Mobile Objekte in einer mobilen Umgebung

Ein Middlewaremodell für Applikationen in mobilen multihop ad-hoc Netzwerken

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
am Fachbereich IV der Universität Trier

vorgelegt von
Diplom-Informatiker Daniel Görgen

August 2006

Zusammenfassung. In dieser Arbeit wird ein Middlewaremodell für Anwendungen in mobilen ad-hoc Netzwerken vorgestellt. Der primäre Fokus liegt dabei auf Netzen, die auf Basis von mobilen Endbenutzergeräten wie PDAs, PocketPCs oder Smartphones gebildet werden und die über eine drahtlose Kommunikationstechnik wie WLAN oder Bluetooth verfügen. Zur Identifizierung der Ansprüche an eine Middleware in diesen Netzen wurden Anwendungen aus dem universitären Umfeld untersucht. Da die Kommunikation in drahtlosen Netzen nicht immer so transparent wie in drahtgebundenen Infrastrukturnetzen erfolgen kann, enthält die Middleware ein Kommunikationsframework, welches die Adaption und auch die Kombination von Kommunikationsmechanismen ermöglicht. Neben der direkten Auslieferung von Nachrichten werden auch Mechanismen zur sukzessiven Weitergabe von Nachrichten, die so genannte Store-and-Forward Kommunikation, angeboten. Hier steht insbesondere die Weitergabe von Informationen im Vorbeigehen (En-Passant) im Vordergrund. Die Komponenten der Middleware und darauf basierenden Applikationen werden als interagierende mobile Objekte modelliert. Die Interaktion dieser Objekte erfolgt aufgrund der Dynamik der mobilen ad-hoc Netze ereignisorientiert. Sowohl die Interaktion, als auch die Adaption und Wahl von Kommunikationsmechanismen erfolgt auf Basis von Informationen über das direkte Umfeld, Applikationswissen über das Netzwerk und Erfahrungen aus der Vergangenheit. Diese Informationen, die von Objekten verbreitet und gesammelt werden können, werden als Spuren bezeichnet. Da eine Aufhebung der Transparenz der Middlewarekomponenten einen höheren Entwicklungsaufwand bedeutet, wird eine Realisierung der Objektinteraktion in Objektrepräsentanten vorgeschlagen. Objektrepräsentanten ermöglichen die Kommunikation mit entfernten Objekten, das lokale Verwalten von Informationen über diese Objekte und schließlich die Repräsentierung von Objekten ohne reale Objektinstanz. Objekte ohne Objektinstanz werden verwendet, um Gerätegruppen mit bestimmten Eigenschaften zusammenzufassen, beispielsweise Objekte an einem bestimmten geographischen Ort. Somit ist auch das Wissen über geographische Orte und deren Identifizierung ein wichtiger Aspekt des Modells. Aufgrund der Anwendungsuntersuchungen ist sukzessive ein Middlewareprototyp entstanden, der in eine Simulationsumgebung für mobile ad-hoc Netze integriert ist. Die Umgebung erlaubt neben der reinen Simulation mobiler ad-hoc Netze und der zu untersuchenden Anwendung auch die Integration realer Geräte und die Ausführung der Anwendungen auf realen Geräten. Somit konnte, neben einer simulativen Untersuchung, auch eine praktische Evaluation der Anwendungen in Feldversuchen durchgeführt werden.

Inhaltsverzeichnis

1

Einleitung

Anhaltender Fortschritt im Bereich der Chipintegration ermöglicht es, dass Computer immer kleiner und leistungsfähiger werden. Schon 1965 stellte Gordon Moore die Vermutung auf, dass sich die Komplexität integrierter Schaltungen alle 18 Monate verdoppelt [83]. Diese Aussage hat bis heute Gültigkeit und wurde in ähnlicher Weise auch für andere Bereiche des technologischen Fortschritts in der Computerindustrie formuliert, beispielsweise in der Speicher- oder der Netzwerktechnologie. Während in früheren Jahren die höhere Integrationsdichte primär zur Leistungssteigerung verwendet wurde, steht seit einigen Jahren die Verkleinerung der Endgeräte im Vordergrund. Damit geht auch ein grundlegender Paradigmenwechsel bezüglich der Computernutzung einher.

Während in den frühen Jahren zentrale Großrechner von vielen Benutzern verwendet wurden, reduzierte sich mit dem Aufkommen des Personal Computers das Computer/Nutzerverhältnis zu einer $1-zu-1$ Beziehung. Diese Entwicklung setzt sich fort, und schon heute ist der Trend hin zu vielen Einzelgeräten pro Benutzer sichtbar. Neben den immer noch vorhandenen Personal Computern interagieren heutige Nutzer mit einer Vielzahl von, in der Regel sehr viel kleineren, Computern, wie sie zum Beispiel in Mobilfunkgeräten, MP3-Playern oder auch Personal Digital Assistants (PDA) enthalten sind. Neben dieser direkten Computerinteraktion existiert auch eine "unsichtbare" Computerinteraktion, da heute in vielen (komplexeren) Alltagsgegenständen Computer integriert sind. So befinden sich in Automobilen eine Vielzahl Kleinstcomputer, die im Verborgenen Aufgaben übernehmen, aber auch in Unterhaltungs- und Haushaltsgeräten wie Fernsehern, DVD-Spielern, Waschmaschinen oder sogar Toastern. Schon 1991 formulierte Marc Weiser die Vision des *Ubiquitous Computing* [109] als logische Konsequenz dieser Entwicklung. Computer stehen Nutzern hierbei zu Hunderten zur Verfügung und sind somit allgegenwärtig. Der primäre Fokus der Benutzeraufmerksamkeit soll dabei nicht wie bisher auf der Bedienung der Geräte liegen, sondern die eigentlich zu erledigende Aufgabe soll im Vordergrund stehen. Der Computer

selbst verschwindet dabei im Hintergrund, Computer werden zu Alltagsgegenständen oder sind Alltagsgegenstände, die um Rechenkapazitäten erweitert werden. Diese Alltagsgegenstände können autonom agieren und mit anderen Gegenständen interagieren.

Ein wichtiger Aspekt in Weisers Vision ist die Möglichkeit der drahtlosen Kommunikation der Geräte untereinander, die eine Interaktion der Geräte ermöglicht. Schon heute besitzt eine Vielzahl mobiler Geräte diese Eigenschaft. Zwar handelt es sich dabei meist um verkleinerte PCs wie Notebooks, PocketPCs oder Smartphones, es werden aber zunehmend auch andere Geräte wie Fotokameras oder MP3-Spieler mit drahtloser Kommunikationstechnik ausgestattet. Eingesetzt werden diese drahtlosen Übertragungstechniken heute meist, um mobile Endgeräte in ein drahtgebundenes Infrastrukturnetz zu integrieren. Zu diesem Zweck werden Zugangspunkte eingerichtet, die mobile Geräte in ihrem Sendebereich drahtlos an das Infrastrukturnetz anbinden. Stand der Technik ist auch die Verwendung der drahtlosen Kommunikation als Kabelersatz. Mobiltelefone können so mit Headsets verbunden und mit PCs abgeglichen werden, oder Fotokameras sprechen Drucker direkt und drahtlos an.

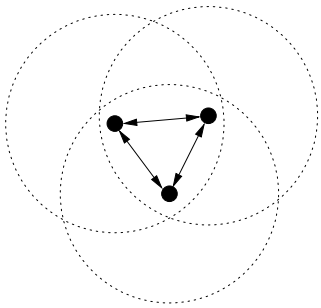


Abb. 1.1. Singlehop Kommunikation: Interaktion erfolgt nur im Sendebereich der drahtlosen Kommunikation.

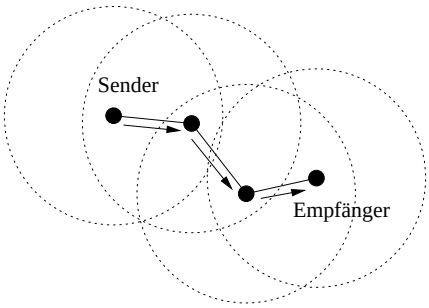


Abb. 1.2. Multihop Kommunikation: Interaktion mit Geräten außerhalb des Sendebereichs erfordert kooperierende Zwischenknoten, die als Router der Kommunikation dienen.

In den letzten Jahren haben so genannte *mobile ad-hoc Netzwerke* (MANet) zunehmend an Interesse in der Forschung gewonnen. Hierbei bilden mobile Geräte auf Basis drahtloser Kommunikation ein Netzwerk, ohne dabei zentrale, drahtgebundene Infrastrukturen verwenden zu müssen. Die Kommunikation zwischen den Geräten des Netzwerks wird also allein von den Geräten des Netzwerks selbst getragen. Unterschieden werden kann hier zwischen *singlehop* und *multihop* ad-hoc Netzwerken. In singlehop Netzen erfolgt eine Kommunikation nur mit Geräten im Sendebereich der drahtlosen Kommunikationstechnik (Abbildung 1.1). Soll aber auch mit Geräten kom-

muniziert werden, die sich nicht im direkten Sendebereich befinden, werden kooperierende Zwischenknoten benötigt, die Nachrichten an andere Knoten weiterleiten (Abbildung 1.2) und somit ein multihop Netzwerk bilden. Wenn diese Zwischenknoten nicht auch Ziel der Kommunikation sind, müssen sie die Nachrichten in altruistischer Weise weiterleiten.

Mobile ad-hoc Netzwerke können aus tragbaren Endbenutzergeräten bestehen, die über traditionelle Benutzerschnittstellen verfügen. Schon heute sind tragbare Kleincomputer wie Taschencomputer, PDAs, Subnotebooks oder Smartphones weit verbreitet und können drahtlos kommunizieren – in der Regel über WLAN [38] oder Bluetooth [82]. Unter der Annahme, dass es möglich ist, diese Geräte nahezu "always on" zu betreiben, lassen diese Geräte die Etablierung mobiler ad-hoc Netzwerke zu, in denen die Geräte auch ohne expliziten Nutzerauftrag oder momentane Nutzerinteraktion autonom interagieren können. Mobile ad-hoc Netze sind geprägt von einer extremen Dynamik, da die Nutzermobilität direkten Einfluss auf die Netztopologie besitzt und darüber hinaus Geräte das Netz spontan betreten oder verlassen können. Gerade in großen Netzen kann die Dynamik zu unvorhersehbaren Netzwerkpartitionen führen, so dass nur ein Teil des Netzes direkt erreichbar ist. Nahezu zufällige Gerätenachbarn, das Auftauchen bisher unbekannter Geräte sowie das spontane und langfristige Wegfallen von Interaktionspartnern prägen die Kommunikation in diesen Netzen. Die Topologie des Netzes kann sich also permanent ändern, so dass traditionelle Routingverfahren dort nicht oder nur sehr bedingt einsetzbar sind.

1.1 Ziele der Arbeit

Ziel der vorliegenden Arbeit ist der Entwurf einer Middleware, die speziell für Anwendungen auf Basis mobiler ad-hoc Netzwerke geeignet ist und die Dynamik dieser Netze im Modell berücksichtigt. Der primäre Fokus liegt dabei auf Netzen, die von mobilen Endbenutzergeräten gebildet werden. Aufgrund der Neuartigkeit dieser Netze wurde für die Entwicklung des Modells ein Topdown-Ansatz gewählt, um zunächst die Anforderungen an eine Middleware zu identifizieren. Zu diesem Zweck wurden Anwendungsprototypen für mobile ad-hoc Netze entworfen und untersucht. Darauf aufbauend wurden Selbstorganisations- und Kommunikationsmuster identifiziert, die in ein Middlewaremodell integriert oder zumindest von diesem unterstützt werden können. Aufgrund der Applikationsuntersuchungen wurden folgende Feststellungen gemacht, die in die Modellierung der vorgestellten Middlewareplattform eingeflossen sind:

- **Ortsinformation als Hilfe zur Interaktion und Kommunikation**

  Abhängig vom Aufenthaltsort des Gerätes können die Interaktionsmöglichkeiten einer Applikation unterschiedlich sein. Daher kann Wissen über den

Ort es ermöglichen, dass sich Applikationen an unterschiedliche Ortsgegebenheiten anpassen. Ortsinformation kann auch als Kommunikationshilfe verwendet werden indem Wissen über Aufenthaltsorte von Kommunikationszielen zu Routingzwecken verwendet werden. Nachrichten können dabei zuerst an den Aufenthaltsort gesendet und erst dort das eigentliche Nachrichtenziel gesucht werden. Aus diesem Grund sind Ortsinformationen ein wichtiger Bestandteil der vorgestellten Middleware.

- **Notwendigkeit neuer Kommunikationsansätze**

  Eine Kommunikation mit beliebigen Geräten im Netz ist aufgrund der aufwendigen Zielsuche [89] und auch aufgrund der verbleibenden Bandbreite pro Gerät [69] in Netzen mit vielen Geräten nicht tragfähig. Die Verwendung von Ortsinformationen kann diese Zielsuche drastisch reduzieren. Diese Art der Kommunikation wird in Infrastrukturnetzen bisher nicht eingesetzt. Um Nachrichtenziele auch trotz der möglichen Netzpartitionen in mobilen ad-hoc Netzen zu erreichen, kann es notwendig sein, Nachrichten auf Geräten zwischenzuspeichern, um die Auslieferung später fortzusetzen (so genannte *Store-and-Forward* Kommunikation). Bei sehr hoher Netzdynamik, kann häufig nur mit direkten Nachbarn interagiert werden. In Kombination mit Store-and-Forward Kommunikation können hier Nachrichten im "Vorbeigehen" (*En-Passant*) ausgetauscht werden, damit diese ihre eigentlichen Nachrichtenziele erreichen. Neben Routingverfahren, die Nachrichten direkt zum Ziel ausliefern, stehen also noch weitere, indirekte Kommunikationsmechanismen in mobilen ad-hoc Netzwerken zur Auswahl. Aufgrund der geringen Bandbreite, die in diesen Netzen für jedes einzelne Gerät zur Verfügung steht, ist eine *Individualkommunikation* nur bedingt möglich. Daher werden vermehrt Verfahren benötigt, die eine Gruppen- oder *Communitykommunikation* realisieren. Eine Middleware muss also nicht nur neue Kommunikationsverfahren anbieten, sondern auch die Integration neuer Verfahren ermöglichen.

- **Einsatz mehrerer Kommunikationsansätze**

  Aufgrund der Vielzahl der möglichen Kommunikationsansätze mit unterschiedlichen Eigenschaften ist es sinnvoll, in mobilen ad-hoc Netzwerken nicht nur ein einzelnes Kommunikationsverfahren einzusetzen. Je nach aktueller Anwendungsanforderung und nach erwarteter oder vorhandener Netzsituation können unterschiedliche Verfahren sinnvoll sein. Diese können auch gleichzeitig betrieben werden, wenn die Applikation unterschiedliche Interaktions- und Nachrichtenverbreitungsmöglichkeiten auch gleichzeitig benötigt. Somit ist es notwendig, dass dem Applikationsentwickler eine Kollektion von Kommunikationsverfahren zur Verfügung steht, aus dieser er die benötigten Verfahren auswählen kann.

- **Kombinier- und Adaptierbarkeit von Kommunikationsansätzen auf Nachrichtenebene**

  Da jedes der möglichen Kommunikationsansätze Vor- und Nachteile hat, können Verfahren auch miteinander kombiniert werden, um die jeweiligen Vorteile zu nutzen. So kann auf Store-and-Forward Verfahren gewechselt werden, falls die direkte Auslieferung einer Nachricht zu ihren Zielen nicht möglich ist. Weitere Beispiele sind im Falle der Unicastkommunikation die Verbindung von positionsbasiertem Greedyrouting mit einer Recoverystrategie im Fehlerfall [9] oder die Verbindung von positionsbasiertem mit linkbasiertem Routing [8]. Da es notwendig ist, Verfahren im Falle einer Kombination mit anderen Verfahren zu adaptieren[1] werden, müssen entweder für jede Kombination spezialisierte oder adaptierbare generische Varianten realisiert werden. Wenn ein Verfahren mit unterschiedlichen Konfigurationen verwendet werden kann, muss für jede zu kommunizierende Nachricht diese Konfiguration entsprechend adaptiert werden. Insbesondere aufgrund der Wiederverwendbarkeit von Verfahren ist also eine Kombinier- und Adaptierbarkeit von Verfahren auf Nachrichtenebene ideal.

- **Applikationswissen über die Netzumgebung**

  Das meist völlige Fehlen zentraler Infrastrukturen und die nahezu unmögliche Administrierbarkeit dieser Netze machen es notwendig, dass diese Netze eine hohes Maß an *Selbstorganisation* besitzen. Entscheidungen, die ein Gerät selbst betreffen, für sein direktes Umfeld wichtig sind oder gar für das gesamte Netz Relevanz besitzen, müssen auf Basis von lokalem Wissen und lokaler Interaktionen mit anderen erfolgen und können in der Regel nicht auf einer globalen Sicht basieren. Der Aufwand, diese Sicht zu erstellen, übersteigt meist die Möglichkeiten des Netzes oder steht in keiner Relation zum erzielten Nutzen für das Gesamtnetz.

  Sobald Geräte in der Lage sind, sich im Raum einzuordnen, also Orte wieder zu erkennen, stehen für rein auf lokalem Wissen basierenden Verfahren weitere Entscheidungskriterien zur Verfügung. Somit können im Idealfall, auch aufgrund von Erfahrungen aus der Vergangenheit, Annahmen über Orte getroffen werden, die als Entscheidungshilfe zur Selbstorganisation dienen können. Neben Erfahrungen kann auch zusätzliches Applikationswissen über die mögliche Netzumgebung verwendet werden. Das Wissen, das an einem bestimmten Ort ein Meeting oder eine Vorlesung stattfindet, kann in gewissem Maße Rückschlüsse auf die Stabilität oder die Dichte des Netzes erlauben. Annahmen können in beiden Fällen zum einen die Eigenschaften des Netzes, zum anderen zu erwartende Dienste oder Geräte an einem bestimmten Ort betreffen. Diese Annahmen können es Applikatio-

[1] Adaption bedeutet in diesem Zusammenhang die Änderung von Konfigurationsparametern der Verfahren oder auch von allgemeineren, eigentlich verfahrensunabhängigen Parametern wie die Beschränkung der Verfahren auf einen Ort oder die Reduzierung der maximalen Hopzahl.

nen erlauben, Kommunikationsverfahren zu selektieren und zu adaptieren. Sie können auch genutzt werden, um zu entscheiden, ob eine Interaktion mit anderen Geräten erfolgen kann.

- **Geringere Transparenz der Middlewareschichten**

  Da eine Vielzahl, zum Teil ähnlicher Kommunikationsverfahren zur Verfügung stehen, benötigt ein Applikationsentwickler zur Wahl und Anpassung der Verfahren ein detaillierteres Wissen über die Eigenschaften der einzelnen Verfahren. Durch Verwendung von Middlewarekomponenten, die spezielle Kommunikationsmuster zur Verfügung stellen, kann die Transparenz auf Anwendungsebene in gewissem Maße wiederhergestellt werden. Dennoch ist ein detaillierteres Wissen des Entwicklers über diese Kommunikationskomponenten und die Auswirkungen der verwendeten Verfahren notwendig. Zum Erstellen von Kommunikationskomponenten, die neue multimodale Kommunikationsverfahren anbieten, ist ein umfangreicherer Zugriff auf die verwendeten Kommunikationsverfahren notwendig, der nicht immer transparent erfolgen kann.

  Dass in mobilen ad-hoc Netzwerken die Protokollschichten nicht mehr völlig transparent aufeinander basieren können, wurde in der Literatur schon vielfach beschrieben [72, 48, 98, 99]. Bisher beschränkt sich diese Aufhebung der Transparenz aber meist nur auf die unteren Schichten. Die vorgestellte Middleware erlaubt diese Aufhebung zusätzlich auch auf Applikationsebene.

- **Modellierung der Middleware und der Applikationen als mobile Objekte**

  Es wurde ein komponentenbasierter Ansatz gewählt, wobei Komponenten als mobile Objekte (MobileObjects) bezeichnet werden. Es werden aber nicht nur Applikationen als Komponenten realisiert, sondern auch die Bestandteile der Middlewareplattform selbst. Insbesondere werden auch Kommunikationsverfahren als Komponenten realisiert. Dies begründet sich in der einfacheren Austauschbarkeit und Kombinierbarkeit von Kommunikationsmechanismen. Organisationskonzepte wie Gruppen, Gerätemengen an identifizierbaren Orten oder die Menge der direkten Nachbarn werden mittels lokaler Komponenten auf einem Gerät repräsentiert, falls das Gerät Teil eines Organisationskonzeptes ist oder damit interagieren will. Zur Interaktion werden von diesen Repräsentantenobjekten Kommunikationsmechanismen zur Verfügung gestellt. Somit kann eine Interaktion mit entfernten Objekten durch eine Interaktion mit lokalen Objekten ermöglicht werden.

- **Ereignisbasierte Interaktion und Programmierung**

  Die hohe Dynamik mobiler ad-hoc Netzwerke führt zu permanenten Änderungen der Netztopologie. Auch die direkte Nachbarschaft eines Gerätes ist permanenten Änderungen unterlegen, so dass spontan neue Interaktions-

partner in direkter Kommunikationsreichweite auftauchen können. Soll mit spontan benachbarten Geräten interagiert werden oder sollen Änderungen in der direkten Umgebung sofort Änderungen in auf lokalem Wissen basierenden Verfahren zur Folge haben, ist der ereignisbasierte Ansatz ideal. Hiermit ist es möglich, die einzelnen Reaktionen auf Umgebungsereignisse direkt zu beschreiben.

Auch entferntere Kommunikationsziele können spontan auftauchen und wieder verschwinden. Somit ist in dieser Umgebung auch die nachrichtenorientierte Kommunikation besser geeignet als eine verbindungsorientierte, da häufig keine längerfristigen Verbindungen zu anderen Geräten aufrechterhalten werden können. Auch eine direkte Antwort auf eine Nachricht ist nicht immer sofort möglich.

