokal verschleiert (x), mit größerem (links) und kleinerem (rechts) r_{max_local}.

Abbildung 6.4 zeigt einen mit zwei verschiedenen Werten für r_{max_local} verschleierten Trace. Zwecks klarer Darstellung wird die globale Transformation in der Abbildung weggelassen und nur die lokale, zeitabhängige Translation gezeigt, welche ja Bewegungsmuster verwischen soll.

Trotz der Transformationen gilt jedoch: Je länger Traces sind, desto identifizierender sind sie prinzipiell auch. Die Durchführung regelmäßiger Pseudonymwechsel der Zielperson ist

aus den genannten Gründen für querverweisende LBSs leider keine geeignete Lösung. Es wird daher vorgeschlagen, den Schlüssel s_G von Zeit zu Zeit auszutauschen. Ohne weitere Vorkehrungen könnte dies leider dazu führen, dass ein Angreifer zwei Schlüssel aufeinander abbilden kann, zum Beispiel indem er beobachtet, dass die Positionen aller Zielpersonen gleichzeitig um eine weite Strecke versetzt werden. Um dies zu vermeiden, wird die Beachtung bestimmter *Schutzzeiten* zwischen der Benutzung verschiedener Schlüssel empfohlen. Während solcher Schutzzeiten dürfen keinerlei Positionen über die Zielpersonen gesammelt werden.

6.1.4 Evaluierung

Im Folgenden wird eine Reihe denkbarer Angriffe auf den vorgeschlagenen Mechanismus diskutiert. Die vorgestellten Angriffe werden alle *ex post* ausgeführt, haben also zum Ziel, die Abbildung zwischen dem verwendeten Pseudonym einer Zielperson und deren wahrer Identität aus einem bereits gesammelten Bestand von Traces abzuleiten. Die Angriffe unterscheiden sich lediglich darin, welche Art von Hintergrundwissen eingesetzt wird.

Typische Online-Angriffe, wie zum Beispiel das Abhören laufender Kommunikationsvorgänge oder Versuche, anhand der verwendeten Netzwerkadressen Rückschlüsse auf die Identitäten zu ziehen, liegen nicht im Fokus dieser Arbeit. Passende Gegenmaßnahmen wie Verschlüsselungstechniken oder so genannte Mix-Netze zur Anonymisierung von Netzwerkadressen [50] werden als gegeben vorausgesetzt. Weiterhin werden Angriffe ausgeschlossen, bei denen die Endgeräte der Zielpersonen gezielt manipuliert werden. Hierfür müssen Vorkehrungen anderer Art getroffen werden.

Abhängig davon, welches Vertrauensmodell verwendet wird (Peer-to-Peer oder zentralisiert), liegt der den Angriffen ausgesetzte, anonymisierte Datenbestand entweder dem Location Provider oder dem LBS Provider vor. Mögliche Angreifer sind: der LBS bzw. Location Provider selbst, beispielsweise mit der Motivation, die de-anonymisierten Daten illegal an Dritte weiterzuverkaufen; bösartige Angestellte des Betreibers, zum Beispiel mit der Absicht, Einsicht in das Privatleben einer bestimmten Person zu gewinnen; externe Hacker, die in das System eingedrungen sind.

Angriff basierend auf bekannten Aufenthaltsorten

Bei diesem Angriff (engl. *Known Whereabouts Attack, KWA*) werden typische Aufenthaltsorte einer gegebenen Person benutzt, um deren verwendetes Pseudonym herauszufinden. Ist zum Beispiel bekannt, dass Person A an einem bestimmten Ort L wohnt und an einem anderen Ort W arbeitet, so ist das Pseudonym P von A wahrscheinlich mit den Traces verknüpft, die eine große Häufung von L und W aufweisen. Je mehr typische Aufenthaltsorte X von A bekannt sind, desto wahrscheinlicher wird es, einen einzelnen Trace zu finden, der alle X enthält. Die gewünschte Abbildung von A nach P ist somit erreicht.

In [67] wird versucht, den KWA durch zeitliche und räumliche Verhüllung zu vermeiden. Wie bereits besprochen ist dieser Ansatz leider nur in der Lage, einzelne Positionsdaten vor Aufdeckung zu schützen und lässt sich nicht allgemein auf Traces anwenden. Andernfalls müssten bei der Berechnung der Verhüllungsfunktion bereits in der Vergangenheit gesammelte Daten miteinbezogen werden. Dies wäre zwar theoretisch möglich, die zur Wahrung

der k-Anonymität eingeführte künstliche Ungenauigkeit müsste aber wohl so stark anwachsen, dass die Daten nicht mehr sinnvoll zu verwenden wären.

In [37] wird der KWA verhindert, indem typische Aufenthaltsorte einfach aus den Anwendungszonen eines LBS ausgeschlossen werden. Das entstehende Problem liegt klar auf der Hand. Der LBS kann schlichtweg die meiste Zeit, zu der sich eine Person ja per Definition an typischen Orten aufhält, nicht benutzt werden.

Die Abwehr des KWA war das vorrangige Ziel bei der Entwicklung des vorgestellten Ansatzes basierend auf Koordinatentransformationen. Offensichtlich enthalten Traces nach der zweistufigen Verschleierung keine typischen Aufenthaltsorte von Zielpersonen mehr, was den KWA prinzipiell unmöglich macht. Im Folgenden soll jedoch untersucht werden, welche anderen statistischen Angriffe denkbar sind, die das Ziel der Aufdeckung der Transformationen haben. Nach erfolgreicher Durchführung eines solchen Angriffs könnte die Abbildung zwischen Identität und Pseudonym wiederum mittels KWA herausgefunden werden.

Campus-Angriff

Während beim KWA der Bestand an Traces nach gegebenen Aufenthaltsorten von Zielpersonen durchsucht wird, wird beim Campus-Angriff (CA) nach Aufenthaltsorten gesucht, die in sehr vielen Traces gleichzeitig auftauchen. Die Annahme ist, dass diese Aufenthaltsorte jeweils öffentlichen, viel besuchten Orten entsprechen, wie etwa einem Universitätscampus. Ausgehend von mehreren solchen Referenzorten lässt sich eine Abbildung zwischen den originalen und den verschleierten Daten finden, und die verwendeten Transformationen können aufgedeckt werden.

Im vorgeschlagenen Ansatz wird ein geheimer Schlüssel jeweils für eine Menge E von Zielpersonen berechnet, deren Ortsinformationen korreliert werden. Der CA setzt jedoch relative große Gruppen miteinander korrelierter Personen voraus. Für sehr kleine Gruppen ist der CA wohl wirkungslos. Ein Beispiel für eine kleine Gruppe wäre die Nahbereichserkennung zwischen einem einzelnen Paar von Personen. In einem anderen Szenario, wo zum Beispiel unter allen Studenten eines Studiengangs die Nahbereichserkennung läuft, scheint der CA jedoch sehr effektiv.

Wie es scheint, ist der vorgeschlagene Ansatz nicht in der Lage, für große Mengen E von Zielpersonen ausreichenden Schutz zu bieten. Für diesen Fall müssten wohl Erweiterungen vorgenommen werden. Möglich wäre eventuell ein hierarchischer Ansatz, bei dem Untermengen von E jeweils andere Transformationsschlüssel zugeteilt bekommen. Während innerhalb einer solchen Untergruppe Aufenthaltsorte exakt korreliert werden, könnte man zum Beispiel zwischen den einzelnen Untergruppen eine nur relativ ungenaue Korrelation zulassen. Die einzelnen Untergruppenschlüssel wären also verschieden, würden aber zu ähnlichen Transformationen führen. Die Entwicklung entsprechender Konzepte sprengt jedoch den Rahmen dieser Arbeit und mag in weiteren Arbeiten verfolgt werden. Einziges Fazit aus dem CA ist somit, dass der Ansatz im jetzigen Zustand höchstens für kleinere Gruppen Sicherheit bietet.

Angriff basierend auf stationären Zielpersonen

Während in der ersten Stufe der Verschleierung eine globale und zeitunabhängige Transformation durchgeführt wird, welche den KWA vermeiden soll (siehe oben), hat die zeitab-

hängige Translation in der zweiten Stufe zum Ziel, die Traces vor Angriffen basierend auf Bewegungs- und Straßenmustern (siehe nächstes Angriffsszenario) zu schützen.

Ziel des Angriffs basierend auf stationären Zielpersonen (engl. *Stationary Users Attack, SUA*) ist es, die zeitabhängige Translation auszuhebeln. Der SUA basiert auf der Annahme, dass sich die Zielpersonen die meiste Zeit stationär verhalten, sich also zum Beispiel während des Arbeitstages kaum vom Büro wegbewegen. Zeigen mehrere Zielpersonen aus derselben Menge E ein sehr ähnliches Bewegungsverhalten, so kann der Angreifer annehmen, dass die Personen in Wirklichkeit stationär sind und sich die beobachtete Bewegung lediglich aus der lokalen Translation ergibt, die für alle Personen aus E gleichermaßen angewendet wird. Die lokale Translation lässt sich dann einfach aus den Traces herausrechnen, und der Angriff basierend auf Bewegungs- und Straßenmustern kann erfolgreich durchgeführt werden.

Zur Vermeidung des SUA ist die zweite Verschleierungsstufe jedoch beschränkt auf Translationen mit einer Maximallänge von r_{max_local}. Gilt $r_{max\_local} \leq d_{sample}$ (vergleiche Abschnitt 6.1.2), so ist gewährleistet, dass stationäre Zielpersonen keine Position Updates auslösen, was den SUA ausschließt. Die Vermeidung des SUA ist daher die Hauptmotivation (neben Effizienzaspekten) dafür, eine Beschränkung für die lokale Translation vorzusehen.

Angriff basierend auf Bewegungs- und Straßenmustern

Der Angriff basierend auf Bewegungs- und Straßenmustern (engl. *Mobility Pattern Attack, MPA*) ist wahrscheinlich der interessanteste unter den behandelten, weshalb er hier auch sehr ausführlich diskutiert wird. Der MPA gründet auf der Annahme, dass die verschleierten Traces ähnliche Bewegungsmuster aufweisen wie die originalen. Ausgehend von verfügbarem Kartenmaterial könnte ein Angreifer beispielsweise versuchen, Straßenmuster in den Traces wiederzuerkennen und so die Transformationen rückgängig machen. Eine weitere Möglichkeit wäre, nach Bewegungsmustern in den Traces zu suchen, die für eine bestimmte Person typisch sind. Angenommen, der Angreifer weiß, dass Person A am Ort L wohnt, am Ort W arbeitet und am Ort F zu Mittag isst, und hat auch grobe Kenntnis über die Pfade auf denen sich A zwischen diesen Stationen bewegt. Dann lässt sich unter Umständen durch gezielte Suche nach einem entsprechenden Muster in den gesammelten Traces das von A verwendete Pseudonym herleiten. Im Folgenden wird nun die Robustheit des vorgestellten Ansatzes gegen den MPA mit Hilfe von Simulationen basierend auf gesammelten GPS-Daten untersucht.

Zuerst werden die grundlegende Herangehensweise sowie die Art und Menge der verwendete Daten beschrieben. Nach der Beschreibung der Parametrisierung der Simulationen folgt ein kurzer Exkurs über verschiedene Methoden zum strukturellen Vergleich von Traces. Dann werden die generierten Simulationsergebnisse beschrieben und bewertet. Schließlich wird die Rolle des Schlüssels s_G kurz diskutiert und ein möglicher Ansatz aufgezeigt, mit dem sich s_G dynamischen ändern lässt, ohne die Anonymität der Traces zu gefährden.

Ziel der Simulationen Aus den Simulationen soll hervorgehen, inwieweit die vorgestellte Verschleierungsmethode es vermag, einem Trace die Ähnlichkeit zu seiner originalen Version zu nehmen. Je weniger ähnlich ein verschleierter Trace seiner originalen Version ist, desto besser ist der erreichte Schutz.

Im Fokus der Evaluation stehen strukturelle Angriffe, die zum Beispiel gesammelte Traces mit den Straßenmustern einer Stadt vergleichen. Angriffe, die auf zeitlichen Aspekten der Traces basieren, wie zum Beispiel das Abgleichen wiederkehrender Ereignisse im täglichen Leben einer gegebenen Zielperson, werden nicht behandelt. Die Entwicklung und Simulation entsprechender Angriffsmodelle, die diese Art von Hintergrundinformationen vorsehen, mag ein Thema zukünftiger Arbeiten sein.

Zur Simulation struktureller Angriffe wurden wiederum die 69 GPS Traces verwendet, welche auch schon zur Evaluation der Update-Strategien aus den vorigen Kapiteln gedient haben. Wie schon erwähnt, enthalten sie fußgängerisches Bewegungsverhalten ebenso wie Fahrtwege von Autos, die alle in München und Umgebung aufgenommen wurden. Die zeitliche Dauer der Traces liegt zwischen mehreren Minuten und einigen Stunden. Gesammelte GPS-Positionsdaten kommen im Sekundentakt auf.

Simulationsparameter Die Traces wurden mittels der vorgestellten Methode verschleiert, und zwar anhand der folgenden Parameter. Der lokale, zeitabhängige Translationsvektor $\vec{v} := (x_{local}, y_{local})$, welcher abhängig vom Schlüssel s_G sowie von der aktuellen Uhrzeit t ist, wird pseudozufällig bestimmt. s_G dient der verwendeten Zufallsfunktion als Initialisierungswert und wird pro Simulationslauf neu und ebenfalls zufällig bestimmt. Alle Traces werden zeitlich gleich ausgerichtet, so dass jeder Trace bei $t := 0$ beginnt. Zu jedem Zeitschritt $t, 0 \leq t \leq t_{max}$, wobei t_{max} der Dauer des längsten Trace entspricht, wird nun der lokale Translationsvektor $\vec{v} := (x_{local}, y_{local})$ mit Hilfe der Zufallsfunktion berechnet. Die Länge von $\vec{v}$ wird gleich verteilt bestimmt innerhalb des Intervalls $0 < |\vec{v}| \leq r_{max\_local}$. Die Richtung von $\vec{v}$ ist ebenfalls gleich verteilt, und zwar im Intervall $[0; 2\pi[$. $\vec{v}$ wird für jeden Zeitschritt t neu berechnet und dann an alle Positionen (pro Trace maximal eine) angehängt, die den Zeitstempel t tragen.

Die Parameter für die globale Transformationen werden beliebig gewählt und sind für jeden Simulationslauf festgesetzt.

Ähnlichkeit zwischen Traces Ziel ist es, herauszufinden, wie ähnlich ein verschleierter Trace λ^* seiner originalen Version λ ist und wie viele andere verschleierte Traces μ^* zu λ eine größere Ähnlichkeit haben als λ^* zu λ. Hat ein bestimmter Anteil der verschleierten μ^* zu λ eine höhere Ähnlichkeit als λ^* zu λ, dann ist ein gewisses Maß an Schutz anzunehmen.

Zum Erhalt eines entsprechenden Ähnlichkeitsmaßes der Struktur zweier Traces sind die folgenden Betrachtungen notwendig. Die Ähnlichkeit von Traces anhand eines formbasierten Vergleichs wird zum Beispiel in [168] besprochen. Ferner stellen [119] rasterbasierte Aggregationsmethoden vor, die eine räumliche, zeitliche und räumlich-zeitliche Analyse zulassen. Leider wird in beiden Arbeiten nicht beschrieben, wie gegeneinander rotierte Traces für den Vergleich ausgerichtet werden sollen. Die Verfahren sind also nur anwendbar für Traces, deren globaler Bezug erhalten bleibt, und können somit nicht zur Evaluierung des vorgestellten Verschleierungsverfahrens hergenommen werden.

Eine mögliche Lösung bieten so genannte Feature-Methoden, die zum Beispiel im Bereich der Ähnlichkeitssuche bei der *computergestützen Konstruktion (CAD)* oder in Moleküldatenbanken zur Verwendung kommen [115] [31]. Diese Algorithmen sind invariant in Bezug auf Translationen und Rotationen, da sie nur relative Änderungen innerhalb der getesteten Objekte betrachten. Die in diesen Ansätzen beschriebenen Feature-basierten Ähn-

lichkeitsmetriken wurden der Evaluation zugrunde gelegt:

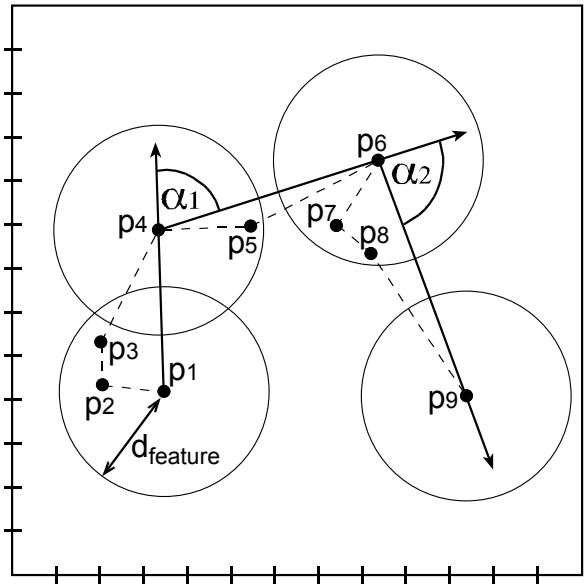

Abbildung 6.5: Berechnung eines Feature-Vektors mit distanzbasiertem Winkelprofil

Für jeden Trace λ lässt sich ein Feature-Vektor $F_\lambda = (\alpha_{\lambda 1}, ..., \alpha_{\lambda m}) \in \mathbb{R}^m$ berechnen, der dem distanzbasierten Winkelprofil von λ entspricht. Abbildung 6.5 illustriert die Berechnung. Parametrisiert durch die so genannte Feature-Distanz $d_{feature}$ wird ein Winkel α_k zu F_λ genau dann hinzugefügt, wenn die Distanz zwischen der aktuellen Position p_c und der letzten betrachteten Position p_l größer oder gleich $d_{feature}$ ist. α_k entspricht der Differenz zwischen der zuletzt betrachteten Bewegungsrichtung und der aktuellen.

Die Ähnlichkeit zweier Feature-Vektoren $F_\lambda = (\alpha_{\lambda 1}, ..., \alpha_{\lambda m})$ und $F_\mu = (\alpha_{\mu 1}, ..., \alpha_{\mu n})$ mit $m \leq n$ wird definiert als das Maximum der euklidischen Distanzen zwischen F_λ und allen möglichen Subsequenzen $(\alpha_{\mu i}, ..., \alpha_{\mu j}) \subseteq F_\mu$, $0 \leq i \leq j \leq n, j - i = m$. Bezogen auf den Angriff ist die Metrik also konservativ, da stets die schlechtestmögliche Übereinstimmung einer Subsequenz eines Traces mit einem anderen Trace herangezogen wird.

Simulationsergebnisse Jeder der 69 Traces wurde mit seiner verschleierten Version und mit allen anderen Traces (ebenfalls verschleiert) auf diese Weise 100-mal verglichen. Wie bereits erwähnt, wurde jeder Simulationslauf mit einem anderen Schlüssel s_G durchgeführt. Bei den Simulationen stellte sich heraus, dass eine Veränderung der Größe der Menge E von Zielpersonen, deren Traces mit demselben s_G verschleiert wurden, nur einen vernachlässigbaren Einfluss aus das Ergebnis hat.

Abbildung 6.6 zeigt Ergebnisse der durchgeführten Simulationen. Die x-Achse liefert die maximale Länge r_{max_local} des lokalen Verschleierungsvektors $\vec{v}$. Die y-Achse gibt den durchschnittlichen Anteil verschleierter Traces μ^* an, die einem gegebenen Trace λ ähnlicher sind als dessen verschleierte Version λ^* es zu λ ist. Anders gesagt entspricht der Wert der y-Achse bezogen auf die verwendeten Testdaten der Größe der in [134] definierten

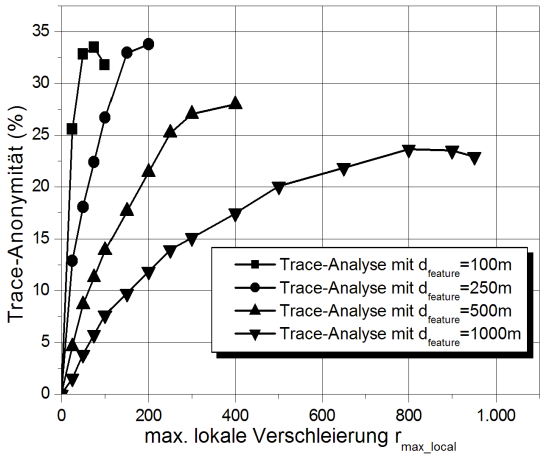

Abbildung 6.6: Ergebnisse der Simulation des strukturellen MPA

Anonymitätsmenge. Ein Wert von 100% steht dabei für die maximal erreichbare Trace-Anonymität, während ein Wert von 0% bedeutet, dass λ^* innerhalb der Daten stets eindeutig auf λ rückführbar ist, also keinerlei Trace-Anonymität gegeben ist.

Die maximale lokale Verschleierung r_{max_local} ist ja nach oben durch die minimale Abtastdistanz d_{sample} beschränkt, um den SUA abzuwehren. d_{sample} darf wiederum nicht größer sein als $\frac{b}{3}$, da anderenfalls keine korrekte Nahbereichs- und Trennungserkennung möglich ist (vgl. Abschnitt 6.1.2). Es gilt $r_{max\_local} \leq d_{sample} \leq \frac{b}{3}$. Je größer also r_{max_local} ist, desto größer muss b mindestens sein und desto schlechter ist die mögliche Präzision der Nahbereichs- und Trennungserkennung.

Die Distanz zwischen zwei nacheinander übermittelten Positionen ist folglich nie kleiner als r_{max_local}. Dadurch ist natürlich auch die Feature-Distanz $d_{feature}$, mit der ein Angreifer die Feature-Berechnung bei der Trace-Analyse parametrisieren kann, nach unten limitiert durch r_{max_local}. Eine feinere Auflösung würde ihm keine zusätzlichen Informationen liefern. Vier verschiedene Kurven sind abgebildet, die sich jeweils auf einen anderen Wert von $d_{feature}$ beziehen und die diesen Betrachtungen zufolge jeweils bei $r_{max\_local} = d_{feature}$ aufhören müssen. Aus Sicht des Angreifers liefert die Feature-Berechnung bei $d_{feature} = 1000m$ die besten Ergebnisse und zwar für alle möglichen Werte von r_{max_local}.

Offensichtlich führt eine Erhöhung von r_{max_local} zur Steigerung der Anzahl ähnlicher Traces. Setzt man r_{max_local} auf 100 m, so lässt sich eine mögliche Grenzlinientoleranz von $b = 300m$ für die Nahbereichs- und Trennungserkennung realisieren. Für diese Konfiguration sind im schlimmsten Fall ($d_{feature} = 1000m$) rund 7,5 % aller verschleierten Traces μ^* einem gegebenen Trace λ ähnlicher als dies der Fall für λ^* ist. Für $r_{max\_local} = 500m$ und somit $b \geq 1500m$ beträgt der Anteil fast 20 %.

Ob diese Werte hinreichende Anonymität gewährleisten, hängt hauptsächlich von der Anzahl an Traces ab, die im angegriffenen Datenbestand vorliegen. Zur Erreichung von k-Anonymität bezüglich eines vorgegebenen k innerhalb eines Datenbestands von x Traces, sollte dieser Anteil mindestens $\frac{k}{x}$ betragen. Die Kurven basieren außerdem auf Durch-

schnittswerten. Sehr charakteristische Traces sind also leichter zurückführbar als durchschnittliche. Schließlich basieren die gezeigten Ergebnisse auf einer begrenzten Anzahl gesammelter Daten. Um noch zuverlässigere Aussagen zu erhalten, müssten wohl mehr und auch andersartige Traces gesammelt werden, zum Beispiel von typischem Fahrverhalten auf Autobahnen, was leider im Rahmen dieser Arbeit nicht möglich war.

Dennoch ist zu bemerken, dass das simulierte Szenario dem ungünstigsten Fall entspricht, bei dem Daten kontinuierlich mit der minimalen Abtastrate von d_{sample} übertragen werden. Wie schon in Abschnitt 6.1.2 besprochen, führen die in dieser Arbeit vorgestellten Strategien typischerweise zu weit weniger gesammelter Positionsdaten, was letztendlich die Anonymität der Traces fördert.

Den MPA zusammenfassend ist zu sagen, dass es mit Hilfe der lokalen Verschleierung wohl grundsätzlich möglich ist, Traces vor strukturellen Angriffen zu schützen. Je größer der entsprechende Translationsvektor r_{max_local}, desto größer ist die Verschleierung. Der vom Ansatz gelieferte Schutz ist also qualitativ mit der Präzision der Nahbereichs- und Trennungserkennung verknüpft. Durch das Inkaufnehmen einer höheren Grenzlinientoleranz lässt sich der gebotene Schutz prinzipiell erhöhen.

Zusammenfassung der Evaluierung

Grundsätzlich ergab die Evaluierung die folgenden Schwachstellen.

- Der Mechanismus ist wohl nur für relativ kleine Gruppen gemeinsam überwachter Zielpersonen geeignet, zum Beispiel, um zwischen zwei Zielpersonen die Nahbereichserkennung durchzuführen. Wenn man hingegen eine größere Menge von Zielperson gemeinsam betrachtet, zum Beispiel zur Erkennung von Cliquen innerhalb einer größeren Gruppe, fallen statistische Angriffe wesentlich leichter.

- Die Robustheit des Ansatzes ist formal schwer beweisbar. Auch lassen sich kaum quantitative Aussagen treffen.

Trotzdem ist der Ansatz im Gegensatz zu allen bestehenden Verfahren tatsächlich anwendbar auf die Nahbereichs- und Trennungserkennung und bietet im Vergleich zu einer unverschleierten Übertragung einen erhöhten Schutz, zum Beispiel vor dem KWA. Außerdem ist er mit relativ einfachen Mitteln umsetzbar.

6.2 Autorisierung des Zugriffs auf Ortsinformationen

Die breite Markteinführung von Mobiltelefonen Anfang der 90er-Jahre führte zum gesellschaftlichen Prinzip der ständigen Erreichbarkeit, welches neben vielen unbestreitbaren Vorteilen auch mit Nachteilen behaftet ist. Ständig für andere *erreichbar* zu sein, bedeutet für viele Personen eine Einschränkung ihrer Privatsphäre. Die in dieser Arbeit behandelten Community-Dienste führen das Prinzip der ubiquitären Erreichbarkeit leider konsequent weiter. Die vorausgesagte breite Verfügbarkeit moderner Mobiltelefone, die mit hochakkuraten Ortungstechnologien wie GPS ausgestattet sind, erlaubt nicht nur die verbreitete Nutzung lokaler ortsbezogener Anwendungen wie Navigation. Plötzlich sind Individuen jederzeit in der Lage, ihren aktuellen Aufenthaltsort über das Mobilfunknetz an andere Perso-

nen weiterzuleiten. Technisch ist es damit möglich, jederzeit für andere exakt *lokalisierbar* zu sein.

Existierende LBCSs wie [2; 15; 136] machen sich diese Technik heute schon zunutze. Beispiele sind Friend-Finder, Child-Tracker und Dating-Dienste sowie ortsbezogenes IM. Um auf dem Markt überhaupt akzeptiert zu werden, stellen LBCSs Mechanismen bereit, die der Zielperson Kontrolle darüber geben, welche Nutzer unter welchen Umständen auf ihre Ortsinformationen zugreifen dürfen. Bislang ist hierfür eine *explizite Autorisierung* vorgesehen, bei der die Zielperson explizit darüber entscheidet, wer Zugriff erhält. Sie kann auf zwei grundlegende Arten umgesetzt werden. Zum einen kann die Zielperson beim LBS Provider so genannte *Datenschutz-Policies* hinterlegen, welche definieren, wer Zugriff auf ihre Ortsinformationen hat und wer vom Zugriff ausgeschlossen ist. Der zweite Ansatz wird als *Ad-hoc-Autorisierung* bezeichnet. Die Zielperson wird hier interaktiv um Zustimmung gebeten, wenn ein Nutzer ihre Ortsinformationen anfragt.

Explizite Autorisierung ist leider mit einigen Nachteilen verbunden. Erstens führt sie zu erheblichem Verwaltungsaufwand, der sich in einem Fall der A-priori-Definition von Policies widmet und im anderen Fall auf die manuelle Interaktion der Zielperson mit dem Autorisierungssystem des LBCS bezieht. Zweitens birgt explizite Autorisierung ein erhöhtes Risiko, soziale Schwierigkeiten herbeizuführen. Ein Nutzer, der den Aufenthaltsort einer Zielperson abrufen will, könnte sich zurückgewiesen fühlen, wenn der Zugriff verwehrt wird, oder er könnte das Gefühl haben, die Zielperson möchte etwas vor ihm verbergen. Auf der anderen Seite könnte sich die Zielperson unter Druck gesetzt fühlen, dem Nutzer den Zugriff nur deswegen zu gestatten, um die mit einer Zurückweisung verbundenen negativen sozialen Auswirkungen zu vermeiden.

Es wird deshalb ein neues Konzept vorgeschlagen, das *implizite Autorisierung* vorsieht und das auch in [159] vorgestellt wurde. Im Gegensatz zu einer Ja/Nein-Entscheidung der Zielperson führen hier bestimmte wohldefinierte Aktivitäten der Zielperson dazu, dass ein bestimmter Nutzer Zugriff auf ihre Ortsinformationen erhält. Die Aktivität bezieht sich dabei *nicht* auf die Verwaltung der eigenen Ortsinformationen. Anstelle einer expliziten Zurücknahme des erteilten Zugriffs werden so genannte *Leases* verwendet, die nach einer bestimmten Zeit automatisch verlöschen. Soll die Möglichkeit zum Zugriff länger bestehen bleiben, muss ihn die Zielperson regelmäßig erneuern.

In einer vorgeschlagenen Realisierung der impliziten Autorisierung erhält ein anfragender Nutzer A für eine bestimmte Zeitdauer Zugriff auf die Ortsinformationen einer Zielperson B, falls im Gegenzug B vorher versucht hat, auf die Ortsinformationen von A zuzugreifen. Durch den Versuch, die Ortsinformationen einer bestimmten Person abzurufen, wird also implizit gestattet, dass diese Person eine Zeit lang auf die eigenen Informationen zugreift.

Der Ansatz generiert weniger Verwaltungsaufwand als die explizite Autorisierung, während eine bessere soziale Verträglichkeit gewährleistet wird. Erstens wird *reziproker Informationsaustausch* gefördert, was für datenschutzkritische Anwendungen wie LBCSs grundsätzlich wünschenswert ist. Zweitens unterstützt der Mechanismus natürlicherweise das Konzept der so genannten *plausiblen Abstreitbarkeit (plausible deniability)* (vgl. Abschnitt 6.2.1), da explizite Zugriffsverweigerungen durch die Zielperson vermieden werden. Stattdessen wird *Mehrdeutigkeit (Ambiguity)* unterstützt [30], weil erfolglose Zugriffsversuche auf die Tatsache zurückgeführt werden können, dass die Zielperson eine bestimmte Aktivität nicht ausgeführt hat. Dies wiederum kann mehrere Ursachen haben, zum Beispiel ökonomische Beweggründe, und muss nicht als „böser Wille“ der Zielperson oder Zurück-

weisung empfunden werden.