Die ereignisbasierte Interaktion und Programmierung entspricht also der stark ereignisbasierten Umgebung eines Gerätes in einem mobilen ad-hoc Netzwerk.

Das in dieser Arbeit entworfene Middlewaremodell bildet Middleware- und Anwendungskomponenten als interagierende mobile Objekte ab, die über das drahtlose Kommunikationsmedium und auch lokal auf den Geräten des Netzes ereignisbasiert kommunizieren. Für die Kommunikation wird ein Kommunikationsframework zur Verfügung gestellt, das es ermöglicht, je nach Anwendungsansprüchen und dabei zu erwartender Netzsituation, verschiedene Kommunikationsverfahren einzusetzen, zu kombinieren und zu adaptieren. Hierbei steht, neben einem Routingframework zur direkten Auslieferung von Nachrichten, auch ein Framework zur Realisierung von Store-and-Forward Kommunikation zur Verfügung. Das Store-and-Forward Framework ermöglicht die Realisierung von Verfahren, die eine Weiterleitung von Nachrichten aufgrund von Änderungen in der Geräteumgebung durchführen. Objektnachbarschaft und Objektumgebung sind die Grundlage zur Objektinteraktion und zur Adaption der Kommunikation. Mobilen Objekten können auch Erfahrungen der Vergangenheit und Wissen über die Zukunft helfen, Entscheidungen in der Gegenwart zu treffen. Dieses Wissen wird als *Spur* bezeichnet. Mobile Objekte, die sich nicht lokal auf dem Gerät befinden oder einem Kommunikations- bzw. Organisationsmuster entsprechen, können durch lokale Repräsentanten dargestellt werden. Diese werden zur Interaktion mit dem eigentlichen Objekt und zum Sammeln von Informationen über das repräsentierte Objekt eingesetzt.

1.2 Aufbau der Arbeit

Das folgende Kapitel 2 gibt einen Überblick über Kommunikationsmechanismen in mobilen ad-hoc Netzwerken. Hierbei werden neben der direkten

Kommunikation auch indirekte Kommunikations- und Informationsverbreitungsformen vorgestellt, die in diesen Netzen von großer Wichtigkeit sind.

Um Anwendungen in einem mobilen ad-hoc Netzwerk zu etablieren, müssen diese die Eigenschaften des Netzes kennen. Meist besitzt die Anwendung selbst ein detaillierteres Wissen über das Netz als es der Middleware möglich ist. Damit kann sie Informationen zur Verfügung stellen, um Middlewarekomponenten zu adaptieren oder sie kann diese Adaption selbst durchführen. Kapitel 3 gibt einen kurzen Überblick über die wichtigsten Eigenschaften dieser Netze.

Die identifizierten Objekttypen, die im vorgestellten Middlewaremodell integriert sind, und deren Eigenschaften werden in Kapitel 4 vorgestellt. Auf das Sammeln, Verbreiten und Darstellen von Spuren wird in Kapitel 5, auf die Repräsentierung von mobilen Objekten in Kapitel 6 eingegangen.

Die Komplexität dieser Netze, insbesondere aber auch die Schwierigkeit ihrer experimentellen Validierung, erfordert die Verwendung einer simulierten Umgebung. Somit wird als Ausgangsbasis der Applikations- und Middlewareuntersuchungen eine Workbench für mobile Anwendungen gewählt (Kapitel 7), die zum einen eine simulative Untersuchung von Anwendungen erlaubt, zum anderen aber auch die Realisierung von Feldversuchen mit diesen Netzen unterstützt. Auf Basis der Workbench wurde ein Middlewareprototyp realisiert und sukzessive erweitert. Dieser bietet Unterstützung zur Realisierung mobiler Objekte, ein adaptierbares und konfigurierbares Kommunikationsframework und weitere Middlewarebasisdienste. Der Middlewareprototyp wird in Kapitel 8 vorgestellt.

Während im Vorfeld die Applikationsuntersuchungen zielführend zum Middlewareentwurf und der Prototyperweiterung eingesetzt wurden, stellt die in Kapitel 9 vorgestellte Applikation *DistScript* eine Realisierung dar, die auf dem aktuellen Stand der Middleware basiert und als "Proof-of-Concept" realisiert wurde. Neben dem vorgestellten Middlewaremodell existiert eine Vielzahl weiterer spezialisierter Middlewareplattformen für mobile ad-hoc Netze. Eine Auswahl dieser Plattformen wird in Kapitel 10 vorgestellt. In Kapitel 11 schließt die Arbeit mit einer Diskussion ab.

1.1 Erklärungen zu den Boxen

Im Laufe der Arbeit finden sich im Text Boxen, die weiterführende Informationen zum Text liefern. Hierbei handelt es sich um Praxisbeispiele, Definitionen und Fakten, Zusammenfassungen sowie Ergebnisse durchgeführter Simulationen. Um die unterschiedlichen Funktionen der Boxen unterscheiden zu können, werden unterschiedliche Boxenstile verwendet:

Beispiele
Die meisten Beispiele stammen aus Applikationsprototypen, die im Rahmen der Arbeit entstanden sind, oder es handelt sich um darauf basierende, weiterführende Überlegungen.

Definitionen und Fakten
In diesen Boxen werden im Text erwähnte Bergriffe genauer definiert oder weitere Fakten zum Text gegeben.

Simulationen
Einige der im Text vorgestellten Mechanismen wurden simulativ genauer untersucht. In diesen Boxen werden die Ergebnisse der Untersuchungen vorgestellt und interpretiert.

Zusammenfassungen
In diesen Boxen werden im Text identifizierte Punkte zusammengefasst aufgelistet.

2

Kommunikation in mobilen ad-hoc Netzwerken

In diesem Kapitel wird ein Überblick von Techniken und Protokollen zur Kommunikation in mobilen ad-hoc Netzwerken gegeben. Neben Verfahren aus der Literatur werden auch einige, in der hier dargestellten Ausprägung bisher noch nicht beschriebene Verfahren vorgestellt. Unterschieden wird im Folgenden zwischen lokaler singlehop Kommunikation, Verfahren zur multihop Kommunikation und der multihop Verbreitung der Nachrichten aufgrund lokaler Interaktion. Die Verbreitung aufgrund lokaler Interaktion benötigt nicht notwendigerweise ein zusammenhängendes Netz, da Store-and-Forward Kommunikation zum Einsatz kommt. Hierbei können Nachrichten auch für längere Zeit auf Geräten zwischengespeichert werden, bevor sie weitergeleitet bzw. weiterpropagiert werden.

2.1 Lokale singlehop Kommunikation

Aufgrund der beschränkten Kommunikationsreichweite können drahtlos kommunizierende Geräte nur benachbarte Geräte direkt erreichen. Als direkt benachbart zu einem Gerät bezeichnet man alle Geräte, die dessen Signale empfangen und interpretieren können. Häufig wird aber darüber hinaus eine gegenseitige Erreichbarkeit gefordert, damit Kommunikationszuverlässigkeit mittels Bestätigungsnachrichten erreicht werden kann. Bestätigungsnachrichten sind aufgrund der drahtlosen Übertragung notwendig, da diese erheblich störanfälliger ist als eine drahtgebundene Übertragung. Darüber hinaus treten neben externen Störfaktoren auch Koordinationsprobleme auf, die so in drahtgebundenen Netzen nicht existieren. Während in drahtgebundenen Netzen Nachrichtenkollisionen und Fehler einfach erkannt werden können, da der Sender einer Nachricht die korrekte Übertragung auf das Medium überprüfen kann, ist diese Kontrolle in drahtlosen Netzen nicht realisierbar. Zwar ist es möglich, ein Gerät so auszustatten, dass es gleichzeitig senden und empfangen

kann[1], zum Erkennen von allen Nachrichtenkollisionen reicht dies aber nicht aus. Wie in Abbildung 2.1 dargestellt, kann der Sender nicht immer erkennen, wenn ein anderes Gerät seine Kommunikation stört (Hidden-Terminal-Problem). Darüber hinaus scheitern beim Hidden-Terminal-Problem auch Carrier-Sense-Mechanismen vor dem Sendern. Es kann also nicht überprüft werden, ob das Medium frei ist. Deshalb werden in vielen Protokollen schon auf MAC[2]-Ebene Bestätigungsnachrichten verwendet, um Kommunikationsfehler erkennen zu können. Das IEEE802.11 Protokoll kann zusätzlich das Request-To-Send/Clear-To-Send (RTS/CTS) Schema einsetzen. Hierbei wird vor einem Unicast eine RTS Nachricht versendet, welche vom adressierten Unicast Empfänger mit einem CTS bestätigt werden muss. Die Nachrichten (und die folgenden Daten- und Bestätigungsnachrichten) beinhalten die Übertragszeit der unmittelbar folgenden Nachrichten. Somit können alle Geräte im

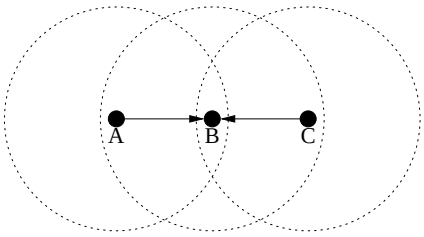


Abb. 2.1. Das Hidden-Terminal-Problem. A und C senden gleichzeitig an B, da A nicht die Übertragung von B hört und umgekehrt. B empfängt keine oder nur eine der beiden Übertragungen.

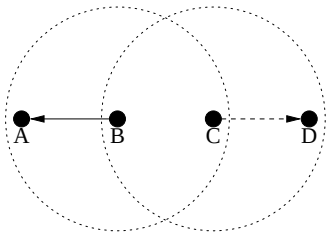


Abb. 2.2. Das Exposed-Terminal-Problem. B sendet an A. C kann aufgrund eines Carrier-Sense-Mechanismus nicht an D senden, obwohl die Übertragungen sich nicht gegenseitig stören.

Kommunikationsbereich des Senders und des Empfängers das Medium für die Übertragungszeit als reserviert markieren (virtuelles Carrier-Sense) und damit das Hidden-Terminal-Problem vermeiden. Ein solches Vorgehen verschärft aber das Exposed-Terminal-Problem (siehe Abbildung 2.2), da auch die Nachbarn des Empfängers blockiert werden. Insbesondere kann bei der lokalen Unicastkommunikation der Fall auftreten, dass Geräte zwar kollisionsfrei senden könnten, sie sich aber unnötig gegenseitig blockieren. Ein Beispiel hierfür ist die stromorientierte multihop Kommunikation, bei der sich aufeinander folgende Nachrichten gegenseitig blockieren können. Daher wird in multihop ad-hoc Netzen häufig auf RTS/CTS Mechanismen verzichtet. Bluetooth löst das Hidden-Terminal-Problem mittels Zeitsynchronisation, wobei den Geräten

[1] In der Praxis wird dies zum einen aus Kostengründen zum anderen aber auch aus technischen Gründen selten realisiert. Aufgrund der Miniaturisierung liegen Sende- und Empfangseinheit sehr nahe beieinander.

[2] Media Access Control

die Rollen *Master* bzw. *Slave* zugeordnet werden. Die Menge der Geräte im Sendebereich eines Masters können zu einem *Pikonetz* zusammengefasst werden. Ein Master teilt den Slaves in seinem Pikonetz Kommunikationszeitscheiben zu, indem er die Knoten explizit anfragt. Mehrere Pikonetze können zu multihop Netzen, so genannten *Scatternetzen*, zusammengefasst werden. Das Hidden-Terminal-Problem tritt dann auf, wenn ein Slave in zwei Pikonetzen enthalten ist und eine Masteranfrage an den Slave erfolgt während der Slave in dem anderen Pikonetz kommuniziert. Reduziert wird die Wahrscheinlichkeit einer Kollision durch unterschiedliche Frequenzsprungmuster in verschiedenen Pikonetzen. Wenn sich zwei Master in Kommunikationsreichweite befinden, kann es auch in Scatternetzen zum Exposed-Terminal-Problem kommen.

In der Praxis können Sendereichweiten vor allem in heterogenen Netzen stark variieren. Variationen treten auch dann auf, wenn die Sendestärke zur effizienteren Unicastkommunikation angepasst wird, um ein bestimmtes Gerät mit größtmöglicher Bandbreite bei gleichzeitig geringstmöglichen Energieverbrauch anzusprechen. In beiden Fällen können Carrier-Sense-Mechanismen fehlschlagen, da auch der Empfänger den störenden Sender unter Umständen nicht erreichen kann bzw. die Kommunikation von anderen nicht bemerkt wird (siehe z.B. Abbildung 2.3). Bei Unicastkommunikation ist dies meist unkritisch, da es möglich ist, das stärkste empfangene Signal herauszufiltern und trotz Störung durch andere Kommunikation den Unicast erfolgreich zu empfangen (*Capture-Effect*). Für die Broadcastkommunikation gilt dies zwar auch, dennoch gehen Nachrichten bei möglichen Empfängern unwiederbringlich verloren, wenn ein anderes Signal stärker ist. Unterschiedliche Sendestärken bei Sender und Empfänger kann eine zuverlässige Kommunikation verhindern, wenn der Empfänger die Nachrichten nicht bestätigen kann.

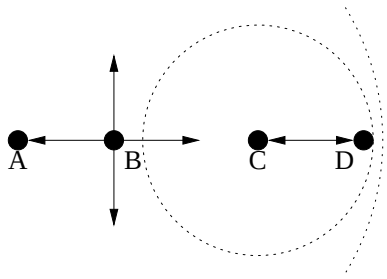


Abb. 2.3. Unterschiedliche bzw. variable Sendestärken sind problematisch. B hört die Kommunikation zwischen C und D nicht, der Broadcast von B kann so nicht von C und D empfangen werden. Darüber hinaus kann die Kommunikation zwischen C und D wiederholt gestört werden.

Das größte Problem ist aber die hohe Dynamik des Netzes, die es mit sich bringt, dass Geräte jederzeit während der Kommunikation verschwin-

den können, d.h. der Kommunikationslink zwischen zwei Geräten jederzeit abbrechen kann. Kommt es zum Abbruch aufgrund von Mobilität, ist meist undefiniert, wann dieser Kommunikationslink wieder hergestellt wird, bzw. ob dies überhaupt passiert. Somit kann es passieren, dass nie eine konsistente Sicht über Erfolg bzw. Misserfolg einer Übertragung erreicht wird. Es kann für jedes Protokoll ein Fall konstruiert werden, bei dem Sender und Empfänger unterschiedlich bzgl. Erfolg oder Misserfolg entscheiden, da der Kommunikationsabbruch jederzeit erfolgen kann. Dies liegt darin begründet, dass die letzte Protokollnachricht unbestätigt bleibt und somit unklar ist, ob die Nachricht erfolgreich ausgeliefert wurde. Der Verlust ist zwar für sehr kurze Protokollnachrichten sehr unwahrscheinlich aber dennoch möglich.

Aufgrund der Störanfälligkeit des Mediums ist es also sinnvoll, schon in der MAC-Schicht die Zuverlässigkeit einer Übertragung durch Bestätigungsnachrichten zu erhöhen. Dies ist bei Unicastnachrichten einfach realisierbar, jedoch bei der Versendung eines Broadcasts nicht möglich, da hier die Empfängermenge unbekannt ist. Unicastkommunikation nutzt aber die Broadcasteigenschaft des Mediums nicht aus. Wenn beispielsweise dieselbe Nachricht zuverlässig an zwei Nachbarn versendet werden soll, wird das Datenpaket bei der Verwendung von Unicasts zweimal übertragen, obwohl beide Geräte diese Nachricht schon bei der ersten Übertragung empfangen könnten. Somit ist es sinnvoll, zwischen einem zuverlässigen Unicast und einem unzuverlässigen Broadcast weitere Kommunikationsprimitive zu definieren, die zum einen die Broadcasteigenschaft des Mediums ausnutzen, zum anderen aber eine höhere Zuverlässigkeit besitzen. Im Folgenden werden die wichtigsten lokalen Kommunikationsprimitiven vorgestellt.

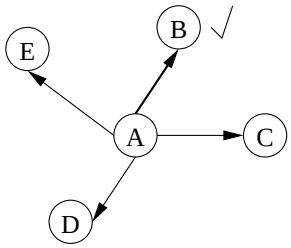


Abb. 2.4. Einfacher Unicast: A sendet an B. B erhält die Nachricht wenn sie korrekt empfangen wurde, alle anderen verwerfen sie.

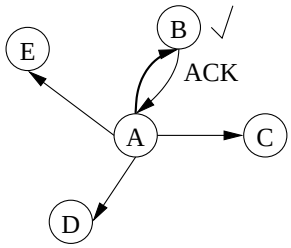


Abb. 2.5. Zuverlässiger Unicast: A sendet an B. B erhält die Nachricht und muss sie bestätigen, alle anderen verwerfen sie.

- *Einfacher Unicast* (Abbildung 2.4)
  Die Nachricht wird an ein bestimmtes Gerät in der Nachbarschaft ausgeliefert. Dafür ist keine gegenseitige Erreichbarkeit notwendig, da der Emp-

fang der Nachricht nicht bestätigt wird. Der Absender besitzt kein Wissen über den Erfolg der Übertragung.

- *Zuverlässiger Unicast* (Abbildung 2.5)
  Dieser Unicast wird durch ein oder mehrere Bestätigungsnachrichten zuverlässiger. Der Sender kann feststellen, ob die Nachricht angekommen ist (alle Bestätigungsnachrichten empfangen), ob sie verloren gegangen ist (trotz Wiederholen bleiben Bestätigungsnachrichten aus) oder ob keine Entscheidung über den Erfolg der Übertragung getroffen werden kann (die letzte Bestätigungsnachricht bleibt aus; wenn sie aufgrund eines Linkabbruchs verloren gegangen ist, hat der Empfänger dennoch die Nachricht erfolgreich empfangen). IEEE802.11 teilt eine Nachricht in einzelne Frames fester Größe auf, die einzeln bestätigt werden. Da dieser Unicasttyp am häufigsten Verwendung findet, wird er im Folgenden als Unicast bezeichnet.

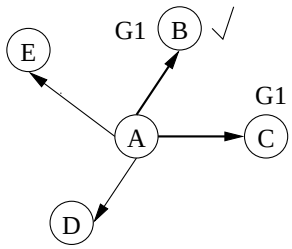


Abb. 2.6. Einfacher Multicast: A sendet an Gruppe G1. Alle aus Gruppe G1 (B und C) empfangen, alle anderen verwerfen die Nachricht. C hat die Nachricht nicht korrekt empfangen.

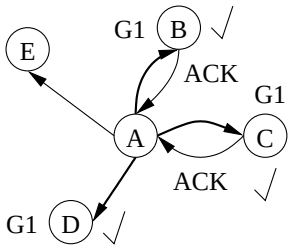


Abb. 2.7. Adressierter Multicast: A sendet an Gruppe G1 und erwartet Bestätigungen von B und C. D ist auch Teil der Gruppe und empfängt die Nachricht ohne Bestätigung. E verwirft die Nachricht.

- *Einfacher Multicast* (Abbildung 2.6)
  Adressiert wird eine bekannte Gruppenkennung (Multicastgruppe). Nachrichten werden nur an Geräte ausgeliefert, die Teil der Gruppe sind. Der Absender benötigt dafür kein Wissen über die Empfänger, insbesondere keine Geräteadressen oder die Anzahl der Empfänger. Der Empfang der Nachricht ist allerdings nicht gesichert, d.h. trotz Gruppenmitgliedern in der Nachbarschaft kann es passieren, das kein Gerät die Nachricht empfängt. Der Sender besitzt keinerlei Wissen über den Erfolg der Nachrichtenübertragung.
- *Adressierter Multicast* (Abbildung 2.7)
  Adressiert wird wie im einfachen Multicast eine bekannte Gruppenkennung. Ausgeliefert wird die Nachricht an alle Mitglieder der Gruppe in der Nachbarschaft und wird von einer Teilmenge der Empfänger bestätigt.

Diese Teilmenge wird vom Sender der Nachricht bestimmt und mitgesendet.

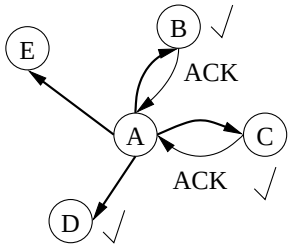


Abb. 2.8. Adressierter Broadcast: A sendet an B und C. B und C bestätigen den Empfang, D und E dürfen die Nachricht empfangen, aber in diesem Fall empfängt nur D sie korrekt.

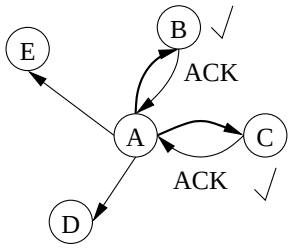


Abb. 2.9. Adressierter Manycast: A sendet an B und C. B und C bestätigen den Empfang, alle anderen verwerfen die Nachricht.

- *Adressierter Broadcast* (Abbildung 2.8)
  Wie im Falle des adressierten Manycast werden Geräte vorgegeben, die den Empfang der Nachricht bestätigen. Die Nachricht wird aber wie bei einem Broadcast von allen Geräten im Sendebereich empfangen und verarbeitet. Im Gegensatz zum adressierten Manycast richtet sich dies Nachricht also auch an Geräte, die nicht in der mitgelieferten Adressliste enthalten sind.
- *Adressierter Manycast* (Abbildung 2.9)
  Der Sender adressiert mehrere Empfänger durch Angabe einer Empfängerliste. Alle Empfänger bestätigen diese Nachricht. Das Datenpaket wird nur an die angegebenen Empfänger ausgeliefert. Ein zuverlässiger Unicast ist ein Spezialfall des Adressierten Manycasts, da dieser allerdings am häufigsten verwendet wird ist, er gesondert aufgeführt.

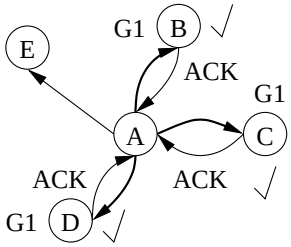


Abb. 2.10. Manycast: A sendet an Gruppe G1 und erwartet von n Bestätigungen. E verwirft die Nachricht da es nicht Teil der Gruppe ist.

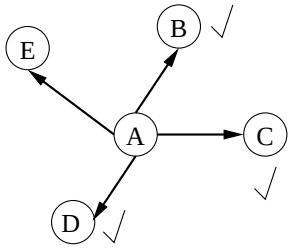
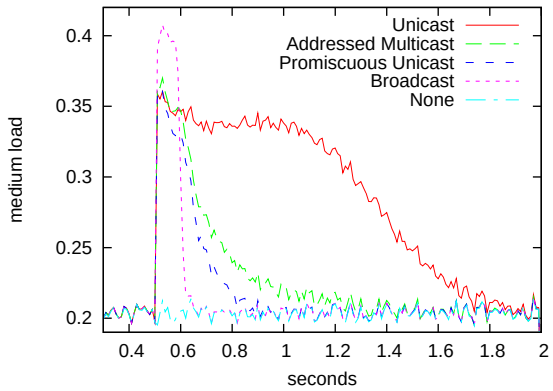


Abb. 2.11. Broadcast: A sendet an alle Nachbarn. E hat die Nachricht nicht korrekt empfangen.

- *Manycast* (Abbildung 2.10)
  Wie der einfache Multicast, nur wird hier die Zahl der Empfänger vorgegeben, welche die Nachricht bestätigen sollen. Alle Empfänger aus der Multicastgruppe bestätigen den Empfang. Wenn nicht genügend Bestätigungen erhalten wurden, wird die Nachricht wiederholt gesendet. Problematisch ist, wenn in der Nachbarschaft weniger Gruppenmitglieder existieren als die vorgegebene Zahl der Empfänger.
- *Broadcast* (Abbildung 2.11)
  Die Nachricht wird an alle Geräte im Sendebereich ausgeliefert, die diese Nachricht korrekt empfangen haben. In diesem Fall kann keine Aussage über den Empfang einer Nachricht bei den Nachbargeräten getroffen werden.
- *Anycast*
  Die Nachricht soll an ein beliebiges Gerät im Sendebereich versendet werden. Hierbei wird gewährleistet, dass die Nachricht von mindestens einem Nachbarn erhalten wird, sofern mindestens einer existiert. Wenn das Senden an den gewählten Nachbarn fehlschlägt, wird das Versenden an einen anderen wiederholt. Auch hier kann die Broadcasteigenschaft genutzt werden und die Wahl des Empfängers rein auf Protokollnachrichten erfolgen.

In mobilen Szenarien häufig benötigte Informationen sind die über momentan benachbarte Geräte, sowie über die Ereignisse des Hinzukommen bzw. des Wegfallens einzelner Geräte. Um diese Information zu erhalten, müssen periodisch Broadcastnachrichten verschickt werden, damit andere Geräte von der Existenz des Gerätes erfahren. Dieses als *Beaconing* bezeichnete Verfahren wird auch in einigen Fällen (z.B. IEEE802.11 oder Bluetooth) schon von der MAC-Schicht des verwendeten Kommunikationsverfahren durchgeführt. Hier werden die Beaconnachrichten zur lokalen Synchronisation der Geräte und auch zur Propagierung der Existenz lokaler ad-hoc Netzwerkzellen verwendet. Im Falle des IEEE802.11 Protokolls ist das Verfahren für andere Dienste aber nicht brauchbar, da zwar mit einer hohen Frequenz (ca. alle 0.1 Sekunden) Beaconnachrichten versandt werden, aber nicht immer von allen Geräten. Sobald ein Gerät eine Nachricht empfängt, wird das Beaconing ausgesetzt, so dass es passieren kann, das einige Geräte in der Nachbarschaft nicht sichtbar sind. Häufig wird auch das Beaconing ausgesetzt, sobald eine andere Nachricht versendet werden soll. Neben der Absenderadresse können in einer Beaconnachricht zusätzliche Informationen per *Piggybacking* angehangen werden. Hierbei handelt es sich um Informationen mit geringem Umfang und entweder einer hohen Änderungshäufigkeit (z.B. die aktuelle Geräteposition) oder von einer hohen Dringlichkeit. Dringlichkeit bedeutet in diesem Fall, dass die Information benötigt wird, sobald das Gerät in Kommunikationsreichweite gelangt bzw. die aktuelle Information immer in Verbindung mit der Existenz des Gerätes steht. Da das Versenden von Broadcasts unzuverlässig ist, ist eine explizite Propagierung von Informationen zu unsicher. Diese Unsicherheit wird

2.1 Addressed Multicast



Um den Adressierten Multicast mit anderen lokalen Kommunikationsprimitiven zu vergleichen, wurden Simulationen durchgeführt. Im simulierten Szenario sind 60 Geräte mit 802.11MAC-Layer* und einer Sendereichweite von etwa $100m$ auf einer Fläche von $300m$ x $300m$ gleich verteilt. Randomisiert versenden alle Geräte Broadcastpakete mit einer Größe von 10KByte und erzeugen eine Grundlast von ca. $19,39\%$ (Kurve *None*). Die dargestellten Lastwerte (zwischen 0 und 1) sind dabei gemittelt über die lokal beobachtete Medienauslastung aller Geräte. Ein zufällig gewähltes Gerät sendet alle 2 Sekunden an 5 Nachbargeräte eine Nachricht der Größe 100KByte. Es werden (1) fünf Unicasts, (2) ein Adressierter Multicast, (3) ein Unicast mit 4 Zusatzadressen an eines der fünf Geräte mit Promiscuousflag$^+$ im Header (4) ein Broadcast mit 5 Zusatzadressen versendet. Kurve *none* stellt die Hintergrundlast im Netz dar. Die Versuche wurden jeweils 200 mal wiederholt und die einzelnen Sendeereignisse einer Simulation überlagert. Es zeigt sich, dass die Zustellrate des adressierten Multicasts sogar höher als die des Unicasts ist. Ein Grund dafür ist, dass bei Nachrichtenwiederholungen immer alle fehlgeschlagenen Empfänger wiederholt werden. Dabei liegt die Mediumsnutzung wie zu erwarten deutlich unter der Auslastung bei Unicast und kaum höher als bei einem Broadcast. Die geringe Broadcastzustellrate ist durch Kollisionen mit Nachrichten der Hintergrundlast aufgrund des Hidden-Terminal Problems zu erklären. Die geringe Zustellrate des Promiscuous Unicast liegt darin begründet, dass bei Verlust der Unicastnachricht immer auch alle anderen Nachbargeräte nicht erreicht wurden.

	Gesamtlast	Zustellrate
Unicast	24.99%	72.51%
Addressed Multicast	21.14%	83.69%
Promiscuous Unicast	20.53%	57.49%
Broadcast	20.30%	16.42%

– Fortsetzung Box 2.1 –

* Soweit nicht anders angegeben, wurden alle Simulationen der Arbeit mit folgenden MAC- und Medienparametern durchgeführt: Empfängerempfindlichkeit (Lucent/Orinoco PCMCIA Silver Gold): receiveThreshold: $-82dB$ ($11MBit/s$), $-94dB$ ($1MBit/s$); signalToNoiseRatioThreshold: 16 ($11MBit/s$), 4 ($1MBit/s$); Two-Ray-Ground-Modell; Übertragungsrate $11MBit$; Noise: exponenzialverteilt $-10dB$. MAC Parameter: fragmentationThreshold: 2346 Byte; rtsThreshold: 2347 Byte; beaconInterval: $0.1s$.

+ Dieses Flag teilt dem Empfänger mit, dass trotz der Unicastadresse die Nachricht weiterverarbeitet werden soll. Dies wird auch *impliziter Promiscuousmode* genannt.

durch das periodische Versenden der Beaconnachricht reduziert. Daher werden Informationen über Nachbargeräte erst nach dem Ausbleiben mehrerer Beaconnachrichten gelöscht.

Verallgemeinert man diesen Piggybackingmechanismus, erhält man den so genannten *Periodicast* [35]. Hierbei werden dynamisch (in der Regel nach Prioritäten) Informationen an die Beaconnachricht angehangen und diese bis zu einer maximalen Größe aufgefüllt. In Umgebungen mit fester Paketgröße wie z.B. Bluetooth, kann somit sogar nicht verwendete Bandbreite genutzt werden. Aber auch hierbei gilt, dass die zu verbreitenden Informationen nur einen geringen Umfang besitzen dürfen.

2.2 Multihop Kommunikation

Um auch andere Geräte als die direkten Nachbarn in einem mobilen ad-hoc Netzwerke erreichen zu können, müssen Nachrichten von Nachbargeräten weitergeleitet werden. In diesem Fall spricht man von *multihop Kommunikation*. Da häufig das weiterleitende Gerät kein Ziel der Kommunikation darstellt, ist für eine multihop Kommunikation ein altruistisches Verhalten der Geräte zwingend erforderlich, d.h. die Geräte müssen Ressourcen zur Verfügung stellen. Neben der benutzten Bandbreite werden daher zusätzlich CPU- und Speicherressourcen sowie die gerade bei mobilen Geräten sehr wichtige Energieressource benötigt. Prinzipiell sind auch bei multihop Kommunikation dieselben Direktiven denkbar wie im Falle der singlehop Kommunikation. Allerdings ist bei multihop Kommunikation die wichtige Vorraussetzung der Broadcasteigenschaft nur lokal gegeben. Somit können nicht alle Direktiven effizient auf den multihop Fall abgebildet werden. Ersichtlich wird dies auch in Abbildung 2.12 im Falle eines multihop Unicasts: A sendet eine Nachricht an D und nutzt dabei B und C als Router. Die Nachricht kann auch von den Geräten E und F empfangen werden, nicht aber von den Geräten G und H. Im Allgemeinen

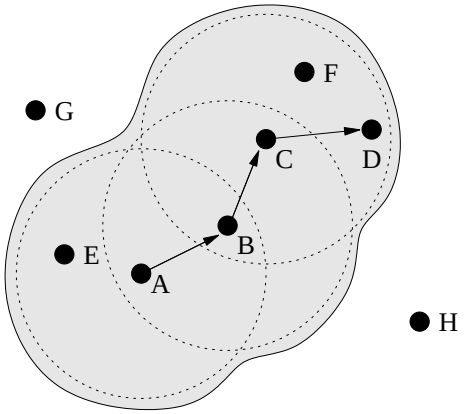


Abb. 2.12. Multihop Kommunikation. Damit Gerät A Gerät D erreichen kann, müssen die Geräte B und C die Nachrichten weiterleiten. Aufgrund der Broadcasteigenschaft des Mediums beschreibt die Kommunikation einen Korridor, indem die Nachricht empfangen werden kann.

beschreibt eine multihop Kommunikation also einen oder mehrere Korridore zwischen dem Sender und dem Ziel bzw. den Zielen einer Nachricht. Innerhalb eines Korridors werden neben den Routern der Nachricht weitere Geräte indirekt in die Kommunikation involviert. Das bedeutet, sie werden einerseits gestört, können aber auch andererseits die Kommunikation mithören. Letztere Eigenschaft kann von Vorteil sein, wenn die kommunizierte Information anderen Geräten zugänglich sein darf und auch von ihnen genutzt werden kann.

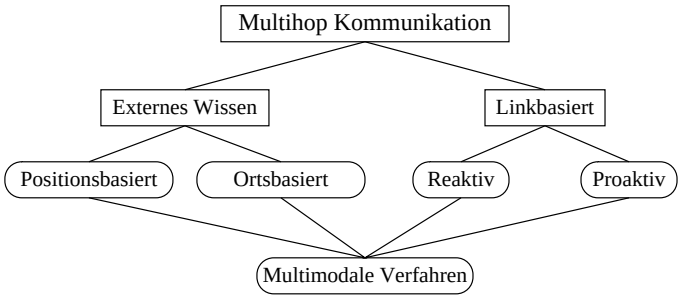


Abb. 2.13. Multihop Kommunikationsklassen

Der entscheidende Aspekt der multihop Kommunikation ist die Zielfindung einer Nachricht, das so genannte Routing. Allgemein ist das Ziel eines Routingverfahrens die Auslieferung einer Nachricht an ein oder mehrere Ziele im

Netz, wobei die Ziele nicht notwendigerweise durch Geräteadressen gegeben sein müssen. Die Notwendigkeit der Zielfindung ist im Falle einer Unicastkommunikation am ersichtlichsten. Die Empfängeradresse des Zielgerätes ist bekannt, der Weg, den die Nachricht durch das Netzwerk nehmen muss um dieses Ziel zu erreichen, muss aber erst gefunden werden. So muss in Abbildung 2.12 Gerät A wissen, dass es die Nachricht an D zuerst an B senden muss, damit B die Nachricht weiter zum Ziel leiten kann. Bekannte Routingverfahren aus drahtgebundenen Netzen lassen sich aufgrund der Dynamik des Netzes nur sehr bedingt in ad-hoc Netzen einsetzen. Daher sind in den vergangenen Jahren Erweiterungen bestehender Verfahren aber auch einige grundsätzlich neue Routingverfahren entstanden. Allgemein können Routingverfahren für mobile ad-hoc Netzwerke in mehrere Teilklassen unterteilt werden (Abbildung 2.13). Rein auf der Basis der Kommunikationsverbindungen arbeiten so genannte *linkbasierten* Verfahren. Eine weitere Differenzierung findet zwischen *proaktiven* und *reaktiven* Verfahren statt. Proaktive Verfahren erstellen Strukturen über das gesamte Netzwerk, welche an die sich ändernden Netzstrukturen angepasst werden. Reaktive Verfahren erstellen diese Strukturen nur bei Bedarf und auch nur die Teile, die momentan benötigt werde, was dazu führt, dass vor der Kommunikation unter Umständen die Strukturen neu erzeugt bzw. repariert werden müssen. Beide Ansätze besitzen bzw. erstellen also in irgendeiner Form globales Wissen über das gesamte Netzwerk, wenn mit diesen Varianten im gesamten Netzwerk kommuniziert werden soll. Das bedeutet, dass linkbasierte Verfahren nicht gut skalieren, da im ersten Fall der Speicheraufwand und im zweiten Fall zumindest der Kommunikationsaufwand zur Neuerstellung der Strukturen mit der Netzwerkgröße wächst. Reaktive Verfahren haben den Vorteil, dass nur ein Teil der Geräte Strukturen speichert bzw. darin enthalten ist, da die Erzeugung vom Kommunikationsbedarf abhängt. Darüber hinaus können reaktive Verfahren besser mit Mobilität umgehen, da bei Änderungen in der Netzstruktur nur die Strukturen verändert werden müssen, die aktuell benötigt werden. Der Nachteil gegenüber proaktiven Verfahren ist aber, dass es unter Umständen lange dauert, bis mit der eigentlichen Kommunikation begonnen werden kann, da die Strukturen zuerst aufgebaut bzw. repariert werden müssen. Proaktive Verfahren sind dann sinnvoll, wenn die Mobilität gering ist bzw. die Strukturen sehr einfach sind. Dies ist z.B. bei stationär installierten Sensornetzwerken der Fall, bei denen alle Geräte nur wenige ausgezeichnete Geräte kennen müssen, zu denen Sensordaten versendet werden (so genannte Senken). Aber selbst hier kann durch wechselseitige Schlafzyklen das Netz eine hohe Dynamik besitzen.

Die zweite Klasse der Kommunikationsverfahren arbeitet mit zusätzlichem *externen Wissen*. Die *ortsbasierten* Verfahren basieren auf der Identifizierbarkeit von Orten, d.h. Geräte können ihren momentanen Aufenthaltsort bestimmen. Werden bei der Kommunikation also keine Geräte sondern Orte adressiert, ist zumindest die Zahl der möglichen Kommunikationsziele in linkbasierten Strukturen nicht mehr abhängig von der tatsächlichen Gerätezahl im Netz. Besitzen die Geräte Wissen über die globale Ortstruktur, z.B. über

eine globale Karte, können diese Topologien verwendet werden, um den Aufwand für die Erstellung bzw. Speicherung der Routingstrukturen linkbasierter Verfahren zu reduzieren. *Positionsbasierte* Verfahren benötigen die globale Positionierbarkeit der Geräte z.B. mittels GPS. Sie sind somit in der Lage, Orte ohne zusätzliches Wissen in Relation zu einander zu setzen. Wenn nur Orte adressiert werden, können diese Verfahren sogar nahezu zustandsfrei arbeiten, d.h. sie benötigen nur lokales Wissen über das Netzwerk und skalieren somit mit der Netzgröße. Werden allerdings auch Geräte adressiert, ist dieser Vorteil nur bedingt vorhanden, da das Wissen über Gerätepositionen vorhanden sein muss und im Netzwerk verbreitet werden muss. Darüber hinaus existieren derzeit keine zustandslosen Verfahren, die eine sichere Auslieferung in beliebigen Netzwerken ermöglichen. Daher ist es sinnvoll, nahezu zustandsfreie auch mit linkbasierten Verfahren zu verknüpfen, um die Vorteile beider Ansätze nutzen zu können.

Im Folgenden werden verbreitete multihop Kommunikationsdirektiven vorgestellt und exemplarisch die wichtigsten Routingverfahren, die diese realisieren, skizziert.

2.2.1 Fluten und Broadcast

Um alle Geräte in einem Netzwerk zu erreichen, kann das Netzwerk mit einer Nachricht geflutet werden. Das bedeutet, das die Nachricht an alle Nachbarn versendet und auch von allen Nachbarn weiterpropagiert wird, wenn die Nachricht noch nicht empfangen wurde. Zur Weiterleitung bietet sich in drahtlosen Netzwerken der Broadcast an. In hochmobilen Netzen kann dieses Verfahren unter Umständen die einzige Möglichkeit sein, weiter entfernte Knoten als die direkten Nachbarn zu erreichen, da keine Routingstrukturen notwendig sind. Darüber hinaus wird auch kein externes Wissen benötigt. Der Nachteil des Flutens ist der exorbitante Kommunikationsaufwand, der dazu führen kann, dass die vorhandenen Netzwerkressourcen vollständig aufgebraucht werden [89]. Um dieses Problem zu reduzieren, wurden eine Reihe von multihop Broadcastverfahren entworfen [110]. Hierbei wird versucht, die Anzahl der Nachrichten zu reduzieren, da im Falle des einfachen Flutens häufig sehr viele redundante Nachrichten erzeugt werden. Probabilistische Verfahren leiten Nachrichten nur aufgrund gegebener Wahrscheinlichkeiten und unter Umständen aufgrund lokal beobachtbarer Informationen, wie z.B. der Zahl der empfangenen Duplikate oder die vermutete Distanz zu den letzten Sendern, weiter. Stehen Positionsinformationen zur Verfügung, kann versucht werden, die zusätzliche Abdeckung der Ebene zu bestimmen, die eine Weiterleitung erreichen würde, und erst ab einem definierten Schwellwert eine Weiterleitung durchzuführen. Diese Varianten verzögern die Auslieferung aufgrund der verwendeten Kriterien, um unter Umständen vorhandene besser geeignete Nachbarn zu bevorzugen und eine Weiterleitung bei schlechteren zu vermeiden. Verfahren auf Basis von single- oder multihop Nachbarschaftsinformationen

versuchen aufgrund der lokal beobachteten Netzstruktur Knoten zu erkennen, die für eine Weiterverbreitung überflüssig sind. Trotz der Reduzierung der Nachrichtenanzahl ist die Nutzung dieser Verfahren nur dann sinnvoll, wenn zum einen die verbreitete Information klein und zum anderen die Relevanz für das Gesamtnetzwerk groß ist.

Häufig ist eine Adressierung des gesamten Netzwerks aber nicht gewollt oder nicht erwünscht, sondern es soll nur ein Teilbereich des Netzwerks erreicht werden. Da die tatsächliche Größe eines Netzwerks unbekannt ist, kann ein unbeschränkter multihop Broadcasts erheblich höhere Kosten verursachen als vorgesehen. Die einfachste Maßnahme ist, die Zahl der Weiterleitungen zu beschränken. Somit werden alle Geräte mit einer bestimmten, in der Regel minimalen, Hopdistanz zum Sender erreicht. Ein multihop Broadcast kann auch auf einen identifizierbaren Ort beschränkt werden, wenn Ortsinformationen zur Verfügung stehen und nur Geräte vor Ort erreicht werden sollen.

Ein spezieller Fall des Flutens ist das *multihop Beaconing*, bei dem den Beaconnachrichten nicht nur lokale Informationen angehangen werden sondern auch Informationen, die von Nachbarn empfangen wurden. Meist handelt es sich um 2-hop Nachbarschaftsinformationen, d.h. Geräte hängen an ihre Beaconnachrichten nur Informationen über ihre direkten Nachbarn an. 2-hop Information ist zum Beispiel notwendig um proaktiv bidirektionale Kommunikationsverbindungen erkennen zu können. Verallgemeinert kann ein multihop Beaconing n-hop Nachbarschaftsinformationen verbreiten.

2.2.2 Unicast

Der multihop Unicast kann ein beliebiges Gerät im ad-hoc Netzwerk adressieren. Unicast Routingverfahren müssen also einen Pfad zu diesem bestimmten Gerät im Netzwerk finden. Wie in drahtgebundenen Netzen können auch in ad-hoc Netzwerken auf Basis der vorhandenen Kommunikationsverbindungen Routingstrukturen realisiert werden. Die Routingstrukturen basieren bei Link-State-Verfahren auf der Speicherung von Routen zu einem Ziel, bei Distance-Vector-Verfahren auf der Speicherung des nächsten Knotens einer Route zu einem Ziel. Proaktive Verfahren, wie z.B. DSDV (Destination-Sequenced Distance Vector routing [91]) müssen Routingstrukturen zu allen Geräten im Netzwerk aufrechterhalten. Somit besitzt jeder Knoten eine globale Sicht auf das gesamte Netzwerk. Die Kommunikationswege werden periodisch auf den aktuellen Stand gebracht und erzeugen daher eine permanente Netzlast, die darüber hinaus von der Netzwerkgröße abhängt. Proaktive Verfahren sind damit nur für Netzwerke von geringer Größe und geringer Dynamik sinnvoll einsetzbar [10].