Die Darstellung des Ansatzes gliedert sich wie folgt. Abschnitt 6.2.1 diskutiert die grundlegenden Anforderungen, die ein geeigneter Autorisierungsmechanismus für Ortsinformationen erfüllen sollte. Wichtig ist zum Beispiel eine einfache Bedienbarkeit sowie die Möglichkeit der Zielperson, ausreichende Kontrolle auszuüben. Letztere Anforderung wird anhand der Begriffe *Korrektheit* und *Vollständigkeit* formal dargestellt. Darüber hinaus wird das Konzept des reziproken Informationsaustausches sowie der Mehrdeutigkeit genauer beschrieben. Abschnitt 6.2.2 überblickt und bewertet bestehende Techniken zur Autorisierung. Abschnitt 6.2.3 beschreibt den eigenen Ansatz der impliziten Autorisierung genauer und stellt zwei mögliche Realisierungen vor, eine kalenderbasierte und eine anfragebasierte Methode. Die anfragebasierte Methode wird in Abschnitt 6.2.4 genauer analysiert und mittels statistischer Simulationen untersucht.

6.2.1 Herausforderungen

Die folgenden Anforderungen sind von zentraler Bedeutung für einen Autorisierungsmechanismus, der sich für LBCSs eignen soll.

Kontrolle

Ein geeigneter Autorisierungsmechanismus sollte der Zielperson jederzeit Kontrolle darüber geben, wer auf ihre Ortsinformationen zugreift. Durch das System kommunizierte Zugriffsentscheidungen sollten also den aktuellen Einstellungen der Zielperson so gut wie möglich entsprechen.

Um diesen Aspekt formal auszudrücken, werden die Begriffe *Korrektheit* und *Vollständigkeit* der Autorisierung eingeführt, die analog definiert sind zu den Metriken *Precision* und *Recall* aus dem Bereich des *Information Retrieval (IR)*. Bezogen auf eine gewisse Anzahl von Ortungsversuchen bezeichnet Korrektheit den Anteil der durch das System erlaubten Zugriffe, die tatsächlich auch so von der Zielperson gewünscht worden sind. Im Gegenzug bezeichnet Vollständigkeit den Anteil der von der Zielperson zu erlauben erwünschten Zugriffsversuche, die vom System tatsächlich erlaubt wurden. Anstelle von *Positiven* und *Negativen*, wie beim IR, spricht man von *Zugriffsbewilligungen* und *Zugriffsverweigerungen*. Die Zustimmung der Zielperson zu einer erfolgten Bewilligung oder Verweigerung wird mit den Adjektiven *wahr* (Zielperson ist einverstanden) bzw. *falsch* (Zielperson ist nicht einverstanden) bezeichnet.

Ebenfalls analog zu Precision und Recall ist die Tatsache, dass es relativ leicht fällt, ein hohes Maß an Korrektheit zu erreichen, wenn die Vollständigkeit dabei nicht berücksichtigt werden muss. Das gilt auch andersherum. Leicht lässt sich eine hohe Vollständigkeit erreichen, wenn keine hohe Korrektheit gefordert ist. Ein Beispiel für die erste Variante ist ein extrem restriktives Autorisierungsschema, welches niemandem den Zugriff auf die eigenen Ortsinformationen gestattet. Offensichtlich würde hier ein hoher Wert für die Korrektheit erreicht, da keine *falschen Zugriffsbewilligungen* auftreten können. Die erreichte Vollständigkeit wäre allerdings schlecht, da potentiell eine große Anzahl *falscher Zugriffsverweigerungen* resultieren würden. Als Beispiel für die zweite Variante dient ein extrem freizügiger Ansatz, der jedem anderen Zugriff gewährt. Nachdem so ganz sicher keine falschen Zugriffsverweigerungen passieren, wäre die erreichte Vollständigkeit sehr hoch.

Ein schlechter Wert würde jedoch für die Korrektheit erreicht, nachdem unter Umständen sehr viele falsche Zugriffsbewilligungen vorkämen. Zusammenfassend lässt sich also sagen, dass ein Autorisierungsmechanismus im Idealfall gleichzeitig hohe Werte für die Korrektheit und für die Vollständigkeit erreichen sollte.

Einfachheit

Neben der Bereitstellung geeigneter Kontrollmechanismen sollte ein Autorisierungsmechanismus einfach genug sein, um sowohl aus Sicht der Nutzer und Zielpersonen als auch des LBS und Location Providers mit annehmbarem Aufwand nutzbar bzw. umsetzbar zu sein.

Einfache Benutzung heißt aus Sicht der Zielperson zum einen, dass diese möglichst wenig zeitlichen Aufwand damit hat, Zugriffsrechte bezüglich ihrer Ortsinformationen zu verwalten. Solcher Aufwand kann zum Beispiel darin bestehen, entsprechende Datenschutz-Policies zu definieren oder, im Fall von Ad-hoc-Autorisierung, interaktiv in die Autorisierungsprozedur eingebunden zu werden. Ein geeigneter Mechanismus sollte selbstverständlich auch intuitiv bedienbar und nicht zu komplex sein, so dass er wirklich von jeder, auch technisch nicht versierten Person genutzt werden kann. Dieser letzte Aspekt stellt insbesondere bei der Definition von Datenschutz-Policies ein Problem dar, vergleiche [84]. Um wirklich den aktuellen Einstellungen einer Zielperson zu entsprechen, sollten Policies fein abgestimmt sein und regelmäßig überarbeitet werden.

Einfachheit ist aus der Sicht des LBS oder Location Providers vor allem wünschenswert in Bezug auf die Umsetzung des Mechanismus, was insbesondere die Integration in bestehende Technologien betrifft. In dieser Hinsicht scheint der Policy-basierte Ansatz der Ad-hoc-Autorisierung überlegen zu sein, da die Definition der Policies von ihrer Durchsetzung prinzipiell entkoppelt ist. Es ist somit keine immer zur Verfügung stehende Nutzerschnittstelle notwendig, mit deren Hilfe die Zielperson um Zustimmung gefragt werden kann. So können wiederum verschiedene Endgerätetypen und Positionierungsverfahren zum Einsatz kommen, die interaktive Dialoge mit der Zielperson schwer machen. Ein Beispiel sind netzbasierte und endgerätunterstützte Verfahren, bei denen das Endgerät relativ wenig Funktionalität bieten muss.

Soziale Verträglichkeit

Sollten LBCSs wirklich ein Teil des täglichen Lebens vieler Menschen werden, so müssen ihre sozialen Implikationen fundiert analysiert werden. Mögliche negative Auswirkungen müssen durch ein geeignetes Design so weit wie möglich reduziert werden. Dies stellt allerdings eine Aufgabe dar, die alleine mit den bekannten Methoden der Softwaretechnik nicht gelöst werden kann. Gefordert ist vielmehr die interdisziplinäre Zusammenarbeit verschiedener Fachbereiche wie der Informatik, der Soziologie und der Sozialpsychologie. Dass sozialen Aspekten bei der Nutzung ubiquitärer Dienste in letzter Zeit immer mehr Bedeutung zukommt, sieht man zum Beispiel an Arbeiten wie [73] und [77], in denen Mechanismen zur sozialen Verträglichkeit datenschutzkritischer Anwendungen behandelt werden. Zwei Konzepte scheinen besonders wichtig, vergleiche [136]: *Reziprozität* und *glaubhafte Abstreitbarkeit* bzw. *Mehrdeutigkeit*.

Reziprozität Dies ist ein wohlbekanntes Prinzip zur Ausbalancierung des Austausches datenschutzkritischer Informationen zwischen zwei Parteien. Das Prinzip besagt, dass die Menge von Informationen, die eine Person A einer anderen Person B preisgibt, in etwa der Menge entsprechen sollte, die B an A überträgt. Nach [81], wo die Anwendung des Prinzips auf kontextsensitive Dienste diskutiert wird, soll damit erreicht werden, negative soziale Situationen zu vermeiden, in denen bestimmte Personen mehr über andere wissen, als sie selbst bereit sind preiszugeben. Diese negativen Situationen werden in dem Artikel mit *ökonomischen Ineffizienzen* verglichen, die in einer Marktwirtschaft entstehen, sobald ein bestimmtes Wissen ungleich verteilt ist, also so genanntes Insider-Wissen bei Kauf- und Verkaufsentscheidungen herangezogen wird. Angewendet auf kontextsensitive Dienste sollen Personen, die eigene Kontextinformationen an andere weitergeben, gleichzeitig eine entsprechende Menge an Informationen über den anderen erhalten.

Diese und ähnliche Arbeiten sehen Reziprozität in Bezug auf reine Informationsflüsse vor, was im Vergleich zu unidirektionalen Flüssen bereits Vorteile bietet. Ein mögliches Risiko entsteht jedoch, wenn das gegenseitige Interesse an den ausgetauschten Informationen nicht ausbalanciert ist. Man nehme zum Beispiel an, zwei Personen A und B seien Mitglieder desselben sozialen Netzes und B würde gern auf den Aufenthaltsort von A mittels eines LBCS zugreifen. Unter der Voraussetzung von klassischer Reziprozität beschließt A, sich von B orten zu lassen, falls der Informationsfluss ausgeglichen ist. Wann immer also B die Position von A erhält, muss B den eigenen Aufenthaltsort ebenfalls preisgeben. Diese Konstellation könnte jedoch für A immer noch nachteilig sein, falls A kein großes Interesse daran hat, über die aktuelle Position von B regelmäßig in Kenntnis gesetzt zu werden. Es könnte zum Beispiel sein, dass es B relativ gleichgültig ist, ob A den Aufenthaltsort von B mehrmals am Tag erfährt, solange nur B regelmäßig Kenntnis über die Position von A erhält. Gleichzeitig könnte sich A kontrolliert fühlen.

Zusammenfassend wird also vorgeschlagen, die Anforderung der Reziprozität so zu erweitern, dass nicht nur symmetrische Informationsflüsse entstehen, sondern gleichzeitig ein gegenseitiges Interesse an der ausgetauschten Information vorliegen soll.

Mehrdeutigkeit Der Begriff *glaubhafte Abstreitbarkeit (plausible deniability)* bezeichnet in der Politik eine Doktrin, die in den USA in den 1950er-Jahren entwickelt wurde und die das Ziel hatte, geheime Aktionen der damals neu gegründeten *Central Intelligence Agency (CIA)* so gut wie möglich zu verschleiern. Glaubhafte Abstreitbarkeit bezeichnet auch ein grundlegendes Datenschutzkonzept in ubiquitären Umgebungen, vergleiche [102]. Hier soll eine Person A, die auf die Daten einer anderen Person B zugreifen oder mit B kommunizieren möchte, nicht feststellen können, ob eine erfolgte Zurückweisung von B beabsichtigt war. Ein offensichtliches Beispiel ergibt sich aus dem täglichen Umgang mit Mobiltelefonen. Möchte man – zum Beispiel zur Vermeidung eines unangenehmen Anrufs – zeitweise nicht erreichbar sein, so bietet es sich an, das Mobilgerät abzuschalten. Aus der Sicht des vergeblich Anrufenden lässt sich dann nicht eindeutig sagen, ob das temporäre Nichterreichen auf absichtliches Abschalten zurückzuführen ist oder ob eventuelle technische Schwierigkeiten, wie fehlender Funkkontakt, dafür verantwortlich sind.

In diesem Kontext ist der Begriff der *Mehrdeutigkeit (Ambiguity)* dem der glaubhaften Abstreitbarkeit sehr ähnlich, er versteht sich aber etwas allgemeiner. Mehrdeutigkeit beschränkt sich nämlich nicht nur auf die Mediation expliziter Zugriffsverweigerung. Sie

ist deshalb auch gut geeignet für die weiter unten vorgeschlagene implizite Autorisierung. Mehrdeutigkeit wird zum Beispiel von [30] als Mittel zur Privacy-Verbesserung bei *persönlichen Kommunikationssystemen (Personal Communication Services, PCSs)* wie Push-to-Talk diskutiert. Wird dem Anrufenden ein erfolgloser Kommunikationsversuch so vermittelt, dass die Ursache des Fehlschlagens auf verschiedene Arten interpretiert werden kann, so lassen sich mögliche, im Nachhinein entstehende soziale Schwierigkeiten zwischen Anrufendem und Angerufenem verhindern. Die Autoren, die sich bei ihren Aussagen auf umfangreiche soziologische Studien stützen, bezeichnen den dabei ablaufenden sozialen Prozess auch als „Face Work“. Durch Mehrdeutigkeit soll der erfolglose Anrufer sein Gesicht wahren können.

Außer möglichen technischen Fehlfunktionen, wie bei der glaubhaften Abstreitbarkeit, werden noch Alternativen untersucht, die Raum zur mehrdeutigen Interpretation bieten. Eine Variante sind mögliche ökonomische Einschränkungen des Angerufenen, welche ihn daran hindern könnten, einen bestimmten Anruf anzunehmen. Wird unter diesem Gesichtspunkt ein Anruf abgelehnt, so muss sich der Anrufende nicht zurückgewiesen fühlen. Interessant ist dabei die Aussage, dass alternative Deutungsmöglichkeiten für einen abgelehnten Anruf nicht extrem glaubhaft sein müssen, um ihren Zweck zu erfüllen (wie es die glaubhafte Abstreitbarkeit fordert). Alleine schon durch die Tatsache, dass das Fehlschlagen nicht unbedingt auf absichtliches Zurückweisen zurückzuführen ist, lässt die beiden Beteiligten ihr Gesicht voreinander wahren. Eine Schlussfolgerung aus [30] ist, dass entsprechende Systeme bereits so entworfen sein sollten, dass möglichst viele Möglichkeiten zur mehrdeutigen Auslegung von fehlgeschlagenen Kommunikationsversuchen bestehen.

Bezogen auf LBCSs bedeutet Mehrdeutigkeit also, dass vergebliche Ortungsversuche aus der Sicht des ortenden Nutzers auf verschiedene Arten ausgelegt werden können. Das mögliche Gefühl einer expliziten Zurückweisung wird somit zumindest gebremst. Die zu ortende Zielperson kann so wiederum freier und unter geringerem sozialen Druck darüber entscheiden, wer auf ihre Ortsinformationen Zugriff erhält. Die Erforschung neuer, geeigneter Mechanismen zur Unterstützung von Mehrdeutigkeit ist ein offenes und überaus wichtiges Thema speziell bei den LBCSs, aber auch bei kontextsensitiven Diensten allgemein. Der zum einen durchwegs erfreuliche Trend, dass die von LBCSs verwendeten Rechnersysteme und -infrastrukturen immer zuverlässiger werden, verschärft die Dringlichkeit der Bereitstellung neuer Mechanismen zur Mehrdeutigkeit dabei nur noch weiter. Eine nahezu perfekte Zuverlässigkeit birgt nämlich das Risiko, dass mögliche technische Ursachen für gescheiterte Nutzerzugriffe nicht mehr akzeptiert werden.

6.2.2 Bestehende Ansätze der Autorisierung

Dieser Abschnitt bietet einen Überblick über bestehende Mechanismen zur expliziten Autorisierung des Zugriffs auf Ortsinformationen, nämlich die Policy-basierte Autorisierung, die Ad-hoc-Autorisierung sowie hybride Autorisierungstechniken.

Policy-basierte Autorisierung

Nach [57] definieren Autorisierungs-Policies die *Bedingungen (Constraints)*, anhand derer eine bestimmte Information oder eine Menge an Informationen an eine bestimmte En-

tität ausgehändigt werden darf. Für Ortsinformationen bestehen Autorisierungs-Policies typischerweise aus mehreren Arten solcher Bedingungen, vergleiche [121]:

- *Aktorenbedingungen (Actor Constraints)* beschränken den Zugriff auf die Ortsinformationen einer Zielperson auf eine bestimmte Gruppe von Nutzern.
- *Zeitbedingungen (Time Constraints)* schreiben vor, dass Ortsinformationen nur zu gewissen Zeiten herausgegeben werden dürfen.
- *Ortsbedingungen (Location Constraints)* beschränken den Zugriff auf bestimmte vordefinierte Orte, an denen sich die Zielperson bei der Ortung aufhalten muss.
- *Genauigkeitsbedingungen (Accuracy Constraints)* ermöglichen es der Zielperson, die Genauigkeit der herausgegebenen Ortsinformation künstlich zu verschlechtern. Wie schon in Kapitel 6.1 beschrieben, sind Genauigkeitsbedingungen als Erweiterung des Geopriv Standards [149], einem (zumindest in wissenschaftlichen Arbeiten) verbreiteten Protokoll zum Austausch von Ortsinformationen, angedacht. Mit ihrer Hilfe lassen sich also Ortsinformationen verschleiern.

Autorisierungs-Policies werden typischerweise von der Zielperson erstellt, sind aber bei einem anderen Akteur, zum Beispiel dem LBS Provider, hinterlegt und werden dort durchgesetzt.

Ad-hoc-Autorisierung

Während Autorisierungs-Policies bereits im Vorhinein hinterlegt werden müssen, bindet Ad-hoc-Autorisierung die Zielperson interaktiv in den Autorisierungsvorgang mit ein. Ad-hoc-Autorisierung wird zum Beispiel favorisiert in Arbeiten, die aus dem PlaceLab-Projekt [100] hervorgingen. Konkrete Anwendungsfälle sind die beiden Systeme Reno und Boise, mit denen Personen innerhalb einer Community ihre Ortsinformationen austauschen können und die die Grundlage für eine Reihe von Feldversuchen und Nutzerstudien lieferten, vergleiche [56; 150; 76].

Zur Steigerung der sozialen Verträglichkeit von Ad-hoc-Autorisierung werden verschiedene Techniken vorgeschlagen und untersucht. Eine ist die Benutzung *stiller Time-outs*. Antwortet die Zielperson auf eine Autorisierungsanfrage nicht innerhalb eines bestimmten Zeitraums, wird dem anfragenden Nutzer der Zugriff verwehrt und die Anfrage wird verworfen, das heißt, die Zielperson bekommt von der Anfrage nichts mit, falls sie sie tatsächlich übersehen hatte. So entsteht also eine gewisse Mehrdeutigkeit, nachdem nicht klar ist, ob die Zielperson den Zugriff absichtlich verweigert hat. Kritisch anzumerken ist hier jedoch, dass es wohl im Falle einer mehrfachen Abweisung immer schwerer wird, die Zielperson von absichtlichem „Wegdrücken“ freizusprechen.

Weiterhin bietet die Ad-hoc-Autorisierung eine sehr feingranulare Möglichkeit zur Täuschung. Die Zielperson kann sich also – zum Beispiel, um gewissen Situationen sozialen Drucks zu entgehen – dazu entscheiden, dem Nutzer eine falsche Information über ihren Aufenthaltsort zu vermitteln.

Auch ist zur Ad-hoc-Autorisierung positiv anzumerken, dass sie der Zielperson die Möglichkeit bietet, den Grund der Ortung vom Nutzer zu erfragen und dann situativ zu entscheiden. Dies fördert den reziproken Informationsaustausch.

Hybridformen

Policy-basierte und Ad-hoc-Autorisierung können auch kombiniert werden. Mittels so genannter *Benachrichtigungsbedingungen (Notification Constraints)*, die ebenfalls Teil einer Autorisierungs-Policy sein können, kann die Zielperson Situationen spezifizieren, in denen sie gerne ad hoc um Zustimmung gebeten möchte. Viele Netzbetreiber bieten diese Kombination auch in einer weiteren Variante an. Wenn Externe versuchen, das Endgerät eines Teilnehmers zu orten, kann der Teilnehmer ad hoc bestimmen, ob dem Anfragenden der Zugriff einmal genehmigt, grundsätzlich genehmigt oder grundsätzlich verboten werden soll oder ob grundsätzlich ad hoc entschieden werden soll. *Benachrichtigungen (Notice)*, die der Zielperson von erfolgten Ortungsversuchen berichten, können auch rein informativer Natur sein, dass heißt, die Zielperson bleibt dabei passiv. Sie werden als grundlegende Technik zur Privacy-Verbesserung gesehen, vergleiche [101].

Erwähnenswert ist schließlich die in [103] beschriebene Studie, welche aussagt, dass die Einstellung einer Zielperson bezüglich der Autorisierung eines bestimmten Nutzers stärker von dessen Identität als von der aktuellen Situation der Zielperson abhängig ist. Eine einmal getroffene Autorisierungsentscheidung bezüglicher eines bestimmten Nutzers bleibt also oft über verschiedene Situationen hinweg konstant.

6.2.3 Implizite Autorisierung

Die beschriebenen Mechanismen zur expliziten Autorisierung geben der Zielperson Kontrolle über ihre Ortsinformationen, führen aber leider zu den beiden folgenden Problemen: Erstens hat ein anfragender Nutzer, dem der Zugriff verweigert wird, trotz der benannten Vorschläge zur besseren sozialen Verträglichkeit (stille Time-outs, situative Täuschungen usw.) guten Grund, anzunehmen, dass die Zugriffsverweigerung auf expliziten Wunsch der Zielperson geschieht. Die Möglichkeiten der Mehrdeutigkeit sind damit stark eingeschränkt. Zweitens erfordern alle genannten Mechanismen erheblichen Zeitaufwand der Zielperson beim Verwalten ihrer Ortsinformationen. Ein damit verbundenes Problem ist die Bereitstellung einer geeigneten Nutzerschnittstelle.

Eine mögliche Lösung, zumindest für bestimmte Arten von LBCSs, könnte der im Folgenden vorgestellte Ansatz der *impliziten Autorisierung* sein. Ähnlich wie von [30] für PCSs vorgesehen, unterliegen hierbei einer Autorisierung zeitbasierte *Leases*, welche durch die Zielperson erteilt werden. Im Folgenden wird genau beschrieben, auf welche Art Leases erteilt werden sollen und warum. Es zeigt sich, dass der Ansatz praktikabler und weniger künstlich ist als der von [30] für PCSs, der diesbezüglich in [41] bemängelt wurde.

Implizite Autorisierung stellt ein völlig neues Konzept bei der Verwaltung eigener vertraulicher Informationen wie Positionsdaten dar. Das Konzept wird daher erst allgemein erklärt und dann anhand zweier möglicher Realisierungen illustriert. Die zweite der beschriebenen Umsetzungen ist wohl breiter einsetzbar und wird deswegen im Anschluss eingehender untersucht.

Bei der impliziten Autorisierung gewährt die Zielperson einem bestimmten Nutzer Zugriff auf ihre Ortsinformationen, indem sie eine vorher definierte, unten näher beschriebene Aktivität ausführt. Anstelle eines expliziten Widerrufs des so erteilten Zugriffsrechts, ist dessen Gültigkeit immer zeitlich beschränkt. Ein solches Zugriffsrecht wird daher auch als *Lease* bezeichnet. Ein Lease ist immer paarweise definiert und gerichtet. Es bezieht sich

also auf genau ein Paar aus Zielperson und Nutzer. Verlängern lässt sich ein Lease nur, wenn die Zielperson die entsprechende Aktivität wiederholt. Verschiedene Realisierungen der impliziten Autorisierung können sich also zum einen darin unterscheiden, wie genau die Gültigkeit des Leases beschränkt ist, zum anderen in der Art der durchzuführenden Aktivität. Im Folgenden werden daher zunächst drei allgemeine Anforderungen motiviert, denen die Aktivität gehorchen muss.

- **Aktivität zeigt auf Nutzer.** Offensichtlich muss die von der Zielperson ausgeführte Aktivität eindeutig auf den potentiellen Nutzer hinweisen. Wahllos im Internet herumzusurfen erfüllt dieses Kriterium beispielsweise nicht. Das Schreiben einer E-Mail an eine bestimmte Person erfüllt sie hingegen schon.

- **Aktivität verursacht Kosten bei der Zielperson.** Wie vorher schon kurz diskutiert, bietet die Knappheit bestimmter Ressourcen auf Seite der Zielperson eine gute Möglichkeit, um erfolgte Zugriffsverweigerung beim Nutzer mehrdeutig interpretierbar zu machen. Verursacht das Aufrechterhalten eines Leases bei der Zielperson nämlich gewisse Kosten, so kann eine Verweigerung auch auf die Knappheit der entsprechenden Ressourcen zurückgeführt werden und muss nicht unbedingt als Zurückweisung empfunden werden. In anderen Worten: Kostet die Zielperson ein „Ja“ etwas, so kann ein „Nein“ vom Nutzer besser akzeptiert werden. Bei den bestehenden expliziten Techniken ist ein „Ja“ leider genauso teuer wie ein „Nein“, weshalb die Zielperson bei einem „Nein“ in größere Erklärungsnöte kommen kann.

  Mögliche Ressourcen der Zielperson, die bei der Durchführung der Aktivität verbraucht werden könnten und mit deren Hilfe Mehrdeutigkeit erzeugt werden könnte, sind folgende:

  – Die Aktivität beansprucht die Zeit der Zielperson.
  – Besondere Aufmerksamkeit der Zielperson ist nötig, um die Aktivität immer rechtzeitig vor Ablauf des Leases auszuführen.
  – Virtuelle Ressourcen, wie zum Beispiel Web Space, werden durch die Aktivität verbraucht.
  – Monetäre Kosten sind mit der Aktivität verknüpft.

- **Aktivität geschieht aus eigenem Antrieb.** Das Risiko, welches dadurch entsteht, dass die Aktivität mit Kosten verknüpft ist, ist, dass ein solcher Mechanismus als künstliche Schikane empfunden und nicht akzeptiert werden könnte. Die Zielperson könnte sich fragen, weshalb sie nicht einfach in der Lage ist, einem bestimmten Nutzer Zugriff zu gewähren, und zwar ohne den gerade beschriebenen, zusätzlichen Aufwand. Um dieses mögliche Problem zu umgehen, sollte die Aktivität mit einer natürlichen Tätigkeit der Zielperson übereinstimmen, die aus eigenem Antrieb durchgeführt wird und nicht den alleinigen Zweck der Verwaltung von Ortsinformationen hat. Statt als zusätzlicher, künstlicher Aufwand soll der Mechanismus von der Zielperson als einfaches Mittel zur Ausübung von Kontrolle empfunden werden, der deshalb so entworfen wurde, um den mit expliziter Autorisierung verbundenen Verwaltungsaufwand zu reduzieren.

Im Folgenden werden zwei Realisierungen impliziter Autorisierung vorgestellt, die jeweils auf einer solchen Aktivität basieren.

Kalenderbasierte Autorisierung

Eine Person A, die zusammen mit Person B dasselbe Kalenderverwaltungssystem benutzt, könnte B zum Zugriff auf ihre Ortsinformationen berechtigen, indem sie B zu einem Treffen einlädt oder eine solche Einladung von B bestätigt. Der Zeitraum, während dessen geortet werden darf, ist sowohl A also auch B bekannt und ist definiert in Bezug auf die Zeit des Treffens, also zum Beispiel angefangen von einer halben Stunde vor dem Treffen bis zu dessen Ende. Ortungsversuche werden also nur dann autorisiert, wenn sie auch sinnvoll erscheinen, zum Beispiel wenn eine Person zum Treffen zu spät kommt. Gleichzeitig wird ein übermäßiger Austausch privater Daten verhindert. Der Mechanismus bietet Kontrolle, ist einfach zu benutzen und sozial verträglich durch implizite Autorisierung.

Nachdem die zugrunde liegende Aktivität (ein Treffen ausmachen) sich auf ein persönliches Treffen zwischen A und B bezieht, wird klar auf den jeweiligen Nutzer A bzw. B verwiesen. Weiterhin verbraucht die Aktivität Ressourcen der jeweiligen Zielperson, deren Anzahl möglicher Kalendertermine natürlich begrenzt sind. Ein anfragender Nutzer, der keine solche Verabredung mit der Zielperson hat, kann dieser nur schwer einen Vorwurf machen, wenn er von der Ortung ausgeschlossen ist. Schließlich geschieht die Aktivität aus eigenem Antrieb der jeweiligen Zielperson, die ihre Kalendertermine sowieso verwalten muss. Die Autorisierung wird daher nicht als zusätzlicher Aufwand empfunden. Nachdem in dem anvisierten Szenario Treffen jeweils beidseitig bestätigt werden müssen, fördert der Mechanismus auch reziproken Informationsaustausch.

Anfragebasierte Autorisierung

Während kalenderbasierte Autorisierung bereits einige Vorteile bietet, ist ihr Anwendungsfeld leider relativ beschränkt. Außerdem wird verlangt, dass ein entsprechender LBCS mit einem Kalenderverwaltungssystem gekoppelt ist. Bezogen auf die in Abschnitt 6.2.1 definierten Begriffe liefert kalenderbasierte Autorisierung ein hohes Maß an Korrektheit, die erreichbare Vollständigkeit ist allerdings schlecht, falls die Zielperson auch für Personen lokalisierbar sein möchte, mit denen sie keine Verabredung hat.

Im Folgenden wird daher *anfragebasierte Autorisierung* vorgestellt und untersucht. Anfragebasierte Autorisierung stellt eine allgemein anwendbare Technik dar, die eine höhere Vollständigkeit als die kalenderbasierte Variante bietet.

Zentrales Konzept ist die Reziprozität. Die Technik eignet sich daher für LBCSs, deren Mitglieder gleichberechtigt, freiwillig und in einem ungefähr gleichen Maße Ortsinformationen miteinander austauschen. Für LBCSs, die vornehmlich auf einseitigem Tracking beruhen, wie Child-Tracker oder Dienste, die zum Beispiel einen Firmeninhaber stets über den Aufenthaltsort seiner Angestellten auf dem Laufenden halten, sind also andere Mechanismen notwendig. Hier ergeben sich teilweise schwerwiegendere Situationen sozialen Drucks, die auch durch Schaffen von Mehrdeutigkeit nicht abzuwehren sind. Grundsätzlich sind einseitige Tracking-Dienste, bei denen nicht unmittelbar hilfsbedürftige Personen von anderen geortet werden, ethisch allerdings fragwürdig.

Das vorgeschlagene Autorisierungssystem lässt sich als Black Box beschreiben: Personen

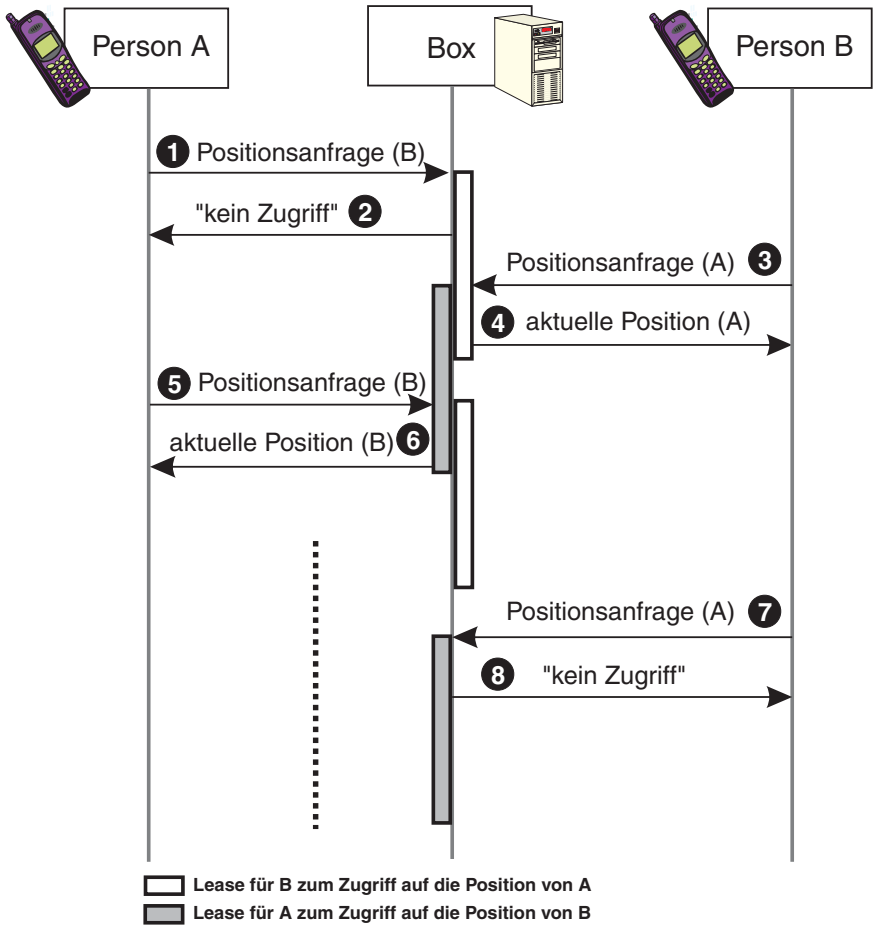

Abbildung 6.7: Typischer Ablauf der anfragebasierten Autorisierung

hinterlegen ihre Ortsinformationen in der Box und fragen die Ortsinformationen anderer aus der Box ab. Jedesmal wenn eine Anfrage bei der Black Box eintrifft, wird über die Bewilligung des Zugriffs anhand folgender einfachen Regel entschieden: Die Anfrage einer Person A nach dem aktuellen Aufenthaltsort einer Person B wird nur dann positiv beantwortet, wenn B zuvor nach der Ortsinformation von A angefragt hat. Selbstverständlich rufen die Personen die Ortsinformationen bei der Box nicht persönlich ab, sondern über einen durch Software realisierten Nutzeragenten. Durch das Abfragen von Ortsinformationen einer bestimmten Zielperson bei der Box wird dieser also implizit ein Lease zum Zugriff auf die eigenen Ortsinformationen erteilt.

Das Prinzip wird in Abbildung 6.7 illustriert: Der erste Versuch von A, Zugriff auf die Ortsinformationen von B zu erhalten, wird abgewiesen (1 und 2), weil B zu diesem Zeitpunkt noch nicht nach der Position von A angefragt hat. Durch die Anfrage von A wird jedoch in der Box ein Lease hinterlegt, welches B den Zugriff auf den Aufenthaltsort von A gestattet. B macht von diesem Lease Gebrauch (3 und 4) und berechtigt dadurch wiederum A, B eine Zeit lang zu lokalisieren (5 und 6). Nachdem A länger nicht mehr versucht, B zu orten, erlöscht das Lease irgendwann, und der spätere Versuch von B, A zu orten, bleibt erfolglos (7 und 8).

Zusätzlich zur zeitlichen Beschränkung kann die Gültigkeit eines Lease an die folgenden Bedingungen sowie Kombinationen daraus geknüpft sein.

- Ein Lease kann nur für eine bestimmte *Anzahl von Anfragen* gelten. Auf die Ortsinformationen des Ausstellers darf also während der zeitlichen Gültigkeit des Lease höchstens n-mal zugegriffen werden.
- Die Gültigkeit eines Lease kann auf ein bestimmtes *räumliches Gebiet* beschränkt sein. Der Nutzer des Lease darf hierbei den Ort des Ausstellers nur dann erfahren, wenn sich Letzterer innerhalb des Bereichs befindet, in dem er sich auch befand, als das Lease erteilt wurde.

Die der anfragebasierten Autorisierung zugrunde liegende Aktivität der Zielperson besteht im Anfragen des Aufenthaltsortes anderer Personen. Die Aktivität zeigt also ganz offensichtlich auf eine andere Person, womit die erste der oben definierten Bedingungen erfüllt ist. Auch der dritten Bedingung, welche besagt, dass die Aktivität aus eigenem Antrieb und unabhängig von der Verwaltung der eigenen Ortsinformation geschehen muss, wird entsprochen. Dadurch, dass das Anfragen anderer Ortsinformationen sicherlich Zeitaufwand bedeutet und auch zu monetären Kosten für mobile Trägerdienste wie GPRS oder UMTS führen kann, wird auch der zweiten Bedingung Rechnung getragen. Wird einem Nutzer der Zugriff verweigert, zum Beispiel wegen eines abgelaufenen Lease, fällt es deshalb relativ leicht zu argumentieren, dass die entsprechenden knappen Ressourcen (Zeit und Geld) die Zielperson davon abhielten, das Lease zu erneuern. Im Gegensatz zur expliziten Autorisierung wird also relativ viel Platz für Mehrdeutigkeit geschaffen.

Abgesehen von reziproken Informationsflüssen setzt der Mechanismus auch gegenseitiges Interesse an den ausgetauschten Informationen voraus, was wünschenswert ist. Die Technik ist sehr einfach anzuwenden und bietet Kontrolle. Vor der Ortung einer anderen Person kann man sich jeweils überlegen, ob sich dies hinsichtlich der im Gegenzug preiszugebenden eigenen Privatsphäre lohnt. Durch gezielte Vermeidung des Ortens gewisser Personen lässt sich ein hoher Korrektheitsgrad erzielen. Gleichzeitig ist es möglich, eine hohe Vollständigkeit zu erreichen, nachdem schließlich jede Person die theoretische Möglichkeit hat, jede andere zu orten. Die Identifikatoren der Personen im System können dabei statisch und öffentlich bekannt sein. Weder zusätzlicher Verwaltungsaufwand noch eine spezialisierte Nutzerschnittstelle ist notwendig.

Ein mögliches Problem ist allerdings, dass initiale Ortungsversuche zwischen zwei Personen per Definition erfolglos bleiben, auch wenn die Zielperson prinzipiell bereit wäre, sich orten zu lassen. Nachdem der Nutzer sich dieses Problems jedoch bewusst ist, lässt sich eine solche Situation einfach auflösen, indem die Zielperson explizit auf die erste Ortung hingewiesen wird, so dass sie, falls gewünscht, die Möglichkeit hat, den Zugriff mittels einer initialen „Gegenortung“ freizuschalten. Um mehr über das Langzeitverhalten des Mechanismus zu erfahren, wurden die im Folgenden präsentierten Untersuchungen durchgeführt.

6.2.4 Evaluierung der anfragebasierten Autorisierung

Die Evaluierung untersucht die anfragebasierte Autorisierung für den Fall, dass sich zwei Personen prinzipiell gegenseitig autorisieren möchten und entsprechend die Ortsinformationen des jeweils anderen anhand einer gewissen Häufigkeitsverteilung anfragen. Ziel ist es,

herauszufinden, welchen Einfluss die Gültigkeitszeiten der Leases sowie die individuellen Zugriffsraten der beteiligten Personen auf den Anteil vom System gewährter Zugriffe haben.

Modell und Szenarien

Die Evaluierung basiert auf einem statistischen Modell, welches in der statistischen Programmiersprache R [135] implementiert wurde. In dem Modell versuchen sich zwei Personen A und B gegenseitig zu orten. Das Zugriffsverhalten beider Personen wird jeweils anhand eines Poisson-Prozesses bestimmt, wobei λ_A bzw. λ_B die erwartete Anzahl von Zugriffen pro Stunde von A bzw. B angeben. Versucht A auf den Aufenthaltsort von B zuzugreifen, so erhält B im Gegenzug für δ_t Stunden das Recht, A zu orten. δ_t entspricht also der Gültigkeitsdauer eines Lease. Die zeitliche Auflösung der Simulationen beträgt eine Stunde pro Simulationsschritt. Für jeden Simulationslauf wurden jeweils eine Million Schritte ausgeführt.

Den Experimenten werden zwei verschiedene Szenarien, die jeweils einem anderen Verhältnis der Zugriffsraten der Personen entsprechen, zugrunde gelegt. Beim *1:1-Szenario* haben beide Personen dieselbe Zugriffshäufigkeit auf den Ort des jeweils anderen. Im *3:1-Szenario* ortet hingegen eine Person die andere dreimal so häufig wie umgekehrt. Beide Szenarien könnten im Rahmen eines typischen Friend-Finder-Dienstes vorkommen. Im ersten Fall orten sich zwei Freunde mit ausgeglichenem Interesse am Aufenthaltsort des jeweils anderen. Im zweiten Fall hat eine Person größeres Interesse am Aufenthaltsort der anderen als umgekehrt. Die im Folgenden beschriebenen Experimente beziehen sich jeweils auf beide Szenarien. Es wird immer ein Parameter (Zugriffsrate oder Lease-Dauer) konstant gehalten und der andere verändert und im Anschluss der Anteil gewährter Zugriffe beobachtet.

Auswirkungen der Lease-Dauer

In der ersten Versuchsreihe wurde die Zugriffsrate konstant gehalten (drei Zugriffe pro Tag im 1:1-Szenario, 4,5 bzw. 1,5 pro Tag im 3:1-Szenario), während die Lease-Dauer zwischen einer Stunde und drei Tagen verändert wurde.

Beim 1:1-Szenario (siehe Abbildung 6.8) sind die Auswirkungen der Veränderung der Lease-Dauer auf den Anteil gewährter Zugriffe natürlich für beide Personen gleich. Ist die Lease-Dauer hier größer als sechs Stunden, so wird etwa die Hälfte aller Zugriffe gewährt. Liegt sie hingegen zwischen 24 und 40 Stunden, so werden zwischen 95% und 99% aller Zugriffe erlaubt. Für dieses Intervall ergibt sich also bereits eine recht hohe Zuverlässigkeit im Falle einer gewünschten Ortung.

Im 3:1-Szenario (siehe Abbildung 6.9) unterscheidet sich der Anteil gewährter Zugriffe bei den beiden Personen. Die erste Person, deren Zugriffsrate mit einem Erwartungswert von 4,5 pro Tag angesetzt wird und deren Ortsinformation von der anderen Person im Mittel 1,5-mal pro Tag angefragt wird, benötigt eine Lease-Dauer von mindestens 11 Stunden, um in 50% aller Fälle Erfolg zu haben. Zwischen 49 und 72 Stunden sind hingegen notwendig, um zwischen 95% to 99% aller Abfragen positiv beantwortet zu bekommen. Die zweite Person, die 1,5-mal pro Tag anfragt und 4,5-mal angefragt wird, erreicht die 50-prozentige Erfolgsrate bereits bei einer Lease-Dauer von vier Stunden. Sollen hier zwischen 95% to 99% aller Abfragen erfolgreich sein, so ist eine Dauer von 16 bis 25 Stunden notwendig.

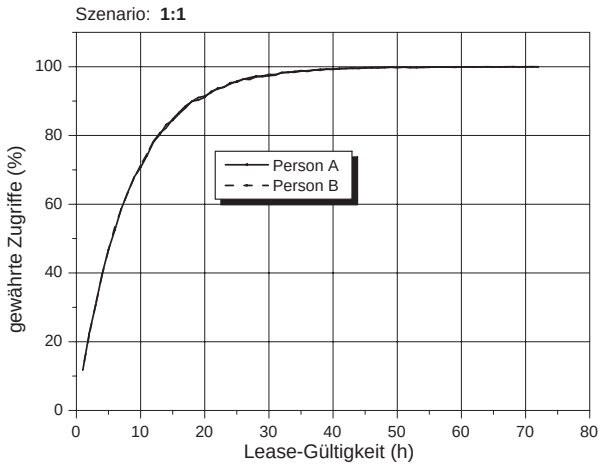

Abbildung 6.8: Anteil gewährter Zugriffe bei gleicher Rate von zwei Zugriffen pro Tag

Auswirkungen der Zugriffsrate

In der zweiten Versuchsreihe wurde die Lease-Dauer konstant bei 48 Stunden gehalten, die akkumulierte Zugriffsrate beider Personen wurde zwischen zweimal wöchentlich und 12-mal täglich variiert. Den beiden Szenarien entsprechend wurden also die folgenden Zugriffsraten verwendet. Beim 1:1-Szenario wurde die Zugriffszahl gleich verteilt, jede Person greift hier zwischen einmal wöchentlich und sechsmal täglich auf die Ortsinformation der anderen zu. Entsprechend hat im 3:1-Szenario die eine Person eine Zugriffsrate von 1,5-mal pro Woche bis zu neunmal pro Tag und die andere von 0,5-mal pro Woche bis zu dreimal pro Tag.

Im 1:1-Szenario (siehe Abbildung 6.10) ist das Verhältnis der Erfolgsraten der beiden Personen wieder ausgeglichen. Es zeigt sich, dass eine Zugriffsrate von 5,4-mal pro Woche ausreichend ist, um in mindestens der Hälfte aller Fälle Erfolg zu haben. Zwischen 22,4 und 38,5 Zugriffe pro Woche sind hingegen notwendig für eine Erfolgsrate zwischen 95% und 99%.

Im *3:1-Szenario* (siehe Abbildung 6.11) unterscheidet sich wiederum die Erfolgsrate der beiden Personen. Für die eine Person ist sie größer als 50%, wenn öfter als 3,1-mal pro Woche zugegriffen wird. Für die andere Person ist dafür lediglich ein Zugriff pro Woche notwendig. Für beide Teilnehmer gleichzeitig ist die Erfolgsrate größer als 50%, wenn der eine mindestens siebenmal pro Woche zugreift und der andere entsprechend dem 3:1-Verhältnis 2,4-mal pro Woche. Eine Bewilligungsrate zwischen 95% und 99% ergibt sich für die eine Person, wenn ihre Zugriffsrate zwischen 10,5-mal und 18,3-mal pro Woche liegt. Für die andere beträgt das Intervall 3,5-mal bis 6,16-mal pro Woche. Beide Erfolgsraten liegen zwischen 95% und 99%, wenn die eine Person zwischen 45,5-mal und 84-mal pro Woche zugreift, was für die andere dann bedeutet, dass sie zwischen 11,4 und 21 Aufrufe pro Woche tätigen müsste.

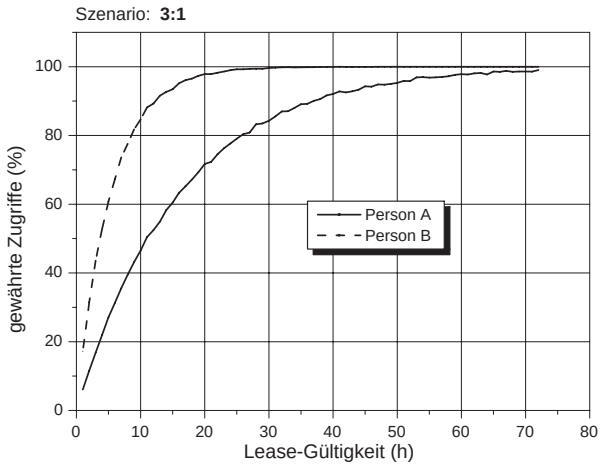

Abbildung 6.9: Anteil gewährter Zugriffe bei einer Rate von 4,5 bzw. 1,5 Zugriffen pro Tag

Zusammenfassung der Evaluierung

Zusammenfassend lässt sich sagen, dass anfragebasierte Autorisierung bei tatsächlich gewünschtem Zugriff diesen auch zuverlässig erlaubt, wenn zwei Person sich relativ häufig orten und wenn das gegenseitige Interesse an Ortsinformation ausgeglichen ist. Andernfalls werden Zugriffsversuche oft abgewiesen, zumindest bei der häufiger ortenden Person. Wie vermutet sind also Anwendungen, die einseitiges Tracking erfordern, wie Child-Tracker- oder Notfalldienste, mit dem Mechanismus wohl nicht kompatibel. Die Technik scheint jedoch sinnvoll, wenn ein ausgeglichenes Interesse am Aufenthaltsort des anderen besteht, eine Autorisierung tatsächlich erwünscht ist und wenn der LBCS häufig genutzt wird, wie es zum Beispiel für einen typischen Friend-Finder-Dienst vorstellbar ist. Die Wahl der Lease-Dauer gestaltet sich als Trade-off. Lange Zeiträume erhöhen die Zuverlässigkeit im Fall einer gewünschten Ortung, während eine kürzere Dauer die Kontrolle in Bezug auf Korrektheit verbessert sowie die Möglichkeiten von Mehrdeutigkeit erhöht, falls eine Zielperson es vorzieht, von einer bestimmten Person nicht mehr geortet zu werden. Anhand der Simulationen scheinen Lease-Dauern von 24 bis 48 Stunden geeignet.

6.3 Zusammenfassung

Diese Kapitel hat sich mit zwei grundlegenden Privacy-Herausforderung bei LBCSs beschäftigt. Die eine Problemstellung betraf die Geheimhaltung der Identitäten von Zielpersonen vor Intermediären wie dem Location Provider und dem LBS Provider. Bei Einzelpersonen- sowie reaktiven LBCSs kann diese relativ einfach realisiert werden. Eine Möglichkeit ist, dass die Ortsinformationen von der Zielperson verschlüsselt und vom Location bzw. LBS Provider lediglich an den Nutzer durchgereicht werden. Sie werden dann erst in der Domäne des Nutzers entschlüsselt und mit statischen Inhalten wie Karten korreliert, ein Ansatz der dem Paradigma der so genannten Web 2.0 Mash-ups entspricht.

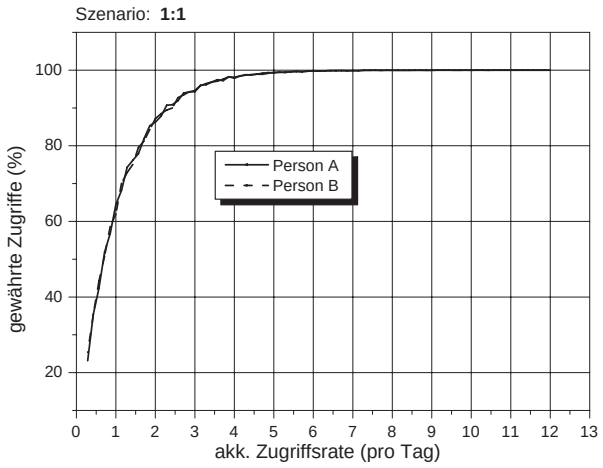

Abbildung 6.10: Anteil gewährter Zugriffe bei gleicher Zugriffsrate und einer Lease-Gültigkeit von 48 h

Bei proaktiven Mehrpersonen-LBCSs müssen die räumlichen Konstellationen mehrerer Zielpersonen jedoch vom Location bzw. LBS Provider ermittelt werden, um ein effizientes Tracking zu ermöglichen. Die Ortsinformationen müssen vor diesen Intermediären daher in einer gewissen Weise offengelegt werden, was wiederum eine geeignete Anonymisierung der Daten erfordert. Die Bereitstellung von Anonymisierungstechniken für proaktive Mehrpersonen-LBCSs stellte ein bislang ungelöstes Problem dar, dessen Schwierigkeit vor allem auf die Notwendigkeit konsistenter Identifikatoren für Zielpersonen zurückzuführen ist. Es wurde daher ein auf die Nahbereichs- und Trennungserkennung maßgeschneiderter Mechanismus vorgestellt, der zwar nicht perfekt aber im Gegensatz zu existierenden Ansätzen tatsächlich anwendbar ist.

Grundsätzlich sind Anonymisierungstechniken relativ aufwändig. Ein praktikablerer Ansatz, der auch der im nächsten Kapitel vorgestellten TraX-Plattform zugrunde liegt, ist die Benutzung des Location Providers als so genannter Mittler [99]. Er dient dabei als einziger vertrauenswürdiger Zugangspunkt für die Zielperson und nimmt die Berechnung räumlicher Konstellationen mehrerer Zielpersonen im Auftrag verschiedener LBS Provider vor.

Das zweite Thema dieses Kapitels war die Beziehung zwischen Nutzer und Zielperson. Hierzu wurden entsprechende Problemstellungen (soziale Verträglichkeit, Verwaltungsaufwand) motiviert. Als eine mögliche Lösung wurde das neuartige Konzept der impliziten Autorisierung entwickelt. Es reduziert den Verwaltungsaufwand bei der Zielperson und gewährleistet gleichzeitig eine höhere soziale Akzeptanz von Zugriffsverweigerungen. Zwei konkrete Ausprägung des Konzepts wurden vorgestellt, wobei die so genannte anfragebasierte Autorisierung näher untersucht wurde. Sie basiert sehr stark auf dem Prinzip der Reziprozität und eignet sich deshalb vor allem für LBCSs, die ein ausgeglichenes Verhältnis an Ortungsversuchen zwischen den teilnehmenden Personen vorsehen, wie typische Friend-Finder-Dienste.

Ein offener Punkt bezüglich der vorgeschlagenen impliziten Autorisierung ist dessen

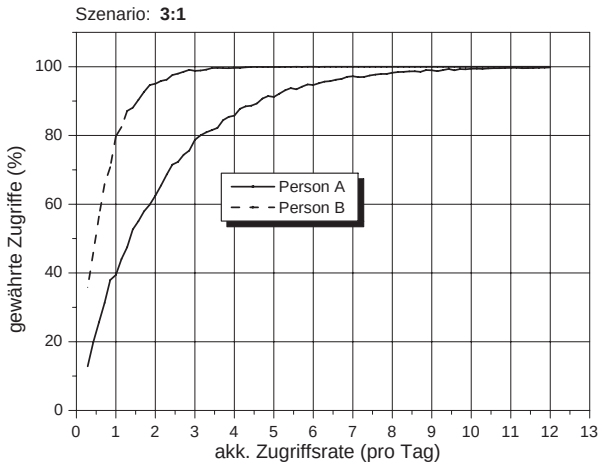

Abbildung 6.11: Anteil gewährter Zugriffe bei einer Zugriffsrate im Verhältnis 3:1 und einer Lease-Gültigkeit von 48 h

juristische Kompatibilität mit bestehenden Regulierungsbeschlüssen wie [20]. Hier wird nämlich vorgesehen, dass Zielpersonen jeweils ihre explizite Einwilligung zum Sammeln persönlicher Daten geben müssen.

Während die durchgeführten Simulationen die grundsätzliche Anwendbarkeit des Mechanismus zeigen, sind weitergehende Nutzerstudien dringend nötig, die jedoch den Rahmen dieser Arbeit sprengen würden. Diese Studien werden so bald wie möglich mit Hilfe der TraX-Plattform durchgeführt, bei deren Entwicklung Privacy-Betrachtungen eine maßgebliche Rolle spielten. Der vorangegangene Abschnitt soll also vor allem als Motivation und Ideengeber sowie zur Darstellung der grundlegenden Problemstellung in dem Bereich dienen. Aus Sicht des Autors ist die Erforschung verwandter Mechanismen dringend notwendig, wenn die für die Zukunft vorhersehbare Verbreitung von LBCSs nicht zu massiven Einschränkungen der Privatsphäre ihrer Benutzer führen soll.

7 Die TraX-Plattform

Die in dieser Arbeit beschriebenen Mechanismen wurden zum großen Teil innerhalb der selbst entwickelten *TraX*-Plattform praktisch umgesetzt und erprobt. TraX steht für *„Tracking and X-change“* und bietet prinzipiell Unterstützung für alle in Kapitel 2 besprochenen Klassen von LBSs. Insbesondere stellt TraX die notwendigen Funktionen zur Realisierung von proaktiven Mehrpersonen-LBCSs bereit, wodurch sich die Plattform von bestehenden LBS-Middleware-Ansätzen abhebt.

Dieses Kapitel gliedert sich folgendermaßen. In Abschnitt 7.1 werden bestehende LBS-Middleware-Ansätze vorgestellt und auf ihre Eignung hinsichtlich proaktiver Mehrpersonen-LBSs untersucht, vergleiche auch [110; 95]. In Abschnitt 7.2 wird dargestellt, wie sich die in Kapitel 2 und 4 vorgestellten Position-Update-Methoden und Strategien zur Nahbereichs- und Trennungserkennung sowie die in Kapitel 5 behandelte Erkennung von Cliquen in die TraX-Architektur eingliedern. Die funktionale Schichtung von TraX erleichtert es außerdem, in dieser Arbeit unbehandelte Mechanismen, die ebenfalls zur Realisierung proaktiver Mehrpersonen-LBCSs benötigt werden, zu identifizieren. Abschnitt 7.2 diskutiert die Verteilung der einzelnen Schichten auf die in Kapitel 2 identifizierten Rollen. Abschnitt 7.3 führt in die konkrete Umsetzung von TraX ein. Der basierend auf J2SE realisierte, zentralisierte TraX-Server verwaltet die Ortsinformationen einer Vielzahl von Zielpersonen, die jeweils eine Instanz des TraX-Clients ausführen. Letzterer ist inzwischen für J2ME, .NET sowie Symbian verfügbar. Außerdem wird die in der Arbeit benutzte Simulationsumgebung vorgestellt, welche auf dem Kern des TraX-Servers aufsetzt. Ferner werden die so genannten TraX-APIs beschrieben, mit deren Hilfe LBS-Anwendungen auf die Funktionalität von TraX zugreifen können. Auch werden zwei konkrete LBS-Anwendungen beschrieben, die auf TraX aufsetzen.

7.1 Verwandte Arbeiten

Ein genereller Überblick über die bei der Realisierung mobiler Dienste auftretenden Probleme und die daraus resultierenden Anforderungen an eine Middleware sowie entsprechende Lösungsmöglichkeiten werden zum Beispiel in [116; 49; 51; 28; 82; 141; 59] dargestellt. Im Folgenden werden ausschließlich Middleware-Ansätze aus der Literatur vorgestellt, die speziell auf LBSs ausgerichtet sind.

MiddleWhere

Der erste bestehende Ansatz, der hier behandelt wird, ist MiddleWhere. MiddleWhere ist eine verteilte LBS-Middleware-Plattform, die unter anderem die folgenden Funktionalitäten anbietet, vergleiche [137]:

- **Integration verschiedener Positionierungsverfahren**. MiddleWhere integriert verschiedene Arten von Positionierungsverfahren, zum Beispiel RFID-Tags, GPS sowie Zugriffskontrollsysteme und Login-Informationen stationärer Rechner. Abgeleitete Positionsdaten werden unter Berücksichtigung der unterschiedlichen Genauigkeiten und Konfidenzen in ein einheitliches Modell eingefügt. Bei widersprüchlichen Positionsdaten aus verschiedenen Quellen findet eine Konfliktauflösung statt. Zur Beschreibung von Orten wird ein Modell verwendet, das sowohl geographische Koordinaten als auch symbolische Bezeichner zulässt.

- **Reaktives und proaktives Interaktionsmodell**. MiddleWhere unterstützt sowohl Ortungsanfragen entsprechend eines reaktiven Dienstmodells (Polling) als auch die Registrierung für bestimmte räumliche Ereignisse (Position Update Request) zur Realisierung proaktiver LBSs.

- **Region- und objektspezifische Anfrageformen**. MiddleWhere erlaubt zwei Anfragetypen: Die objektspezifische Anfrage zielt auf die momentane Position eines bekannten Zielobjekts ab („Wo befindet sich Person X?“), während sich die regionsspezifische Anfrage auf alle Objekte innerhalb einer gegebenen Region bezieht („Welche Personen befinden sich in Raum Y?“).

- **Räumliche Nähe zwischen Objekten**. Zu den räumlichen Beziehungen, die von MiddleWhere erkannt werden, zählt zum Beispiel auch die Nähe zweier mobiler Objekte.

MiddleWhere teilt sich in verschiedene Schichten auf. Die von den verschiedenen Sensoren ermittelten Ortsinformationen werden in einer räumlichen Datenbank (*Spatial DB*) vereint. Diese bietet eine einheitliche Schnittstelle zur Auswertung der Daten, vergleiche Abbildung 7.1.

Die Art der Übermittlung der Ortsinformationen von den Sensoren zur Datenbank ist einstellbar, so dass zum Beispiel Übertragungen nur bei bestimmten Ereignissen bzw. anhand einer definierbaren Periodizität erfolgen. Die *Reasoning Engine* verarbeitet empfangene Rohdaten und schätzt die Position eines Objekts mit einer bestimmten Wahrscheinlichkeit ab, wozu die Sensordaten mit einem räumlichen Modell kombiniert werden. Auch berechnet diese Engine die Beziehungen zwischen mobilen Objekten und Regionen.

In MiddleWhere werden Aufenthaltsorte durch ein hierarchisches Format namens *Gaia Location Byte-String (GLOB)* abgebildet. Ein GLOB kann einen Ort in Form eines Punktes, einer Linie oder eines Polygons sowohl mittels Koordinaten als auch mittels symbolischer Bezeichner repräsentieren.

Der Location Service ist der zentrale Anfragepunkt aller auf MiddleWhere aufsetzenden LBSs. Folgende Funktionalitäten werden bereitgestellt:

- Konsolidierte Sicht auf Ortsdaten unterschiedlicher Quellen

- Auswertung sowohl objekt- als auch regionsbezogener Anfragen

- Benachrichtigung beim Betreten einer Region durch ein Objekt

- Funktionen zur Erkennung von räumlichen Beziehungen

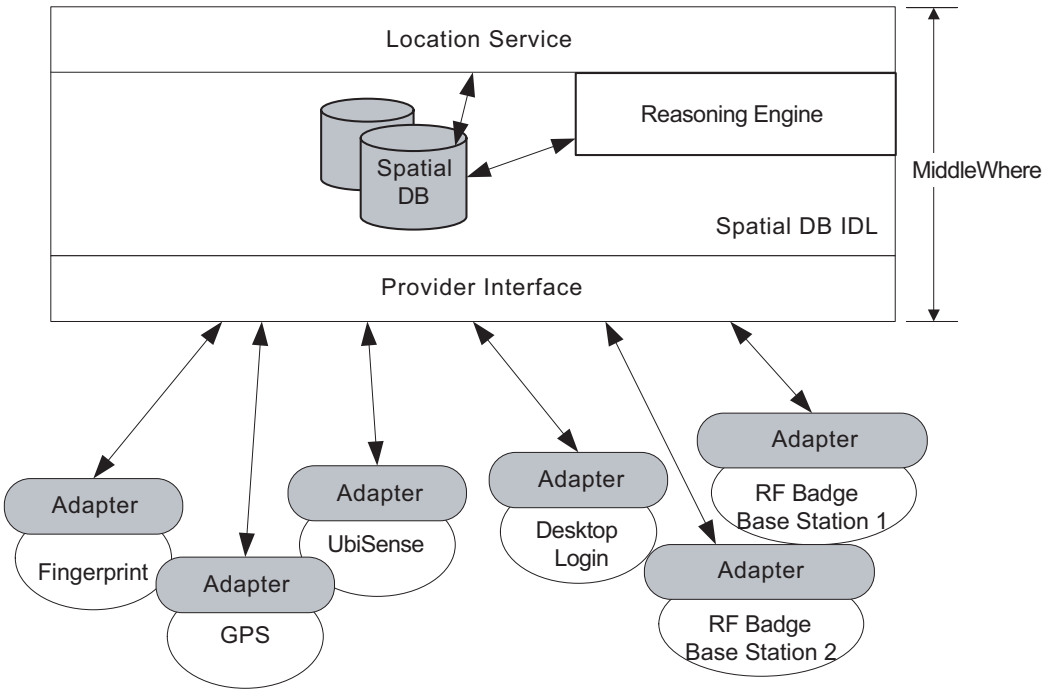

Abbildung 7.1: Architektur der MiddleWhere-Plattform. Quelle: [137]

- Beziehung zwischen zwei Regionen, zum Beispiel Teilmenge oder Überschneidung
- Beziehung zwischen einem Objekt und einer Region, zum Beispiel „Objekt X ist enthalten in Region Y“
- Beziehung zwischen zwei Objekten, zum Beispiel Nähe oder „Co-Location“ (Objekte befinden sich in der gleichen symbolischen Region)

Die räumliche Datenbank basiert auf PostgreSQL [22] mit der PostGIS-Erweiterung [12]. MiddleWhere ist als Erweiterung der Gaia-Plattform [142] implementiert worden. Die einzelnen Module der Plattform kommunizieren mittels der *Common Object Request Broker Architecture (CORBA)* [108] miteinander.