Unter der Annahme, dass innerhalb eines Zeitfensters nur ein Bruchteil der möglichen Routen benötigt wird, erstellen reaktive Verfahren die Routen nur bei Bedarf. Deshalb muss vor Beginn der Kommunikation eine Route zum Ziel gesucht werden, falls lokal keine Informationen vorhanden sind.

Dies kann die Auslieferung einer Nachricht stark verzögern, da die Suche je nach Netzgröße sehr lange dauern kann. Prinzipiell arbeiten alle Verfahren auf der Basis von Fluten der Routediscoverynachrichten. Ist der Zielknoten erreichbar, empfängt er die Routediscoverynachricht und kann mit einer Routereplynachricht antworten. Unterschieden werden kann auch hier zwischen Link-State- (z.B. Dynamic Source Routing DSR [56]) und Distance-Vector-Verfahren (z.B. Ad-hoc On demand Distance Vector routing AODV [92]). Bei DSR beinhaltet jede Nachricht die Route zum Ziel, d.h. Geräte auf der Route benötigen keine Informationen über die Route. Nachteilig ist die wachsende Nachrichtengröße bei steigender Entfernung. Um Routen auch über längere Zeit und für Zwischenknoten zur Verfügung zu haben, werden gefundene bzw. mitgehörte Routen in einem Cache abgelegt, wobei mehrere Routeneinträge pro Ziel möglich sind. Ist keine Route bekannt oder die bekannte Route fehlerhaft, wird mit einer Routediscoverynachricht das Netz geflutet und dabei der zurückgelegte Pfad in der Nachricht gespeichert. Der Empfänger kann also direkt mit dieser Route antworten. DSR ermöglicht auch die Verwendung unidirektionaler Verbindungen. Dann muss auch das Routereply im Netzwerk geflutet werden. AODV speichert in jedem Gerät den letzten Hop einer Routediscovery Nachricht als Route zum Sender der Nachricht. Der Routereply kann somit ausgeliefert werden und dadurch der notwendigerweise bidirektionale Routingpfad hergestellt werden. Auf Basis von Beaconing wird die Gültigkeit der lokalen Routingeinträge überprüft und bei Wegfall eines Nachbars die entsprechenden Ziele gelöscht. Empfängt ein Gerät eine Nachricht zu einem fehlerhaften Ziel, wird ein erneutes Routediscovery ausgelöst. Routediscovery Nachrichten können auch von anderen Knoten als dem Empfänger beantwortet werden. Um Routingkreise zu vermeiden, werden Routen mit logischen Zeitstempeln, bestehend aus Zieladresse und ziellokaler Sequenznummer versehen. Ein Routereply wird nur dann gesendet, wenn der Zeitstempel in der Routerequestnachrichten undefiniert oder älter ist.

Aufgrund des Aufwandes und der Dauer der Routenfindung sind linkbasierten Verfahren nur für Netzwerke mit geringer Dynamik und einer Größenordnung bis etwa 200 Geräte geeignet [10, 94, 97]. Dennoch können reaktive Verfahren auch in größeren Netzen eingesetzt werden, wenn die maximale Entfernung zwischen Sender und Empfänger bekannt ist. Dann können die aufwendigen Routenfindungsprozesse beschränkt werden und das Verhalten der Verfahren ist mit dem Verhalten in kleinen Netzen gleichzusetzen. Ist eine ungefähre geographische Aufenthaltsregion des Empfängers bekannt, kann das Routediscovery beispielsweise auch mittels gerichtetem Fluten [60] oder einem Locationcast durchgeführt werden (siehe Abschnitt 2.2.3). Sind solche Informationen bekannt bzw. ist eine Anwendung dazu in der Lage, diese Angaben zu machen, sind linkbasierte Verfahren auch in großen Netzen sinnvoll einsetzbar. Für DSR wurde von den Autoren eine maximale Entfernung von 5 bis 10 Hops vorgeschlagen [56], für die anderen Verfahren sind ähnliche Größenordnungen denkbar. Somit kann das initiale Fluten durch die Zahl der Hops oder auch, wie oben erwähnt, durch Vorgabe einer Verbreitungsregion beschränkt

werden. Inkrementelles Fluten, wie es beispielsweise bei AODV beschrieben wird, stellt allerdings nur bedingt eine Verbesserung der Routenfindung dar. Hierbei wird der Hopcount des Flutens beschränkt und im Fehlerfall erhöht. Bei nahen Knoten kann dadurch der Aufwand der Routenfindung drastisch reduziert werden, ohne dass die Netzwerkdistanz des Knotens bekannt sein muss. Falls der gesuchte Knoten aber nicht erreichbar ist, ist der Aufwand im Vergleich zum einfachen Fluten höher, um dieselbe Netzdistanz zu durchsuchen.

Steht auf jedem Gerät die aktuelle Positionen als externes Wissen zur Verfügung können so genannte *Positionsbasierte Routingverfahren*[3] eingesetzt werden. Ist die Position des Zielgerätes bekannt, ist, zumindest theoretisch, eine zustandslose Kommunikation möglich. Das einfachste positionsbasierte Routingprotokoll basiert auf dem Greedyprinzip. Dabei wird in jedem Routingschritt versucht, aufgrund von lokalem Wissen dem (geographischen) Ziel näher zu kommen. In der einfachsten Variante wird die Entfernung minimiert [24], es existieren aber eine Vielzahl weitere Varianten (Most Forward within Radius (MFR [104]), Compass routing (DIR [63]), eine Übersicht ist in [28] zu finden). Das benötigte lokale Wissen beschränkt sich in der Regel auf das Wissen über die eigene Position und die Positionen der Nachbargeräte. Aufgrund des so genannte *Greedyroutingfehler* können Greedyverfahren allerdings die Auslieferung einer Nachricht in einem zusammenhängenden Netz nicht immer ermöglichen. Der Greedyroutingfehler liegt in der Tatsache begründet, das Nachrichten in lokale Optima gelangen und somit verworfen werden müssen. Ein Beispiel hierfür ist in Abbildung 2.14 zu sehen. Anders ausgedrückt ist Greedyrouting nicht in der Lage, Wege um beliebige "Löcher" im Netzwerk zu finden. Diese Löcher entstehen in dünnen oder ungleichmäßig dichten Netzwerken, insbesondere aber auch durch Hindernisse (z.B. Gebäude) in der Umgebung oder Bereiche, in denen sich keine Geräte befinden können (z.B. Seen).

Um dennoch eine Paketauslieferung zu ermöglichen, wurden Verfahren entworfen die in zusammenhängenden Netzen garantiert einen Weg zum Ziel finden und dennoch zustandsfrei bleiben. All diese Verfahren beruhen auf dem Prinzip des *Planargraphenroutens* [63]. Aus dem ursprünglichen Netzwerkgraphen wird mittels lokaler Verfahren ein planarer Subgraph erzeugt. Dieser planare Subgraph wird benötigt, um kreisfrei entlang der dabei entstehenden Flächen zu routen. Die gängigste Variante verwendet die Folge der Flächen, die von einer Gerade zwischen Start und Zielknoten geschnitten werden. Zur Planarisierung werden sehr harte Kriterien an das Netzwerk gestellt. So muss das Netzwerk die so genannte Unitdiscgrapheigenschaft besitzen, d.h. die Senderadien aller Geräte müssen exakte Kreise gleicher Radien sein. Darüber hinaus werden exakte Gerätepositionen benötigt. In gewissem Maße lassen sich die harten Kriterien aber abschwächen. So erlaubt Beispielsweise der Ansatz in [5] eine Variation des Senderadius um $\sqrt{2}$. Dennoch sind Planargraphenroutingverfahren in realen Netzen nur bedingt einsetzbar. Darüber hinaus

[3] Synonym ist auch der Begriff geographische Routingverfahren

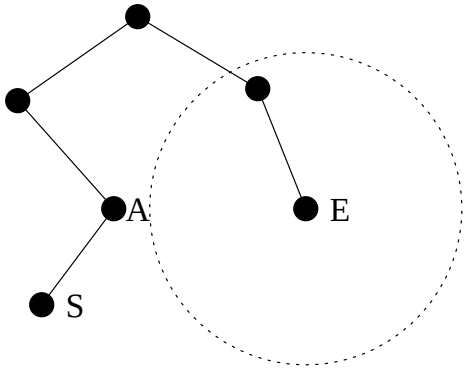


Abb. 2.14. Der Greedyroutingfehler. Die Nachricht von S nach E kann in A nicht weitergeleitet werden, da keiner der Nachbarn von A einen Fortschritt für die Nachricht bedeutet.

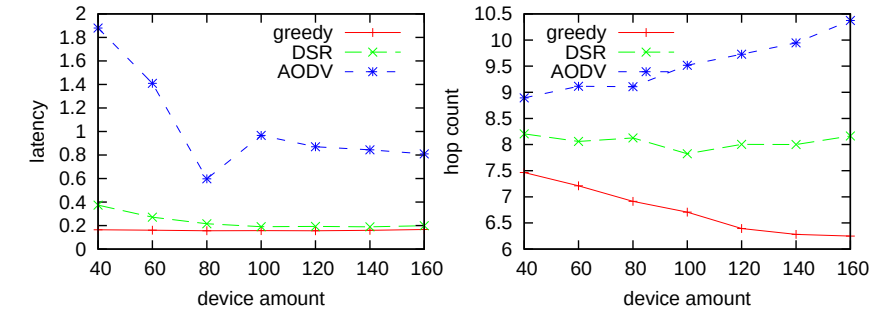
sind sie auch anfällig bezüglich Mobilität [36]. Es existieren jedoch einige vielversprechende Ansätze, die eine höhere Robustheit bzgl. Senderadienvariation, Positionsungenauigkeit und Mobilität besitzen. Ein Beispiel ist das Planargraphenrouten auf Basis von geographischen Clustern (GCR [30]). Dieses Verfahren teilt die Ebene in reguläre Hexagone, welche die Ausgangsbasis zur Konstruktion eines geographischen Overlaygraphen bilden. Knoten dieses Graphen sind die Zentren der Hexagone, in denen sich Geräte befinden, Kanten entstehen, wenn Kommunikationsverbindungen zwischen Geräten in verschiedenen Hexagonen bestehen. Der Overlaygraph wird planarisiert und auf Basis des planaren Overlaygraphen das Planargraphenrouting durchgeführt. In der ursprünglichen Variante werden zwar dieselben harten Kriterien an das zugrunde liegende Netzwerk gestellt, allerdings ist zu vermuten, dass eine Robustheit bzgl. Senderadienvariation und Positionsungenauigkeit mit geringerem Aufwand erreichbar ist als in [5]. Gründe dafür sind die Knotenaggregation und die regelmäßigen Struktur des Overlaygraphen. GCR ist auch erheblich robuster bzgl. Mobilität [28] als andere bekannte Planargraphroutingverfahren. Die einfache Variante SGCR [29, 31] stellt keinerlei Vorraussetzungen an das zugrunde liegende Netzwerk. Wenn es möglich ist, die Geräte ohne Positionsinformationen eindeutig Clustern zuzuordnen, kann hier sogar auf Positionsinformationen verzichtet werden. Allerdings kann das Verfahren nicht immer einen Weg zwischen Start und Zielknoten finden. Alle Planargraphenroutingverfahren lassen sich in Kombination mit Greedyroutingvarianten verwenden, wobei sie als Rückfallstrategie im Falle des Greedyroutingfehlers verwendet werden. Die unbestreitbare Stärke dieser Strategien ist ihre Zustandslosigkeit, so dass diese Verfahren in beliebig großen Netzen einsetzbar sind, falls die Position des Ziels bekannt und fest ist. Sie eignen sich daher dafür, Nachrichten über größere Entfernungen zu transportieren.

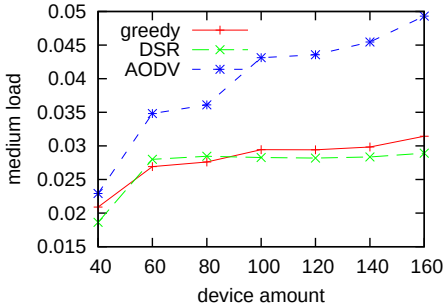
2.2 Linkbasiert vs. Positionsbasiert I

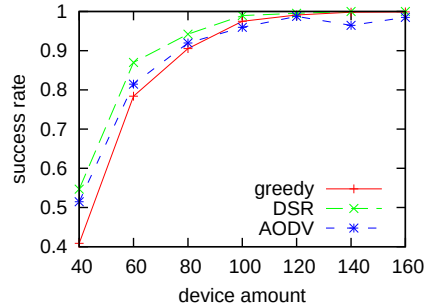
In dieser Untersuchung werden die linkbasierten Unicastkommunikationsverfahren AODV und DSR mit einfachem positionsbasiertem Greedyrouting verglichen. Es liegt die Annahme zugrunde, dass der Applikation oder der Middleware die Zielposition bekannt ist, also keine Adressabbildung auf die Position erfolgen muss. Das Szenario entspricht auch einem positionsbasierten Anycast, da die linkbasierten Verfahren auch hier nach einem Gerät in der Zielregion suchen müssen. In dem Szenario werden auf einer Fläche von $500m$ x $500m$ zwischen 40 und 160 Geräte mit dem Randomwaypointmodell (Pausezeit $0s$, Geschwindigkeit $0.5m/s$-$20.0m/s$) bewegt. Die Geräte verfügen über einen 802.11MAC-Layer mit einer Reichweite von $100m$. Ein fixes Gerät auf der einen Seite sendet an ein fixes Gerät auf der gegenüberliegenden Seite der Fläche mit einer konstanten Bitrate (100 KByte-Pakete alle $0.5s$).
In den Abbildungen sind die Latenz, die benötigten Routingschritte, die verursachte Medienlast und die Erfolgsrate der Verfahren über die Gerätezahl bei $1m/s$ und $10m/s$ abgetragen. Es ist erkennbar, dass in dünnen Netzen die Auslieferungsrate von Greedy aufgrund des Routingfehlers schlechter ist, sie aber bei steigender Netzdichte und Dynamik besser als die linkbasierten Verfahren wird. Gründe hierfür sind die höhere Netzlast des Routediscoveries aufgrund der steigenden Dichte und Dynamik und die mit der Dynamik steigenden Fehlschläge bei Routediscovery und -repair.
AODV verursacht die höchste Netzlast, da Beaconing und Fluten zusätzliche Last verursacht. DSR benötigt kein Beaconing, dagegen sind aber die gefluteten Nachrichtenpakete aufgrund der darin enthaltenen Route größer. Somit wird DSR mit steigender Dynamik schlechter. Aufgrund des Routerepairs entstehen bei den linkbasierten Verfahren längere Routen als bei Greedy, das immer nahe beim kürzesten Pfad liegt. Längere Routen und höhere Last verursachen eine höhere Latenz. Somit zeigt AODV die höchste Latenz bei $1m/s$ und wird bei steigender Gerätezahl und Dynamik von DSR überholt. Greedy liegt deutlich darunter, da auch kein initiales Routediscovery nötig ist.
Greedy verursacht also die geringste Latenz, benötigt die wenigsten Routingschritte, verursacht bei steigender Dynamik und Netzdichte die geringste Last und besitzt bei geringer Netzdichte eine kaum schlechtere, bei steigender Netzdichte eine bessere Auslieferungsrate als die linkbasierten Verfahren. In diesem Szenario liegt allerdings auch eine idealisierte Umgebung vor, in der der Greedyroutingfehler nur bei geringer Gerätedichte auftritt. In allgemeinen Netzen ist eine Recoverystrategie notwendig, da aufgrund von Hindernissen auch wiederholte Kommunikationsversuche scheitern können. Dann können aber auch linkbasierte Verfahren als Fallback- oder Recoverystrategie verwendet werden. So kann im Fehlerfall vollständig bzw. bis zum Erreichen eines Knotens, der dem Ziel näher ist, auf diese Verfahren gewechselt werden.

– Fortsetzung Box 2.2 –

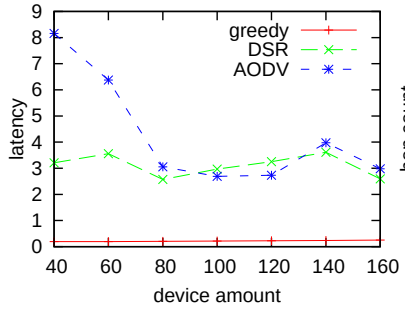
$1.0m/s$

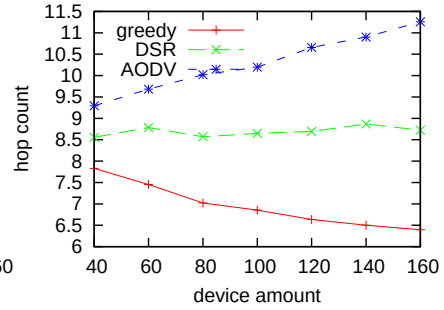


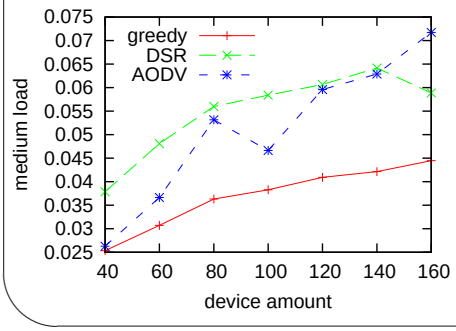


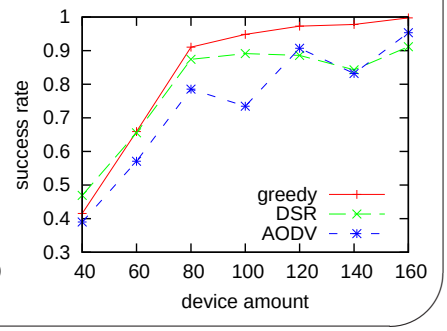


$10.0m/s$









Neben den zustandsfreien Verfahren existieren weitere, zustandsbehaftete Verfahren als Greedyrückfallstrategie. GRA (Geographical Routing Algorithm) [55] nutzt für den Fall des Greedyroutingfehlers Routingtabellen, die das nächste Nachbargerät für Zielpositionen speichern. Gewählt wird derjenige Nachbar, dessen Zielpositionseintrag dem gesuchten Ziel am nächsten liegt. Die Routingtabellen werden im Falle eines Greedyroutingfehlers wie bei linkbasierten Verfahren durch Wegfindungsmechanismen bestimmt. Allgemein lassen sich auch linkbasierte Verfahren wie DSR als Rückfallstrategie für den Greedyroutingfehler verwenden. Dieses Vorgehen ist mit einer Breitensuche im Netz gleichzusetzen, es existieren aber auch Verfahren auf Basis der Tiefensuche [102]. Umgekehrt gibt es auch Ansätze, Positionsinformationen zur Verbesserung von linkbasierten Verfahren einzusetzen. LAR (Location Aided Routing [40]) beispielsweise nutzt ungefähre Positionsinformationen über das Ziel, um das Routediscovery durch Richtungsbasiertes Fluten einzuschränken.

Stehen den Geräten Karteninformationen zur Verfügung, kann in speziellen Szenarien diese Information auch zum Routing verwendet werden. Bewegen sich die Geräte auf bekannten Wegen, kann das durch eine Karte gegebene Wegnetz verwendet werden, um den Greedyroutingfehler zu umgehen. Ein Beispiel ist GSR (Geographic Source Routing [71]), welches zur Interfahrzeugkommunikation eingesetzt werden soll. Hier werden aufgrund vorhandener Karten Wegpunkte definiert, die jeweils durch Greedyrouting erreicht werden.

Eine andere Möglichkeit, linkbasierte Verfahren in Kombination mit positionsbasierten Verfahren zu verwenden, ist das Verknüpfen der Verfahren. Ist vor Beginn des Routings die Region des Zielgerätes bekannt, kann mit positionsbasiertem Routing in die Nähe des Zielgerätes gelangt werden. Vor Ort kann dann linkbasiert das Ziel erreicht werden. Somit kann über größere Distanzen nahezu zustandsfrei Nachrichten weitergeleitet werden, die exakte Position des Ziels muss dafür aber nicht bekannt sein [8].

Damit positionsbasierte Verfahren Geräte adressieren können, muss die Position des Zielgerätes bekannt sein. Ist diese Information nicht immer lokal vorhanden, müssen Positionsverzeichnisdienste eingesetzt werden, welche die Positionsinformationen im Netz propagieren und zur Verfügung stellen. Je höher die Dynamik des Netzes desto aufwendiger sind diese Dienste. Meist werden die Positionen proaktiv in bestimmten Netzteilbereichen verbreitet. Beispielsweise kann die aktuelle Geräteposition mittels der Geräteadresse und einer Hashfunktion auf eine Position abgebildet und in einer bestimmten Region um diese Position von allen Geräten gespeichert werden [40]. Ohne das Wissen über die Zielposition sind positionsbasierte Verfahren genauso wie linkbasierte Verfahren nur in kleinen Netzen einsetzbar. Eine Ausnahme bilden Netze mit stationären Knoten, da keine Positionsänderungen im Netz propagiert werden müssen. Diese Netze können dennoch eine hohe Dynamik (z.B. durch Schlafzyklen) besitzen, da sie nur Einfluss auf die Linkstruktur des Netzes hat und somit insbesondere Greedyverfahren kaum von ihr beeinflusst werden.

2.3 Ortsbestimmung

Geräte eines ad-hoc Netzwerks können in einen globalen Kontext eingeordnet werden, in dem sie in der Lage sind, ihren momentanen Aufenthaltsort zu bestimmen. Aufgrund dieser Zusatzinformation ist es meist einfacher, gerätelokale Entscheidungen zu treffen. Neben Kommunikationsstrategien, die auf externen Wissen basieren, können auch Applikationen Entscheidungen auf Basis dieser Informationen treffen. Unterschieden werden kann zwischen einfacher *Lokalisierung* und *Positionierung* [52].

Ortslokalisierung

Ortslokalisierung bedeutet, dass ein Gerät verschiedene Orte unterscheiden kann. Die Unterscheidung kann mittels fest installierter Geräte erfolgen, die periodisch eine eindeutige Kennung senden, die den Installationsort identifizierbar macht. Die Kennung wird entweder über Funk (RF-Beacons), Infrarot (IR-Beacons) oder Ultraschall verbreitet. Geräte können also erkennen, wenn sie einen bestimmten Ort betreten oder verlassen. Diese Geräteeigenschaft wird auch *Lokalitätsbewusstsein* genannt, Kommunikationsverfahren die darauf basieren werden auch *ortsbasiert* genannt. Ortslokalisierung lässt sich auch mit einem Positionierungssystem erreichen.

Positionierung

Geräte können sich aufgrund eines Positionierungssystems in ein globales Koordinatensystem einordnen. Der Vorteil gegenüber einfacher Lokalisierung ist, dass die Positionskoordinaten in Relation zueinander gestellt werden können, um z.B. die Distanz zweier Geräte zu bestimmen. Verwendet wird hierfür häufig GPS (Global Positioning System), das zur Positionierung der Geräte Satelliten verwendet. Von Nachteil ist, dass Positionen in Gebäuden nicht oder nur sehr eingeschränkt zur Verfügung stehen. Werden IR-/RF-Beacons mit Koordinaten versehen, kann bei geeigneter Abdeckungen eine Positionierung erfolgen. Allgemein lässt sich durch Triangulierung und fest installierten Funkstationen eine Gerätepositionierung erreichen. Beispielsweise kann in Mobilfunknetzen je nach Zellenüberlappung aufgrund der gerichteten Antennen eine hohe Positionsgenauigkeit erreicht werden. Schließlich ist es im multihop ad-hoc Netzwerk möglich, aufgrund der Signalstärke der verwendeten Funktechnik ein virtuelles Koordinatensystem auf Basis der Linkstruktur des Netzes aufzubauen [14]. Diese Möglichkeit ist aber mit hohem Kommunikationsaufwand verbunden und aufgrund der häufig schlechten Signalabschätzung sehr ungenau. Dennoch lassen sich mittels solcher Verfahren Lücken in der Abdeckung durch andere Positionierungssysteme schließen.

2.2.3 Ortsbasierte Kommunikationsverfahren

Sind Orte identifizierbar, lassen sich auch völlig neue Kommunikationsverfahren definieren. So kann beispielsweise ein *ortsbasierter Anycast* ein beliebiges Gerät adressieren, welches sich aber an einem bestimmten Ort aufhalten muss. Stehen geographische Positionen zur Verfügung und ist auch der Ort durch Positionen gegeben, können positionsbasierte Unicastverfahren direkt

angewendet werden, wobei auf die vorherige Abbildung von Geräteadressen auf Positionen verzichtet werden kann. Die Nachricht kann an das erstbeste Gerät innerhalb der geographischen Region ausgeliefert werden. Orte lassen sich aber auch mit linkbasierten Verfahren adressieren. Hierbei werden Routen zu Orten vorgehalten, und das Routediscovery wird von Geräten vor Ort beantwortet, wobei mehrere gültige Routen entstehen können. Sollen mehrere Orte parallel angesprochen werden, handelt es sich um einen ortsbasierten Manycast. Es können, wie bei einem Multicast, der nur registrierte Geräte anspricht, Wegredundanzen genutzt werden, um Nachrichtenduplikationen zu optimalen Zeitpunkten durchzuführen.

Eine Erweiterung des ortsbasierten Anycasts ist der ortsbasierte Broadcast, bei dem alle Geräte an einem bestimmten, identifizierbaren Ort adressiert werden. Dazu kann im einfachsten Fall ein ortsbasierter Anycast verwendet werden, um dann in dem Zielknoten ein ortsbeschränktes Fluten zu starten. Werden geographische Positionen verwendet spricht man auch von einem *Geocast* oder auch *Locationcast*. In der ursprünglichen Beschreibung [61] handelt es sich hier aber um ein geographisch beschränktes Fluten ausgehend vom Nachrichtensender, welches die Zielregion beinhaltet bzw. um ein gerichtetes Fluten. Hierbei wird die Nachricht an alle Geräte in Richtung der geographischen Region weitergeleitet, wobei beispielsweise die Distanz zur Zielregion verringert werden muss. Gerichtetes Fluten kann auch unabhängig von einer Zielregion eingesetzt werden, indem ein Korridor und eine maximale Tiefe des Korridors, in dem die Nachricht geflutet wird, definiert wird.

2.2.4 Multicast

Im Falle eines Multicast wird eine Nachricht an eine Empfängergruppe, in der die Geräte explizit ihre Gruppenmitgliedschaft bekannt gegeben haben, gesendet. Dem Sender müssen die Gruppenmitglieder nicht bekannt sein, er muss aber zumindest den Anfang einer Route zu allen Mitgliedern kennen. Linkbasierte Verfahren nutzen meist Multicastbäume, wobei sich jedes Gerät die Nachbarn, über die Mitglieder einer Gruppe zu erreichen sind, speichert. Eine reaktive Variante ist das auf AODV basierende MAODV(Multicast-AODV [96]), welches den Multicastbaum bei Bedarf durch Fluten erstellt. Meshbasierte Multicastverfahren können auch mehrere Pfade zwischen jedem Sender und Empfängerpaar nutzen und sind deshalb in Netzen mit höherer Dynamik besser geeignet. ODMRP(On Demand Multicast Routing Protocol [39]) flutet wie MAODV das Netzwerk, erlaubt aber das Antworten der Gruppenmitglieder über mehrere Pfade. AMRoute(Ad-hoc Multicast Routing Protocol [111]) verwendet eine Kombination aus beiden Ansätzen. Eine Übersicht linkbasierter Multicastroutingverfahren ist in [84] zu finden.

Stehen Positionsinformationen zur Verfügung, können sie zur Reduzierung des Protokollaufwandes verwendet werden. SPBM (Scalable Position-based Multicast [106]) nutzt geographisch hierarchische Strukturen um die

Existenz von Gruppenmitgliedern zu propagieren. Zu diesem Zweck wird die Ebene in einen Quadtree unterteilt, wobei 4 Quadrate in der nächsthöheren Ebene zu einem Quadrat der doppelten Kantenlänge zusammengefasst werden. Geräte wissen lediglich, ob sich in den eigenen und in den direkt benachbarten Quadraten Mitglieder einer Gruppe befinden. Dazu müssen die entsprechenden Regionen mit diesen Informationen geflutet werden. Die Auslieferung kann bei direkt erreichbaren Gruppenmitgliedern sofort erfolgen, für alle anderen werden die benachbarten Quadrate der Quadtreehierarchie mit einem positionsbasierten Anycast adressiert. Nachrichten werden erst bei Bedarf dupliziert, um sie an benachbarte Gruppenmitglieder auszuliefern oder wenn Zielquadrate nicht über dasselbe Nachbargerät erreicht werden. Im Vorfeld ist also nur ein positionsbasierter Multicastbaum auf Basis der Quadtreehierarchie vorhanden, der Multicastbaum auf Linkebene entsteht erst bei der Nachrichtenauslieferung.

2.4 Paketisierung

Wird nachrichtenorientiert in einem multihop ad-hoc Netzwerk kommuniziert, kann die Nachricht Punkt-zu-Punkt (auch *Hop-by-Hop*) oder Ende-zu-Ende paketisiert werden, falls die MTU* der drahtlosen Kommunikationstechnik überschritten wird. Bei der Übertragung großer Nachrichten ist eine Ende-zu-Ende Paketisierung aufgrund der dann durchführbaren Flusskontrolle sinnvoll (Zahl der Pakete > Zahl der Hops). Darüber hinaus werden vor Abbruch jeglicher möglicher Verbindungen weniger Daten überflüssig übertragen, wenn noch nicht alle Pakete abgeschickt wurden. Auf der anderen Seite kann bei einer Hop-by-Hop Übertragung das Paket unter Umständen dennoch ausgeliefert werden, insbesondere wenn die direkte Auslieferung mit Store-and-Forward Kommunikation kombiniert wird. Ende-zu-Ende Paketisierung ist im Falle der Store-and-Forward Kommunikation eigentlich ungeeignet. Häufig ist hier der Weg das Ziel, d.h. die Nachricht wird auch auf Zwischenknoten interpretiert. Ebenso ist es ist möglich, dass das Ziel nicht mit einem oder mehreren bestimmten Geräten gleichzusetzen ist und deshalb ein Wiederherstellen der Nachricht nicht möglich ist, wenn einzelne Pakete auf unterschiedlichen Geräten ankommen. In manchen Fällen ist ein Wiederherstellung dennoch möglich: werden z.B. bei einem Locationcast auf Basis der Store-and-Forward Kommunikation die Pakete in einer Region für längere Zeit von einer variablen Gerätemenge vorgehalten, können mit dem Eintreffen des letzten Paketes alle Geräte die Nachricht wiederherstellen. Allerdings ist hier der Verlust eines Paketes kritischer, da ein Nachfordern von Paketen nicht immer möglich ist.

* Maximum Transmission Unit.

2.3 Store-and-Forward Netze

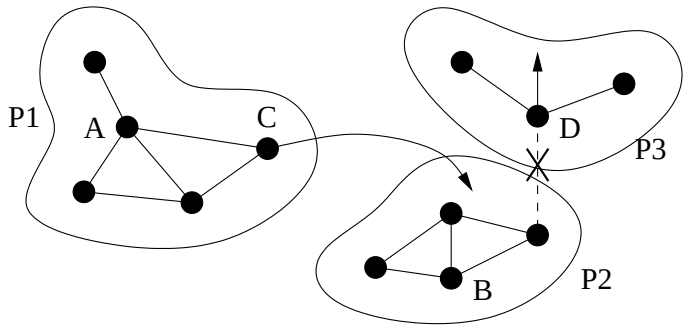


Abb. 2.15. Netzpartitionen können in mobilen ad-hoc Netzwerken jederzeit auftreten. Durch die Bewegung von Gerät D sind $P2$ und $P3$ nicht mehr verbunden, die Geräte in $P2$ können nicht mehr direkt mit Geräten in $P3$ kommunizieren. Gerät A kann aber dennoch Gerät B in einer anderen Partition erreichen, da Gerät C in diesem Beispiel Nachrichten zu Partition $P2$ mitnehmen kann.

Die beschriebenen multihop Kommunikationsverfahren basieren auf der sofortigen Auslieferung der Nachricht an ein oder mehrere Ziele. Ist eine Weiterleitung nicht sofort (bzw. nach einem Routediscovery) möglich, wird die Nachricht verworfen. Daher können nur Geräte innerhalb einer Netzpartition erreicht werden. Soll die Nachricht auch über die Partitionsgrenze hinaus verbreitet werden, müssen Nachrichten zwischengespeichert und ihre Weiterleitung später durchgeführt werden. Falls Partitionen nur sporadisch zu beobachten sind, kann die Weiterleitung erfolgen, sobald das Ziel erreichbar ist. Wenn Regionen nahezu permanent unverbunden sind, ist dennoch eine Weiterleitung aufgrund der Mobilität möglich (siehe Abbildung 2.15). Die Gerätemobilität kann also genutzt werden, um Nachrichten weiterzuleiten. Dann kann der Netzwerkzusammenhang über die Zeit definiert werden. Ein Gerät ist also dann erreichbar, wenn über die Zeit eine Folge von Kanten im Netzwerk existiert, über die eine Nachricht vom Sender zum Empfänger geleitet werden kann.

Die einfachsten Varianten entsprechen dem Fluten des Netzwerks, d.h. alle Geräte sollen erreicht werden. Da dies aufgrund von Netzpartitionen nicht sofort möglich ist, müssen die Nachrichten über einen längeren Zeitraum weiterverbreitet werden. Dieses Vorgehen bezeichnet man auch als *Verbreitung durch Infektion*, da die Nachricht nach und nach die Geräte des Netzwerks erreicht, indem sich die Geräte untereinander mit der Nachricht “anstecken”. Im einfachsten Fall geschieht dies durch periodisches Wiederholen der Nachricht von allen Geräten. Das Informationsradio [37] verbreitet so Informationen von geringem Umfang und geringer Änderungshäufigkeit in zugeordneten

geographischen Regionen. Die Wiederholungsperiode der Nachrichten ist sehr gering und adaptiert sich an die induzierte Netzlast, so dass nur eine geringe konstante Hintergrundlast erzeugt wird. Es können aber auch alle neue Nachbarn angefragt werden, ob die zu verbreitende Information schon vorhanden ist [88]. Informationen können in diesem Fall auch Gruppen zugeordnet sein, so dass Nachrichten nur innerhalb einer Gerätegruppe weitergegeben werden [44]. Dadurch wird ein Multicast über die Zeit realisiert. *Gossiping* basiert auf der zufälligen periodischen Wiederholung von Nachrichten, sobald neue Geräte in der Nachbarschaft erscheinen. Autonomous Gossiping [18] schränkt die zufällige Verbreitung aufgrund von Regeln ein, die in den Nachrichten vorgegeben sind.

2.5 Smallworldnetze

Unter dem Begriff Smallworld Models werden Netzwerke zusammengefasst, die sich durch einen geringen durchschnittlichen Abstand zwischen zwei beliebigen Knoten und einen hohen Clusteringkoeffizienten auszeichnen. Hubs, also Knoten mit sehr vielen Kanten, sind in Smallworldnetzen überrepräsentiert. Auf diese Eigenschaften hin, die Stanley Milgram ursprünglich in sozialen Netzen untersuchte [81], wurden in der letzten Jahren auch weitere Netze wie das WWW, die Routerstruktur des Internet oder Stromnetze untersucht [3].
Betrachtet man aus mobilen Endbenutzergeräten bestehende ad-hoc Netze, werden die Eigenschaften des sozialen Netzes zumindest auf Basis von Personendistanz an das Netz vererbt. Somit lässt sich Milgrams Experiment, in dem Briefe auf Grundlage eines Zielortes und Empfängernamens nur durch Weitergabe an Bekannte ausgeliefert wurden, als Routingidee in Store-and-Forward Netzen verwenden. Hier kann durch Hinzunahme geographischer Informationen das Routing unterstützt und mittels Duplikation die Auslieferungslatenz reduziert werden. Der Nachweis, ob dieses über die Zeit definierte Netz tatsächlich Smallworldeigenschaften besitzt, steht aber noch aus.

In Sensornetzwerken kann die Store-and-Forward Technik zur Datenaggregation eingesetzt werden, indem empfangene Informationen zusammengefasst und weitergeleitet werden. Ein Beispiel ist die Maximumsbildung von Sensorwerten. Die SPIN Protokolle [50] verwenden ein vorgelagertes Protokoll, um die Notwendigkeit der Weiterverbreitung von Informationen zu überprüfen. Hierbei wird zuerst die Existenz eines Informationstyps propagiert. Die Nachbarn antworten darauf, wenn sie die Information benötigen. Der Sender sammelt die Anfragen und leitet die Nachricht bei Bedarf per Broadcast weiter. Eine Variante propagiert die Existenz periodisch, so dass auch Nachrichtenverluste und Mobilität berücksichtigt werden. Die SPIN Protokolle verbinden diese Verbreitungsart mit der Datenaggregation.

Schließlich sind auch Unicast und Anycast mittels Store-and-Forward Technik realisierbar. Auf Basis positionsbasierten Greedyroutings wurde dies in [45] unter Berücksichtigung der Gerätemobilität beschrieben. Hier wird im Falle des Greedyroutingfehlers keine Rückfallstrategie verwendet, sondern auf bessere Geräte in der Nachbarschaft gewartet, die sich entweder zum Ziel hinbewegen oder schon näher am Ziel sind. Dabei steht die Überwindung von Netzpartitionen im Vordergrund, d.h. eine Rückfallstrategie würde bei Netzpartitionen fehlschlagen, da kein Pfad zum Ziel existiert. Hier kann es passieren, dass trotz eines existierenden Pfades zum Ziel eine Nachricht nicht ausgeliefert bzw. die Auslieferungslatenz sehr groß wird. Allgemein kann bei Verfahren, die Netzwerkpartitionen mit der Zeit überwinden, keine Aussage über Auslieferungsgarantie oder -dauer getroffen werden, da diese von einer häufig unvorhersagbaren Netzwerkdynamik abhängen. Aussagen sind dann im gewissem Maße möglich, wenn Annahmen an die Netztopologie und -dynamik getroffen werden können.

3

Eigenschaften mobiler Netze

Betrachtet man ein beliebiges ad-hoc Netzwerk sind die Kommunikationsmöglichkeiten äußerst beschränkt, da z.B. unklar ist, wie viele Knoten beteiligt sind, wie groß die Dynamik oder wie stark der Zusammenhang des Netzes ist. Wenn keine Annahmen über das Netz gemacht werden können, ist es nur sinnvoll, mit momentan benachbarten Geräten zu kommunizieren oder aber Nachrichtenverbreitungsverfahren per Infektion zu betreiben. Multihop Kommunikationsverfahren, die eine direkte Nachrichtenauslieferung erfordern, sind in einem solchen Szenario, zumindest theoretisch, auszuschließen. Damit dennoch weitere ad-hoc Kommunikationsverfahren verwendet werden können, müssen einige Eigenschaften des Netzes bekannt sein oder zumindest angenommen werden können. Es ist notwendig, die Eigenschaft des Netzes zu kennen, um zu entscheiden, welche Kommunikationsverfahren verwendet werden können bzw. um die eingesetzten Verfahren an die momentane Netzsituation anzupassen. In der Praxis können diese Entscheidungen im Vorfeld aber auch zur Laufzeit getroffen werden, je nachdem, ob sich die Eigenschaften des Netzes über die Zeit ändern oder erhalten bleiben.

Eine im ad-hoc Routingbereich häufig getroffene Annahme ist, dass das Netz in soweit einen *Zusammenhang* besitzt, dass eine Kommunikation zwischen dem Startknoten und dem Ziel einer Nachricht mit hoher Wahrscheinlichkeit möglich ist. Es wird also davon ausgegangen, dass mindestens ein Pfad vom Start zum Ziel für gewisse Zeit im Netz existiert. Ist das Netz allerdings häufig unzusammenhängend oder existieren gar permanente Netzpartitionen, ist eine direkte Kommunikation zwischen Punkten in verschiedenen Partitionen nicht möglich. In diesem Fall kann aber ein *Mobiler Zusammenhang* einen Nachrichtenaustausch über Partitionsgrenzen ermöglichen. Aufgrund der Gerätemobilität und mittels der Store-and-Forward Kommunikation können Nachrichten über Partitionsgrenzen transportiert werden.

Auch die *Dichte* eines Netzes hat einen direkten Einfluss auf die zu verwendenden Kommunikationsstrategien. In dichten Netzregionen ist aufgrund von Alternativpfaden zwischen Kommunikationspartnern eine höhere Kom-

munikationszuverlässigkeit möglich. Allerdings ist in dichten Netzen die Konkurrenz beim Mediumszugriff höher und somit kann jeder Knoten im Schnitt weniger kommunizieren. Die verwendete Kommunikationsstrategie muss also die momentane Dichte berücksichtigen bzw. die Wahl der Kommunikationsstrategie muss entsprechend erfolgen. Um mit sehr dichten Netzen besser umgehen zu können, ist es auch möglich, das Netz künstlich auszudünnen indem Knoten ihre Sendestärke reduzieren und damit an die Gegebenheiten anpassen.

3.1 Eigenschaften von ad-hoc Netzwerken

- Zusammenhang
- Netzdichte
- Mobilität und Dynamik
- Heterogenität
- Netzausdehnung
- Netzgröße
- Selbstorganisation
- Kooperatives Verhalten und Altruismus

Die *Ausdehnung* eines Netzes im Sinne des Verhältnisses von betrachteter Fläche zur Kommunikationsreichweite der Knoten hat einen direkten Einfluss auf die Länge des kürzesten Pfades zwischen zwei beliebigen Knoten. Je länger ein Pfad, desto größer ist der Einfluss der Dynamik des Netzes. Insbesondere linkbasierte Routingverfahren können nicht beliebig lange Pfade verwenden, weil die Dynamik des Netzes einen zu hohen Protokollaufwand verursacht [10, 94, 97]. Beliebige Ende-zu-Ende Kommunikation in Netzen mit großer Ausdehnung ist auch aufgrund der Kapazitätsaufteilung unter den Knoten nicht sinnvoll realisierbar [69]. Damit in Zusammenhang steht die *Netzwerkgröße* des Netzes, also die Zahl der betrachteten Knoten und die Dichte des Netzes. Ein Netzwerk besitzt bei geringer Ausdehnung und hoher Gerätezahl eine hohe Dichte. Die Netzwerkgröße hat deshalb auch einen direkten Einfluss auf die Wahl der Kommunikationsstrategie. Bei wenigen und lokal interagierenden Geräten kann eine Anwendungsmodellierung einfacher sein, da es möglich ist, alle teilnehmenden Geräte zu kennen und Kommunikationspfade zu diesen aufrecht zu halten. Bei großer Gerätezahl ist es umgekehrt notwendig, Entscheidungen rein aufgrund lokalen Wissens zu treffen, da es nahezu unmöglich ist, eine globale Netzwerksicht zu erstellen.