MiddleWhere sieht also bereits proaktive Interaktionsmuster vor und kann auch räumliche Beziehungen zwischen Objekten ermitteln. Die effiziente Realisierung proaktiver Mehrpersonen-LBCSs ist mit MiddleWhere jedoch bislang nicht möglich.

Nexus

Aufgrund des hohen Aufwands, der zum Aufbau eines globalen und gleichzeitig detaillierten Kontextmodells nötig wäre, verwenden die meisten der Arbeiten im Bereich kontextsensitive Anwendungen bislang entweder nur relative grobe Kontextmodelle oder solche, die speziell auf den jeweiligen Dienst zugeschnitten sind. Die Nexus-Plattform hat hingegen zum Ziel, alle möglichen kontextsensitiven Dienste mittels eines offenen, globalen Kontextmodells zu unterstützen [62]. Teile dieses Modells können von unterschiedlichen Dienstanbietern eingebracht und unter dem Dach der Nexus-Plattform zusammengefügt werden, so

dass ein möglichst detailliertes Modell der realen Welt entstehen kann. Das Modell kann dabei auch stark veränderliche Daten enthalten, wie den aktuellen Aufenthaltsort von Personen. Die Daten werden von Sensoren aktualisiert und auf so genannten *Context Servern* abgelegt. Die Daten werden von Sensoren aktualisiert und auf so genannten *Context Servern* abgelegt. Ein Prototyp dieser Plattform sowie darauf aufsetzende Beispielanwendungen werden in [127; 128] beschrieben.

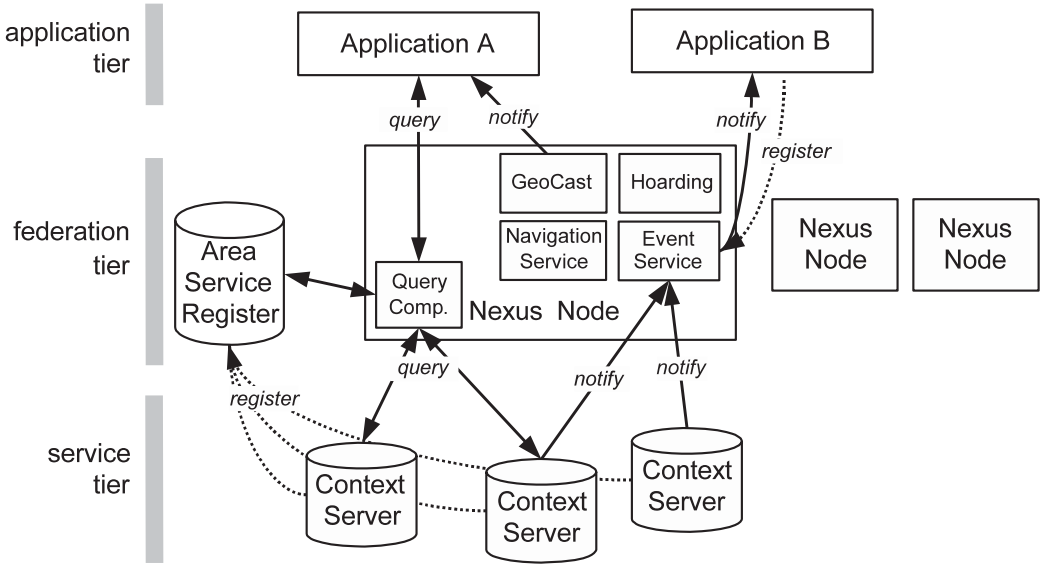

Abbildung 7.2: Architektur der Nexus-Plattform. Quelle: [62]

Abbildung 7.2 zeigt die Architektur der Nexus-Plattform: Ein Context Server beantwortet jeweils Anfragen über ein bestimmtes lokales Kontextmodell, welches beim *Area Service Register (ASR)* für ein bestimmtes räumliches Abdeckungsgebiet registriert ist.

Das ASR stellt eine Art räumliches *Domain Name System (DNS)* dar, da es ein Verzeichnis aller verfügbaren lokalen Kontextmodelle und den dazugehörigen Context Server samt deren Adressen, den Objekttypen und den Abdeckungsgebieten vorhält.

Ein *Nexus Node* hält selbst keine Daten vor, sondern analysiert und verteilt die Anfragen verschiedener kontextsensitiver Anwendungen auf die entsprechenden Context Server. Die einzelnen Rückantworten werden vor der Herausgabe in einem konsistenten Datensatz zusammengefügt. Für das Auffinden der geeigneten Context Server greift der Node auf das ASR zu. Zusätzlich bietet der Nexus Node weitere Dienste an. Der *Event-Dienst* beobachtet räumliche Ereignisse und unterstützt somit proaktive Interaktionsmuster. Der *Navigationsdienst* berechnet Routen über die Grenzen eines lokalen Kontextmodells hinweg. *Geocast* bietet den Versand von Nachrichten an Objekte, die sich momentan in einer bestimmten geographischen Region befinden.

Aus LBS-Anwendungssicht kann die Nexus-Plattform also auf drei verschiedene Arten genutzt werden. Zum einen können reaktive Anfragen (Pollings) über einen Nexus Node an das geeignete Kontextmodell weitergegeben werden. Als Zweites kann sich die Anwendung für bestimmte (räumliche) Ereignisse registrieren und wird bei deren Eintreten benachrichtigt (Position Update Request). Die dritte Option stellen die zusätzlichen Dienste (zum Beispiel Navigation) dar.

Neben einer auf Berechtigungszertifikaten und asymmetrischen Schlüsseln basierenden Zugriffskontrolle unterstützt Nexus auch die Verwendung von Pseudonymen für die preiszugebenden Kontextinformationen der Zielpersonen. Nähere Details zu Sicherheits- und Privacy-Mechanismen in Nexus finden sich in [70].

Neben weiteren wichtigen Aufgaben lassen sich also mit Nexus insbesondere die in Abschnitt 2.4 beschriebenen Position-Update-Methoden realisieren. Die besprochenen höherwertigen Funktionen der effizienten Nahbereichs- und Trennungserkennung sowie der Erkennung von Cliquen sind in Nexus jedoch nicht vorgesehen.

Nimbus

Auch das Nimbus-Rahmenwerk [143; 144] bietet einen einheitlichen Ortungsdienst, der verschiedene dezentrale Ortsmodelle vereint und sowohl symbolische als auch geographische Ortsbeschreibungen zulässt. Reaktive und proaktive Interaktionsmuster sind vorgesehen, wobei Letztere durch so genannte *Trigger* realisiert werden, die ausgelöst werden, wenn eine Zielperson eine bestimmte Zone betritt. Der in [69] beschriebene *Proximity Service* ist ein in Nimbus integrierter Mechanismus, mit dessen Hilfe räumliche Nähe zwischen Objekten effizient erkannt werden kann, indem die Objekte in einem hierarchischen Ortsmodell organisiert werden. Im Gegensatz zur hier beschriebenen Nahbereichserkennung erfolgt eine solche Erkennung jedoch nur reaktiv, und die Effizienz bezieht sich nur auf die reduzierte Rechenlast am Server. Nimbus stellt LBS-Anwendungen eine Palette weiterer hilfreicher Dienste bereit, zum Beispiel Geocasts sowie bestimmte wiederverwendbare Sicherheitsmechanismen.

PoLoS

Die *Platform for Location-Based Services (PoLoS)* [153] enthält als zentrales Element den *PoLoS Kernel*, in dem LBS-Anwendungen ausgeführt werden. Die Anwendungen registrieren sich dazu beim so genannten *Service Deployer* mit Hilfe eines *Service Specification Documents*. Das Dokument durchläuft verschiedene Integritätstests und wird dann dem *Service Invocation Modul* übergeben. Letzteres ist für die Abarbeitung sämtlicher Anfragen an die Anwendung verantwortlich. Zur Kommunikation nach außen unterstützt PoLoS bislang die Protokolle bzw. Protokollfamilien *Short Message Service (SMS)*, WAP und das *Hypertext Transfer Protocol (HTTP)* [78].

Das *Positioning-Modul* bietet dem Kernel eine einheitliche Schnittstelle zur Positionsermittlung und abstrahiert von den technischen Spezifika und Anforderungen der unterschiedlichen Ortungsverfahren. Die gegenwärtige Implementierung unterstützt den Abruf der Ortsinformationen von GMLCs bei 2G und 3G Netzen und die Ansteuerung von GPS-fähigen Endgeräten mittels einer speziellen, auf dem Endgerät installierten Software. Letzteres ist sowohl reaktiv (Polling) als auch proaktiv (Position Update Request) vorgesehen.

Die *GIS-Komponente* ermöglicht die Interaktion zwischen dem Kernel und externen GIS-Datenbanken. Zusätzlich ist sie für die Beschaffung von textuellen oder visuellen Daten und der graphischen Aufbereitung von Informationen (Karten, etc.) verantwortlich. Hierfür implementiert sie Algorithmen zur Routenberechnung und zum Transformieren von Positionsdaten (Geocoding und Reverse Geocoding).

In der *Service-Creation-Umgebung* können LBS-Entwickler mittels einer graphischen

Oberfläche ihre Dienste gestalten und spezifizieren, die im Anschluss daran im PoLoS Kernel installiert werden. Die Dienstlogik wird in einer *Extensible Markup Language (XML)*-basierten Sprache, der *Service Control Language (SCL)*, angegeben [79].

Spatial Publish/Subscribe System for Intelligent Location-Based Services

Bisherige Publish/Subscribe-Middlewaresysteme bieten keine effiziente Unterstützung von räumlichen Ereignissen, da ihnen unter anderem ein räumlicher Index zur Verwaltung von zwei oder drei Dimensionen fehlt. Das in [52] beschriebene System stellt daher eine Erweiterung des klassischen Publish/Subscribe-Paradigmas um räumliche Ereignisse und Filter dar. Der Ansatz ermöglicht es auch, räumliche Abonnements und Filter teilweise auf das mobile Endgerät auszulagern, falls dieses über ausreichend Rechenleistung und über ein endgerätbasiertes Ortungsverfahren verfügt.

Das Abonnement räumlicher Ereignisse entspricht also den in Abschnitt 2.4 beschriebenen Position-Update-Methoden, welche hier als Tupel bestehend aus einem räumlichen Prädikat (*Spatial Predicate*) und einem Diensttyp (*Type of Service, ToS*) modelliert werden. Bislang werden zwei Arten räumlicher Prädikate unterstützt:

- *Within* besitzt folgende Syntax: $(oid_1, oid_2, ..., oid_n)within(zone_1, zone_2, ..., zone_m)$. Das Prädikat ist genau dann wahr, wenn eine der Zielpersonen oid_i sich innerhalb einer der geographischen Zonen $zone_j$ befindet.

- *Distance* besitzt folgende Syntax: $(oid)distance(d, oid_1, oid_2, ..., oid_n)$. Das Prädikat ist genau dann wahr, wenn die Distanz zwischen oid und einem der oid_i kleiner als d ist.

Mit dem Within-Prädikat lassen sich also sowohl zonenbasierte als auch distanzbasierte Position Updates umsetzen. Das Distance-Prädikat formalisiert hingegen Trigger zur Nahbereichserkennung, genau wie sie auch diese Arbeit vorsieht. Wie die Nahbereichserkennung jedoch konkret umgesetzt werden soll, wird in dem Artikel nicht behandelt. Mehr Details zur verwendeten Software-Architektur, die auf dem *Java Messaging Service (JMS)* [6] aufsetzt und zum Teil Module aus dem CAMEL-Projekt [53] wiederverwendet, finden sich in [110].

AMPROS-Plattform

Diese Plattform entstand im Rahmen des Projekts *Adaptive Middleware Platform for Proactive Reconfigurable Systems (AMPROS)* [55]. Der geographische Ort einer Zielperson wird hierbei wie jede andere Kontextinformation behandelt. Die von der Middleware abgedeckten Bereiche umfassen Kontext-, Disconnection- und Gruppen-Management. Zusätzlich werden Funktionen zum Deployment von Software auf mobilen Endgeräten bereitgestellt, was das Verteilen, die Installation, das Aktualisieren und die Konfiguration von Anwendungen umfasst. Zur Kommunikation der verteilten Komponenten im Festnetz wird CORBA benutzt.

Um aktuelle Kontextwerte zu ermitteln, greift jeweils ein *Resource Monitor* pro Ressourcentyp auf einen Treiber im System des Endgeräts der Zielperson zu. Kontextsensitive Dienste werden vom System jeweils nur dann benachrichtigt, wenn sich für die spezifizierten Kontextquellen relevante Änderungen ergeben.

Ein besonders positives Merkmal des Ansatzes ist die Erkennung von Verbindungsabbrüchen und entsprechend, die Zwischenspeicherung von Anfragen und die Synchronisation wieder verbundener Endgeräte (*Reconciliation*). Falls das mobile Endgerät mit dem Server verbunden ist, werden mögliche Anfragen also direkt an dieses gesendet. Besteht jedoch keine Verbindung, so werden Nachrichten im Cache bis zur Wiederverbindung vorgehalten.

Trotz der schönen, ereignisgesteuerten Benachrichtigung von Anwendungen wird bei AMPROS zur Übermittlung von Kontextinformationen zwischen Endgerät und Server nur auf ein einfaches periodisches Protokoll zurückgegriffen. LBS-Anwendungen ist es bislang auch nicht möglich, Anforderungen an die Genauigkeit und Periodizität der übermittelten Positionsdaten zu spezifizieren.

OpenGIS Location Services

Das *Open Geospatial Consortium (OGC)* veröffentlichte im Oktober 2002 erstmals eine Spezifikation für eine LBS-Middleware zum Einsatz im Mobilfunkbereich [113], die *OpenGIS Location Services (OpenLS)* [130]. Ziel der OpenLS ist es, die Komplexität von *geographischen Informationssystemen (GIS)* sowie die Gewinnung von Positionsdaten vor dem LBS-Anwendungsentwickler zu verbergen. Die Kernkomponente der Spezifikation stellt der *GeoMobility Server* dar, der in Kooperation mit einem LBS-Anwendungsserver den ortsbezogenen Dienst erbringt. Zum Erhalt der Ortsinformationen der Zielpersonen kommuniziert der GeoMobility Server über MLP mit den GMLCs verschiedener Netzbetreiber (siehe Abbildung 7.3). Die OpenLS basieren also auf der klassischen, netzwerkzentrischen Wertschöpfungskette aus Abschnitt 2.3.

Der GeoMobility Server befindet sich dabei in der Verwaltungsdomäne des LBS Providers und kommuniziert mit dessen Anwendungsserver, der Aufgaben wie Authentifizierung und Accounting übernimmt. Zusätzlich verfügt der GeoMobility Server über eine GIS-Datenbank und kann mit Datenbanken externer Inhalteanbieter verbunden sein.

LBS-Anwendungen unterstützt der GeoMobility Server mit so genannten *Core Services*, die durch XML-Schemata beschrieben sind. Die aktuelle Spezifikation umfasst die folgenden Core Services:

- **Verzeichnisdienst (Directory Service)**. Der Verzeichnisdienst ermöglicht es, nach umgebenden PoIs zu suchen. Als Parameter für die Suche können der Typ des PoI oder Namen von Produkten und Dienstleistungen ähnlich den Verzeichnissen der „Gelben Seiten“ angegeben werden.

- **Gateway Service**. Dieser Dienst agiert als Proxy für den Zugriff auf die GMLCs. Daher sind auch die Parameter und Nachrichten, die an diese Schnittstelle gerichtet werden, denen des MLP sehr ähnlich.

- **Geocoding Service**. Dieser Dienst transformiert eine Beschreibung eines Ortes, wie zum Beispiel eine Adresse, den Namen eines Platzes oder eine Postleitzahl, in eine geographische Koordinatendarstellung. Die Daten werden mittels der *Geography Markup Language (GML)* des OGC [129] repräsentiert.

- **Reverse Geocoding Service**. Der Reverse Geocoding Service ermöglicht die Umwandlung in die entgegengesetzte Richtung, also die Transformation von geographischen Koordinaten in einen beschreibenden Bezeichner, meist eine Adresse.

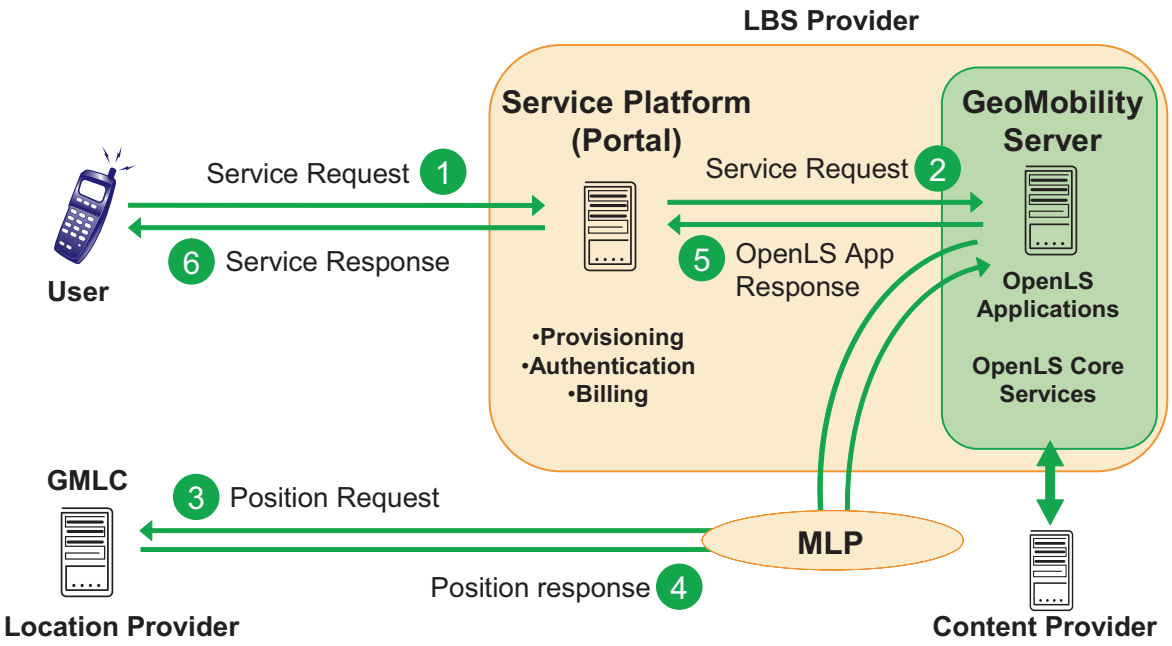

Abbildung 7.3: Kommunikation innerhalb der OpenLS. Quelle: [130]

- **Präsentationsdienst (Presentation Service)**. Der Präsentationsdienst generiert Karten, die auf mobilen Endgeräten dargestellt werden können. Es werden sowohl Karten der realen Infrastruktur, wie Straßen und Häuser (*Basemap Layer*), als auch so genannte *Overlays* generiert, die den momentanen Aufenthaltsort der Zielperson und seine Route einzeichnen.

- **Navigationsdienst (Route Service)**. Dieser Dienst berechnet Verbindungen zwischen zwei oder mehreren gegebenen Punkten. Dadurch lassen sich Routenplaner sehr einfach in bestehende LBS-Anwendungen integrieren. Die Anwendung kann sowohl textuelle Schritt-für-Schritt-Anweisungen als auch eine Karte über den Visualisierungsdienst anfordern.

Abbildung 7.4 zeigt die geschichtete Architektur der OpenGIS Location Services.

Mittlerweile existieren einige kommerzielle Implementierungen, die zumindest Teile der OpenLS-Spezifikation realisieren. Aufgrund des Umfangs, der Komplexität und der relativ starken Ausrichtung auf reine Mobilfunknetze hat sich OpenLS jedoch nicht weitflächig durchgesetzt.

Ganz im Gegensatz dazu sind momentan andere Spezifikationen auf dem Weg der Quasi-Standardisierung. Die Internetsuchmaschinenbetreiber Google und Yahoo! bieten gegenwärtig proprietäre Programmierschnittstellen *(Application Programming Interfaces, APIs)* zu ihren Projekten Google Maps bzw. Yahoo! Maps an [3; 19]. Über diese kann man kostenfrei auf Visualisierungsdienste und Routenplanungsdienste zugreifen, welche dem Präsentationsdienst und dem Navigationsdienst der OpenLS-Spezifikation sehr ähnlich sind. Auch wird inzwischen Geocoding angeboten.

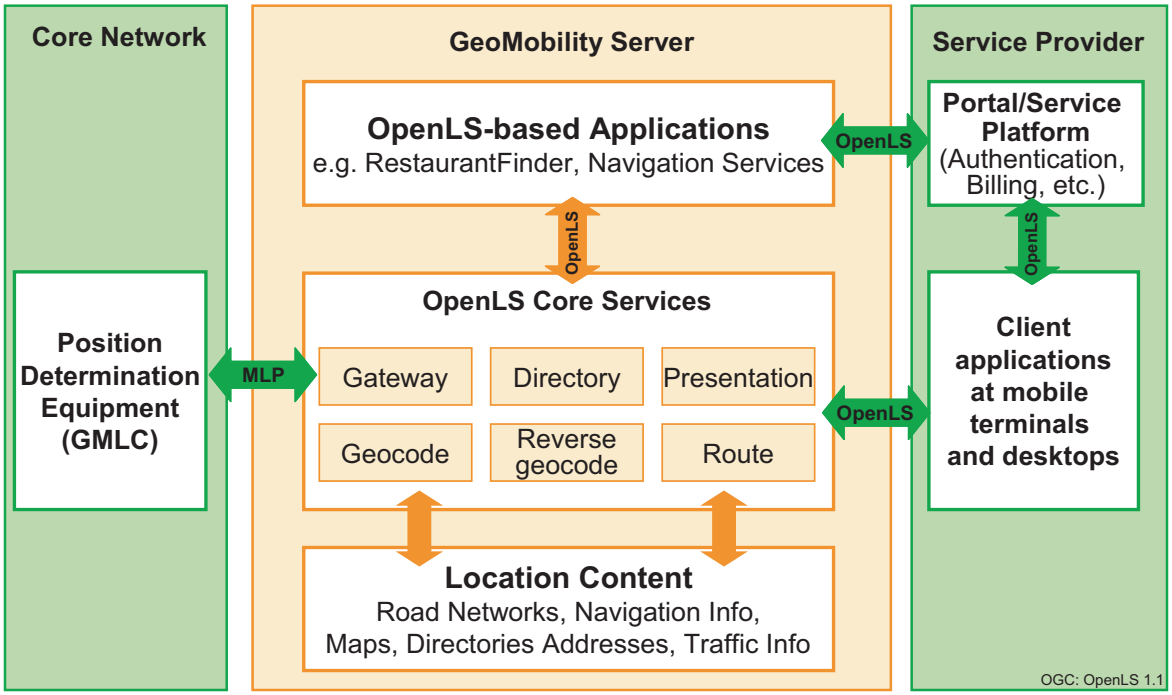

Abbildung 7.4: Architektur der OpenGIS Location Services. Quelle: [130]

MANET- und Peer-to-Peer-Ansätze

Scalable Timed Events And Mobility (STEAM) ist eine ereignisbasierte Middleware für *mobile Ad-hoc-Netzwerke (MANETs)* [118], die einen Peer-to-Peer-Ansatz verfolgt. Ereignisbenachrichtigungen können sowohl beim Publisher als auch beim Subscriber gefiltert werden. STEAM verwendet für diesen Zweck so genannte *Distributed Event Notification Filter*. Soll ein bestimmtes Ereignis veröffentlicht werden, so kündigt der Publisher den Typ des Ereignisses und die geographische Ausdehnung (*Proximity*) an, welche das Ausbreitungsgebiet des Ereignisses bestimmt. Ein Subscriber registriert sich entsprechend für bestimmte Ereignistypen innerhalb einer bestimmten Proximity. Ein mobiles Endgerät kann sich gleichzeitig in mehreren Proximities befinden.

Ein Ereignis besteht in STEAM aus einem Thema (Subject), einem Inhaltstyp und einer Liste von Attributen. Die Ereignisfilter unterstützen die Einschränkung der Weitergabe von Ereignissen anhand des Themas (*Subject Filter*), des Wertes des Inhaltstyps (*Content Filter*) und des Ausbreitungsgebiets (*Proximity Filter*). Mittels des *Proximity-based Group Communication Service* verwaltet STEAM die einzelnen Nutzer einer Anwendung [116]. Diese werden zu *Proximity Groups* zusammengefasst, um das für die Ausbreitung von Ereignissen notwendige 1:n-Kommunikationsmuster anbieten zu können [86]. Anstelle eines Naming-Dienstes zum Auffinden von anderen Entitäten in der Nähe verwendet das System einen *Proximity Discovery Service*, der mittels Funkbaken andere Proximities erkennt. Da in STEAM das geographische Ausbreitungsgebiet eines Ereignisses unabhängig von der Übertragungsreichweite definiert werden kann, wird sowohl eine Single-Hop- als auch eine Multi-Hop-Ausbreitungsstrategie unterstützt.

Im Rahmen des Projekts *COoperating Real-time senTient objects: architecture and EX-*

perimental evaluation (CORTEX) entstand eine Middleware für ortsbezogene Anwendungen in MANETs [156]. Schwerpunkt lag hierbei auf Ad-hoc-Netzen, die eine Fahrzeug-zu-Fahrzeug-Kommunikation ermöglichen. Die Positionsdaten werden von GPS-Empfängern und lokalen Abstandssensoren kontinuierlich an die Middleware weitergegeben. Die Plattform nutzt für die Kommunikation eine Abwandlung des Publish/Subscribe-Modells für MANETs, welche dem Konzept der STEAM-Plattform ähnelt [117]. Dieses wird durch zusätzliche Multicastprotokolle für die Gruppenkommunikation unterstützt.

Es existiert eine Vielzahl weiterer, auf Peer-to-Peer basierender Plattformen, wie zum Beispiel *Proem* [91], *WSAMI* [109] oder AGAPE [42; 43]. Proem ist in J2ME entwickelt worden und bietet neben einer ereignisorientierten Kommunikation und dem Deployment der Anwendungen auch einen so genannten *Community Manager*, mit dessen Hilfe der Anwender die Mitgliedschaft in Gruppen verwalten kann. Die Gruppenzugehörigkeit der Nutzer wird hierbei hauptsächlich zur Darstellung von Vertrauensbeziehungen verwendet. Die AGAPE Middleware basiert auf SOMA, einer in Java implementierten Plattform für mobile Agenten, und nutzt deren Dienste zur Adressierung, Kommunikation, Mobilitätsunterstützung und Absicherung des Systems [35].

Auf Peer-to-Peer ausgerichtete Systeme sind für bestimmte LBS-Anwendungsgebiete sehr interessant, reichen aber aus Sicht des Autors bislang nicht an das technische Potential einer zentralisierten LBS-Architektur heran, wie sie in dieser Arbeit angenommen wird.

Sonstige Middleware-Konzepte

Es besteht eine Vielzahl weiterer verwandter Arbeiten, die Ortsinformationen nicht gesondert, sondern lediglich als eine von vielen möglichen Kontextinformationen betrachten, zum Beispiel [54; 40; 71]. Durch diese Betrachtungsweise sind die Optimierungsmöglichkeiten bei der Positionsgewinnung und -übertragung leider stark eingeschränkt, so dass insbesondere die effiziente Realisierung proaktiver Mehrpersonen-LBCSs schwer fällt.

Auch beschäftigen sich Arbeiten mit dem Problem der Lastbalancierung bei häufig angefragten Datensätzen bei LBSs [169; 170]. *CARISMA* [48] und *ReMMoC* [64] sind Middleware-Konzepte für die Anpassung der LBS-Anwendung an den jeweiligen Kontext des Nutzers. Hierfür verwenden beide Ansätze die Programmiertechnik *Reflection* [47].

Bewertung

Wie dieser Überblick zeigt, existiert bereits eine Reihe von LBS-Middleware-Ansätzen, deren Zielsetzungen sich zum Teil überschneiden und zum Teil ergänzen. Insbesondere sind die folgenden Bereiche von bestehenden Ansätzen bereits relativ gut abgedeckt:

- **Ortungstransparenz.** Mit der Verschattung der speziellen Eigenheiten der benutzten Ortungstechnologie vor der LBS-Anwendung haben sich bereits viele Arbeiten beschäftigt. Die Location API für J2ME [7], die in Abschnitt 7.2.1 noch genauer behandelt wird, stellt sogar einen entsprechenden Standard dar. Auch das Problem des dynamischen *Handovers* zwischen verschiedenen Ortungstechnologien wird in der Literatur aufgegriffen und zumindest ansatzweise gelöst. Ein solcher Handover wird notwendig, wenn das aktuell benutzte System plötzlich nicht mehr verfügbar ist, was bei GPS durch das Betreten eines Gebäudes ausgelöst werden kann.

- **Darstellung und Austausch von Ortsinformationen und räumlich ausgedehnten Objekten.** Hier gibt es eine Fülle von wissenschaftlichen Arbeiten sowie mehrere Standards, zum Beispiel die GML [129], mit der sich räumliche Objekte anhand vorgefertigter XML-Elemente beschreiben lassen. Um Ortsinformationen beim GMLC des Mobilfunkbetreibers abzuholen, wird klassisch MLP [131] verwendet. Geopriv [58] ist ein weiterer Protokollstandard, dessen Wurzeln aus der Internet-Welt stammen und der den Austausch von Ortsinformationen zwischen beliebigen Parteien vorsieht. Auch betont Geopriv den Datenschutz stärker als MLP.

  Zur Domänen-internen Modellierung und Verarbeitung von Ortsinformationen und räumlichen Objekten existieren ebenfalls geeignete Systeme, zum Beispiel in Form von GIS-Erweiterungen für bestehende DBMSs wie PostGIS [12]. Die effiziente Ausführung von Rechenoperationen auf räumlichen Objekten, zum Beispiel zur Berechnung von Enthaltenseinsbeziehungen oder Überschneidungen wird in der Forschung seit langem behandelt.

- **Kartendarstellung und Routenplanung.** Ein verwandtes Thema ist die Zusammenstellung von Karteninformationen und anderer navigationsunterstützender Funktionen, wie zum Beispiel der Routenplanung. Wie oben diskutiert sind hier die standardisierten OpenLS [130] sowie entsprechende proprietäre APIs, wie sie zum Beispiel Google und Yahoo! erfolgreich betreiben, zu nennen.

- **Bereitstellung von IDEs.** Zur Erleichterung der relativ komplexen Entwicklung von LBSs ist auch eine Vielzahl spezialisierter *integrierter Entwicklungsumgebungen (Integrated Development Environments, IDEs)* geschaffen worden, die meist auf existierenden Rahmenwerken wie den *Enterprise Java Beans (EJB)* aufsetzen.