Der wichtigste Aspekt bei der Betrachtung mobiler ad-hoc Netze ist deren *Dynamik*. Selbst das im Bereich der Sensornetze häufig betrachtete stationäre Netzwerkszenario besitzt Dynamik, da Knoten hinzukommen und wegfallen

können, Knoten kurzzeitig zum Energiesparen aus dem Netz ausfallen oder eine dynamische Sendestärkenanpassung die vorhandenen Kommunikationsverbindungen verändert. Sind die Knoten darüber hinaus auch selbst mobil, erhöht sich die Dynamik des Netzes weiter. Dies kann soweit gehen, dass keinerlei Kommunikation im Netz möglich ist, weil keine Kommunikationsverbindung lang genug existiert, um darüber Nachrichten auszutauschen. Die Dynamik hat aber auch starken Einfluss auf andere Netzeigenschaften. Netzpartitionen können aufgrund der Dynamik plötzlich entstehen oder verschwinden – gerade dünne Netze sind hierfür anfällig. Je länger ein Kommunikationspfad ist, desto wahrscheinlicher wird es, dass er sich während des Nachrichtentransports verändert. Auch bezüglich der Dynamik werden einschränkende Annahmen an das Netz getroffen. So machen es einige Routingverfahren erforderlich, dass sich während der Auslieferung einer Nachricht ein Pfad selten oder sogar nie ändert (so z.B. bei positionsbasierten Faceroutingstrategien). Der Begriff der Gruppenmobilität setzt voraus, dass zwar eine hohe Knotenmobilität vorhanden ist, Kommunikationspartner sich aber einigermaßen gleichförmig bewegen, so dass die Dynamik innerhalb der Gruppe geringer ist. Anwendungen, die auf ad-hoc Interfahrzeugkommunikation [27] oder auf Kommunikation zwischen tragbaren Endbenutzergeräten basieren, setzen zum Teil auf die Tatsache, dass sich die Mobilität auf feste Straßen bzw. Wege beschränkt, um z.B. Geographic Source Routing (GSR [71]) betreiben zu können. Kommunikation von Endbenutzergeräten wird häufig auch innerhalb von Räumen betrachtet, so dass die Bewegungsgeschwindigkeit geringer und die Zahl der längerfristig stationären Geräte höher ist. Das Besprechungsraumszenario, bei dem sich Nutzer über längere Zeit zusammensetzen und nur sporadisch Personen bzw. Geräte hinzukommen oder wegfallen, ist dafür ein Beispiel.

Mobilität verursacht aber nicht nur Probleme. Je nach Anwendung ist es sogar möglich, Vorteile aus der Mobilität zu ziehen. Das Auftauchen von vielen neuen Knoten in Kommunikationsreichweite kann es erübrigen, Nachrichten über große Entfernung im Netz zu routen oder gar zu fluten. Als Beispiel sei hier die En-Passant Kommunikation genannt (siehe Kasten 3.2). Im Falle der Store-and-Forward Kommunikation kann die Knotenmobilität dazu genutzt werden, Nachrichten auch über Netzwerkpartitionsgrenzen hinaus zu transportieren. Die Knotenmobilität wird also als langsamer, aber unter Umständen einzig möglicher Nachrichtentransportmechanismus verwendet.

Allen ad-hoc Netzen gemein ist die Notwendigkeit der *Selbstorganisation*. Alle beteiligten Komponenten wie z.B. Kommunikationsverfahren oder Applikationen, die auf diesem Netz basieren, müssen in der Lage sein, selbstorganisierend mit der Dynamik des Netzes umzugehen. Es ist schlicht unmöglich durch administrative Eingriffe von außen Anpassungen an geänderte Netzwerksituationen vorzunehmen. In großen Netzen bedeutet Selbstorganisation auch, dass globale Ziele primär nur durch lokal vorhandene Information, lokale Operationen und der Kooperation der Geräte zu erreichen sind. Selbstorganisierende Netze sind deshalb durch Dezentralität geprägt. Das bedeutet, dass

3.2 En-Passant Kommunikation [44]

In Netzwerken mit sehr geringer Dichte und hoher Mobilität ist die Kommunikation mit anderen Geräten auf kurze Interaktionsphasen mit temporär benachbarten Geräten beschränkt. Um diese kurze Phase effizient auszunutzen, wird hierbei frühzeitig festgestellt, ob das andere Gerät Interesse an denselben Informationsräumen besitzt. Daraufhin kann dann ein effizienter Abgleich der Informationsräume erfolgen. Diese Kommunikationsart ist daher für Applikationen geeignet, die Informationen verbreiten wollen, die für viele Geräte von Interesse sind. Als Beispielapplikation dient hier *UbiQuiz*, die Prüfungsfragen mit benachbarten Geräten austauscht, wobei die Informationsräume den Prüfungsfächern entsprechen.
Bei der En-Passant Kommunikation wird davon ausgegangen, dass das zusammenhängende Netz, welches die beteiligten Knoten umgibt, aus sehr wenigen, häufig nur aus den kommunizierenden Knoten selbst, besteht. Das Verfahren ist ohne Vorkehrungen in großen zusammenhängenden Netzen ungeeignet, da es hier zu einem kontrollierten Fluten des Netzes degeneriert.

zwar Geräte mit besonderen Aufgaben existieren, sie aber einfach austauschbar sind, da andere Knoten ihre Aufgaben übernehmen können. Aufgaben können Knoten von selbst übernehmen oder sie durch andere Knoten erhalten. Dies kann soweit gehen, dass die Knoten des Netzwerks autonom Fehler erkennen und lösen können und somit ein so genanntes selbstheilendes Netzwerk [100] bilden. Da Fehler durch Kommunikationsabbrüche und Ausfall von Knoten keine Ausnahme sind, sollte jede Komponente Fehlerfälle von vorneherein berücksichtigen und mögliche Lösungen bereitstellen. Häufig wird aber ein geringer, initialer Administrationsaufwand benötigt, um ein Netzwerk, bzw. die darauf betriebenen Anwendungen an die jeweilige Umgebung anzupassen. Im Idealfall beschränken sich administrative Eingriffe zur Laufzeit auf das Installieren weiterer Anwendungen bzw. die Aktualisierung bestehender Software auf einigen wenigen Geräten. Diese Änderungen können dann im restlichen Netzwerk propagiert und die Operationen von den restlichen Geräten autonom durchgeführt werden.

Gerade bei multihop Kommunikationsstrategien, aber auch in den meisten Anwendungen, die auf größeren Netzen mit einer größeren Zahl von Knoten basieren, wird zwingend vorausgesetzt, dass die Knoten sich *kooperativ* verhalten. Das bedeutet, dass Knoten Ressourcen für andere zu Verfügung stellen, z.B. als Router für fremde Nachrichten fungieren oder Dienstleistungen für andere bereitstellen. Häufig wird zusätzlich die Annahme getroffen, dass sogar *altruistisches* Verhalten vorliegt, d.h. die Knoten kooperieren und stellen Ressourcen für andere zur Verfügung, ohne eine direkte Gegenleistung dafür zu erhalten. Während diese Annahme z.B. in Sensornetzen nur natürlich ist, da ihr Zweck ein kooperatives Gesamtziel ist, kann sie bei Endbenutzergeräten eigentlich nicht getroffen werden. Der Einsatz von Anreizsystemen [12, 80] für

die Gerätebesitzer kann es aber ermöglichen, das statt altruistischem Verhalten nur noch kooperatives Verhalten vorhanden sein muss. Nutzer können dabei für das Bereitstellen von Ressourcen mit einer virtuellen Währung bezahlt werden.

Neben den genannten existieren noch eine Vielzahl weiterer, das Verhalten des Netzwerks und die Modellierung von Anwendungen und Kommunikationsstrategien beeinflussender Eigenschaften. Die verwendete Drahtloskommunikationstechnik kann die Nachbarschaftsbeziehungen von Geräten stark beeinflussen, da sie von der Kommunikationsreichweite aber auch davon, ob omnidirektional oder gerichtet kommuniziert werden kann, abhängen. Letzteres hat zur Folge, dass geographisch benachbarte Geräte unter Umständen auf Kommunikationsebene nicht benachbart sind. Sind die betrachteten Netzwerkknoten bzgl. ihrer zur Verfügung stehenden Ressourcen stark unterschiedlich, muss diese *Heterogenität* des Netzwerks berücksichtigt werden. Kommunikationsstrategien können ressourcenärmere Geräte schonen, und Anwendungen können ressourcenreichere Geräte bevorzugt zum Bereitstellen von Diensten nutzen. Dies kann soweit gehen, dass z.B. in Sensornetzwerken gewisse Knoten nur als Informationsquelle dienen und nicht als Ziel oder nur als Router einer Kommunikation verwendet werden.

Ad-hoc Netze können rein auf ad-hoc Kommunikation aufbauen oder sie können auch Infrastrukturen integrieren, und somit *hybride Netzstrukturen* besitzen. Der Bergriff "Infrastrukturen" umfasst in ad-hoc Netzwerken bestehende stationäre Netzwerke, die über fixe Zugangspunkte erreicht werden können, aber auch den Zugang zu zellbasierter Mobilfunkkommunikation. Sind Infrastrukturen vorhanden, werden sie in einigen Anwendungsszenarien als Ausweichlösung verwendet, falls das Ziel der Anwendung im ad-hoc Netzwerk nicht erreicht werden kann. Im Allgemeinen sind damit aber auch Kosten verbunden, so dass der Schwerpunkt auf dem ad-hoc Netzwerk liegt. Infrastrukturen können aber auch wieder selbstorganisierende Netzwerke sein, die für andere Infrastrukturleistungen erbringen und ein fest installiertes System bilden. Ein Beispiel hierfür sind Meshnetzwerke, die häufig als selbstorganisierender Geräteverbund anderen Geräten Zugang zu einem dahinter liegenden, traditionellen Infrastrukturnetz wie dem Internet bieten. Die Verwendung spezieller ad-hoc Netzwerkgeräte, die selbstorganisierend oder administriert Dienste zur Verfügung stellen, kann ebenfalls als Infrastruktur bezeichnet werden. Die Geräte sind meist fest installiert und besitzen nicht notwendigerweise Anbindung an weitere Infrastrukturen.

Sind Geräte in der Lage zu entscheiden, ob sie sich an einem identifizierbaren Ort befinden oder nicht, bezeichnet man sie als *lokalitätsbewusst*. Wenn die Geräte ihre globale Position bestimmen können, spricht man von eindeutiger *Positionierbarkeit*. Dadurch ist nicht nur Lokalitätsbewusstsein möglich, sondern man ist auch in der Lage, sich in einem globalen Koordinatensystem einzuordnen. Positionsbasierte Routingprotokolle benötigen diese Eigenschaft, um überwiegend zustandslos operieren zu können.

Ist Lokalitätsbewusstsein vorhanden, kann dies auch für weitere Anforderungen an die Netzstruktur verwendet werden. So können in einigen Netzen *stabile Orte* identifiziert werden, an denen permanent oder zumindest für abschätzbare Zeit eine gleich bleibende Mindestgerätedichte vorhanden ist und häufig darüber hinaus auch eine geringere Gerätemobilität. An einem stabilen Ort kann zuverlässiger kommuniziert werden, da er sich durch ein geringere Dynamik auszeichnet. Orte lassen sich auch durch ortsbasierte Kommunikation ansprechen, so dass sich dort befindende Geräte oder Dienste einfacher adressieren lassen. Diese Eigenschaften werden beispielsweise von der Marktplatzbasierten Kommunikation (Kasten 3.3) verwendet. Solche Orte können auch durch fest installierte Geräte ihre Stabilität erhalten. Im Falle von Endbenutzergeräten können stabile Orte öffentliche Orte sein, an denen sich häufig viele Personen für längere Zeit aufhalten.

3.3 Marktplatzbasierte Kommunikation [45]

Ein Marktplatz ist ein geographischer Ort, an dem zu bestimmten Zeiten eine hohe Gerätedichte zu erwarten ist. Beispielorte im universitären Umfeld sind Mensen, Cafeterien, Aufenthaltsbereiche und Hörsäle. Applikationen können Objekte an diesen Ort senden, die dort auf Objekte von anderen Geräten treffen und mit diesen autonom vor Ort interagieren können. Somit können vorher unbekannte Kommunikationspartner gefunden werden, ohne dass dafür das gesamte Netzwerk geflutet werden muss. Als Beispielanwendung wurde das Auktionssystem *UbiBay* [34, 32] entwickelt, bei dem sich Bietagenten und Auktionsagenten auf einem Marktplatz treffen und unabhängig vom Nutzer in Verhandlung treten. Das Ergebnis wird an vorher mit dem Agenten vereinbarten Orten, so genannten *Homezones*, hinterlassen, an denen das Benutzergerät in Zukunft anzutreffen ist.

Positionierbarkeit erlaubt es, Geräte in einen globalen Zusammenhang zu setzen. Aufgrund der Tatsache, dass geographische Positionen eine globale Ordnung besitzen, können verschiedene Positionen lokal in Relation zueinander gesetzt werden können. Dies wird bei positionsbasierten Routingverfahren angewendet. Existiert weiteres globales Wissen, wie Umgebungskarten, die Wege und Orte enthalten, lassen sich Annahmen an die Struktur des Netzes stellen. Bewegen sich die Geräte fast ausschließlich auf diesen bekannten Strukturen, lässt sich die Information zum Routing und auch zur Strukturierung des Netzes nutzen, wenn z.B. die oben genannten, stabilen Orte bekannt sind.

Bei der Modellierung von Anwendungen für ad-hoc Netzwerke ist es also essenziell, Annahmen über das gegebene Netz treffen zu können und auch Aussagen über die Ziele einer Anwendung zu treffen. Ohne Vorraussetzungen treffen zu können, sind selbst einfachste ad-hoc Anwendungen nicht realisier-

3.4 Mobile ad-hoc Netzwerke im universitären Umfeld

Die im Vorfeld dieser Arbeit untersuchten Anwendungen basieren auf Netzwerkannahmen wie sie in einem universitären Umfeld in naher Zukunft vorzufinden sein könnten. Die Knoten des ad-hoc Netzes sind hierbei mobile Endbenutzergeräte wie Smartphones, PDAs, PocketPCs oder Subnotebooks, die über eine einheitliche drahtlose Kommunikationsschnittstelle wie IEEE802.11 oder Bluetooth verfügen. Die Geräte sind dabei nahezu "always-on", und die Benutzer verhalten sich altruistisch oder zumindest kooperativ. Eine hinreichende Verbreitung der Gräte vorausgesetzt, kann man an einer Campusuniversität Annahmen über die Netzstruktur treffen. Es lassen sich zum Beispiel Orte identifizieren, an denen zumindest zu bestimmten Zeiten höhere Gerätedichten anzutreffen sind. Als Beispiel dienen hier Veranstaltungsräume, öffentliche Aufenthaltsbereiche, Mensen und Cafeterien. Diese Regionen lassen sich als abgeschlossenes Netzwerk betrachten, das heißt, Applikationen und deren Kommunikation finden nur dort statt, und Geräte können dynamisch hinzukommen oder wegfallen. In den restlichen Bereichen ist das Netz häufig sehr dünn, so dass nur kurzfristige, spontane Interaktion mit anderen Geräten möglich sein kann. Geht man allerdings davon aus, dass Benutzer sich nur auf fest definierten Wegen zwischen den oben genannten Orten bewegen, kann auch eine multihop Kommunikation zwischen diesen Orten bzw. von nahezu beliebigen Geräten zu diesen Orten erfolgen. Hierfür ist häufig das Vorhandensein einer Positionierungstechnologie (z.B. GPS) notwendig. Schließlich lässt sich auch eine Reihe von ad-hoc fähigen stationären Geräten identifizieren, beispielsweise Arbeitsplatzrechner von Dozenten, die selbstorganisierend oder administriert Dienste, wie die Bereitstellung von Veranstaltungsmaterial, zur Verfügung stellen können. Darüber hinaus können auch stationäre Geräte integriert werden, die keinerlei Benutzerschnittstelle zur Verfügung stellen und selbstorganisierend Aufgaben im Netzwerk übernehmen, beispielsweise um stabilen Orten permanent eine Mindeststabilität zu verleihen.

bar. Dient das Netzwerk beispielsweise schlicht als Kabelersatz zur Synchronisation zweier Geräte und ist es aber nicht möglich, eine Kommunikationsverbindung lange genug aufrechtzuerhalten, so ist selbst diese einfache und heute am häufigsten anzutreffende ad-hoc Anwendung nicht durchführbar. Aufgrund der Kombinierbarkeit der verschiedenen Eigenschaften ergeben sich eine Vielzahl, vom Verhalten und der Nutzbarkeit sehr unterschiedlicher Netzwerktypen. Allen gemein ist, dass sie eine Dynamik besitzen, mit der die zu entwerfende Anwendung umgehen muss. Da aus Sicht der Knoten häufig die Netzwerkdynamik mit Mobilität gleichzusetzen ist, können die meisten ad-hoc Netzwerke auch als *mobile ad-hoc Netzwerke* bezeichnet werden, und man kann den Aspekt Mobilität, der eigentlich nur ein Teilaspekt der Dynamik ist, mit der Dynamik gleichsetzen. Somit werden auch Netze berücksichtigt, deren

Dynamik nur aus dem spontanen Entstehen und Verschwinden von Kanten entsteht.

Anwendungen in mobilen ad-hoc Netzwerken können in der Regel also nicht auf traditionelle, d.h. transparente Art, realisiert werden. Im Allgemeinen kann der Umgang mit der Mobilität nicht in einer darunter liegenden Schicht versteckt werden, sondern muss in der Anwendung berücksichtigt werden. Das erfordert zum einen eine andere Herangehensweise bei der Modellierung, zum anderen aber auch neue Anwendungskonzepte, da es in vielen Anwendungsfällen nicht möglich ist, diese aus stationären Netzen zu übertragen. Viele Anwendungen sind in den meisten ad-hoc Netzwerkvarianten nicht realisierbar, weil sie z.B. hohe Anforderungen an Zuverlässigkeit stellen oder hohe Bandbreiten des Netzwerks voraussetzen. Im folgenden Kapitel wird ein Modellierungsansatz für Anwendungen in mobilen ad-hoc Netzwerken vorgestellt und die Anwendbarkeit mit neue Anwendungskonzepte beispielhaft skizziert.

4

Mobile Objekte

In diesem Kapitel wird das der Middleware zugrunde liegende Objektmodell vorgestellt. Hierbei werden die grundlegenden Objekteigenschaften und die identifizierten Objekttypen beschrieben, sowie deren mögliche Ausprägungen diskutiert. Es können, je nach Netzsituation und Anwendung, unterschiedliche Varianten sinnvoll sein.

4.1 Eigenschaften

Die Dynamik eines Netzes wird in den meisten Fällen als gravierendes Problem angesehen, ist sie doch Ursache für unvorhersehbare Topologieänderungen im Netz, spontane Linkabbrüche, Netzpartitionen und das Auftauchen von bisher unbekannten Kommunikationspartnern. Weil die Dynamik quasi allgegenwärtig ist, muss sie auch von Grund auf berücksichtigt werden, sei es bei der Realisierung von Kommunikationsprotokollen oder insbesondere auch bei der Entwicklung von Anwendungen für diese Netzwerke. Jede beteiligte Komponente muss sich der Dynamik bewusst sein und damit umgehen können. Daher ist es nahe liegend, diese Komponenten als mobiles Objekt [103] zu modellieren und somit inhärent den Umgang mit der Mobilität zu berücksichtigen. Sich über die Dynamik des Netzes bewusst zu sein bedeutet aber auch, seine Umgebung *ereignisbasiert* wahrzunehmen. Aktive Objekte müssen deshalb jederzeit in der Lage sein, Ereignisse in der Umgebung wahrzunehmen und unter Umständen darauf zu reagieren. Darüber hinaus ist auch die Kommunikation mit anderen Objekten asynchron und nachrichtenorientiert zu modellieren, da synchrone Kommunikation in einigen Fällen nicht möglich oder aber häufig aus Applikationssicht nicht notwendig ist. Eine synchrone Kommunikation lässt sich ebenso durch eine Folge von asynchronen Kommunikationsschritten realisieren.

Neben der Mobilität ist die *Kommunizierbarkeit* von Objekten eine wichtige Eigenschaft. Objekte können andere Objekte empfangen, selbst an ande-

re Objekte kommuniziert werden oder beides. Da die Kommunikation stark von den Netzwerkeigenschaften, insbesondere der Mobilität, abhängt, können je nach Objekttyp, intendiertem Einsatzgebiet, momentanem Umfeld usw. unterschiedlichste Kommunikationsstrategien zum Einsatz kommen. Darüber hinaus ist es häufig sinnvoll, die Wahl der Kommunikationsstrategie nicht transparent einer Zwischenschicht, also den Kommunikationsschichten oder einer Middleware, zu überlassen, sondern diese Entscheidungen auf Anwendungsebene zu treffen.

Für die Kommunizierbarkeit von Objekten ist es notwendig, dass Objekte global eindeutig identifizierbar sind, damit sie als Ziel einer Kommunikation dienen können und um Objektduplikate bei einem Empfänger aufgrund von Kommunikationsfehlern erkennen zu können. Geräte lassen sich auf zwei Arten global eindeutig identifizieren entweder durch die Vergabe einer global eindeutigen Netzadresse[1] oder durch eine eindeutige Identifikationsnummer. Letztere wird aufgrund individueller Geräteeigenschaften, Uhrzeit und Zufallswerten mittels einer kryptographischen Hashfunktion erzeugt. Lokal erzeugte Objekte können dann mit Hilfe lokal eindeutiger Zeitstempel und der eindeutigen Erzeugerkennung identifiziert werden.

Kommunikation und entfernte Objektrepräsentierung kann mittels *Objektrepräsentanten* realisiert werden. Hierbei wird objektspezifische Information nicht nur im Objekt selber und auf dem momentanen Objektwirt, sondern auch auf anderen Geräten gespeichert. Das hat zum einen den Vorteil, dass ein Teil der Objektinteraktion nicht durch Nachrichtenkommunikation erfolgen muss, zum anderen kann dadurch jedes Gerät lokal relevantes Wissen über dieses Objekt repräsentieren. *Repräsentantenkommunikation* bedeutet, dass die Nachrichteninteraktion mit dem eigentlichen Objekt über den lokalen Repräsentanten erfolgt. Somit ist es möglich, dass bei der Realisierung eines Objektes auch Strategien realisiert werden können, wie in dem Netzwerk mit dem Objekt interagiert werden kann. Darüber hinaus kann der Repräsentant auch Informationen bezüglich der Dynamik oder Mobilität des eigentlichen Objektes zur Verfügung stellen. Damit kann abgeschätzt werden, ob eine Interaktion überhaupt oder mit hoher Wahrscheinlichkeit möglich ist, oder ob es unklar ist, ob eine Interaktion erfolgreich sein kann. Repräsentantenkommunikation kann sogar soweit gehen, dass ein Teil der Interaktion ausschließlich mit dem Repräsentanten erfolgt und deshalb weniger oder gar keine Nachrichtenkommunikation verursacht. So kann ein intelligenter Repräsentant einen Teil der Verhandlungen mit anderen Objekten durchführen und nur das Ergebnis dem eigentlichen Objekt mitteilen oder nur sporadisch Rückfragen stellen.

Wenn ein Objekt sich über seinen momentanen Aufenthaltsort bewusst ist, bedeutet dies, dass es feststellen kann, ob es sich an einem bestimmten

[1] Die 48-bit MAC-Adresse von Ethernetgeräten ist eigentlich aufgrund einer Herstellerkennung und einer Seriennummer eindeutig. Aufgrund von Fehlern, Fälschungen und der Möglichkeit, die Adressen nachträglich zu ändern, ist die Eindeutigkeit aber nur bedingt gegeben.

4.1 Repräsentanten von benachbarten Geräten

Werden direkt benachbarte Geräte durch Beaconnachrichten erkannt, können automatisch Repräsentanten auf den Geräten in Kommunikationsreichweite entstehen. Wenn in der Beaconnachricht per Piggybacking weitere Geräteinformationen verbreitet werden (z.B. Position, Stationarität, Ressourceninformationen, dort vorhandene, wichtige Objekte,...), können diese Informationen über die Repräsentanten direkt zur Verfügung gestellt werden. Kommunikation mit den Geräten kann dann direkt über den Zugriff auf den Repräsentanten erfolgen, und bei Bedarf kann auch die singlehop Kommunikation über wenige Hops erweitert werden. Die Repräsentanten können nach Ausbleiben des Beaconings von selbst wieder verschwinden oder auch bei Bedarf weiterhin auf einem Gerät verbleiben. Handelt es sich z.B. um ein fix positioniertes Gerät, kann das Wissen über die Position eine weitere Kommunikation ermöglichen.

(räumlichen) Ort befindet. Dann ist es dem Objekt möglich, diesem Ort einen Kontext zu geben und darauf aufbauend Kommunikationsentscheidungen zu treffen oder die Kommunikationsstrategie an den Ort anzupassen. Ist sogar Positionierbarkeit vorhanden, ist es möglich, eine ganze Klasse von, in der Regel nahezu zustandslosen, positionsbasierten Routingverfahren einzusetzen. In beiden Fällen sind damit auch Orte zum Zwecke der Kommunikation adressierbar. Neben der Kommunikation kann man das Lokalitätsbewusstsein auch dazu verwenden, den Zustand des Objektes dem Ort entsprechend anzupassen. So können an manchen Orten bestimmte Dienste vorhanden sein, so dass dem Objekt eine viel umfangreichere Funktionalität zur Verfügung steht, als es unter Umständen an anderen Orten der Fall ist. Ein Objekt kann auch bestimmte Erwartungen an einen Ort knüpfen, indem es die Wahrscheinlichkeit, andere Geräte als Interaktionspartner anzutreffen für einen Ort höher erachtet, als an anderen Orten.

Eine Strategie, mit Mobilität besser umgehen zu können oder sogar Nutzen aus ihr zu ziehen, ist, Erfahrungen der Vergangenheit zu verwenden, um Entscheidungen in der Zukunft besser treffen zu können. Daher können mobile Objekte Spuren bei anderen hinterlassen und auch selber eine Spur besitzen (*traceable*). Die Spuren können sich nicht nur auf die Vergangenheit beziehen sondern auch auf die Zukunft. Falls für ein Objekt vorherbestimmt ist, wo es sich im Netzwerk in Zukunft befinden (z.B. geographisch), welchen anderen Objekten es begegnen oder einfach nur zu welchem Gerät es kommuniziert werden soll, kann dies mit einer Spur über die Zukunft dargestellt werden. Eine Spur muss nicht notwendigerweise aus Fakten bestehen, da dies im Allgemeinen nicht garantiert werden kann, z.B. kann ein Objekt nicht immer zu einem bestimmten Gerät kommuniziert werden. Daher wird die Zukunft einer Spur in der Regel als Absichtserklärung interpretiert. Schließlich spiegelt die Gegenwart einer Spur die momentan für ein Objekt wichtige Umgebung wi-

der, die unter anderem zurzeit benachbarte oder assoziierte Objekte oder den momentanen Aufenthaltsort beinhalten kann. Ein wichtiger Teilaspekt von Objektspuren ist die *Objektnachbarschaft*. Benachbarte Objekte sind in der Lage, direkt und mit einem sehr geringen Kommunikationsaufwand miteinander zu interagieren. Benachbarte Objekte befinden sich also auf demselben Gerät oder auf Geräten in direkter Kommunikationsreichweite, oder es liegen wenige und zuverlässige Geräte dazwischen.

4.2 Spuren und verbesserter Altruismus

Die En-Passant Kommunikation gleicht nur Informationen ab, die für beide beteiligte Knoten von Interesse sind. Erweitert man das Verfahren um altruistisches Verhalten, damit auch Informationen gespeichert und verteilt werden können, die für ein Gerät nicht von Interesse sind, müssen die dafür zur Verfügung gestellten Ressourcen effizient verwaltet werden. Zu diesem Zweck können Erfahrungen der Vergangenheit verwendet werden. So werden z.B. häufiger gefragten Informationsräumen mehr Ressourcen zur Verfügung gestellt. Für Geräte, die in der Vergangenheit am häufigsten getroffen worden sind und damit sehr wahrscheinlich wieder getroffen werden, können explizit Informationen aus deren bekannten Interessenbereichen gesammelt werden und somit quasi "mitgebracht" werden.

Objekte besitzen auch die Eigenschaft, dass sie bzw. ihre Repräsentanten dynamisch erzeugt werden können. Darin enthalten ist auch die Eigenschaft, selbstorganisierend entstehen zu können, ohne dass Benutzerinteraktion notwendig ist, bis hin dazu, dass auch dem Gerät vorher nicht bekannte Objekte entstehen können. Wenn eine Applikation einen fehlenden bzw. nicht mehr vorhandenen Dienst bemerkt, kann dieser also neu erzeugt werden, wenn dies mit dem lokal vorhandenen Wissen möglich ist. Dies muss nicht notwendigerweise auf demselben Gerät geschehen, auf dem der fehlende Dienst bemerkt wurde, es ist auch möglich, dass das Starten des Dienstes auf einem benachbarten Gerät initiiert wird.

Aufgrund der Dynamik des ad-hoc Netzwerks sind Informationen über andere Objekte, die nicht auf demselben Gerät vorhanden sind, häufig sehr schnell veraltet. Dasselbe gilt auch für Kommunikationsverbindungen, weil häufig keine Annahme an die Permanenz einer Verbindung getroffen werden kann. Oft ist es aber nicht möglich, ohne weitere Vorkehrungen Aussagen über die Gültigkeitsdauer von Informationen, die Lebensdauer von Objekten oder die Erreichbarkeitsdauer zu treffen. Um diesen Unwägbarkeiten entgegenzuwirken, kann jedes Objekt Lebensdauerinformationen besitzen. Gerade im Falle von Nachrichten bzw. der darin verbreiteten Information ist es sinnvoll, deren Gültigkeit einzuschränken, damit diese nicht beliebig lange im Netzwerk kursieren können. Auch für ein mobiles Gerät, das aus sich heraus meist kei-

4.3 Marktplätze und selbstorganisierende Diensterzeugung

Einige Anwendungen, die marktplatzbasierte Kommunikation verwenden, benötigen Dienste auf dem Marktplatz, um dort Verwaltungsaufgaben durchzuführen. Beispielsweise kann es passieren, dass Geräte lokal nicht in der Lage sind, nicht erlaubte Nachrichtenduplikate aufzulösen. Wenn ein solcher Dienst benötigt wird, aber noch nicht vorhanden ist, kann dieser selbstorganisierend entstehen. Das bedeutet, dass ein geeignetes Gerät gefunden werden muss, welches diesen Dienst zur Verfügung stellen kann. Sobald ein solcher Dienstwirt nicht mehr geeignet ist, muss dieser Dienst selbstorganisierend auf einen anderen Wirt migrieren. (siehe auch [41])

ne eindeutig feststellbare Lebensdauer besitzt, kann aus Sicht anderer Geräte dennoch eine Lebensdauer identifiziert, und die lokal vorhandene Gräteinformationen hiermit verknüpft werden. Damit lassen sich mehrere verschiedene Realisierungen der Lebensdauer identifizieren. Die *Tatsächliche Lebensdauer* eines Objektes beschreibt den Zeitraum, in dem das Objekt existiert. Nach Ablauf dieser Lebensdauer darf es gelöscht werden und existiert somit nicht mehr weiter. *Reaktivierbare Lebensdauer* heißt, dass ein Objekt nach Ablauf verschwinden kann, aber unter Umständen eine Verlängerung der Lebensdauer möglich ist. Dies ist notwendig, wenn keine Abschätzung über die Gültigkeitsdauer von Objektinformationen getroffen werden kann und vor Ablauf der festgelegten Gültigkeit festgestellt wird, dass die Informationen dennoch weiterhin benötigt werden. Objektrepräsentanten können nach Ablauf ihrer Lebensdauer beendet, aber in naher Zukunft wieder erzeugt werden, falls sie wieder benötigt werden oder das repräsentierte Objekt wieder in ihre Nähe kommt. In diesem Falle ist die Objektlebensdauer eingeschränkt auf die lokale Sicht des Gerätes und entspricht somit einer *Beobachteten Lebensdauer*. Schließlich können Objekte wiederholt eine Abschätzung ihrer Lebensdauer propagieren und damit auf anderen Geräten vorhandene Informationen reaktivieren. Dieses Vorgehen bezeichnet man als *propagierte Lebensdauer*. Komplexere Modellierungen können es ermöglichen, Zeitintervalle der Objektgültigkeiten zu definieren, d.h. Objekte können für mehrere bekannte Zeiträume existieren und zwischen den Zeiträumen nicht vorhanden sein.

Schließlich kann noch zwischen *aktiven* und *passiven* Objekten unterschieden werden. Aktive Objekte können mit anderen interagieren, während passive Objekte reine Informationsträger sind. Diese Eigenschaft ist zwar häufig mit der Kommunizierbarkeit eines Objektes verbunden. Dies ist aber nicht notwendigerweise der Fall, da auch aktive Objekte kommuniziert werden und passive Objekte immer lokal auf einem Gerät verbleiben können.

Der Eigenschaftenkatalog für mobile Objekte umfasst nicht nur Kriterien, die zur Modellierung zwingend erforderlich sind. So ist z.B. das Lokalitätsbewusstsein nicht immer notwendig, insbesondere wenn, wie bei der reinen

4.4 Eigenschaften mobiler Objekte

- Mobil
- Mobilitätsbewusst
- Kommunizierbar
- Adressierbar
- Eindeutig identifizierbar
- Hinterlassen oder sammeln Spuren (traceable)
- Lokalitätsbewusst
- Dynamisch erzeugbar
- Aktiv oder passiv
- Ereignisbasiert
- Beschränkte Lebensdauer

En-Passant Kommunikation, nur lokal kommuniziert wird. Auch muss nicht jedes Objekt eine Spur besitzen. Da Spuren auch mit einem Datenaufwand verbunden sind, gibt es Fälle, bei denen die Kosten, z.B. durch das Versenden des Objektes, den daraus gewonnen Nutzen überwiegen. Die tatsächlich benötigten Eigenschaften der mobilen Objekte sind abhängig vom Einsatzgebiet, den Netzeigenschaften und insbesondere von der Anwendung, die sie benötigt.

4.2 Objekttypen

Die größte Dynamik in einem ad-hoc Netzwerk wird durch die beteiligten Geräte verursacht. Sie entsteht zum einen aus der Mobilität der Geräte, zum anderen durch deren spontanes Auftauchen bzw. Wegfallen, was in diesem Modell auch als Mobilität interpretiert wird. Somit stellen Geräte eine Klasse von mobilen Objekten dar. Geräte sind nicht die einzigen Bestandteile, die ein ad-hoc Netzwerk ausmachen und darin Dynamik verursachen bzw. mit dieser Dynamik umgehen müssen. Es lassen sich in diesem Umfeld noch weitere mobile Objekttypen identifizieren.

Insbesondere der Austausch von Nachrichten wird stark durch die Mobilität des Netzwerks beeinflusst. Neben einfacher singlehop Kommunikation, bei der Nachrichten nur zwischen Geräten in Kommunikationsreichweite ausgetauscht werden, existieren zahlreiche multihop Kommunikationsverfahren, die es Nachrichten ermöglichen, sich über das ad-hoc Netzwerk bis zu ihrem eigentlichen Ziel zu “bewegen”. Daher lassen sich auch Nachrichten als mobile Objekte modellieren, unabhängig davon, ob sie nur an direkte Nachbarn oder multihop kommuniziert werden. Erlaubt man über die Nachrichtenkommunikation hinaus auch die Migration von Diensten, sowohl in der Form von

Zustands- als auch echter Kodemigration, erhält man eine weitere Klasse mobiler Objekte. Diese kann ähnlich modelliert werden wie die Klasse der Nachrichten, weil die Migration auf Nachrichtenkommunikation abgebildet werden kann. Dabei sind aber zusätzliche Mechanismen notwendig, die die Migration der Objekte unterstützen.

4.5 Mobiler Zustand

Eine Anwendungsmöglichkeit für den mobilen Zustand sind Dienste, die an geographische Regionen gebunden sind. Dabei muss ein bestimmtes Gerät nicht notwendigerweise permanent vor Ort sein. Der mobile Zustand sucht sich stattdessen vor Ort ein passendes Wirtsgerät und wechselt dieses bei Bedarf. Dies wird zum Beispiel dann notwendig, wenn der aktuelle Wirt die dem Dienst zugeordnete Region verlässt. Der mobile Zustand lässt sich dann ohne Kodemigration modellieren, wenn der Kode auf allen zu nutzenden Geräten vorhanden ist, bzw. nur Geräte genutzt werden, die den Kode schon besitzen, da sie Teil derselben Applikation sind.

Sind die Knoten des Netzwerks in der Lage, Orte zu identifizieren, ist es möglich, den Orten einen Kontext zu verleihen. Ein Beispiel aus dem universitären Umfeld sind Veranstaltungen, die dem Veranstaltungsort zu bestimmten Zeiten einen Kontext geben. Der Kontext ergibt sich daraus, dass dort ein bestimmtes Thema eines bestimmten Fachs behandelt wird, und dass die dort vorhandenen Geräte bzw. deren Nutzer gewisse Eigenschaften besitzen, z.B. Hörer dieses Fachs sind. Dieser Ortskontext lässt sich durch ein Objekt modellieren, welches nicht als spezifische Instanz realisiert sein muss. Dynamik entsteht hier dadurch, dass ein Ortskontext nicht zu jeder Zeit gültig sein muss und darüber hinaus verschiedenen geographischen Orten zugeordnet sein kann. Ortskontexte können auch durch das Binden von Diensten an Orte entstehen. Die Dienste können solange an einem Ort verbleiben, wie das ad-hoc Netzwerk dies ermöglicht, unter Umständen auch mittels Dienstmigration. Ortskontexte können es Anwendungen ermöglichen, Unterstützung zur Selbstorganisation zu erhalten, da ein Ortskontext zusätzliches externes Wissen in das Netz bringen.

Geräte werden häufig in Gruppen eingeteilt, um somit Interesse an bestimmten Informationen bekunden zu können oder Applikationsteilnehmer in einem Kommunikationsschritt durch Adressierung der Gruppe anzusprechen. In ad-hoc Netzwerken lassen sich Gruppenzugehörigkeiten auch zur automatischen Interaktion verwenden. Hierbei werden Gruppeninformationen im Beaconing mitgeteilt, die Interaktionen oder einen automatischen Informationsabgleich anstoßen können. Gruppeneinteilungen werden sowohl bei single- als auch bei multihop Kommunikation verwendet, um die Auslieferung der Nachrichten auf Gruppenmitglieder zu beschränken. Im multihop Fall wird meist

4.6 Ortskontext und Verteilte Skripterstellung [47]

Diese Applikation nutzt den Kontext einer Veranstaltung, um den Nutzern die gemeinschaftliche Erstellung eines Veranstaltungsskripts zu ermöglichen. Aufgrund des Wissens, dass die Applikation nur innerhalb der Veranstaltungen durchgeführt wird, können Annahmen an das Netzwerk getroffen werden, beispielsweise darüber, dass die Wahrscheinlichkeit einer erfolgreichen Interaktion mit anderen Geräten sehr hoch ist und im Fehlerfall eine lokale Lösung möglich ist. Geräte können aufgrund des eigenen geographischen Ortes, der damit feststellbaren Distanz zur Veranstaltung und aufgrund von Annahmen an das Netzwerk über die Möglichkeit und Sinnhaftigkeit einer Teilnahme entscheiden. Die Applikation ist also in der Lage, aufgrund ihrer momentanen Situation bzgl. des Ortskontextes ihre Kommunikation und ihr Applikationsverhalten anzupassen.

das Erreichen aller Mitglieder einer Gruppe im Netz angestrebt. Gruppen sind äußerst dynamisch, da zum einen jederzeit Geräte diese Gruppe betreten und verlassen können, zum anderen in der Regel nur ein kleiner Teil der Gruppe sichtbar bzw. adressierbar ist. Gerätegruppen sind somit auch mobile Objekte.

Schließlich existieren Applikationskomponenten, die sich keiner dieser Gruppen zuordnen lassen. Sie sind in der Regel fix an ihr Gerät gebunden und repräsentieren eine Applikation auf diesem Gerät. Hierunter fallen z.B. graphische Benutzerschnittstellen und auch Verbindungslogik zwischen Objekten, die diese nicht selbst herstellen können. Ebenso sind Teile einer Middlewareplattform zu nennen, auf denen die Applikation basiert und die auf allen Geräten vorhanden sein müssen, wie z.B. Implementierungen von Routingstrategien und Middlewarebasisdienste. Allen diesen Komponenten ist gemein, dass sie sich der Dynamik ihrer Umgebung bewusst sind und selbstorganisierend auf Änderungen der Umgebung reagieren können.

4.7 Typen mobiler Objekte MobileObjects

- Mobiles Gerät – MobileDevice
- Nachricht – MobileMessage
- Mobiler Zustand – MobileState
- Mobiler Ortskontext – MobileLocation
- Mobile Gruppe – MobileGroup
- Mobilitätsbewusste aber fixe Applikationskomponenten – MobileApplication

Nach der Identifizierung einiger Objektklassen wird im Folgenden jede dieser Klassen auf ihre möglichen Ausprägungen bezüglich der Objekteigenschaften untersucht und dargestellt.

4.2.1 Mobile Geräte – MobileDevice

Zur dieser Klasse gehören sämtliche mobilen und auch stationären Geräte des ad-hoc Netzwerks. Geräte sind aufgrund ihrer Mobilität zu bestimmten Zeitpunkten an bestimmten Orten. In der Regel sind die Orte aber nicht vorhersehbar, im allgemeinsten Fall auch nicht identifizierbar. Ob zukünftige Orte vorhersagbar sind, hängt von dem tatsächlichen Typ des Gerätes ab. Sowohl stationäre Geräte, die für lange Zeit an ein und demselben Ort verbleiben, als auch mobile Geräte mit definierten und bekannten Wegen, erlauben eine Vorhersage zukünftiger Aufenthaltsorte. Besitzen die Geräte kein klar definiertes Verhalten, wie es unter anderem bei Endbenutzergeräten der Fall ist, ist die Vorhersage ohne weitere Informationen und Annahmen nicht möglich. Stehen Informationen über potenzielle zukünftige Aufenthaltsorte, z.B. Stundenpläne des Benutzers, zur Verfügung, lassen sich in gewissem Rahmen auch Vorhersagen treffen. Ähnliches gilt für Geräte, bei denen von vergangenem auf zukünftiges Bewegungsverhalten geschlossen werden kann, insbesondere dann, wenn Eigenschaften der Umgebung bekannt sind, z.B. aufgrund von lokal vorhandenem Kartenmaterial der Umgebung.

Geräte können Objekte anderer Typen hosten, d.h. jedes Gerät besitzt eine Menge von Objekten, die auf ihm gespeichert und unter Umständen ausgeführt werden. Dabei kann unterschieden werden, ob ein Objekt dem Gerät gehört, also auf ihm erzeugt worden ist, oder ob es von anderen Geräten erzeugt wurde. Eine weitere Unterscheidung kann in der Zuordnung getroffen werden. Objekte können permanent, d.h. für ihre gesamte Lebensdauer, einem Gerät zugeordnet sein, oder nur für einen bestimmten Zeitabschnitt. Auf Zeit zugeordnete Objekte können aufgrund einer implementierungsspezifischen Entscheidung auf einem Gerät verbleiben und damit dem Gerät tatsächlich zugeordnet sein, oder es kann sich um einen kommunikationsbedingten Aufenthalt handeln, wenn das Objekt Hop-by-Hop übertragen wird. Im letzteren Fall befindet sich ein Objekt quasi “auf der Durchreise”, das eigentliche Ziel des Objektes ist noch nicht erreicht. Der Aufenthalt kann sehr kurz sein, wenn das nächste Empfängergerät sofort nach Empfang gefunden werden muss, oder im Falle von Store-and-Forward Kommunikation auch längerfristiger, da die Suche nach dem nächsten Gerät der Objektweiterleitung ein bestimmtes Zeitintervall in Anspruch nehmen darf.