Dank dieser und anderer umfassender Konzeptarbeiten und den resultierenden, immer reifer werdenden Implementierungen lassen sich mittlerweile reaktive Einzelbenutzer-LBSs auf einfache und schnelle Weise entwickeln und bereitstellen. Auch zeigt sich, dass viele Arbeiten schon proaktive Dienstinteraktionsmuster vorsehen und dem LBS-Anwendungsentwickler dafür effiziente Methoden bereitstellen. Diese proaktiven Methoden beschränken sich leider derzeit auf Einzelpersonen-LBSs. Für Mehrpersonen-LBSs stehen entweder nur recht ungenaue oder ineffiziente Funktionen bereit, oder sie funktionieren eben nur reaktiv.

Im Folgenden wird daher die selbst entwickelte TraX-Plattform vorgestellt, welche die in dieser Arbeit entwickelten Konzepte umsetzt und LBS-Anwendungen möglichst einfach zur Verfügung stellt.

7.2 Architektur

Dieser Abschnitt beschreibt die Architektur von TraX, und zwar unabhängig von einer konkreten Software-Realisierung. Diese wird in Abschnitt 7.3 behandelt. Zuerst werden die von TraX zur Verfügung gestellten Funktionen in mehreren Schichten organisiert, die funktional aufeinander aufbauen. Davon verspricht man sich ein sehr hohes Maß an Flexibilität. Anschließend wird erklärt, wie sich diese Schichten auf die in Kapitel 2 definierten Rollen bzw. entsprechende Komponenten verteilen lassen.

7.2.1 Funktionale Schichtung

Abbildung 7.5 zeigt die geschichtete Architektur von TraX. Auf jede der Schichten wird im Folgenden detailliert eingegangen. Zunächst ein kurzer Überblick:

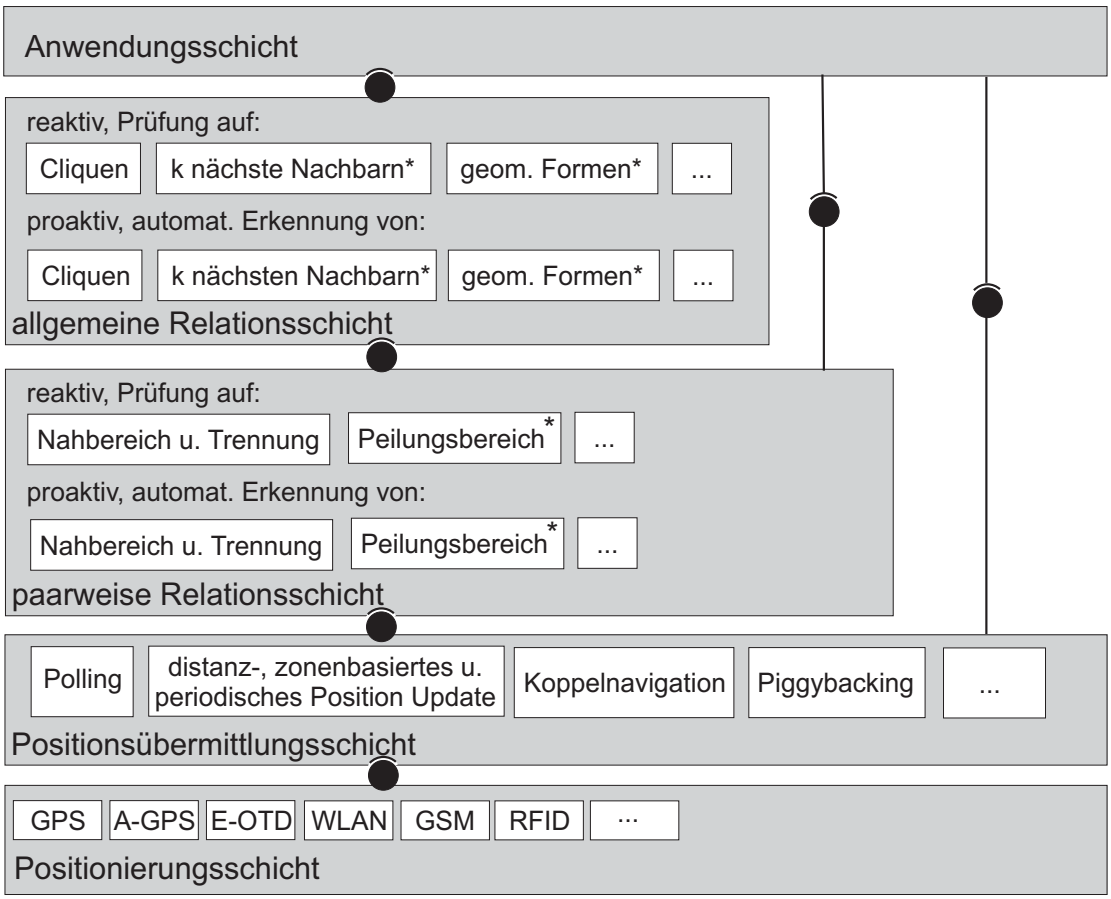

Abbildung 7.5: Funktionale Schichtung von TraX

- **Positionierungsschicht.** Diese Schicht bietet Ortungstransparenz, das heißt, sie verschattet die dynamische Auswahl sowie die technischen Spezifika verschiedener Ortungsverfahren hinter einer einheitlichen, wohldefinierten Schnittstelle.

- **Positionsübermittlungsschicht.** Die Positionsübermittlungsschicht setzt auf der Positionierungsschicht auf und realisiert die in Abschnitt 2.4 vorgestellten Position-Update-Methoden.

- **Paarweise Relationsschicht.** Wie in Kapitel 4 für den Fall der Nahbereichs- und Trennungserkennung erklärt, bedient sich die paarweise Relationsschicht der verschiedenen Position-Update-Methoden der Positionsübermittlungsschicht, um räumliche Ereignisse zwischen Paaren von Zielpersonen festzustellen.

- **Allgemeine Relationsschicht.** Am Beispiel der Erkennung von Cliquen (Kapitel 5) kann man sehen, wie die allgemeine Relationsschicht arbeitet. Unabhängig von der Positionsübermittlungsschicht benutzt sie nur die paarweise Relationsschicht, um räumliche Konstellationen zwischen zwei und mehr Zielpersonen zu detektieren.

- **Anwendungsschicht.** Die Anwendungsschicht bietet LBS-Anwendungen einen einheitlichen Zugriffspunkt zu den darunter liegenden Schichten. Zu ihren Aufgaben zählt die Autorisierung und Authentifizierung, die Bereitstellung von Abrechnungs- und Protokollierungsmechanismen sowie die Verwaltung von Dienstsitzungen.

Positionierungsschicht

Ziel der Positionierungsschicht ist es, die technischen Spezifika der verfügbaren Ortungsverfahren sowie deren dynamische Auswahl vor den höheren Schichten zu verbergen. Zum Erhalt von Ortsinformationen stellt sie einen einheitlichen und einfachen Dienstzugriffspunkt bereit. Die Bestimmung des verwendeten Ortungsverfahrens ist dabei nicht explizit, sondern erfolgt anhand allgemeiner, von der höheren Schicht vorgegebenen Auswahlkriterien. Beispiele für solche Kriterien sind mögliche monetäre Kosten, die bei der Benutzung des Verfahrens entstehen könnten, eine erforderliche Mindestgenauigkeit, ein maximaler, vom Ortungsverfahren erzeugter Batterieverbrauch usw.

Aufgabe der Positionierungsschicht ist es außerdem, ermittelte Positionsdaten in ein vorher vereinbartes Referenzsystem zu wandeln, sofern dies möglich ist. Transformationen zwischen verschiedenen geographischen Referenzsystemen und Kartenprojektionen, zum Beispiel zwischen WGS84 und UTM, sind relativ einfach durchzuführen. Anders verhält es sich bei der Umwandlung zwischen symbolischen und geographischen Ortsrepräsentationen, wie zum Beispiel zwischen Raumnummern innerhalb eines Gebäudes und WGS84-Koordinaten. Hierfür werden weiterführende Konzepte benötigt, deren Behandlung den Rahmen dieser Arbeit sprengen würden. Statt dessen wird auf [144] verwiesen, der sich näher mit dem Thema auseinandersetzt. Die Positionierungsschicht wurde in dieser Arbeit nicht vertieft behandelt, da bereits mehrere geeignete Konzepte und Implementierungen vorliegen. Unter ihnen befinden sich *PlaceLab* und die *Location API for J2ME*.

PlaceLab [147; 100] wird vom *Intel Research Laboratory* in Seattle entwickelt. Der Schwerpunkt liegt bei PlaceLab auf der endgerätbasierten Ortung, und zwar in Abwesenheit von satellitengestützten Verfahren wie GPS oder Galileo. PlaceLab funktioniert sowohl in Innenräumen von Gebäuden, in die GPS-Signale nicht eindringen können, als auch im Freien, falls das Endgerät zum Beispiel nicht mit GPS ausgestattet ist. Die Ortung basiert auf empfangenen Signalen umliegender WLAN Access Points, GSM-Basisstationen oder fest installierten Bluetooth-Sendern. Mittels einer lokal vorgehaltenen Datenbank, welche die Standorte der verschiedenen Sendestationen enthält, schätzt das Endgerät der Zielperson die eigene Position ab. Ein großer Vorteil dieses Verfahrens ist, dass es ohne zusätzliche Positionierungs-Hardware funktioniert. Ein Nachteil von PlaceLab ist jedoch, dass sich bei nur wenigen Einträgen in der Datenbank nur relativ ungenaue Ortsbestimmungen durchführen lassen. Außerdem ist zum Füllen und Pflegen der Datenbank teilweise erheblicher Aufwand nötig, dessen rechtliche Grundlage zudem teilweise unklar ist (für das nicht autorisierte Verzeichnen von Access Points und GSM-Basisstationen hat sich auch der Begriff „War Driving“ etabliert.).

Die *Location API for J2ME* wurde im Rahmen des *Java Community Process* als *Java Specification Request (JSR)* verabschiedet und als Standard zur Steuerung und Verschattung der auf dem mobilen Endgerät verfügbaren Ortungsverfahren erhoben [7]. Alle verfügbaren Ortungsverfahren registrieren sich bei der Location API und spezifizieren dabei eine Liste abstrakter Diensteigenschaften, zum Beispiel hinsichtlich der erreichbaren Genauigkeit oder der bei der Ortung entstehenden Kosten. Die auf dem Endgerät ablaufenden Java-Anwendungen richten wiederum Lokalisierungsanfragen an die API, welche eine oder mehrere solcher zu erfüllender Kriterien enthalten können. Der Abgleich von gewünschten und verfügbaren Eigenschaften der Ortungssysteme geschieht in der Location API. Ein Ortungsverfahren wird ausgewählt, welches der Anwendung gegenüber dann als so genannter *Location Provider* auftritt.

Mittlerweile ist die Location API auch in der Version 2.0 [8] verfügbar. Unter anderem bietet die API nun Funktionen zum Geocoding, zur Kartenerzeugung sowie zur Navigation. Eine weitere mögliche Erweiterung der Location API, die insbesondere im Rahmen dieser Arbeit von Interesse ist, sind die in [133] vorgestellten Friend-Finder-Funktionen, die unter anderem auch einen *FriendProximityListener* enthalten, bei dem man sich zum Empfang von Nahbereichsereignissen registrieren kann. Auf die technische Umsetzung der Funktionen wird in dem Artikel jedoch nicht genau eingegangen.

Positionsübermittlungsschicht

Die Positionsübermittlungsschicht setzt die in Abschnitt 2.4 behandelten Position-Update-Methoden um. Aufgabe der Schicht ist es, den Austausch von Ortsinformationen zwischen mobilen Endgeräten und fest installierten Infrastrukturkomponenten möglichst effizient zu gestalten. Um Ressourcen zu sparen, soll die Übertragung von Ortsinformationen oder der zur Ableitung von Ortsinformationen benötigten Signale über die Luftschnittstelle nur dann erfolgen, wenn die Informationen von der Anwendung auch wirklich benötigt werden. Für reaktive LBSs werden hierfür typischerweise Pollings, eventuell anhand einer bestimmten Caching-Strategie, verwendet. Bei proaktiven LBSs kommt distanzbasiertes- und zonenbasiertes Updating sowie die Koppelnavigation zum Einsatz.

Um die gewünschte Effizienzsteigerung zu gewährleisten, wird die Positionsübermittlungsschicht wie unten beschrieben auf verschiedene Komponenten verteilt, die mittels mobiler Trägerdienste wie GPRS, UMTS oder IEEE 802.11 verbunden sind.

Es muss sich nicht unbedingt um endgerätbasierte Ortungsverfahren wie GPS handeln, die von der Positionsübermittlungsschicht gesteuert werden. In [87] wird zum Beispiel beschrieben, wie anstelle von GPS Location Fingerprinting effizient gesteuert werden kann, welches in [32] erstmalig erforscht wurde. Location Fingerprinting wird dabei im so genannten *endgerätunterstützten* Modus durchgeführt, bei dem das Endgerät die *Received-Signal-Strength-(RSS)*-Werte der umliegenden Access Points empfängt und die Messungen dann an den Location Server berichtet. Dieser leitet den Aufenthaltsort des Endgeräts anhand der berichteten Werte in Kombination mit einer vorgehaltenen Datenbank von Signalstärkemustern, den Fingerprints, ab.

Die Positionsübermittlungsschicht wurde in [87] also für endgerätunterstützte Ortungsverfahren erweitert und ermöglicht nun auch hier die effiziente Umsetzung von distanz- und zonenbasierten Position Update Requests. Abschnitt 8.1 greift dieses noch einmal ausblickend auf. Die Schnittstelle, die die Positionsübermittlungsschicht den höher liegenden

Schichten anbietet, bleibt dabei identisch. Der Umstand, ob ein endgerätbasiertes oder ein endgerätunterstütztes Verfahren zum Einsatz kommt, wird für die höheren Schichten also verschattet. Dies ermöglicht es, bestehende Algorithmen innerhalb der höheren Schichten ohne Anpassungen zu übernehmen und bietet somit ein erhöhtes Maß an Flexibilität.

Paarweise Relationsschicht

Die paarweise Relationsschicht erkennt, ob zwei mobile Zielobjekte in einer vorher definierten, räumlichen Relation zueinander stehen, und zwar sowohl auf reaktive als auch auf proaktive Weise. Als Beispiel für die Feststellung einer solchen räumlichen Relation wurde in Kapitel 4 die Nahbereichs- und Trennungserkennung bezogen auf Luftlinien-Distanzen beschrieben.

Wie bereits erwähnt, lässt sich von den entworfenen Algorithmen insbesondere die DCC-Strategie ohne großen Aufwand auf die Erkennung topologischer Distanzen anwenden, beispielsweise anhand von Straßen- oder Gebäudeplänen. Hier ist die aktuelle Distanz zwischen zwei Objekten definiert als der kürzeste (oder schnellste) Pfad innerhalb des zugrunde liegenden Straßennetzes bzw. der Topologie des Gebäudes, der die beiden Objekte verbindet.

Die Überwachung von Distanzen (topologisch oder entlang der Luftlinie) ist jedoch nicht die einzige räumliche Beziehung, die zwischen zwei mobilen Objekten von Interesse ist. Für viele mögliche Anwendungen, zum Beispiel bei der Navigation, ist die *räumliche Lagerichtung* zweier Objekte von Wichtigkeit. Ein einfaches Beispiel ist die Verfolgung eines mobilen Zielobjekts durch ein anderes. Entspricht die *Fahrt- bzw. Gehrichtung* (engl. *Heading*) des verfolgenden Objekts der so genannten *Peilung* (engl. *Bearing*) der beiden Objekte, so ist es auf dem richtigen Kurs, kann also zur Verfolgung seine Richtung beibehalten. Die Peilung bezeichnet den Winkel zwischen der Verbindung der beiden mobilen Objekte und einer festen Bezugsrichtung, die innerhalb eines bestimmten Bezugssystems wie den Himmelsrichtungen definiert ist.

Während die Fahrtrichtung eines Objekts lokal festgestellt werden kann, erfordert die Überwachung der Peilung zwischen zwei Objekten den kontinuierlichen Austausch und Abgleich der ermittelten Ortsinformationen der Objekte. Analog zur Nahbereichs- und Trennungserkennung sind hierfür effiziente Algorithmen denkbar, die den Gesamtaufwand an Nachrichten reduzieren und zum Beispiel ein entsprechendes Ereignis liefern, sobald sich die aktuelle Peilung der beiden Objekte innerhalb oder außerhalb eines bestimmten Bereichs, dem *Peilungsbereich*, bewegt. Wiederum können Methoden zum Erkennen des Peilungsbereichs sowohl reaktiv als auch proaktiv funktionieren. In der Literatur finden sich bislang noch keine entsprechenden Strategien. Vorstellbar wäre wiederum eine Lösung basierend auf zentrierten Kreisen, ganz ähnlich der DCC-Strategie.

Die Peilung zweier Zielpersonen spielt nicht nur bei der Navigation eine Rolle. Die Erfassung bestimmter geometrischer Formen und Konstellationen eines Verbundes von Zielpersonen könnte zum Beispiel im Katastropheneinsatz bei der Koordination von Rettungskräften sinnvoll sein. Ein anderes, weniger ernsthaftes, wirtschaftlich jedoch hochinteressantes Einsatzgebiet ist Mobile Gaming. Die Art und Weise, wie sich die Teilnehmer eines ortsbezogenen Spiels zueinander ausrichten, könnte zum Beispiel den Spielfluss beeinflussen. Mit Hilfe geeigneter Mechanismen zur Erkennung solcher Konstellationen ließe sich eine Reihe bestehender Brett- und Strategiespiele auf das mobile Umfeld übertragen. Dabei sind dann

möglicherweise nicht nur die räumlichen Beziehungen eines Paares von Zielpersonen von Interesse, sondern die einer größeren Gruppe von Personen, womit sich die im Folgenden behandelte allgemeine Relationsschicht auseinandersetzt.

Allgemeine Relationsschicht

In Kapitel 5 wurde am Beispiel der Erkennung von Cliquen ersichtlich, wie die allgemeine Relationsschicht arbeitet. Sie wendet die Funktionen der paarweisen Relationsschicht dynamisch an, um räumliche Zusammenhänge zwischen zwei und mehr Zielpersonen festzustellen.

Dies gilt sowohl für den reaktiven als auch für den proaktiven Fall, wobei der reaktive meist einfacher zu lösen ist. Soll zum Beispiel reaktiv erkannt werden, ob eine Menge von drei Zielpersonen $S = \{t_1, t_2, t_3\}$ eine Dreier-Clique bildet, so ergibt sich basierend auf der Cliquen-Definition die entsprechende Aussage $Clique(t_1, t_2, t_3, d_c)$ durch Konjunktion der paarweisen Nahbereichs-Tests $Nah(t_i, t_j, d_c)$ als folgt:

$$Clique(t_1, t_2, t_3, d_c) \equiv Nah(t_1, t_2, d_c) \wedge Nah(t_1, t_3, d_c) \wedge Nah(t_2, t_3, d_c) \qquad (7.1)$$

Die Auswertung des Ausdrucks kann angehalten werden, sobald einer der Nahbereichs-Tests $falsch$ zurückliefert oder wenn sich alle als $wahr$ herausstellen. Dieses Vorgehen bringt zum Teil erhebliche Einsparungen bezüglich ausgetauschter Nachrichten im Vergleich zu einem naiven Ansatz, der erst die Positionen aller beteiligter Zielpersonen feststellt und dann das Zutreffen einer bestimmten räumlichen Beziehung entscheidet.

Wie bei der Erkennung von Cliquen gezeigt wurde, bringt eine geschickte Kombination paarweiser räumlicher Bedingungen auch für den proaktiven Fall hohe Einsparungen. Als Anregung für zukünftige Arbeiten wird im Folgenden grob skizziert, wie zwei weitere räumliche Ereignisse zwischen mehreren Personen proaktiv mit Hilfe der paarweisen Relationsschicht erkannt werden könnten.

Das erste Ereignis ist die Formation einer Reihe von Zielpersonen $S = \{t_1, ..., t_s\}$. Eine solche Personenreihe wird bei Rettungsmaßnahmen notwendig, beispielsweise bei der Bergung von Lawinenopfern. Auch bei speziellen ortsbezogenen Spielen sind Reihen als Spielelement vorstellbar. Die Bildung einer Reihe bedingt, dass alle möglichen Paare aus S zueinander dieselbe Peilung aufweisen. Umgekehrt bedeutet dies, dass keine Reihe existieren kann, solange zwei beliebige Paare von Objekten aus S nicht dieselbe Peilung zueinander aufweisen. Es wird also erst die Peilung α eines beliebigen Paares (t_i, t_j) aus S ermittelt und dann überwacht, ob das Paar den Peilungsbereich α verlässt. Gleichzeitig wird ein weiteres Paar (t_i, t_k) daraufhin geprüft, ob es den Peilungsbereich α zueinander einnimmt. Solange beides nicht der Fall ist, kann eine Reihe ausgeschlossen werden, ohne die Positionen weiterer Zielpersonen zu betrachten.

$$(Peilung(t_i, t_j) = \alpha) \wedge (Peilung(t_i, t_k) \neq \alpha) \Rightarrow \neg Reihe(t_1, ..., t_s) \qquad (7.2)$$

Erst wenn die paarweise Relationsschicht zum Beispiel meldet, das $Peilung(t_i, t_k) \neq \alpha$ nicht mehr zutrifft, dass also (t_i, t_k) nun auch die Peilung α zueinander eingenommen haben, könnte ein weiteres Paar (t_i, t_l) bezüglich des Enthaltenseins im Peilungsbereich α

untersucht werden. Wie bei der proaktiven Erkennung von Cliquen entsteht ein ereignisgesteuerter Algorithmus, der hier allerdings nicht ausgeführt wird.

Das zweite Ereignis, welches proaktiv überwacht werden kann, ist eine Änderung in der Zusammensetzung der k nächsten Nachbarn einer bestimmten Zielperson. In [120] wurde ein entsprechendes Verfahren bereits behandelt, jedoch ohne die in dieser Arbeit besprochene Schichtung zu berücksichtigen und ohne auf bestehende Algorithmen zur Nahbereichs- und Trennungserkennung aufzusetzen, wodurch der Ansatz komplexer wird als eigentlich nötig. Der Artikel geht außerdem von der Anwendung spezieller Broadcast-Strategien zur initialen Ermittlung der k nächsten Nachbarn aus, die innerhalb heutiger zellulärer Mobilfunknetze technisch nicht möglich sind.

Der vorgeschlagene proaktive Überwachungsalgorithmus geht davon aus, dass die geordneten k nächsten Nachbarn $(t_1, ..., t_k)$ einer bestimmten Zielperson t_i zu einem bestimmten Zeitpunkt bekannt waren und dass dann auch die aktuellen Distanzen $(d_1, ..., d_k)$ zu t_i ermittelt wurden. Ferner sei d_x die Distanz des zu dem Zeitpunkt zu t_i am $k+1$-nächsten gelegenen Objekts.

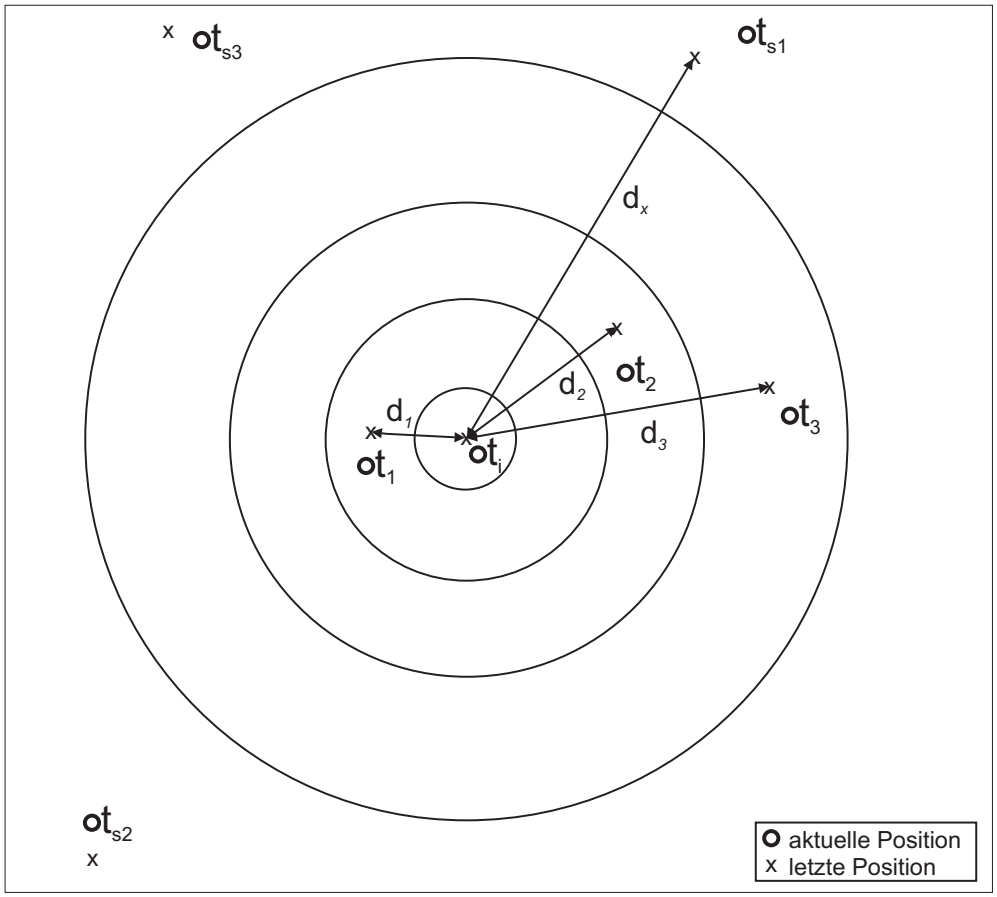

Abbildung 7.6: Überwachung der drei nächsten Nachbarn von t_i

Soll sich die Zusammensetzung und Ordnung der k nächsten Nachbarn nicht unbemerkt ändern können, so übergibt die allgemeine Relationsschicht der paarweisen Relationsschicht

die folgenden Klauseln zur Prüfung:

$$Nah(t_i, t_1, \frac{d_1 + d_2}{2})$$
$$Fern(t_i, t_2, \frac{d_1 + d_2}{2})$$
$$Nah(t_i, t_2, \frac{d_2 + d_3}{2})$$
$$...$$
$$Fern(t_i, t_k, \frac{d_{k-1} + d_k}{2})$$
$$Nah(t_i, t_k, \frac{d_k + d_x}{2})$$

Für alle anderen Zielobjekte t_{si} in S, die zum Messzeitpunkt nicht unter den k nächsten Nachbarn von t_i waren, muss außerdem geprüft werden, ob gilt:

$$Fern(t_i, t_{si}, \frac{d_k + d_x}{2})$$

Erst wenn eine der Klauseln nicht mehr gilt, werden aus Sicht der allgemeinen Relationsschicht weitere Berechnungsschritte nötig. Intuitiv lässt sich der Ansatz anhand von Abbildung 7.6 verstehen, die den Fall $k = 3$ darstellt. Die an die paarweise Relationsschicht übergebenen Klauseln stellen sicher, dass keine Zielperson einen der virtuellen Kreisränder übertritt, wodurch wiederum gewährleistet wird, dass die drei nächsten Nachbarn von t_i gleich bleiben.

Im Gegensatz zu dem Artikel [120], der die dargestellten Kreise direkt als Position Update Requests an die überwachten Zielpersonen verteilt, werden durch die vorgestellte Schichtung von TraX neue Möglichkeiten eröffnet. Nachdem an die paarweise Relationsschicht nur obige Klauseln übergeben werden, kann diese die zur Nahbereichs- und Trennungserkennung für die aktuelle Situation am besten passende Strategie auswählen. Eine zweite Möglichkeit ist die Überwachung der k nächsten Nachbarn in Bezug auf topologische Distanzen. Die Klauseln müssten hierfür nicht geändert werden. Die konkrete Erforschung eines ereignisgesteuerten Algorithmus zur dynamischen Definition der Klauseln sprengt leider den Rahmen dieser Arbeit.

Zusätzlich zu den genannten Beispielen lassen sich mit Hilfe der allgemeinen Relationsschicht prinzipiell alle möglichen geometrischen Formen zwischen verfolgten Zielpersonen überwachen, und zwar reaktiv und proaktiv. Ein Beispiel ist die Formation von gleichseitigen Dreiecken oder von Quadraten. Die entsprechenden Anforderungen zukünftiger LBSs sind im Moment jedoch schwer abzuschätzen, weshalb die Erforschung geeigneter Algorithmen nicht Bestandteil dieser Arbeit ist. Stattdessen versteht sich dieser Abschnitt hauptsächlich als Ideengeber.

Anwendungsschicht

Die Anwendungsschicht bietet LBS-Anwendungen Zugriff auf die Funktionen der Positionsübermittlungsschicht sowie der paarweisen und der allgemeinen Relationsschicht. Besonderes Ziel ist die Bereitstellung einer möglichst einfachen und flexibel einsetzbaren

Schnittstelle. Diese soll die Benutzung eines LBS sowohl von mobilen als auch von stationären Endgeräten aus ermöglichen. Während selbstverweisende LBSs natürlich nur sinnvoll sind, wenn der Nutzer selbst mobil ist, so hat die stationäre Nutzung für querverweisende LBSs durchaus Berechtigung. Ein Beispiel sind Child-Tracker-Dienste, mit deren Hilfe Erziehungsberechtigte den aktuellen Aufenthaltsort eines Kindes abrufen können.

Wie an diesem Beispiel deutlich wird, ist die Autorisierung und Authentifizierung von Ortungsanfragen eine wichtige Funktion der Anwendungsschicht. Zum einen müssen entsprechende Verschlüsselungstechniken bereitgestellt werden. Zum anderen wird eine Schnittstelle erforderlich, mit deren Hilfe Zielpersonen ihre Privacy-Einstellungen verwalten können. Dies kann klassisch durch explizite Autorisierung geschehen, also mittels Datenschutz-Policies oder Ad-hoc-Autorisierung. Alternativ kann die in Abschnitt 6.2.3 vorgestellte implizite Autorisierung angewendet werden.

Im Zusammenhang mit Datenschutz steht die Zugriffsprotokollierung, eine weitere zentrale Aufgabe der Anwendungsschicht. Für die Zielperson soll es im Nachhinein nachvollziehbar sein, wer versucht hat, auf ihren Aufenthaltsort zuzugreifen und wem der Zugriff gestattet wurde. Protokollierungsfunktionen sind gleichzeitig für andere Bereiche von sehr hoher Wichtigkeit. Ein Beispiel ist die Suche und Behebung von Fehlern im System. Ein anderes umfangreiches Thema ist die Kundenabrechnung bzw. das Accounting.

Die größte technische Herausforderung auf Ebene der TraX-Anwendungsschicht ist die Verwaltung von proaktiven Dienstsitzungen. Mit Hilfe der TraX-APIs (vergleiche unten) registriert sich der Nutzer bzw. der LBS Provider beim TraX-Server für bestimmte räumliche Ereignisse, die beim Eintreten asynchron zugestellt werden. Die Registrierung für die Ereignisse und der Erhalt der proaktiven Benachrichtigungen sollen aus Sicht des LBS-Anwendungsentwicklers möglichst einfach umzusetzen und mit bestehenden Anwendungen kombinierbar sein. Dabei treten verschiedene technische Schwierigkeiten auf, zum Beispiel die Berücksichtigung bestehender Sicherheitsvorkehrungen wie Firewalls beim Nutzer bzw. beim LBS Provider. Eine weitere Herausforderung ist die effiziente und rechtzeitige Erkennung von Verbindungsabbrüchen und Ausfällen der für die Benachrichtigung registrierten Komponente. In diesem Fall müssen proaktive Positionierungsanfragen abgebrochen werden, um TraX nicht unnötig zu belasten.

7.2.2 Verteilung auf Rollen

TraX unterscheidet sich von den meisten bestehenden LBS-Middleware-Ansätzen dadurch, dass TraX auf mehrere Rollen entlang der in Kapitel 2 vorgestellten LBS-Wertschöpfungskette verteilt ist. Es werden also nicht nur einzelne, von einer LBS-Rolle bereitgestellte Funktionen berücksichtigt, wie es zum Beispiel bei der J2ME Location API oder den Open-LS der Fall ist.

Wie Abbildung 7.7 zeigt, wird die indirekte endgerätzentrische Wertschöpfungskette für TraX zugrunde gelegt. Die TraX-Middleware teilt sich in drei grundlegende Komponenten auf: den TraX-Client, der auf dem mobilen Endgerät der Zielperson installiert ist, den zentralen TraX-Server, welcher die Ortsinformationen einer Vielzahl von Zielpersonen aufbereitet und miteinander in Beziehung setzt, sowie die TraX-APIs. Die Trax-APIs sind für unterschiedliche Nutzungsszenarien konzipierte Software-Bibliotheken, die ortsbasierten Anwendungen die Funktionen von TraX zur Verfügung stellen. Abbildung 7.8 zeigt, wie die zuvor besprochenen Schichten auf die drei TraX-Komponenten verteilt werden.

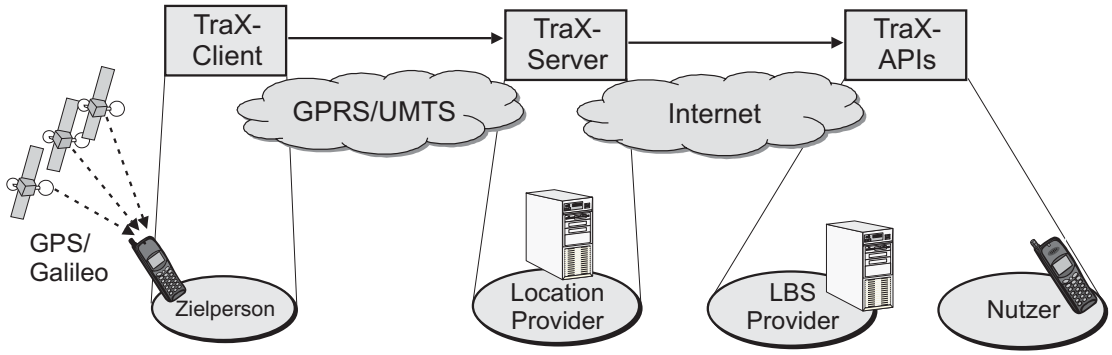

Abbildung 7.7: Verteilung der TraX-Komponenten entlang der LBS-Wertschöpfungskette

- **TraX-Client.** Der TraX-Client ist ein Software-Daemon, der auf dem mobilen Endgerät der Zielperson abläuft und die dort verfügbaren Positionierungstechnologien kontrolliert. Der Daemon steht mit dem TraX-Server, der ihn entsprechend den Anwendungsanforderungen mit Position-Update-Methoden konfiguriert, in ständiger Verbindung, vergleiche auch Abbildung 2.5. Hierfür wird ein proprietäres Protokoll verwendet, welches auf dem *Transmission Control Protocol (TCP)* aufsetzt. Ziel des Protokolls ist es, den Austausch von Ortsinformationen besonders effizient zu gestalten. Als mobile Trägerdienste kommen zum Beispiel GPRS, UMTS oder IEEE 802.11 in Frage. Alternativ zu TCP ist auch SMS als Trägerdienst vorstellbar.

  Im Falle von rein endgerätbasierter Positionierung (wie zuvor beschrieben, sind auch endgerätunterstützte Verfahren denkbar) realisiert der TraX-Client also selbständig die Funktionen der Positionierungsschicht. Die Positionsübermittlungsschicht ist zwischen TraX-Client und TraX-Server aufgeteilt.

  Die sichere Kommunikation sowie die Authentifizierung werden ebenfalls gewährleistet. Auch werden der Zielperson rudimentäre Mechanismen zur Privacy-Kontrolle bereitgestellt. Insbesondere soll die Zielperson sicherstellen können, dass der Client an- bzw. ausgeschaltet ist. Für erweiterte Privacy-Einstellungen ist ein vom TraX-Client unabhängiges Werkzeug erforderlich, welches per TraX-API an den TraX-Server angebunden ist. Ziel dieser Aufteilung ist es, den TraX-Client möglichst leichtgewichtig zu halten, so dass er rasch auf neue Endgeräte-Plattformen portiert werden kann.

- **TraX-Server.** Der TraX-Server wird von einem spezialisierten Location Provider betrieben und steuert die TraX-Clients einer Vielzahl von Zielpersonen dynamisch mit Position Update Requests. Er übernimmt somit den server-seitigen Teil der Positionsübermittlungsschicht. Der TraX-Server realisiert außerdem die paarweise und die allgemeine Relationsschicht. Im Vergleich zu einem Ansatz, wo diese Schichten beim LBS Provider liegen, wird so eine Vielzahl von Synergien möglich. Außerdem müssen bei höherwertigen Mechanismen wie der Nahbereichserkennung keine „rohen“ Ortsinformationen an den LBS Provider ausgeliefert werden, was aus Sicht der Zielperson Privacy-Vorteile bietet. Verschiedene LBSs können benutzt werden, ohne den einzelnen LBS Providern in hohem Maße vertrauen zu müssen. Der TraX-Server verwaltet

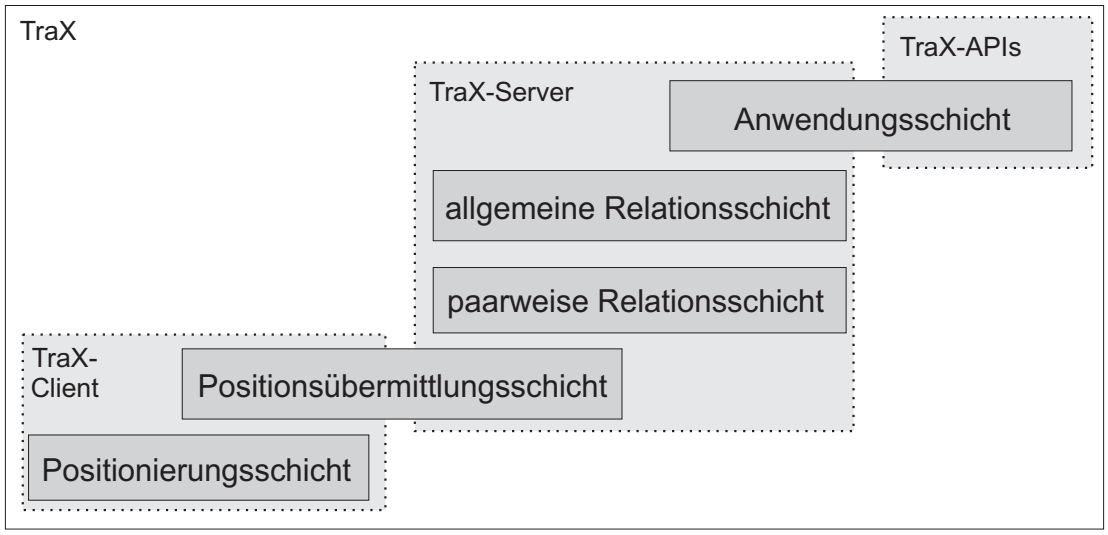

Abbildung 7.8: Verteilung der TraX-Schichten auf die TraX-Komponenten

überdies die Nutzerkonten der Zielpersonen und Nutzer und implementiert entsprechende Authentifizierungs- und Privacy-Mechanismen.

Trotz des Namens, der eher einen monolithischen Aufbau andeutet, ist der TraX-Server ein verteiltes System, das eine hohe Skalierbarkeit gewährleistet. Diese wird insbesondere dadurch erreicht, dass sich die vom Server realisierten Schichten (vergleiche Abbildung 7.8) aufgrund der klaren funktionalen Trennung sehr leicht auf verschiedene Rechner verteilen und replizieren lassen. Zu diesem Zweck kann zum Beispiel eine Kommunikations-Middleware wie CORBA oder Java *Remote Method Invocation (RMI)* eingesetzt werden.

- **TraX-APIs.** LBSs greifen auf die Funktionalitäten von TraX zu, indem sie eine der bereitgestellten Software-Bibliotheken einbinden, die als TraX-APIs bezeichnet werden. Die Bibliotheken sind für verschiedene Programmiersprachen und Service-Modelle erhältlich und decken mobile und nicht mobile Nutzungsfälle ab. Ein Ziel der TraX-APIs ist es, eine möglichst einfache Realisierung proaktiver Mehrpersonen-LBCSs zu ermöglichen. Mit Hilfe der TraX-APIs kann sich ein LBS bei TraX für den Erhalt eines bestimmten räumlichen Ereignisses, zum Beispiel der räumlichen Annäherung zweier Zielpersonen, registrieren. Beim Eintreffen des Ereignisses wird der Dienst asynchron darüber informiert.

  Insbesondere die asynchrone Kommunikation stellt bekannte Anforderungen an das Verbindungs- und Session-Management, für deren Lösung Middleware-Ansätze wie CORBA im Festnetz oder MundoCore [28] im mobilen Umfeld konzipiert wurden. Ziel bei TraX ist jedoch die Bereitstellung einer möglichst leichtgewichtigen Komponente sowie die einfache Integrierbarkeit mit vielen bestehenden Anwendungen. Auf die Benutzung einer generischen Kommunikations-Middleware wurde deshalb verzichtet. Auch die Benutzung von Webservices [17], denen grundsätzlich eine synchrone Kommunikation zugrunde liegt, ist für das beabsichtigte asynchrone Kommunikationsmuster nicht geeignet. Die TraX-APIs werden in der Domäne des Nutzers

bzw. des LBS-Providers ausgeführt und tauschen mit einem entsprechenden Proxy-Dienst in der Domäne des Location Providers basierend auf einem proprietären Protokoll Daten aus.

Zwei grundlegende Nutzungsmöglichkeiten werden gewährleistet:

- **Indirekte Nutzung von TraX.** Ganz wie in der Wertschöpfungskette in Abbildung 2.4 vorgesehen, wird die TraX-API ausschließlich in der Domäne des LBS Providers ausgeführt. Dieser stellt basierend auf den empfangenen Ortsinformationen und höherwertigen Ereignissen seinen Nutzern Informationen bereit. Beim Nutzer installierte Komponenten kommunizieren lediglich mit denen des LBS Providers.
- **Direkte Nutzung von TraX.** Hinter der direkten Nutzung verbirgt sich derselbe Gedanke, der vielen so genannten *Mash-ups* zugrunde liegt: In der Domäne des Nutzers läuft ein Nutzer-Agent, typischerweise ein Web-Browser, der sowohl in Kontakt mit dem LBS Provider als auch dem Location Provider, also TraX, steht. Die endgültige, dem Nutzer präsentierte Informationen wird erst in dessen Domäne zusammengestellt. Dieser Ansatz birgt den großen Vorteil, dass der LBS Provider keinerlei Kenntnis über den Aufenthaltsort der Zielperson haben muss, was eine höhere Nutzerakzeptanz bewirken könnte.

  Solche Mash-ups sind typisch für das so genannte Web 2.0, in dem Communities eine immer größere Rolle spielen. Der Anbieter Qype [13] stellt zum Beispiel einen Empfehlungsdienst von PoIs zur Verfügung, wobei die dargestellten WWW-Seiten dynamisch auf dem Browser des Nutzers zusammengestellt werden – unter Einbindung der Google Maps API [3]. Der bei Qype installierte Anwendungsserver muss dafür nicht mit Google kommunizieren. Weiter unten wird eine auf Ajax basierte TraX-API vorgestellt, mit deren Hilfe sehr schnell und einfach Ortsinformationen mobiler Nutzer in WWW-Seiten integriert werden können.

  Bei dem direkten Nutzungsmodell müssen zum einen für den PC konzipierte Anwendungen unterstützt werden, bei denen ein stationärer Nutzer eine oder mehrere mobile Zielpersonen beobachtet (querverweisende LBSs). Zum anderen sollen die TraX-APIs natürlich für mobile Endgeräte verfügbar sein, so dass der Nutzer selbst mobil sein kann.

7.3 Implementierung von TraX

Im Folgenden wird ein kurzer Überblick über die Implementierung von TraX gegeben. Der TraX-Client sowie die TraX-APIs sind bereits für eine Reihe verschiedener Plattformen verfügbar. Beide sind im Sinne einer möglichst einfachen Portierung sehr leichtgewichtig gehalten. Der TraX-Server ist in J2SE implementiert und kann auf mehrere Rechner verteilt werden. Momentan wird er in der Domäne *trax.mobile.ifi.lmu.de* betrieben. Kurz beschrieben wird auch, wie der TraX-Simulator funktioniert, mit dessen Hilfe zum Beispiel die in Kapitel 4 und 5 beschriebenen Strategien simuliert wurden.

7.3.1 Der TraX-Client

Der leichtgewichtige TraX-Client ist mittlerweile in mehreren Versionen verfügbar, unter anderem für J2ME in der *Connected Limited Device Configuration (CLDC)* [1], für das .NET Compact Framework [11], das für Geräte mit Windows Mobile [10] verfügbar ist, sowie als native Symbian-Anwendung [16] in C++, was genauer in [26] beschrieben ist (TraX hieß zu der Zeit noch *LAMA, Location-Aware Mobility Architecture*).

Unter den getesteten Geräten sind verschiedene Modelle von Mobiltelefonen bzw. Smartphones: das Siemens S60, das Nokia 9500, der HTC Magician sowie das Siemens SXG75. Das Siemens SXG75 ist dabei das einzige Gerät mit eingebautem GPS-Sensor. Bei den anderen Geräten wurde ein externer GPS-Empfänger entweder per Bluetooth oder per *Secure Digital Input/Output (SDIO)*-Schnittstelle angekoppelt. Außerdem wurde der Client auf *Personal Digital Assistants (PDAs)* von Siemens, Compaq und HP getestet, zum einen ebenfalls unter Anbindung einer externen GPS-Maus, zum anderen unter Benutzung von WLAN Fingerprinting als Ortungssystem. Momentan wird der TraX-Client auf eine spezielle Hardware-Plattform portiert, die nur aus einem GPS-Empfänger und einer Mobilfunkkomponente besteht und die sich mittels Python programmieren lässt, vergleiche [4]. Angedacht ist es außerdem, den TraX-Client für das SIM (Subscriber Identity Module) Application Toolkit bereitzustellen. Die Ortung könnte anhand der gemessenen Signalstärken und Identifikatoren umliegender GSM-Basisstationen erfolgen. In dem Fall würde der Client also auf der SIM-Karte eines Mobiltelefons installiert. Der Vorteil wäre eine bereits jetzt sehr hohe Durchdringung geeigneter Geräte, vergleiche [68].

Der Slogan ”Write once, run anywhere”, mit dem die Firma Sun Microsystems darauf anspielt, dass einmal erstellte Java-Programme auf jeglichen Java-fähigen Endgeräte-Plattformen funktionieren, trifft insbesondere bei J2ME nicht zu. Trotz der standardisierten Syntax der bereitgestellten APIs verhalten sich diese auf verschiedenen Endgerätemodellen zum Teil sehr unterschiedlich. Ein besonders negatives Beispiel war die Implementierung der Bluetooth API [9] auf dem Siemens S60. Auch wenn man von diesen Einschränkungen absieht, eignet sich J2ME für die Realisierung des TraX-Clients nur sehr bedingt. Der Client soll nämlich als Software-Daemon im Hintergrund laufen, um mit dem TraX-Server zu kommunizieren, was die meisten der getesteten Geräte für J2ME nicht vorsehen und was auch in keiner der bekannten J2ME-Spezifikationen behandelt wird.

Besonders einfach war im Gegensatz dazu die Entwicklung unter .NET. Die in C# erstellte Software konnte mit minimalem Aufwand direkt auf dem Gerät getestet werden, die bereitgestellten APIs verhalten sich auf verschiedenen Geräten gleich, und der Client kann ständig im Hintergrund laufen.

7.3.2 Der TraX-Server

Der TraX-Server ist eine verteilte J2SE-Anwendung und realisiert die oben vorgestellten Schichten: den server-seitigen Teil der Positionsübermittlungsschicht, die paarweise und allgemeine Relationsschicht sowie einen Teil der Anwendungsschicht. Jede Schicht verfügt dabei über zwei so genannte *Service Access Points (SAPs)*, die jeweils die Schnittstelle zur darunter- bzw. darüber liegenden Schicht darstellen. Die Schichten kommunizieren also untereinander nur über die entsprechenden SAPs, was es sehr einfach macht, die Schichten auf verschiedene Rechner zu verteilen und zu replizieren. Insbesondere können Instanzen von

Schichten dynamisch erzeugt und zerstört werden.

Eine Herausforderung bei der Realisierung war die *Interprozesskommunikation (Inter-Process Communication, IPC)* zwischen den Schichten. Als mögliche Lösung wurde zuerst Java RMI getestet, später dann aber aus Effizienzgründen verworfen. Insbesondere die Implementierung von Callbacks, die bei der asynchronen Benachrichtigung über bestimmte räumliche Ereignisse erforderlich ist, schien bei RMI unvorteilhaft. Gleichzeitig sollte aber die erhöhte Komplexität von CORBA vermieden werden. Es wurde daher eine eigene Lösung zur IPC implementiert, die die Kommunikation der Schichten über Sockets ermöglicht. Auch war bei RMI nicht zufrieden stellend, wie Verbindungsabbrüche bzw. Ausfälle erkannt wurden. In der bestehenden Implementierung wurde daher eine möglichst einfache und robuste Lösung mit Hilfe von Keep-alive-Nachrichten umgesetzt. Nachdem sich alle Schichten des TraX-Servers im selben Netzwerk befinden, wurde auch auf den Einsatz einer zentralen Service Registry wie bei RMI verzichtet. Stattdessen werden die Schichten mit einem global gültigen Initialisierungsskript gestartet, welches die Adressen der entsprechenden Rechner enthält. Ziel war wiederum eine möglichst einfache Fehlererkennung und -behebung.

Will eine Schicht auf die Funktionen der darunter liegenden Schicht zugreifen, so registriert sie sich direkt bei dieser und erhält im Gegenzug ein so genanntes *Handle* zugeteilt. Mit Hilfe des Handles können dann im Folgenden synchrone wie auch asynchrone Abfragen, zum Beispiel Position Update Requests an die Positionsübermittlungsschicht, abgesetzt werden. Solange das Handle besteht, werden etwaige asynchrone Callbacks, zum Beispiel Position Updates, korrekt an die anfragende Schicht zugestellt. Fällt die anfragende Schicht aus oder bricht die Verbindung für längere Zeit ab, so wird dies automatisch erkannt, und alle mit dem Handle verbundenen Anfragen werden abgebrochen.

Listings 7.1 und 7.2 verdeutlichen das Zusammenspiel zweier Schichten anhand der Positionsübermittlungsschicht: Eine darüber liegende Schicht (nach Abbildung 7.5 kann das die paarweise Relationsschicht oder die Anwendungsschicht sein) implementiert dafür die Schnittstelle *App_SAP_PosTrans* (das steht für den SAP, den die Schicht der Positionsübermittlungsschicht bietet) und registriert sich bei der Schnittstelle *PosTrans_SAP_App* (das steht für den SAP, den die Positionsübermittlungsschicht darüber liegenden Schichten bietet) mittels *createHandle(...)*.

Zurückgeliefert wird ein entsprechendes Handle, mit dem nun Anfragen an die Positionsübermittlungsschicht möglich sind. Mit der Funktion *pollPositions(...)* kann zum Beispiel (synchron) der aktuelle Aufenthaltsort einer oder mehrerer Zielpersonen ermittelt werden. Die Zielpersonen werden innerhalb von TraX mit einem globalen Identifikator, zum Beispiel einer E-Mail-Adresse, gekennzeichnet. Außerdem besteht die Möglichkeit, mit Hilfe von *addPURequest(...)* asynchrone Position Update Requests bezüglich einer Zielperson abzusetzen. Bislang sind periodische und distanz- wie zonenbasierte Position Update Requests möglich. Löst das Endgerät der Zielperson ein entsprechendes Position Update aus, so wird die anfragende Schicht über die Callback-Funktion *positionUpdated(...)* darüber informiert.

Für die paarweise und allgemeine Relationsschicht stehen in TraX ebenfalls entsprechende Schnittstellen zur Verfügung, die an dieser Stelle jedoch nicht weiter ausgeführt werden. Praktisch umgesetzt und getestet wurden die DCC- und DSC-Strategien für die Nahbereichs- und Trennungserkennung sowie die in Kapitel 5 behandelte Erkennung von Cliquen.

Listing 7.1: Schnittstelle der Positionsübermittlungsschicht zu den oberen Schichten

```
public interface PosTrans_SAP_App{
    /**
     * @param app: das Callback-Objekt
     * @return: ein gueltiges Handle
     */
    public String createHandle(App_SAP_PosTrans app);

    /**
     * Setzt einen Position Update Request ab.
     * @param targetID: Identifikator der Zielperson
     * @param reqID: Identifikator der Anfrage
     * @param handle: Handle, zuvor mit createHandle() erzeugt
     * @param req: der Position Update Request
     */
    public void addPURequest(String targetID, String reqID, PURequest req,
            String handle);

    /**
     * Nimmt einen Position Update Request zurueck.
     * @param targetID: Identifikator der Zielperson
     * @param reqID: Identifikator der Anfrage
     * @param handle: Handle mit dem die Anfrage abgesetzt wurde
     * @return: der zurueckgenommene Position Update Request
     */
    public PURequest removePURequest(String targetID, String reqID, String handle);

    /**
     * Fragt synchron die Positionen einer od. mehrerer Zielpersonen an.
     * @param targetIDs: Identifikatoren der Zielpersonen
     * @param handle: Handle, zuvor mit createHandle() erzeugt
     * @param pr: die Polling-Request-Objekte
     * @return: die gelieferten Polling Responses
     */
    public PollingResponse [] pollPositions(String [] targetIDs,
    PollingRequest [] pr, String handle);
}
```

Nachdem der TraX-Server für eine Vielzahl von TraX-Clients eingesetzt werden soll, muss er hinsichtlich der Anzahl gleichzeitig verwalteter Datenverbindungen sehr gut skalierbar sein. Aus diesem Grund wurde die Positionsübermittlungsschicht in der vorliegenden Implementierung noch feiner unterteilt: Die unterste Schicht ist besonders leicht replizierbar und dient dem Austausch von Rohdaten. Sie verschattet die Verwaltung der Socket-Verbindungen vor den höheren Schichten. Darauf aufbauend liegt eine Schicht, die die auf dem Endgerät platzierten Position Update Requests über Verbindungsabbrüche hinweg vorhält und das Endgerät bei einer Neuverbindung entsprechend synchronisiert.

Listing 7.2: Schnittstelle der oberen Schichten zur Positionsübermittlungsschicht

```
public interface App_SAP_PosTrans
{
    /**
     * Das Endgeraet einer Zielperson hat in Folge eines
     * Position Update Request ein Position Update berichtet.
     * @param targetID: Identifikator der Zielperson
     * @param reqID: Identifikator der Anfrage
     * @param up: das berichtete Position Update
     */
    public void positionUpdated(String targetID, String reqID, PositionUpdate up);
}
```

Der Einfachheit halber werden bislang im TraX-Server die aktuell bearbeiteten Anfragen nicht persistent gespeichert. Fällt eine Server-Komponente aus, so werden alle ausstehenden asynchronen Anfragen gelöscht, und die LBS-Anwendung wird über den Ausfall benachrichtigt.

Die persistente Verwaltung von Nutzerdaten basierend auf einer *Lightweight Directory Access Protocol (LDAP)*-Datenbank wird zur Zeit implementiert. Außerdem ist derzeit eine geeignete Implementierung zur Definition und Durchsetzung von Privacy-Policies in Bearbeitung. Als Ausgangspunkt für diese praktischen Arbeiten dienen zum Teil Konzepte aus [158].

7.3.3 Der TraX-Simulator

Ein Ziel bei der Entwicklung des TraX-Simulators, mit dem unter anderem die in Kapitel 4 und 5 entwickelten Strategien evaluiert wurden, war es, bereits einen möglichst großen Teil des auch im TraX-Server eingesetzten Codes zu verwenden. So lassen sich schon anhand der zahlreichen durchgeführten Simulationen viele Fehler erkennen und beheben, die sonst erst beim Betrieb des produktiven Systems auftauchen würden. Die vorgestellten Algorithmen zur Nahbereichs- und Trennungserkennung wurden also gegen dieselbe Schnittstelle der Positionsübermittlungsschicht programmiert, die auch im TraX-Server vorliegt, vergleiche Listings 7.1 und 7.2. Anstelle einer Anbindung realer TraX-Clients über Netzwerk-Sockets werden für die Simulationen die Bewegungen der Teilnehmer jedoch simuliert. Der TraX-Simulator läuft daher in einem einzelnen Prozess ab.

Zur dynamischen Instantiierung verschiedener Simulationsszenarien wurde eine spezielle Skriptsprache entwickelt, die ein auf *JavaCC (Java Compiler Compiler)* [5] basierter Parser verarbeitet, vergleiche Listing 7.3:

Listing 7.3: Skript zur Instantiierung eines Simulationslaufs

```
/*Ursprung, Breite und Hoehe des Spielfelds */
SetGeoConstraints(32,N,680km,5310km,20km,20km);

/* Zeitaufloesung und simulierte Zeit */
SetTimeResolution(5.0 seconds);
```

```
SetRunTime(2.0 hours);

/* Definition von Bewegungsmodellen, z.B. Smooth Random, Random Waypoint
oder GPS Traces*/
AddMicroscopicModel model1(SmoothRandom,top_speed=15.0 km/h,max_slow_down=\\
 20.0 km/h/s,max_speed_up=10.0 km/h/s,max_torque=0.1/s, speed_thresh =0.1/s,\\
 dir_thresh =0.01/s, min_turn_time =20.0 minutes, max_turn_time =2.0 hours);
AddMicroscopicModel model2(RndWaypoint,20.0 km/h);
AddMicroscopicModel model3(GPSTraces , "../ traces /" , keep);

/* Definition von Zielpersonen */
AddMobileEntities (5, model1);
AddMobileEntities (7, model2);
AddMobileEntities (3, model3);

/* Parametrisierung und Instantiierung des Simulationsszenarios */
 SetInt    (... , ...);
 SetStr    (... , ...);
Simulate(de.lmu.lbs . simulator .MyScenario);
```

Die Konfiguration eines Szenarios umfasst also immer die folgenden Schritte:

1. **Geographische Definitionen.** Die geographischen Ausmaße des simulierten Spielfelds werden im UTM-Koordinatensystem definiert. Benutzt man GPS Traces als Bewegungsmodell, so muss hierbei darauf geachtet werden, dass das Spielfeld die Orte, an denen die Traces aufgenommen wurden, mit einschließt.

2. **Zeitliche Definitionen.** Es wird angegeben, wie viel simulierte Zeit einem Simulationsschritt entspricht und wie viel Zeit insgesamt simuliert werden soll. Der TraX-Simulator betrachtet also diskrete Zeitschritte und unterstützt sowohl ereignisgesteuerte [152] als auch taktgesteuerte Simulation. Die erste Variante ist effizienter, wenn die für die Simulation relevanten Ereignisse eher unregelmäßig und selten auftreten. Die zweite bietet sich insbesondere bei der gewünschten Simulation mobiler Nutzer an, da Bewegungsmodelle einfacher zu implementieren sind, wenn sie auf Taktsteuerung basieren, vergleiche [125].

3. **Definition von Bewegungsmodellen.** Eine Reihe von Bewegungsmodellen lassen sich mit Hilfe der selbstentwickelten Skriptsprache konfigurieren. Unter den künstlichen Modellen befinden sich zum Beispiel das Smooth-Random und das Random-Waypoint-Modell. Es lassen sich aber auch GPS Traces einbinden, die im NMEA-Format vorliegen und die zum Beispiel mit dem in [148] entwickelten Tool von einem GPS-fähigen mobilen Endgerät aufgenommen werden können.

4. **Definition von Zielpersonen.** In das Spielfeld lässt sich eine beliebige Anzahl simulierter Zielpersonen einsetzen, die jeweils mit unterschiedlichen Bewegungsmodellen assoziiert sein können.

5. **Parametrisierung und Instantiierung des Simulationsszenarios.** Schließlich können weitere Parameter an die Simulation übergeben werden, zum Beispiel die getestete Nahbereichsdistanz bei der Nahbereichserkennung. Die Steuerung der Simulation übernimmt dann eine spezielle Java-Klasse, die ebenfalls im Skript angegeben wird, vergleiche Listing 7.3, letzte Zeile.

Optional kann der TraX-Simulator mit einem Visualisierungsmodul gekoppelt werden. Als Beispiele dienen die auf http://www.mobile.ifi.lmu.de/TraX/ verfügbaren Java-Applets.

7.3.4 Die TraX-APIs

Zur Einbindung der Funktionen von TraX in LBS-Anwendungen wurden bislang TraX-APIs für die folgenden Plattformen implementiert:

- **J2SE API.** Die J2SE API unterstützt die indirekte Nutzung von TraX (vergleiche oben). Sie wird also in der Domäne des LBS Providers server-seitig als Bibliothek in Java-Programme bzw. Servlets eingebunden. Die Kommunikation mit dem TraX-Server erfolgt mit RMI.

- **J2ME API.** Die J2ME API unterstützt die direkte Nutzung von TraX und wird auf dem mobilen Endgerät des Nutzers installiert, wo sie von verschiedensten Anwendungen genutzt werden kann. Die Kommunikation mit dem TraX-Server erfolgt über eine Socket-Verbindung basierend auf einem proprietären Protokoll, welches auch die Authentifizierung des Nutzers gewährleistet. RMI ist in J2ME nicht verfügbar und würde auch zu einem erhöhten Datenaufkommen über die Luftschnittstelle führen, da das zugrunde liegende Protokoll nicht sehr effizient ist.

- **.NET API.** Dasselbe proprietäre Protokoll wie bei der J2ME API wird bei der .NET API, die für das *.NET Compact Framework* entwickelt wurde, eingesetzt. Die .NET API unterstützt ebenfalls die direkte Nutzung von TraX und kann auf Endgeräten basierend auf dem Windows-Mobile-Betriebssystem installiert werden. Sowohl die J2ME als auch die .NET API können separat auf dem Endgerät installiert werden oder mit dem ebenfalls für diese Plattformen verfügbaren TraX-Client kombiniert werden. Im letzten Fall erfolgen beide Kommunikationsverbindungen zum TraX-Server – einmal zum Abruf der TraX-Funktionen und einmal zum Aktualisieren der eigenen Position auf dem Server – über dasselbe Socket, was eine Effizienzsteigerung bewirkt.

- **Ajax API.** Die Ajax API ist ebenfalls für die direkte Nutzung von TraX konzipiert. Ajax steht für *Asynchronous JavaScript and XML* und ist in Web-2.0-Anwendungen derzeit sehr verbreitet. Hauptmerkmal von Ajax ist, dass innerhalb einer HTML-Seite eine HTTP-Anfrage durchgeführt werden kann, ohne die Seite komplett neu laden zu müssen. Nutzdaten können also sukzessiv bei Bedarf nachgeladen werden, und zwar ohne dass die Seite beim Laden für Nutzereingaben blockiert ist. Der Datenaustausch verläuft also asynchron. Mittels Javascript werden die Nutzdaten verschiedener Inhalteanbieter auf dem Web-Browser verknüpft, was die einfache Realisierung von Mash-ups (engl. für Vermanschung) ermöglicht.

Die Ajax API von TraX kann mit sehr einfachen Mitteln in jede HTML-Seite eingebunden werden. Mit nur wenigen JavaScript-Befehlen kann dann die Positionsübermittlungsschicht sowie die paarweise und allgemeine Relationsschicht angesprochen werden. Weiter unten wird am Beispiel der Google Maps Demo gezeigt, wie durch TraX gewonnene Ortsinformationen mit Hilfe der Ajax API ganz einfach mit Kartenmaterial von Google „vermanscht“ werden.

Server-seitig wird die Ajax-API realisiert durch Java-Servlets, die wiederum mit Hilfe des selbst entwickelten IPC-Mechanismus an den TraX-Server angebunden sind.

7.3.5 LBS-Prototypen

Im Folgenden werden kurz zwei auf TraX basierende, prototypisch realisierte LBSs vorgestellt. Der erste demonstriert, wie die beschriebene Ajax API mit Google Maps zusammenspielt, um die Positionen mobiler Zielpersonen auf einer Karte anzuzeigen. Die zweite Anwendung, ein Buddy Tracker, basiert auf der J2ME API von TraX und ermöglicht die in Kapitel 4 beschriebene Nahbereichserkennung innerhalb einer Gruppe mobiler Freunde.

Google Maps Demo

Anhand der Google Maps Demo wird die besonders einfache Nutzung der Ajax API von TraX klar. Mit den wenigen, in Listing 7.4 enthaltenen Zeilen HTML-Code lassen sich in jedem Ajax-fähigen Web-Browser die Positionen mobiler Zielpersonen durch TraX ermitteln und auf einer Karte anzeigen. Die Position auf der Karte aktualisiert sich auch automatisch, sobald ein ausgegebener Position Update Request mit einem Position Update quittiert wird.

Wie im Listing ersichtlich, wird erst die Ajax API und dann die Google Maps API in die HTML-Seite eingebunden, jeweils mit einer Zeile JavaScript-Code. Es folgt die Definition der Karte, wie durch Google Maps vorgeschrieben (aus Platzgründen weggelassen). Dann wird die Funktion $pollPosition()$ definiert: Im Methodenrumpf wird erst die Positionsübermittlungsschicht, welche durch das von der API bereitgestellte Objekt $PosTrans$ repräsentiert wird, synchron nach der Position einer bestimmten Zielperson angefragt. Dann wird die Position in der Karte eingezeichnet. Beide Aktionen bestehen aus lediglich einer Zeile Programmcode.

Wie die Funktion $addPURequest()$ zeigt, können auch asynchrone Position Update Requests an die Positionsübermittlungsschicht abgesetzt werden. In drei Zeilen HTML-Code wird ein entsprechendes Request-Objekt konstruiert, mit einer bestimmten Update-Distanz konfiguriert und an das globale Objekt $PosTrans$ übergeben. Der Position Update Request wird dann im Hintergrund über den TraX-Server an den TraX-Client der Zielperson übermittelt und dort ausgewertet. Meldet der TraX-Client zu einem späteren Zeitpunkt ein Position Update an den Server, so wird dieses mittels Ajax an die HTML-Seite durchgereicht. Die lokale Ereignisbehandlungsroutine $onPositionUpdate()$ stellt die Position schließlich auf der Karte dar.

Listing 7.4: Die Ajax API von TraX in Kombination mit Google Maps

<**html**> <**head**>

<!−−Einbinden der Ajax API von TraX−−>

```
<script type="text/javascript"
src="http://trax.mobile.ifi.lmu.de:8080/TraX/js/trax.js"></script>

<!--Einbinden der Google Maps API-->
<script type="text/javascript"
src="http://maps.google.com/maps?file=api&v=2&key=[nicht gezeigt]"></script>

<script>
<!--Kartendefinition siehe Google Maps-->

<!--Abfrage einer Position am TraX-Server und Anzeige auf der Karte-->
function pollPosition() {
    PosTrans.pollPosition(document.getElementById('p.mt').value, function(update) {
        draw(update.mt, "mt", update, true);
    });
}

<!--Hinzufuegen eines Position Update Request-->
function addPURequest() {
    var req = new PURequest();
    req.distance = parseInt(id('sdj.distance').value);
    PosTrans.addPURequest(id('sdj.mt').value, "req_001", req);
}

<!--Definition der Callback-Funktion:
Anzeige eines Position Update auf der Karte-->
TraX.attach({
    onPositionUpdate: function(mt, reqid, update) {
        draw(mt, "mt", update, true);
    }});
</script> </head>
<body>
<!--HTML-Textfelder zur Steuerung nicht gezeigt-->
</body> </html>
```

Der Rest der HTML-Seite definiert einfache Textfelder zur Ansteuerung der genannten Funktionen durch den Nutzer. Abbildung 7.9 zeigt einen Screenshot der Google Maps Demo um das Institutsgelände in der Oettingenstr. 67 in München. Die Positionen von vier Zielpersonen sind sichtbar, wobei der Name der Person Axel in der typischen Google-Maps-Sprechblase angezeigt wird. Weiterhin wird das Endgerät von Georg durch einen distanzbasierten Position Update Request mit 500-m-Update-Distanz verfolgt.

In der Google Maps Demo stellt die Ajax API also die Funktionen der Positionsübermittlungsschicht bereit, was sich auch gut nachvollziehen lässt, indem man den HTML-Code aus Listing 7.4 mit Listings 7.1 und 7.2 vergleicht. Natürlich lassen sich mit der Ajax API auch die höherwertigen Funktionen, zum Beispiel die der Nahbereichserkennung, in Web-Seiten

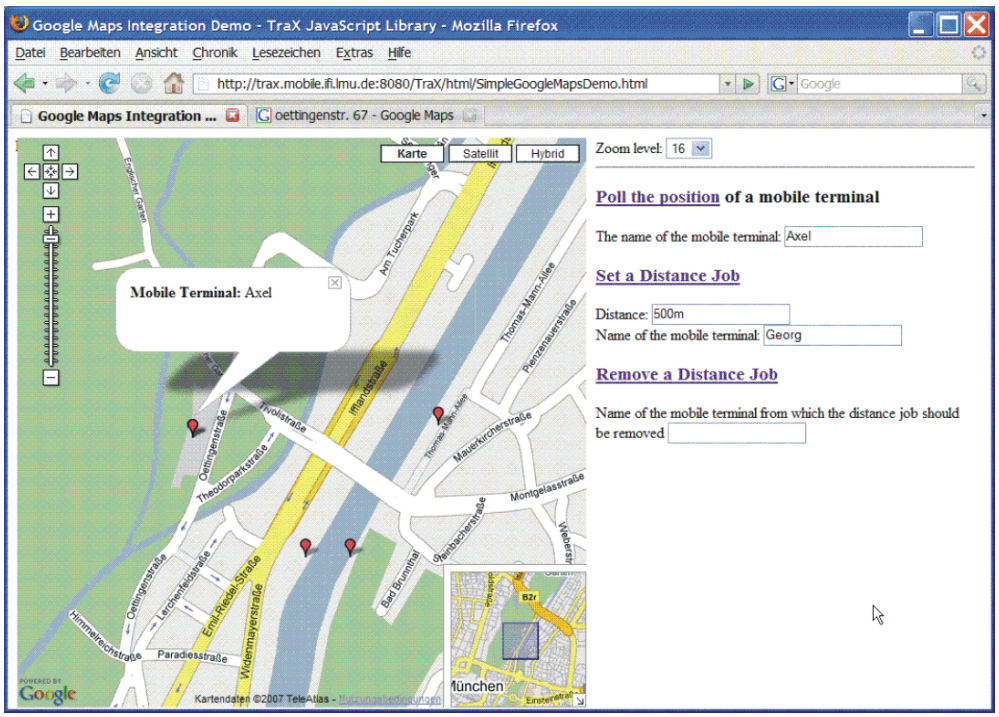

Abbildung 7.9: Screenshot der Google Maps Demo

einbinden. Ein entsprechender Demonstrator wird gerade umgesetzt.

Buddy Tracker

Der Buddy Tracker entspricht dem bereits in Abschnitt 4.4.2 besprochenen Prototypen zur Nahbereichs- und Trennungserkennung. Er ermöglicht es einem Mitglied einer Community, sich für Nahbereichsereignisse zwischen ihm und anderen Community-Mitgliedern zur registrieren. Tritt ein solches Ereignis ein, wird er zum Beispiel per Vibrationsalarm seines Endgeräts darauf hingewiesen. Abbildung 7.10 zeigt eine solche Alarm-Situation, bei der dem Nutzer auch die aktuell zwischen beiden Endgeräten gemessene Distanz angezeigt wird. In dem konkreten Versuch betrug die Nahbereichsdistanz 50 m und die Grenzlinientoleranz 20 m. Der gemessene Wert zeigt also ein korrektes Verhalten an.

Bei der Realisierung des Buddy Trackers sollte vor allem die einfache Benutzbarkeit der TraX-API getestet werden. Zu diesem Zweck entwickelten mehrere Gruppen von jeweils zwei Studenten an einem Übungsnachmittag des Praktikums *Mobile und Verteilte Systeme* eigenständig eine entsprechende Anwendung, wozu ihnen die J2ME API von TraX bereitgestellt wurde. Nachdem die API die Komplexität der Nahbereichserkennung verschattet, konnten sich die Studenten voll auf die Programmierung der graphischen Nutzerschnittstelle konzentrieren. Es zeigte sich, dass fast jede Zweiergruppe das gesteckte Ziel innerhalb weniger Stunden erreichte, so dass noch genügend Zeit zum gegenseitigen Tracking im Englischen Garten blieb. Die API schien also ihren Zweck zu erfüllen. Die bei den Tests gesammelten Erfahrungen wurden bereits in Abschnitt 4.4.2 dokumentiert.

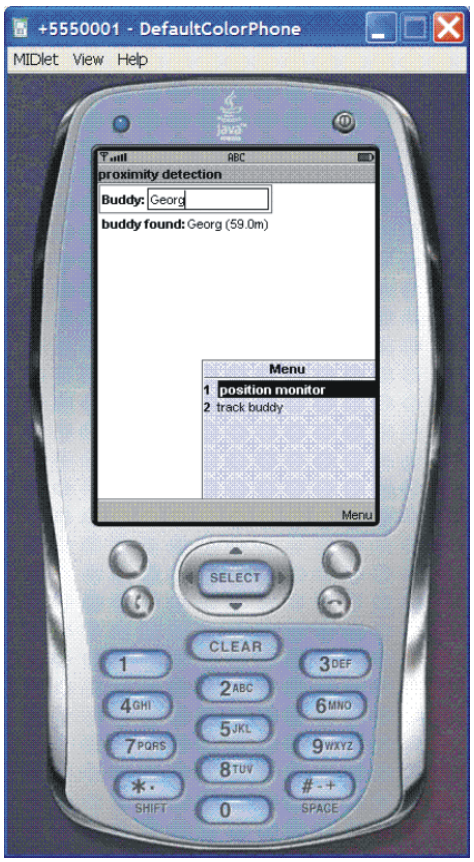

Abbildung 7.10: Screenshot des Buddy Trackers

7.4 Zusammenfassung

Die TraX-Plattform stellt LBS-Anwendungen eine Art Baukastensystem zur Verfügung, mit dem sich insbesondere proaktive Mehrpersonen-LBSs sehr einfach realisieren lassen. Die Flexibilität von TraX ergibt sich durch die Identifikation verschiedener funktionaler Schichten (Positionsübermittlungs-, paarweise und allgemeine Relationsschicht). Die Möglichkeiten, die die vorliegende Schichtenaufteilung bietet, wurden bereits anhand der Erkennung von Cliquen in Kapitel 5 angedeutet. Zu betonen ist jedoch, dass sich durch die beschriebene Aufteilung eine Vielfalt verwandter Probleme lösen lassen (k nächste Nachbarn, Erkennung von geometrischen Formen ...). Ein vergleichbarer Ansatz der Schichtenorganisation wurde in der Literatur bislang nicht beschrieben.

Bei der Architekturkonzeption war außerdem wichtig, den Zugriff auf die Funktionen von TraX (TraX-APIs) klar von denen zur Ortung der Zielpersonen und zur Realisierung der Position-Update-Methoden (TraX-Client/TraX-Server) zu trennen. Dieser modulare Ansatz erlaubt die Kombination verschiedenster Technologien und Plattformen, von denen mehrere implementiert und praktisch getestet wurden. Auch wurden die bereitgestellten TraX-APIs bereits erfolgreich von Dritten (Studenten) erprobt.

8 Zusammenfassung und Ausblick

Die Dissertation hat das Thema *Community-Dienste* unter dem neuen Blickwinkel des Ortsbezugs behandelt. Durch die Spezialisierung auf das so genannte *Position Management* wurden grundlegende Probleme identifiziert, die bei der Integration von Ortsinformationen in Community-Dienste auftreten. Die entwickelten Lösungen sind generisch gehalten und daher auf ein weites Feld denkbarer Community-Dienste anwendbar. Beispiele sind neuartige MoSoSo-Anwendungen, die proaktiv arbeiten, Friend-Finder-Dienste sowie Mobile-Gaming-Anwendungen. Aber auch seriösere Anwendungen im Bereich CSCW und Logistik sind denkbar.

Ein Merkmal dieser Arbeit ist, dass, wie es der technischen Informatik entspricht, zunächst auf rein technische Problemstellungen eingegangen wird (begrenzte Luftschnittstelle, begrenzte Batterie- und Rechenressourcen auf dem Endgerät, Skalierbarkeit des Servers usw.). Trotz der experimentell nachgewiesenen Effizienz der entwickelten Lösungen werden diese jedoch nicht nur unter einem algorithmischen Blickwinkel behandelt. Vielmehr misst die Arbeit organisatorischen und sozialen Aspekten eine erhöhte Bedeutung zu, die bei der Erbringung eines LBCS hervortreten. Beispiele sind das vorliegende LBS-Rollenmodell, die LBS-Klassifikation sowie die Betrachtungen zu den LBS-Wertschöpfungsketten aus Kapitel 2. Durch die stete Überprüfung algorithmischer Lösungen an solchen organisatorischen Modellen wird der Arbeit hoffentlich eine hohes Maß an praktischer Relevanz zuteil.

Als erstes wurden proaktive Mehrpersonen-LBCSs behandelt, die besondere Herausforderungen an das Tracking von Zielpersonen stellen. Für den dabei zentralen Mechanismus der proaktiven Nahbereichs- und Trennungserkennung wurden eine Palette eigener Strategien entwickelt und diese sowohl simulativ als auch anhand von Prototypen miteinander verglichen. Ferner wurde ein gleichzeitig sehr effizienter und flexibler Mechanismus zur Erkennung von Cliquen beschrieben. Anhand des Zusammenspiels von Nahbereichs- und Trennungserkennung und der Erkennung von Cliquen wurde schließlich eine funktionale Schichtung entwickelt, die die einfache Lösung anderer wichtiger Probleme in Aussicht stellt und somit einen wichtigen Ausgangspunkt für weitere Arbeiten auf dem Gebiet darstellt.

Das zweite behandelte Thema war der Datenschutz, der sowohl auf einer rein technischen Ebene (Anonymisierung vor Intermediären) als auch auf einer nutzerfokussierten Ebene (einfache und sozial akzeptable Autorisierung von Nutzern) behandelt wurde. Während die praktische Tragfähigkeit der dazu entwickelten Konzepte sich noch zeigen muss, liegt ein wichtiger Beitrag bereits in der Herausarbeitung der entsprechenden Problemstellungen und der darauf basierenden Diskussion bestehender Ansätze.

Die in dieser Arbeit entwickelten Konzepte wurden innerhalb der TraX-Plattform praktisch umgesetzt und erprobt. Dabei wurde Wert auf den modularen Aufbau der TraX-Komponenten gelegt, so dass verschiedenste Hard- und Software-Plattformen flexibel kombinierbar sind. Auch wurde darauf geachtet, dass die Funktionen von TraX möglichst einfach in bestehende Dienste integriert werden können. Die dafür vorgesehenen TraX-APIs, mit

deren Hilfe sich die Realisierung komplexer Funktionen wie der Nahbereichs- und Trennungserkennung vor dem LBCS verschatten lässt, sind bereits für eine Reihe verschiedener Plattformen verfügbar.

Ein Beispiel für die Anwendbarkeit von TraX ist auch das vom Lehrstuhl ausgerichtete Praktikum Mobile Gaming. Basierend auf TraX wurde hier schon zweimal, jeweils während einer viertägigen Blockveranstaltung mit sechs Studenten, eine lauffähiges, ortsbezogenes Mehrbenutzerspiel entwickelt. Das erste auf diese Weise entwickelte Spiel, das den Namen *VsGolf* trägt, wird in [163] beschrieben. Mit diesem Praktikum ist nicht zuletzt also auch der Transfer von Konzepten aus der Forschung (TraX) in die Lehre gelungen.

Im Folgenden wird eine Auswahl zukünftiger Arbeiten präsentiert.

8.1 Integration endgerätunterstützter Ortungsverfahren

Die in dieser Arbeit entwickelten Strategien für proaktive Mehrpersonen-LBCSs basieren auf den in Abschnitt 2.4 beschriebenen Position-Update-Methoden. Diese übermitteln die durch endgerätbasierte Verfahren wie GPS berechneten Positionen nur bei Bedarf an den Location Server und sparen so Ressourcen ein.

Innerhalb von Gebäuden ist jedoch kein GPS verfügbar, weshalb dort oft Location Fingerprinting zur Anwendung kommt. Location Fingerprinting wurde bislang sowohl für 802.11 Netze als auch für GSM vorgeschlagen, vergleiche [32; 132]. Anhand einer Datenbank gemessener Signalstärkemuster umliegender Basisstationen (den Fingerprints), die im Vorhinein in verschiedenen Bereichen eines Gebäudes aufgenommen wurden, kann die aktuelle Position eines mobilen Endgeräts abgeschätzt werden. Die aktuellen Messungen *empfangener Signalstärken (RSS)* des Endgeräts werden dazu mit den Einträgen in der Datenbank abgeglichen. Weil die Speicherung und Pflege der Datenbank auf mobilen Endgeräten schwierig ist, wird für Location Fingerprinting meist ein endgerätunterstützer Ansatz gewählt. Anstelle geographischer Positionen übermittelt das Gerät hierbei die aktuellen RSS-Messungen an den Server, welcher dann die Position des Geräts abschätzt.

Wie zonenbasiertes Updating auf solche Systeme übertragen werden kann, wird genauer in [87] beschrieben. Prinzipiell transformiert der Location Server hierbei geographische Zonen in eine RSS-basierte Darstellung und übermittelt sie dem Endgerät. Dieses vergleicht dann stets aktuelle RSS-Messungen mit dieser Konfiguration und meldet die Werte dem Server nur dann, wenn die Zone betreten oder verlassen wurde. Anstelle einer geographischen Update-Zone wird diese also in einer RSS-basierten Darstellung übertragen. Ausgehend von diesem Basismechanismus lässt sich die Nahbereichs- und Trennungserkennung auch innerhalb von Gebäuden realisieren. Die DCC-Strategie wird hierzu auf die beschriebenen RSS-Zonen angewendet und gleichzeitig für topologische Distanzen angepasst.

Während also bezüglich einer effizienten Positionsübermittlung für Location-Fingerprinting-Systeme bereits entsprechende fortführende Arbeiten bestehen, bieten auch noch andere endgerätunterstützte Verfahren Raum für Verbesserungen. Sehr interessant sind in dieser Hinsicht Verfahren, die Pfadverlustmodelle zur Positionsbestimmung nutzen, vergleiche [112].

8.2 Effiziente Steuerung des GPS-Empfängers

Ein weiteres wichtiges Thema, bei dem wiederum von GPS bzw. Galileo als (endgerätbasiertes) Ortungsverfahren ausgegangen wird, ist die effiziente Ansteuerung des GPS-Empfängers auf dem Endgerät. In Bezug auf das Erkennen des Betretens oder Verlassens von Update-Zonen bzw. des Überschreitens der Update-Distanz bei distanzbasiertem Updating entsteht nämlich folgendes Problem. Um eine möglichst hohe zeitliche Genauigkeit der Erkennung zu gewährleisten, müsste sich GPS quasi im Dauerbetrieb befinden, was allerdings zu einem nicht vertretbaren Energieverbrauch auf dem Endgerät führt.

Ein mögliches Ziel weiterer Arbeiten ist es also, das Betreten und Verlassen von Update-Zonen weiterhin mit hoher zeitlicher und räumlicher Auflösung zu erkennen, aber das GPS-Gerät dabei so lange wie möglich ausgeschaltet zu lassen. Ein Ansatzpunkt wäre die Einbeziehung umliegender 802.11 Access Points: Man nehme an, dass anhand der aktuell gemessenen Position errechnet werden kann, dass zum Betreten bzw. Verlassen der nächstgelegenen Update-Zone noch eine Mindestdistanz d zurückzulegen ist. Stellt das Endgerät nun fest, dass mindestens einer der Access Points, die schon bei der letzten Positionsbestimmung in Funkreichweite waren, immer noch sichtbar ist, so kann der GPS-Empfänger ausgeschaltet bleiben. Dies gilt natürlich nur, wenn d größer ist als die maximal anzunehmende Sendereichweite eines Access Point. Natürlich sind weitere Verfahren denkbar, die zum Beispiel gemessene Signalstärken, mehrere Access Points oder auch Informationen aus dem GSM-Netz mit einbeziehen.

8.3 Weitere Funktionen proaktiver Mehrpersonen-LBCSs

Abschnitt 7.2 wird an dieser Stelle nur kurz zusammengefasst. Anhand der entwickelten TraX-Architektur wurde dort beschrieben, wie sich weitere Funktionen, die von proaktiven Mehrpersonen-LBCSs benötigt werden, möglicherweise relativ leicht lösen lassen. Für die paarweise Relationsschicht sind Algorithmen denkbar, die proaktiv erkennen, wenn sich die räumliche Lagerichtung (Peilung) zweier Zielobjekte um einen bestimmten Wert verändert. Solche Informationen sind für eine Reihe von Anwendungsszenarien interessant. Unter ihnen sind die Verfolgung eines Objekts durch ein anderes und das Erkennen bestimmter geometrischer Figuren, zum Beispiel bei der Koordination von Rettungskräften eines Katastropheneinsatzes. Das Erkennen solcher Figuren fällt wiederum in das Aufgabengebiet der allgemeinen Relationsschicht, für die auch noch weitere Funktionen denkbar sind. Ein Beispiel ist die proaktive Überwachung der k nächsten Nachbarn einer Zielperson, zum Beispiel um mobilen Nutzern stets sortierte Listen nahe gelegener Buddies vorzuhalten.

8.4 Unterstützung neuartiger Nutzerschnittstellen

Ein Thema, das in dieser Arbeit nicht explizit behandelt wurde, ist die Bereitstellung geeigneter Nutzerschnittstellen für LBSs. Dies beinhaltet zum Beispiel die Darstellung von Kartenmaterial auf dem mobilen Endgerät. Auch ortsbezogene Anfragen seitens des Nut-

zers werden zum Teil basierend auf Karten formuliert, wozu neue und möglichst einfache Arten der Interaktion erforscht werden müssen.

Eine im mobilen Umfeld besonders viel versprechende Eingabemodalität ist die natürliche Sprache. Die in der Computerlinguistik entwickelten Verfahren der *Spracherkennung* (*Automatic Speech Recognition* und *Language Understanding*) und *Sprachgenerierung* (*Language Generation*) [114] sind schon auf heutigen mobilen Endgeräten praktisch umsetzbar. Eine besondere Herausforderung in diesem Gebiet ist die Unterstützung proaktiver Dienste, bei denen das Endgerät, durch räumliche Ereignisse angestoßen, den Nutzer automatisch anspricht. Mehr zu diesem Thema lässt sich aus [123; 124] erfahren, wo auch die praktische Implementierung eines proaktiven, sprachbasierten Touristenführers auf einem mobilen Endgerät beschrieben wird.

Literaturverzeichnis

[1] *Connected Limited Device Configuration (CLDC).* http://java.sun.com/products/cldc/, Abruf: 2007-04-06

[2] *Dodgeball.com :: mobile social software.* http://www.dodgeball.com, Abruf: 2007-04-06

[3] *Google Maps API.* http://www.google.com/apis/maps/, Abruf: 2007-04-06

[4] *GSM-GPS Modem/Module.* http://www.roundsolutions.com/gsm-modem/gm862-gps.htm, Abruf: 2007-04-06

[5] *Java Compiler Compiler (JavaCC) - The Java Parser Generator.* https://javacc.dev.java.net/, Abruf: 2007-04-06

[6] *Java Message Service (JMS).* http://java.sun.com/products/jms/, Abruf: 2007-04-06

[7] *JSR 179: Location API for J2ME 1.0.* http://jcp.org/en/jsr/detail?id=179

[8] *JSR 293: Location API for J2ME 2.0.* http://jcp.org/en/jsr/detail?id=293

[9] *JSR 82: Java APIs for Bluetooth.* http://jcp.org/en/jsr/detail?id=82

[10] *Microsoft Windows Mobile.* http://www.microsoft.com/windowsmobile/default.mspx, Abruf: 2007-04-06

[11] *.NET Compact Framework.* http://msdn.microsoft.com/mobility/netcf/, Abruf: 2007-04-06

[12] *PostGIS.* http://www.postgis.org, Abruf: 2007-04-06

[13] *Restaurants, Handwerker, Geschäfte entdecken und Menschen treffen. - Deutschland - QYPE.* http://www.qype.com/, Abruf: 2007-04-06

[14] *sixsense - Welcome to sixsense!* http://www.sixsense.com/, Abruf: 2007-04-06

[15] *Socialight / Home.* http://socialight.com, Abruf: 2007-04-06

[16] *Symbian OS*. http://www.symbian.com/, Abruf: 2007-04-06

[17] *WebServices - Axis*. http://ws.apache.org/axis/, Abruf: 2007-04-06

[18] *Welcome at Freever*. http://www.freever.com, Abruf: 2007-04-06

[19] *Yahoo! Maps Web Services*. http://developer.yahoo.net/maps/, Abruf: 2007-04-06

[20] Directive 2002/58/EC of the European Parliament and of the Council of 24 October 1995 Concerning the Processing of Personal Data and the Protection of Privacy in the Electronic Communications Sector. In: *Official Journal of the European Communities of 31 July 2002* (2002), Juli, Nr. L 201/37

[21] *NMEA 0183 Standard*. Version: 3.01, 2002. http://www.nmea.org/pub/0183/

[22] *PostgreSQL*. Version: 8.2, 2007. http://www.postgresql.org, Abruf: 2007-04-06

[23] 3GPP: Location Services (LCS); Functional Description — Stage 2 (03.71). – TS

[24] 3GPP: Location Services (LCS); Service Description — Stage 2 (22.071). – TS

[25] ACZEL, A.: *The Riddle of the Compass. The Invention That Changed the World*. Harcourt Inc., 2001. – ISBN 3–498–00056–X

[26] ADLER, C.: Portierung des LAMA-Prototyps auf Symbian / LMU München. 2005. – Forschungsbericht

[27] AGARWAL, P. K. ; ARGE, L. ; ERICKSON, J.: Indexing Moving Points. In: *Journal of Computer and System Sciences* 66 (2003), Nr. 1, S. 207–243

[28] AITENBICHLER, E. ; KANGASHARJU, J. ; MUHLHAUSER, M.: Experiences with MundoCore. In: *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*. Kauai, Hawaii, USA : IEEE Computer Society, März 2005, S. 168–172

[29] AMIR, A. ; EFRAT, A. ; MYLLYMAKI, J. ; PALANIAPPAN, L. ; WAMPLER, K.: Buddy Tracking – Efficient Proximity Detection among Mobile Friends. In: *Proceedings of IEEE INFOCOM 2004*. Hong Kong : IEEE Computer Society, März 2004, S. 298–309

[30] AOKI, P. ; WOODRUFF, A.: Making space for stories: ambiguity in the design of personal communication systems. In: *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*. Portland, Oregon, USA : ACM Press, 2005, S. 181–190

[31] BADEL, A. ; MORNON, J. ; HAZOUT, S.: Searching for geometric molecular shape complementarity using bidimensional surface profiles. In: *Journal of Molecular Graphics* 10 (1992), Dezember, Nr. 4, S. 205–211

[32] BAHL, P. ; PADMANABHAN, V.: RADAR: an in-building RF-based user location and tracking system. In: *Proceedings of INFOCOM*. Tel Aviv, Israel : IEEE Computer Society, März 2000, S. 775–784

[33] BAKKEN, D. ; PARAMESWARAN, R. ; BLOUGH, D. ; FRANZ, A. ; PALMER, T.: Data Obfuscation: Anonymity and Desensitization of Usable Data Sets. In: *IEEE Security and Privacy* 2 (2004), Nr. 6, S. 34–41

[34] BECKMANN, N. ; KRIEGEL, H.-P. ; SCHNEIDER, R. ; SEEGER, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: *Proceedings of the ACM SIGMOD international conference on management of data*. Atlantic City, New Jersey, United States : ACM Press, 1990, S. 322–331

[35] BELLAVISTA, P. ; CORRADI, A. ; STEFANELLI, C.: Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment. In: *IEICE Transactions on Communications, Special Issue on Autonomous Decentralized Systems* E83-B (2000), Mai, Nr. 5, S. 961–972

[36] BERESFORD, A.: *Location privacy in ubiquitous computing*. Cambridge, United Kingdom, University of Cambridge, Diss., 2005

[37] BERESFORD, A. ; STAJANO, F.: Location Privacy in Pervasive Computing. In: *IEEE Pervasive Computing* 2 (2003), Nr. 1, S. 46–55

[38] BETTSTETTER, C.: Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks. In: *Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Rome, Italy : ACM Press, 2001, S. 19–27

[39] BETTSTETTER, C. ; RESTA, G. ; SANTI, P.: The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. In: *IEEE Transactions on Mobile Computing* 2 (2003), Nr. 3, S. 257–269

[40] BISDIKIAN, C. ; BOAMAH, I. ; CASTRO, P. ; MISRA, A. ; RUBAS, J. ; VILLOUTREIX, N. ; YEH, D. ; RASIN, V. ; HUANG, H. ; SIMONDS, C.: Intelligent pervasive middleware for context-based and localized telematics services. In: *Proceedings of the 2nd International Workshop on Mobile Commerce*. Atlanta, Georgia, USA : ACM Press, 2002, S. 15–24

[41] BOEHNER, K. ; HANCOCK, J.: Advancing ambiguity. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI)*. Montreal, Canada : ACM Press, 2006, S. 103–106

[42] BOTTAZZI, D. ; CORRADI, A. ; MONTANARI, R.: AGAPE: a location-aware group membership middleware for pervasive computing environments. In: *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC)*. Kiris-Kemer, Turkey : IEEE Computer Society, Juni 2003, S. 1185–1192

[43] BOTTAZZI, D. ; CORRADI, A. ; MONTANARI, R.: A context-aware group management middleware to support resource sharing in MANET environments. In: *Proceedings of the 6th International Conference on Mobile Data Management (MDM)*. Ayia Napa, Cyprus : ACM Press, Mai 2005, S. 147–151

[44] BOTTAZZI, D. ; CORRADI, A. ; MONTANARI, R.: Enabling context-aware group collaboration in MANETs. In: *Proceedings of the 2nd International Symposium on Autonomous Decentralized Systems (ISADS 2005)*. Chengdu, China : IEEE Press, April 2005, S. 310–318

[45] BRAKEL, O. ; NEY, M. ; REICHWALD, R.: Pilotierung mobiler Community-Dienste am Beispiel der Community jetzt.de. In: REICHWALD, R. (Hrsg.) ; KRCMAR, H. (Hrsg.) ; SCHLICHTER, J. (Hrsg.) ; BAUMGARTEN, U. (Hrsg.): *Community Services: Lifestyle*. Eul Verlag, 2005, S. 181–257

[46] CAI, W. ; LEE, F.B.S. ; CHEN, L.: An Auto-Adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation. In: *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS)*. Atlanta, Georgia, USA : IEEE Computer Society, Mai 1999, S. 82–89

[47] CAPRA, L.: Mobile computing middleware for context-aware applications. In: *Proceedings of the 24th International Conference on Software Engineering*. Orlando, Florida, USA : ACM Press, 2002, S. 723–724

[48] CAPRA, L. ; BLAIR, G. ; MASCOLO, C. ; W.EMMERICH ; GRACE, P.: Exploiting reflection in mobile computing middleware. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6 (2002), Nr. 4, S. 34–44

[49] CAPRA, L. ; EMMERICH, W. ; MASCOLO, C.: Middleware for Mobile Computing / Universty College of London. 2001. – Forschungsbericht

[50] CHAUM, David L.: Untraceable electronic mail, return addresses, and digital pseudonyms. In: *Communications of the ACM* 24 (1981), Nr. 2, S. 84–90

[51] CHEN, G. ; KOTZ, D.: A Survey of Context-Aware Mobile Computing Research / Dartmouth College. 2000 (TR2000-381). – Forschungsbericht

[52] CHEN, X. ; CHEN, Y. ; RAO, F.: An efficient spatial publish/subscribe system for intelligent location-based services. In: *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems*. San Diego, CA, USA : ACM Press, 2003, S. 1–6

[53] CHEN, Y. ; RAO, F. ; YU, X. ; LIU, D.: CAMEL: A Moving Object Database Approach for Intelligent Location Aware Services. In: *Proceedings of the 4th International Conference on Mobile Data Management (MDM)*, Springer, Januar 2003, S. 331–334

[54] CHEUNG, R.: An adaptive middleware infrastructure for mobile computing. In: *Special interest tracks and posters of the 14th international conference on World Wide Web*. Chiba, Japan : ACM Press, Mai 2005, S. 996–997

[55] CONAN, D. ; TACONET, C. ; AYED, D. ; CHATEIGNER, L. ; KOUICI, N. ; BERNARD, G.: A Pro-Active Middleware Platform for Mobile Environments. In: *Proceedings of the Mobile Computing Systems in Dynamic Environments Workshop*, ACTA Press, Februar 2004, S. 701–706

[56] CONSOLVO, S. ; SMITH, I. ; MATTHEWS, T. ; LAMARCA, A. ; TABERT, J. ; POWLEDGE, P.: Location disclosure to social relations: why, when, & what people want to share. In: *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*. Portland, Oregon, USA : ACM Press, 2005, S. 81–90

[57] CUELLAR, J.: Location Information Privacy. In: SARIKAYA, B. (Hrsg.): *Geographic Location in the Internet*. Kluwer Academic Publishers, 2002, S. 179–212

[58] CUELLAR, J. ; MORRIS, J. ; MULLIGAN, D. ; PETERSON, J. ; POLK., J.: *Geopriv Requirements, RFC 3693*

[59] DA COSTA, C. ; DA SILVA STRZYKALSKI, M. ; BEMARD, G.: A reflective middleware architecture to support adaptive mobile applications. In: *Proceedings of the ACM symposium on Applied computing*. Santa Fe, New Mexico : ACM Press, 2005, S. 1151–1154

[60] DUCKHAM, M. ; KULIK, L.: A Formal Model of Obfuscation and Negotiation for Location Privacy. In: *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive)*. Munich, Germany : Springer, Mai 2005, S. 152–170

[61] DÖRING, N.: *Sozialpsychologie des Internet*. Hogrefe, 1999. – ISBN: 3-8017-1255-9

[62] DÜRR, F. ; HÖNLE, N. ; NICKLAS, D. ; BECKER, C. ; ROTHERMEL, K.: Nexus– A Platform for Context-Aware Applications. In: ROTH, J. (Hrsg.): *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, Fernuniversität Hagen, Juni 2004, S. 15–18

[63] FRIKKEN, K. ; ATALLAH, M.: Privacy preserving route planning. In: *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. Washington DC, USA : ACM Press, 2004, S. 8–15

[64] GRACE, P. ; BLAIR, G. ; SAMUEL, S.: A reflective framework for discovery and interaction in heterogeneous mobile environments. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 9 (2005), Januar, Nr. 1, S. 2–14

[65] GROH, G. ; HILLEBRAND, D.: LBCS – Location Based Community Services: Proactive LBS Services for Mobile Communities in the Research Project COSMOS / TU München. 2004. – Forschungsbericht

[66] GRUTESER, M. ; BREDIN, J. ; GRUNWALD, D.: Path Privacy in Location-aware Computing. In: *Proceedings of MobiSys 2004 Workshop on Context Awareness*. Boston, Massachusetts, USA, Juni 2004

[67] GRUTESER, M. ; GRUNWALD, D.: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In: *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys)*. San Francisco, CA, USA, Mai 2003

[68] GUTHERY, S. ; CRONIN, M.: *Mobile Application Development with SMS and the SIM Toolkit*. McGraw-Hill Professional, 2001. – ISBN 0071375406

[69] HADIG, T. ; ROTH, J.: Proximity Services with the Nimbus Framework. In: *International Conference on Applied Computing*. Lissabon, Portugal : IADIS Press, März 2004, S. 437–444

[70] HAUSER, C. ; LEONHARDI, A. ; KÜHN, P.: Sicherheitsaspekte in Nexus - einer Plattform für ortsbezogene Anwendungen. In: *it+ti, Zeitschrift für Informationstechnik und Technische Informatik* 44 (2002), Oktober, Nr. 5, S. 268–277

[71] HAWICK, K. ; JAMES, H.: Middleware for context sensitive mobile applications. In: *Proceedings of the Australasian information security workshop conference on ACSW frontiers* Bd. 21. Darlinghurst, Australia : Australian Computer Society, 2003, S. 133–141

[72] HILLEBRAND, C. ; BAUMGARTEN, U.: Location Based (Community) Services & terminalbasierte Positionsbestimmung. In: REICHWALD, R. (Hrsg.) ; KRCMAR, H. (Hrsg.) ; SCHLICHTER, J. (Hrsg.) ; BAUMGARTEN, U. (Hrsg.): *Community Services: Lifestyle*. Eul Verlag, 2005, S. 263–305

[73] HONG, J. ; NG, J. ; LEDERER, S. ; LANDAY, J.: Privacy risk models for designing privacy-sensitive ubiquitous computing systems. In: *Proceedings of the 2004 Conference on Designing interactive systems*. Cambridge, MA, USA : ACM Press, 2004, S. 91–100

[74] HU, H. ; XU, J. ; LEE, D. L.: A generic framework for monitoring continuous spatial queries over moving objects. In: *Proceedings of the ACM SIGMOD international conference on Management of data*. Baltimore, Maryland : ACM Press, 2005, S. 479–490

[75] HYYTIÄ, E. ; VIRTAMO, J.: Random waypoint mobility model in cellular networks. In: *Wireless Networks* 13 (2007), Nr. 2, S. 177–188

[76] IACHELLO, G. ; SMITH, I. ; CONSOLVO, S. ; ABOWD, G. ; HUGHES, J. ; HOWARD, J. ; POTTER, F. ; SCOTT, J. ; SOHN, T. ; HIGHTOWER, J. ; LAMARCA, A.: Control, Deception, and Communication: Evaluating the Deployment of a Location-Enhanced Messaging Service. In: *Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp)*, Springer, September 2005, S. 213–231

[77] IACHELLO, G. ; SMITH, I. ; CONSOLVO, S. ; CHEN, M. ; ABOWD, G. D.: Developing privacy guidelines for social location disclosure applications and services. In: *Proceedings of the 2005 symposium on Usable privacy and security (SOUPS)*. Pittsburgh, Pennsylvania : ACM Press, 2005, S. 65–76

[78] IOANNIDIS, A. ; PRIGGOURIS, I. ; MARIAS, Y. ; HADJIEFTHYMIADES, S. ; KASSAPOGLOU-FAIST, C.: Demo: PoLoS: integrated platform for location based services. In: *Proceedings of the 6th international conference on Mobile Data Management (MDM)*. Ayia Napa, Cyprus : ACM Press, Mai 2005, S. 313–315

[79] IOANNIDIS, A. ; SPANOUDAKIS, M. ; SIANAS, P. ; PRIGGOURIS, I. ; HADJIEFTHYMIADES, S. ; MERAKOS, L.: Using XML and related standards to support Location Based Services. In: *Proceedings of the ACM Symposium on Applied computing*. Nicosia, Cyprus : ACM Press, März 2004, S. 1629–1633

[80] ISAACS, E. ; WALENDOWSKI, A. ; RANGANTHAN, D.: Hubbub: a sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions. In: *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*. Minneapolis, Minnesota, USA : ACM Press, April 2002, S. 179–186

[81] JIANG, X. ; HONG, J. ; LANDAY, J.: Approximate Information Flows: Socially-Based Modeling of Privacy in Ubiquitous Computing. In: *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp)*. Göteborg, Sweden : Springer, 2002, S. 176–193

[82] JÄRVENSIVU, R. ; PITKÄNEN, R. ; MIKKONEN, T.: Object-oriented middleware for location-aware systems. In: *Symposium on Applied Computing (SAC)*. Nicosia, Cyprus : ACM Press, 2004, S. 1184–1190

[83] KAEMARUNGSI, K. ; KRISHNAMURTHY, P.: Modeling of Indoor Positioning Systems Based on Location Fingerprinting. In: *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM) 2004*. Hong Kong : IEEE Computer Society, März 2004

[84] KARAT, C. ; KARAT, J. ; BRODIE, C. ; FENG, J.: Evaluating interfaces for privacy policy rule authoring. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI)*. Montreal, Canada : ACM Press, 2006, S. 83–92

[85] KARP, R.: Reducibility among combinatorial problems. In: MILLER, R. (Hrsg.) ; THATCHER, J. (Hrsg.): *Complexity of Computer Computations*. Plenum Press, 1972, S. 85–103

[86] KILLIJIAN, M.-O. ; CUNNINGHAM, R. ; MEIER, R. ; MAZARE, L. ; CAHILL, V.: Towards Group Communication for Mobile Participants. In: *Proceedings of Principles of Mobile Computing*. Newport, Rhode Island, USA : ACM Press, August 2001, S. 75–82

[87] KJAERGAARD, M. ; TREU, G. ; LINNHOFF-POPIEN, C.: Zone-based RSS Reporting for Location Fingerprinting. In: *Proceedings of the Fifth International Conference on Pervasive Computing (Pervasive)*, Springer, Mai 2007

[88] KOCH, M.: *Community-Unterstützungssysteme - Architektur und Interoperabilität*, TU München, Habilitationsschrift, 2003

[89] KOLLIOS, G. ; GUNOPULOS, D. ; TSOTRAS, V. J.: On Indexing Mobile Objects. In: *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. Philadelphia, Pennsylvania, USA : ACM Press, 1999, S. 261–272

[90] KOLODZIEJ, K. W. ; HJELM, J.: *Local Positioning Systems*. CRC, 2006. – ISBN 0849333490

[91] KORTUEM, G.: Proem: a middleware platform for mobile peer-to-peer computing. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6 (2002), Nr. 4, S. 62–64

[92] KRISHNA, P. ; VAIDYA, N. ; CHATTERJEE, M. ; PRADHAN, D.: A cluster-based approach for routing in dynamic networks. In: *ACM SIGCOMM Computer Communication Review* 27 (1997), Nr. 2, S. 49–64

[93] KUMAR, V. ; DAS, S.: Performance of Dead Reckoning-based Location Service for Mobile Ad Hoc Networks. In: *Wireless Communications and Mobile Computing Journal* 4 (2004), März, Nr. 2, S. 189–202

[94] KÜPPER, A.: *Location–based Services — Fundamentals and Operation*. John Wiley & Sons, 2005. – ISBN 0–470–09231–9

[95] KÜPPER, A.: *Location-based Services. Middleware and Applications*, LMU München, Habilitationsschrift, Juni 2006

[96] KÜPPER, A. ; TREU, G.: From Location to Position Management: User Tracking for Location-based Services. In: *Tagungsband der ITG/GI-Fachtagung Kommunikation in Verteilten Systemen (KiVS)*. Kaiserslautern, Germany, Februar 2005, S. 81–88

[97] KÜPPER, A. ; TREU, G.: Efficient Proximity and Separation Detection among Mobile Targets for Supporting Location-based Community Services. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 10 (2006), Juli, Nr. 3, S. 1–12

[98] KÜPPER, A. ; TREU, G. ; LINNHOFF-POPIEN, C.: TraX: a Device-centric Middleware Framework for Location-based Services. In: *IEEE Communications Magazine* 44 (2006), September, Nr. 9, S. 114–120

[99] KÖLSCH, T. ; FRITSCH, L. ; KOHLWEISS, M. ; KESDOGAN, D.: Mit IDM und Mittler zu mehr Privatsphäre in LBS. In: *2. GI/ITG KuVS Fachgespräch Ortsbezogene Anwendungen und Dienste*. Stuttgart, Deutschland, Juni 2005

[100] LAMARCA, A. ; CHAWATHE, Y. ; CONSOLVO, S. ; HIGHTOWER, J. ; SMITH, I. ; SCOTT, J. ; SOHN, T. ; HOWARD, J. ; HUGHES, J. ; POTTER, F. ; TABERT, J. ; POWLEDGE, P. ; BORRIELLO, G. ; SCHILIT, B.: Place Lab: Device Positioning Using Radio Beacons in the Wild. In: *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive)*. Munich, Germany : Springer, Mai 2005, S. 116–133

[101] LANGHEINRICH, M.: Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems. In: *Proceedings of Ubicomp*. Atlanta, Georgia, USA : Springer, 2001, S. 273–291

[102] LEDERER, S. ; HONG, I. ; DEY, K. ; LANDAY, A.: Personal privacy through understanding and action: five pitfalls for designers. In: *Personal Ubiquitous Computing* 8 (2004), Nr. 6, S. 440–454

[103] LEDERER, S. ; MANKOFF, J. ; DEY, A.: Who wants to know what when? privacy preference determinants in ubiquitous computing. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI)*. Ft. Lauderdale, Florida, USA : ACM Press, 2003, S. 724–725

[104] LEE, H. L. ; BUCKLAND, M. A. ; SHEPHERDSON, J. W.: A Multi-Agent System to Support Location-Based Group Decision Making in Mobile Teams. In: RALPH, Daniel (Hrsg.) ; SEARBY, Stephen (Hrsg.): *Location and Personalisation: Delivering Online and Mobility Services* Bd. 8. London, United Kingdom : The Institution of Engineering and Technology (IET), November 2003, Kapitel 14, S. 197–209

[105] LEONHARDI, A. ; NICU, C. ; ROTHERMEL, K.: A Map-Based Dead-Reckoning Protocol for Updating Location Information. In: *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*. Washington, DC, USA : IEEE Computer Society, 2002, S. 193–200

[106] LEONHARDI, A. ; ROTHERMEL, K.: Protocols for Updating Highly Accurate Location Information. In: BEHCET, A. (Hrsg.): *Geographic Location in the Internet.* Kluwer Academic Publishers, 2002, S. 111–141

[107] LIANG, B. ; HAAS, Z.: Predictive Distance-based Mobility Management for Multidimensional PCS Networks. In: *IEEE/ACM Transactions on Networking (TON)* 11 (2003), Nr. 5, S. 718–732

[108] LINNHOFF-POPIEN, C.: *CORBA, Kommunikation und Management*. Springer, 1998. – ISBN 3540640134

[109] LIU, J. ; SACCHETTI, D. ; SAILHAN, F. ; ISSARNY, V.: Group management for mobile Ad Hoc networks: design, implementation and experiment. In: *Proceedings of the 6th international conference on Mobile Data Management (MDM)*. Ayia Napa, Cyprus : ACM Press, 2005, S. 192–199

[110] MARTENS, J.: *Eine Middleware für proaktive ortsbezogene Community Dienste*, LMU München, Diplomarbeit, 2006

[111] MARTENS, J. ; TREU, G. ; RUPPEL, P. ; WEISS, D. ; KÜPPER, A. ; LINNHOFF-POPIEN, C.: Eine Plattform zur Unterstützung von proaktiven ortsbezogenen Mehrbenutzer-Anwendungen. In: *3. GI/ITG KuVS Fachgespräch Ortsbezogene Anwendungen und Dienste*. Berlin, Germany : Freie Universität Berlin, September 2006

[112] MCGUIRE, M. ; PLATANIOTIS, K. ; VENETSANOPOULOS, A.: Data Fusion of Power and Time Measurements for Mobile Terminal Location. In: *IEEE Transactions on Mobile Computing* 4 (2005), Nr. 2, S. 142–153

[113] MCKEE, L.: LBS Interoperability Through Standards. In: SCHILLER, J. (Hrsg.) ; VOISARD, A. (Hrsg.): *Location-Based Services*. Morgan Kaufmann Publishers, Elsevier, 2004, Kapitel 6, S. 149–172

[114] MCTEAR, M.: Spoken dialogue technology: enabling the conversational user interface. In: *ACM Computing Surveys (CSUR)* 34 (2002), Nr. 1, S. 90–169

[115] MEHROTRA, R. ; GARY, J.: Similar-shape retrieval in shape data management. In: *IEEE Computer* 28 (1995), September, Nr. 2, S. 57–62

[116] MEIER, R.: Communication paradigms for mobile computing. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6 (2002), Oktober, Nr. 4, S. 56–58

[117] MEIER, R. ; CAHILL, V.: Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications. In: *Proceedings of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)*. Paris, France : Springer, 2003, S. 285–296

[118] MEIER, R. ; CAHILL, V.: Location-Aware Event-Based Middleware: A Paradigm for Collaborative Mobile Applications? In: *Proceedings of the 8th CaberNet Radicals Workshop*. Ajaccio, Corsica, Oktober 2003

[119] MERATNIA, N. ; DE BY, R.: Aggregation and comparison of trajectories. In: *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. McLean, Virginia, USA : ACM Press, 2002, S. 49–54

[120] MOURATIDIS, K. ; PAPADIAS, D. ; BAKIRAS, S. ; TAO, Y.: A Threshold-based Algorithm for Continuous Monitoring of k Nearest Neighbors. In: *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), November, Nr. 11, S. 1451–1464

[121] MYLES, G. ; FRIDAY, A. ; DAVIES, N.: Preserving Privacy in Environments with Location-Based Applications. In: *IEEE Pervasive Computing* 2 (2003), Nr. 1, S. 56–64

[122] MYLLYMAKI, J. ; KAUFMAN, J.: High-performance Spatial Indexing for Location-based Services. In: *Proceedings of the 12th international conference on World Wide Web*. Budapest, Hungary : ACM Press, 2003, S. 112–117

[123] NEPPER, P.: *Spoken Dialogue Infrastructure for Location-based Services*, LMU München, Diplomarbeit, 2006

[124] NEPPER, P. ; TREU, G. ; KÜPPER, A.: Adding Speech to Location-based Services. In: *Wireless Personal Communications, Special Issue on Towards Global & Seamless Personal Navigation (to appear)*

[125] NEUKUM, O.: Implementierung Mikroskopischer Bewegungsmuster im LAMA-Framework / LMU München. 2005. – Forschungsbericht

[126] NEUKUM, O.: *Effiziente Clustererkennung mobiler Endgeräte*, LMU München, Diplomarbeit, 2006

[127] NICKLAS, D. ; GROSSMANN, M. ; SCHWARZ, T.: NexusScout: An Advanced Location-Based Application On A Distributed, Open Mediation Platform. In: *Proceedings of the 29th Very Large Databases Conference (VLDB)*. Berlin,Germany : Morgan Kaufmann Publishers, September 2003

[128] NICKLAS, D. ; MITSCHANG, B.: On building location aware applications using an open platform based on the NEXUS Augmented World Model. In: *Software and Systems Modeling, Special section on OOIS 01* 3 (2004), Dezember, Nr. 4, S. 303–313

[129] OPEN GEOSPATIAL CONSORTIUM: *GML - the Geography Markup Language Version 3.1.1*. 2004

[130] OPEN GEOSPATIAL CONSORTIUM: *OpenGIS® Location Service (OpenLS) Implementation Specification: Core Services Version 1.1*. 2005

[131] OPEN MOBILE ALLIANCE (OMA): *OMA Mobile Location Protocol V3.1 Candidate Enabler Specification*. März 2004

[132] OTSASON, V. ; VARSHAVSKY, A. ; MARCA, A. L. ; LARA, E. de: Accurate GSM Indoor Localization. In: *Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp)*. Tokyo, Japan : Springer, September 2005, S. 141–158

[133] PARSONS, D.: Extending the location API for J2ME to support friend finder services. In: *Proceedings of the ACM symposium on Applied computing*. Dijon, France : ACM Press, 2006, S. 992–996

[134] PFITZMANN, A. ; KÖHNTOPP, M.: Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In: *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*. Berkeley, CA, USA : Springer, Juli 2001, S. 1–9

[135] R FOUNDATION FOR STATISTICAL COMPUTING (Hrsg.): *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2006. `http://www.R-project.org`

[136] RAENTO, M. ; OULASVIRTA, A.: Privacy management for social awareness applications. In: *Proceedings of the workshop on Context Awareness for Proactive Systems (CAPS)*, 2005, S. 105–114

[137] RANGANATHAN, A. ; AL-MUHTADI, J. ; CHETAN, S. ; CAMPBELL, R. ; MICKUNAS, M.: MiddleWhere: a middleware for location awareness in ubiquitous computing applications. In: *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Toronto, Canada : Springer, Oktober 2004, S. 397–416

[138] REICHWALD, R. (Hrsg.) ; KRCMAR, H. (Hrsg.) ; SCHLICHTER, J. (Hrsg.) ; BAUMGARTEN, U. (Hrsg.): *Community Services: Lifestyle*. Eul Verlag, 2005. – ISBN 3–89936–403–1

[139] REICHWALD, R. ; NEY, M.: Das Forschungsprojekt COSMOS Lifestyle. In: REICHWALD, R. (Hrsg.) ; KRCMAR, H. (Hrsg.) ; SCHLICHTER, J. (Hrsg.) ; BAUMGARTEN, U. (Hrsg.): *Community Services: Lifestyle*. Eul Verlag, 2005, S. 5–25

[140] RHEINGOLD, H.: *The Virtual Community: Homesteading on the Electronic Frontier*. Perseus Books, 1993. – ISBN 0201608707

[141] ROMAN, M. ; ISLAM, N.: Dynamically programmable and reconfigurable middleware services. In: *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Toronto, Canada : Springer, Oktober 2004, S. 372–396

[142] ROMÁN, M. ; HESS, C. ; CERQUEIRA, R. ; RANGANATHAN, A. ; CAMPBELL, R. ; NAHRSTEDT, K.: Gaia: a middleware platform for active spaces. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6 (2002), Nr. 4, S. 65–67

[143] ROTH, J.: Accessing Location Data in Mobile Environments – the Nimbus Location Model. In: *Proceedings of the Mobile HCI 03 Workshop on Mobile and Ubiquitous Information Access*. Udine, Italy : Springer, September 2003

[144] ROTH, J.: *A Decentralized Location Service Providing Semantic Locations*, Fernuniversität Hagen, Habilitationsschrift, Januar 2005

[145] RUPPEL, P.: *Anonymisierung ortsabhängiger Community Dienste*, LMU München, Diplomarbeit, 2005

[146] RUPPEL, P. ; TREU, G. ; KÜPPER, A. ; LINNHOFF-POPIEN, C.: Anonymous User Tracking for Location-based Community Services. In: *Proceedings of the 2nd International Workshop on Location- and Context-Awareness (LoCA)*. Dublin, Ireland : Springer, Mai 2006

[147] SCHILIT, B. ; LAMARCA, A. ; BORRIELLO, G. ; GRISWOLD, W. ; MCDONALD, D. ; LAZOWSKA, E. ; BALACHANDRAN, A. ; HONG, J. ; IVERSON, V.: Challenge: ubiquitous location-aware computing and the "place lab" initiative. In: *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*. San Diego, CA,USA : ACM Press, September 2003, S. 29–35

[148] SCHNEIDER, S.: Entwicklung eines Tools zur Aufzeichnung von GPS-Traces / LMU München. 2006. – Forschungsbericht

[149] SCHULZRINNE, H. ; TSCHOFENIG, H. ; MORRIS, J. ; CUELLAR, J. ; POLK, J. ; ROSENBERG, J.: *Common Policy: A Document Format for Expressing Privacy Preferences, RFC 4745*

[150] SMITH, I. ; CONSOLVO, S. ; LAMARCA, A. ; HIGHTOWER, J. ; SCOTT, J. ; SOHN, T. ; HUGHES, J. ; IACHELLO, G. ; ABOWD, G.: Social Disclosure Of Place: From

Location Technology to Communications Practices. In: *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive)*. Munich, Germany : Springer, Mai 2005, S. 134–151

[151] SNEKKENES, E.: Concepts for personal location privacy policies. In: *Proceedings of the 3rd ACM conference on Electronic Commerce, Tampa*. Tampa, Florida, USA : ACM Press, Oktober 2001, S. 48–57

[152] SPANIOL, O. ; HOFF, S.: *Ereignisorientierte Simulation Konzepte und Systemrealisierung*. International Thomson Publishing, 1995. – ISBN 3–8266–0122–X

[153] SPANOUDAKIS, M. ; BATISTAKIS, A. ; PRIGGOURIS, I. ; IOANNIDIS, A. ; HADJIEFTHYMIADES, S. ; MERAKOS, L.: Extensible platform for location based services provisioning. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops*, IEEE, Dezember 2003, S. 72–79

[154] STRASSMAN, M. ; COLLIER, C.: Case Study: Development of the Find Friend Application. In: SCHILLER, J. (Hrsg.) ; VOISARD, A. (Hrsg.): *Location-Based Services*. San Francisco, CA, USA : Morgan Kaufmann Publishers, Elsevier, Mai 2004, Kapitel 2, S. 27–39

[155] SWEENEY, L.: k-anonymity: a model for protecting privacy. In: *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10 (2002), Nr. 5, S. 557–570

[156] SØRENSEN, C.-F. ; WU, M. ; SIVAHARAN, T. ; BLAIR, G. ; OKANDA, P. ; FRIDAY, A. ; DURAN-LIMON, H.: A context-aware middleware for applications in mobile Ad Hoc environments. In: *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. Toronto, Canada : ACM Press, 2004, S. 107–110

[157] TANG, J. ; YANKELOVICH, N. ; BEGOLE, J. ; VAN KLEEK, M. ; LI, F. ; BHALODIA, J.: ConNexus to awarenex: extending awareness to mobile users. In: *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*. Seattle, Washington, USA : ACM Press, 2001, S. 221–228

[158] TEUBER, M.: *Konzeption und Implementierung von Datenschutz-Policies für ortsbezogene Community Dienste*, LMU München, Diplomarbeit, 2006

[159] TREU, G. ; FUCHS, F. ; DARGATZ, C.: Implicit Authorization for Accessing Location Data in a Social Context. In: *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES)*. Wien, Austria : IEEE CS, April 2007

[160] TREU, G. ; KÜPPER, A.: Efficient Proximity Detection for Location Based Services. In: *Proceedings of the 2nd Workshop on Positioning, Navigation and Communication 2005 (WPNC)*. Hannover, Germany : SHAKER, März 2005

[161] TREU, G. ; KÜPPER, A. ; RUPPEL, P.: Anonymization in Proactive Location Based Community Services. In: *Advances in Pervasive Computing. Adjunct Proceedings of the 3rd International Conference on Pervasive Computing*. Munich, Germany : Österreichische Computer Gesellschaft, Mai 2005

[162] TREU, G. ; MARTENS, J. ; KÜPPER, A.: TraX — Eine Plattform zur Unterstützung ortsbezogener Community Dienste. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 29 (2006), Februar, Nr. 1, S. 19–24

[163] TREU, G. ; MARTENS, J. ; SCHICKER, M. ; BREISINGER, M. ; KÜPPER, A.: Vs Golf – Developing Location-based Multi-Player Games. In: *Proceedings of the First International Workshop on Engineering Mobile-Based Software and Applications*. Beijing, China : IEEE, Juli 2007

[164] TREU, G. ; WILDER, T. ; KÜPPER, A.: Efficient Proximity Detection among Mobile Targets with Dead Reckoning. In: *Proceedings of the 4-th international workshop on mobility management & wireless access protocols (MobiWac)*. Torremolinos, Malaga, Spain : ACM Press, Oktober 2006, S. 75–83

[165] WILDER, T.: *Effiziente Nachbarschaftserkennung mobiler Endgeräte mit Dead Reckoning*, LMU München, Diplomarbeit, April 2006

[166] WOLFSON, O. ; SISTLA, A. P. ; CHAMBERLAIN, S. ; YESHA, Y.: Updating and Querying Databases that Track Mobile Units. In: *Distributed and Parallel Databases* 7 (1999), Nr. 3, S. 257–387

[167] XU, Z. ; JACOBSEN, H.-A.: Efficient Constraint Processing for Location-aware Computing. In: *Proceedings of the 6th International Conference on Mobile Data Management (MDM)*. Ayia Napa, Cyprus : ACM Press, 2005, S. 3–12

[168] YANAGISAWA, Y. ; AKAHANI, J. ; SATOH, T.: Shape-Based Similarity Query for Trajectory of Mobile Objects. In: *Proceedings of the 4th International Conference on Mobile Data Management (MDM)*. Melbourne, Australia : Springer, Januar 2003, S. 63–77

[169] YU, S. ; AUFAURE, M.-A. ; CULLOT, N. ; SPACCAPIETRA, S.: A Collaborative Framework for Location-Based Services. In: *Short Paper Proceedings of the 15th Conference On Advanced Information Systems Engineering*. Klagenfurt, Austria : CEUR-WS.org, 2003

[170] YU, S. ; AUFAURE, M.-A. ; CULLOT, N. ; SPACCAPIETRA, S.: Location-Based Spatial Modelling Using Ontology. In: *Proceedings of the 6th AGILE conference on Geographic Information Science*. Lyon, France : AGILE, April 2003

[171] ZHU, M. ; LEE, D. L. ; ZHANG, J.: k-Closest Pair Query Monitoring Over Moving Objects. In: *Proceedings of the 7th International Conference on Mobile Data Management (MDM)*. Nara, Japan : IEEE Computer Society, Mai 2006, S. 14–21

[172] ZOU, X. ; RAMAMURTHY, B. ; MAGLIVERAS, S.: *Secure Group Communications over Data Networks*. Springer, 2005. – ISBN 0387229701