Das Sammeln von Spuren kann bei diesem Objekttyp am ausgeprägtesten sein, da hierbei in erster Linie nur lokale Ressourcen verwendet werden müssen und keine, im Vergleich dazu sehr viel knapperen, Kommunikationsressourcen. Aufgrund von Mobilität sind andere Geräte für eine bestimmte Zeit direkt benachbart, so dass mit diesen eine Verbindung eingegangen, d.h. kommuni-

ziert werden kann. Somit ist die am häufigsten anzutreffende Spurinformation die der sich momentan in Kommunikationsreichweite befindlichen Geräte. Zu den benachbarten Objekten gehören die momentan gehosteten Objekte, unter Umständen auch Objekte in der direkten Nachbarschaft wie Dienste auf benachbarten Geräten. Zusammengefasst kann festgehalten werden, dass der gegenwärtige Aspekt einer Spur die nähere Umgebung eines Gerätes beschreiben kann, inklusive der eventuell vorhandenen Ortsinformation und des Zeitpunkts, oder allgemeiner, eines kurzen Zeitraums, der die Gegenwart umfasst. Die Vergangenheit einer Spur kann als eine Aggregation der gegenwärtigen Informationen angesehen werden. Im Falle von Gerätenachbarschaften kann die Vergangenheit die Statistik der am häufigsten benachbarten Geräte beinhalten. Angereichert mit Orts- und Zeitinformationen können daraus Schlüsse darüber gezogen werden, ob ein bestimmtes Gerät an einem bestimmten Ort erneut getroffen werden kann, was zum Beispiel bei Nutzern, die dieselben universitären Veranstaltungen besuchen, mit hoher Wahrscheinlichkeit der Fall sein kann. Können für Dienste Aussagen über die Mobilität und die Lebensdauer getroffen werden, kann die Speicherung dieser Information das Wiederfinden eines bestimmten Dienstes ermöglichen. Schließlich lassen Erfahrungen über besuchte Orte in der Vergangenheit Rückschlüsse auf zukünftige Aufenthaltsorte zu. Durch Hinzunahme weiterer Informationen wie Terminkalendern, Stundenplänen aber auch Umgebungskarten können Vorhersagen sehr zuverlässig werden. Wird z.B. festgestellt, dass ein Gerät sich in Richtung eines im Stundenplan vermerkten Raumes bewegt und nutzt es dabei auch noch den Weg, den es meistens zu diesem Raum nimmt, kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass der Nutzer den vermerkten Termin tatsächlich wahrnehmen wird – insbesondere wenn er diesen in der Vergangenheit selten versäumt hat. Zum Themenkomplex der Nutzerbewegungsvorhersage existieren in der Literatur einige viel versprechende Arbeiten [93, 49, 78, 2]. Bewegungsvorhersagen sind bei Geräten die primären Aspekte der Zukunftsspur. Handelt es sich um stationäre oder nach einem vorgegebenen Plan bewegte Geräte wie z.B. Roboter, stehen zukünftige Aufenthaltsorte fest. Stationäre Geräte können entweder ihre Stationarität erlernen oder sie wurde ihnen durch Konfiguration mitgeteilt. Andere Objekte können ebenfalls Teil der Spurzukunft sein. Stationäre und ad-hoc fähige PCs (Heim- oder Bürogeräte), die von mobilen Nutzergeräten häufig getroffen werden, können somit zur Datensicherung oder zur temporären Auslagerung von Daten verwendet werden.

Geräte bzw. darauf befindliche Applikationen sind das primäre Ziel von Kommunikation. Daher sind Geräte inhärent adressierbar und damit auch notwendigerweise eindeutig identifizierbar. Eine Möglichkeit, die Adressierbarkeit zu modellieren, sind Objektrepräsentanten, die einige Kommunikationsdetails, wie z.B. die Wahl des Kommunikationsverfahrens, zum Teil verbergen aber auch weitere Informationen über das Gerät enthalten und zur Verfügung stellen können. Hierunter fallen auch Spurinformationen wie die momentane Position, öffentlich bereitgestellte Dienste, Gruppenzugehörigkeiten, die

4.8 Nutzung von Heim-PCs als Datensenke

Die Applikation UbiQuiz tauscht Prüfungsfragen mit anderen Geräten unter Verwendung der En-Passant Kommunikation. Die Fragen unterliegen bestimmten Nutzungscharakteristiken. So werden sie nur benötigt, solange der Student eine Veranstaltung besucht und sich auf die Semesterabschlussprüfung vorbereitet. Sie werden unter Umständen einige Monate später wieder benötigt, wenn der Student mit Zwischen-/Abschlussprüfungsvorbereitungen in diesem Themengebiet beginnt. In der Zwischenzeit können die Fragen auf dem heimischen PC ausgelagert werden, um auf dem Mobilgerät Platz für andere Information zu schaffen. Falls andere Nutzer diese Daten benötigen, kann die UbiQuiz Applikation entscheiden, Teile dieser Daten wieder an die Universität zu transportieren. Eine Auswahl der Daten kann hier genauso erfolgen wie bei der altruistischen Verwaltung der UbiQuiz Daten auf dem mobilen Gerät.

altruistisch zur Verfügung gestellten Ressourcen, die Gesamtressourcen und deren Auslastung. Wenn es erlaubt ist, können auch nutzerspezifische Informationen wie Namen oder Matrikelnummern enthalten sein. Darüber hinaus kann der Repräsentant für die Speicherung lokal relevanter Daten verwendet werden, beispielsweise lokale Reputationswerte eines Anreizsystems und Referenzen zu Informationen über dieses Gerät in der lokalen Spur.

Die Verwendung von Repräsentanten zur Kommunikation oder zur Aufrechterhaltung einer Kommunikationsverbindung ist den Gegebenheiten des Netzwerks unterworfen. Eine Variante ist hier, dass Gerätерepräsentanten auf benachbarten Geräten erscheinen, sobald Geräte in Kommunikationsreichweite gelangen. Aufgrund das Gerätebeaconings kann der Repräsentant selbstorganisierend entstehen und im Beaconing vorhandene Informationen bereitstellen. Bei Bedarf kann er weitere Informationen vom direkt benachbarten Gerät beziehen. Kommunikation kann in diesem Fall immer über lokale Unicasts ablaufen. Sobald die Kommunikationsreichweite verlassen wird, verschwinden die Repräsentanten in der Regel wieder und können nicht mehr zur Kommunikation verwendet werden. Repräsentanten können auch längerfristig Kommunikation zur Verfügung stellen, wenn das Netzwerk auch zuverlässige multihop Kommunikation erlaubt. Dies ist nur in Ausnahmefällen sinnvoll, unter anderem, wenn eine begonnene Interaktion mit einem nicht mehr benachbarten Gerät abgeschlossen werden soll. Weitere mögliche Ausprägungen der Repräsentantenkommunikation werden im Kapitel 6 dieser Arbeit vorgestellt.

4.2.2 Nachrichten – MobileMessage

Nachrichten bewegen sich mittels der im Netzwerk vorhandenen Geräte von einer Quelle zu einem oder mehreren Zielen. Ziele sind beliebige andere, adres-

sierbare Objekte. Hierbei können je nach Ziel und auch je nach den Entscheidungen des Absenders oder der dort vorhandenen Zielrepräsentanten unterschiedliche Kommunikationsstrategien zum Einsatz kommen. Allen Kommunikationsstrategien ist gemein, dass Nachrichtenobjekte für eine gewisse Zeit bestimmten Geräten zugeordnet werden. Es wird zwischen Nachrichtenerzeuger, als Router verwendeten Zwischenknoten und den eigentlichen Zielgeräten unterschieden. Zwischenknoten, die keine Zielknoten sind, müssen sich altruistisch verhalten, damit Nachrichten empfangen, zwischengespeichert und weitergeleitet werden können. Die Zuordnung der Nachrichten zu einem Gerät ist immer temporär. Nachrichten, die aufgrund der Kommunikationsstrategie sofort weitergeleitet werden müssen, müssen verworfen werden, falls dies nicht möglich ist. Nachrichten, die aufgrund einer Store-and-Forward Kommunikationsstrategie weitergeleitet werden, werden erst nach Ablauf der Nachrichtenlebensdauer verworfen, falls keine Weiterleitungsmöglichkeit gefunden wurde. Die Lebensdauer eines Nachrichtenobjekts ist also für die Behandlung der Nachricht auf altruistisch handelnden Kommunikationszwischenstationen essenziell. Darüber hinaus erlaubt die Lebensdauer aber auch dem sendenden Gerät abzuschätzen, wann spätestens mit einer möglicherweise versendeten Bestätigungsnachricht zu rechnen ist - insbesondere im Falle der Store-and-Forward Kommunikation ist hier ein längeres Zeitintervall möglich. Modellierbar sind dadurch auch Gültigkeitszeiträume für Informationen. Werden Informationen über Kommunikationsstrategien basierend auf Infektion verbreitet, ist es notwendig, dass die Informationen eine bestimmte Gültigkeitsdauer besitzen, damit die Verbreitung danach eingestellt werden kann. Ist die Gültigkeitsdauer nicht abschätzbar, sollte dennoch eine Beschränkung vorgenommen werden. Dann sind Mechanismen notwendig, die die Gültigkeit von Informationen verlängern oder diese Informationen müssen vor Ablauf dieses Zeitpunktes erneut in das Netz eingespeist werden.

Es kann zwischen *aktiver* und *passiver* Gültigkeit unterschieden werden. Die aktive Gültigkeit bezeichnet den Zeitraum, in dem eine Information aktiv verbreitet wird, z.B. durch Fluten oder das Informationsradio. Unter passiver Gültigkeit versteht man den Zeitraum, in dem eine Information gespeichert wird, ohne dass sie weiterverbreitet wird. Dennoch kann sie auf Anfrage anderen Geräten zur Verfügung gestellt werden. Passive Gültigkeit muss nicht notwendigerweise beschränkt sein, da das Verwerfen einer Information in erster Linie nur gerätelokale Auswirkungen hat. Damit ist aber auch eine Abfrage dieser Informationen von anderen nicht mehr immer gewährleistet. Die Modellierung eines Gültigkeitszeitraums kann *relativ* oder *absolut* erfolgen. Relative Gültigkeit wird durch eine Restlebenszeit realisiert. Diese wird vor dem Versenden um die gemessene Aufenthaltszeit auf einem Gerät reduziert und unter Umständen durch eine Transportzeitabschätzung zum nächsten Gerät vermindert. Wird die Nachricht dupliziert, können so für dasselbe Objekt unterschiedliche Restlaufzeiten entstehen, da die Transportzeit pro Zwischenstation nicht genau bestimmt werden kann, und die lokalen Uhren nicht gleichförmig laufen. Absolute Gültigkeit verwendet eine absolute Uhrzeit, an der die Le-

benszeit eines Objekts beendet ist. Wenn keine gemeinsame Zeitbasis existiert, kann es aber zu noch größeren Abweichungen als bei relativer Gültigkeit kommen. Sie kann vernachlässigt werden, wenn die Objektgültigkeit in der Größenordnung von Stunden oder Tagen liegt. Kommunikationsstrategien, die keine echte Zwischenspeicherung erlauben, verwenden häufig alternativ oder zusätzlich einen Gültigkeitszähler für die Anzahl der erlaubten Zwischenstationen zum Ziel. Wird der vorgegebene Wert überschritten, wird auch hier die Nachricht verworfen.

Aufgrund der Lebensdauer bzw. des Lebenszyklus lassen sich zwei Nachrichtentypen identifizieren, *Ereignisse* (oder auch entfernte Ereignisse) und *Datenobjekte*. Ereignisse werden nach dem Ausliefern am Ziel gelöscht. Dieser Nachrichtentyp besitzt nur eine sehr kurze Lebensdauer und muss in der Regel direkt ausgeliefert werden. Store-and-Forward Kommunikation ist daher in den meisten Fällen ungeeignet. Einsatzzwecke sind unter anderem das Versenden einer Mitteilung an andere Objekte, das Mitteilen lokaler Ereignisse oder das Auslösen von Ereignissen bei anderen Objekten. Im Gegensatz dazu besitzen Datenobjekte eine längere Lebensdauer, können auch mit Store-and-Forward verbreitet werden und werden nach dem Empfang nur gelöscht, wenn der Empfänger kein Interesse an dem Objekt besitzt. Hierbei handelt es sich meist um gemeinsam genutzte Informationen, die dementsprechend auch häufig dupliziert werden. Ungewollte Duplikation ist bei diesem Nachrichtentyp unkritisch.

4.9 Ereignisse und Datenobjekte

UbiQuiz tauscht Prüfungsfragen aus, die dupliziert werden und eine lange Lebensdauer besitzen. Die Prüfungsfragen sind also Datenobjekte. Der Empfang einer Prüfungsfrage erzeugt zwar Ereignisse auf einem Gerät, die Frage selbst ist aber kein Ereignis. Lokale Interaktion mit benachbarten Quizapplikationen, beispielsweise die Verwendung eines benachbarten Nutzers als Fragenjoker, basiert auf dem Austausch von Ereignissen. Die Auslieferung einer Anfrage hat in diesem Beispiel sofort zu erfolgen, da hier nach Zeit gespielt wird. Die multihop Auslieferung der Ereignisse ist möglich, wenn sie sofort erfolgt. Die Store-and-Forward multihop Auslieferung ist nur dann möglich, wenn das Spielkonzept geändert wird und der Einsatz eines Jokers weder die Garantie einer Antwort noch eine vorgegebene Antwortzeit besitzt. Dennoch handelt es sich nicht um ein Datenobjekt, da der Empfänger das Objekt nach der Abarbeitung nicht lokal speichert und es bei der Übertragung nicht dupliziert werden sollte. Somit gibt es auch Ereignisse, die eine lange Lebensdauer besitzen können.

Aufgrund von Kommunikationsfehlern, die in einem ad-hoc Netzwerk mit hoher Wahrscheinlichkeit auftreten können, ist es jederzeit möglich, dass unge-

wollt Nachrichtenduplikate entstehen. Ungewollte Duplikation lässt sich leicht auflösen, wenn alle Duplikate dasselbe Geräte als Ziel haben. Dann können durch Caching von eindeutigen Nachrichtenidentifikationen die Nachrichtenwiederholungen aufgelöst werden. Im Falle des Anycasts können Duplikate verschiedene Geräte erreichen. Deshalb die Auflösung von Nachrichtenwiederholungen schwierig, in den meisten Fällen gar unmöglich. Duplikation von Nachrichten kann aber auch bewusst eingesetzt werden, z.B. wenn die Nachricht mehrere Ziele erreichen soll, wie es bei Multicast- oder Broadcaststrategien der Fall ist. Duplikation wird auch eingesetzt, wenn die Wahrscheinlichkeit des Nachrichtenverlusts hoch ist und der Auslieferungserfolg verbessert werden soll. Ein und demselben Ziel wird dabei auf unterschiedlichen Wegen die Nachricht zukommen gelassen (so genannte *Multipathstrategien*). Ein extremes Beispiel hierfür ist das gerichtete Fluten, bei dem die Nachricht in einem Korridor in Richtung des geographischen Ziels auf alle Geräte dupliziert wird.

Egal ob gewollte oder ungewollte Duplikation, es muss immer eine Duplikationsbeschränkung existieren oder auf allen Geräten Vorkehrungen getroffen werden, die eine beliebige Duplikation verhindern. Entsprechende Mechanismen sind die Beschränkung des Duplikationsbaums (also Zahl der Kopien pro Knoten und der maximalen Tiefe), lokales Caching von Nachrichtenkennungen und die Beschränkung der Nachrichtenlebensdauer.

Nachrichten müssen also eindeutig identifizierbar sein, damit überflüssige Nachrichtenwiederholungen erkannt werden können. Unicastverfahren benötigen Kennungen auch, um Bestätigungsnachrichten zuordnen zu können. Zur Kennzeichnung existieren verschiedene Verfahren, die je nach Einsatzzweck sinnvoll sind. Kann eine bestimmte Nachricht einem bestimmten Sendegerät zugeordnet werden, kann die Identifikation mittels eindeutiger Identifizierung des Sendegerätes und einem lokal eindeutigen Zeitstempel erfolgen. Dieses Verfahren lässt sich nicht einsetzen, wenn Nachrichten als identisch angenommen werden können, auch wenn sie aus unterschiedlichen Quellen stammen. Dann muss die Identifizierung über den tatsächlichen Informationsgehalt der Nachricht erfolgen. Um dennoch eine im Vergleich zum Inhalt kurze Identifizierung zu erhalten, können kryptographische Hashverfahren verwendet werden, die aus dem Inhalt eine Kennung erzeugen. Diese ist allerdings nur noch mit sehr hoher Wahrscheinlichkeit eindeutig, da die Möglichkeit von Kollisionen besteht, die aber in der Regel zu vernachlässigen ist. Um mit Hashverfahren eine Kennung erzeugen zu können, muss die Information in gewissem Maße interpretiert werden können. Es muss mindestens entscheidbar sein, welche Teile einer Nachricht zur Bestimmung der Eindeutigkeit gehören und welche nicht.

Nachrichten eigen sich am wenigsten dafür, zusätzliche Informationen über das sie umgebende Netz zu sammeln und damit eine Spur zu besitzen, denn Mehrinformation bedeutet immer einen höheren Kommunikationsaufwand. In den meisten ad-hoc Netzwerken ist Kommunikationsbandbreite nach der Energie die knappste Ressource. Dennoch besitzt eigentlich jede Nachricht eine

Spur, die zumindest den Sender und das Ziel beinhaltet. Soll die Nachricht mehrere Ziele nacheinander oder parallel durch gewollte Duplikation besuchen, ist auch diese Information Teil der Nachrichtenspur. Einige Routingverfahren setzen voraus, dass Nachrichten Spuren sammeln bzw. längere Spuren besitzen. Gerade linkbasierte Verfahren verwenden Routediscoverynachrichten, die Informationen über die besuchten Geräte sammeln müssen. Speziell das Dynamic Source Routing speichert die Route der Nachricht zum Ziel in der Nachricht selbst. Auch aus Applikationssicht können Spuren in Nachrichten sinnvoll sein, da in den Spuren Informationen über das Netzwerk gesammelt werden können, die für die Applikation wichtig sein können, z.B. wenn die Nachricht Dienste oder andere Applikationsteilnehmer entdeckt, die für den Empfänger einer Nachricht interessant sind. Der Nachricht können aber auch Informationen über die Spur des sendenden Gerätes mitgegeben werden, so dass daraus alternative Antwortpfade ersichtlich werden. Dies ist insbesondere dann sinnvoll, wenn eine kurzfristige Vorhersage darüber getroffen werden kann, an welchem Ort sich der Empfänger in naher Zukunft aufhalten wird. Eine Ortsvorhersage ist dann notwendig, wenn entweder der Nachrichtentransport aufgrund von Store-and-Forward Mechanismen im Vergleich zur Gerätemobilität langsamer ist, oder aber die Bearbeitungszeit für eine Antwort längere Zeit in Anspruch nimmt, so dass der Antwortpfad des Routingprotokolls bzw. die Position des Senders veraltet ist.

Auch Nachrichten können eine Art Lokalitätsbewusstsein besitzen. Werden ortsbasierte Kommunikationsverfahren eingesetzt, kann die Spur der Nachricht Lokalitätsinformationen beinhalten, die durch das Kommunikationsverfahren aber auch durch die empfangenden Ziele interpretiert werden. Werden Nachrichten “aktiv” realisiert, wie es beim aktiven Routing [77] praktiziert wird, können Routingentscheidungen von der Nachricht selbst oder mittels Kodierung von Entscheidungsinformationen durch auf dem Gerät vorhandene Routingverfahren getroffen werden. Beispielsweise kann aufgrund von Informationen über die aktuelle Position und Zeit entschieden werden, ein alternatives Ziel anzusteuern. Eine Variante des aktiven Routings ist die Kodierung von Kommunikationsalternativen für den Fehlerfall. So kann beim Nachrichtentransport zu bekannten Orten primär das zustandslose, positionsbasierte Greedyrouting eingesetzt werden und beim Auftreten des Greedyroutingfehlers auf linkbasierte Verfahren zurückgegriffen werden. Der Aufwand, eine Route zu einem bekannten fixen Ziel zu finden, ist in der Regel geringer, als eine Route zu einem beliebigen Gerät zu finden. Nachrichten, die selbst über die Anpassung und Auswahl von Kommunikationsstrategien während der Kommunikation entscheiden können, sind prinzipiell auch als MobileState modellierbar.

4.10 Homezones

Bei der Marktplatzbasierten Kommunikation kann der fixe Ort des Marktplatzes als Nachrichtenziel verwendet werden. Antworten auf eine Nachricht sind aufgrund der Mobilität des ursprünglichen Senders schwer zu realisieren, wenn diese nicht sofort erfolgen können, z.B. aufgrund einer längeren Bearbeitungszeit. Je nach Entfernung des Senders ist eine umfangreiche Routensuche eines linkbasierten Verfahrens nicht mehr tragbar. Kann daher das Zielgerät nicht sofort erreicht werden, ist es möglich, Nachrichten an einem oder mehreren Orten, so genannten *Homezones*, zu hinterlegen. Homezones sind dem Zielgerät bekannte Orte, an denen es sich mit hoher Wahrscheinlichkeit in Zukunft wiederholt aufhalten wird, oder die es mittels Nachrichtenkommunikation mit geringem Aufwand erreichen kann. Diese Orte können den an den Marktplatz gerichteten Nachrichten als Antwortspur mitgegeben werden. Wenn die Antwortzeit in etwa bekannt ist, kann der Anfragesteller aber auch rechtzeitig seine momentane Position mitteilen oder eine Route vom Marktplatz zu seinem Gerät aufbauen. Dies kann bei sehr lange andauernden Anfragen oder Anfragen ohne abschätzbare Antwortzeit auch periodisch erfolgen. Um bei größeren Entfernungen den Kommunikationsaufwand zu reduzieren, kann die aktuelle Geräteposition auch an einem oder mehreren, dem Anfrageobjekt bekannten Orten hinterlegt werden, bis hin dazu, dass hierbei eine Folge von Orten verwendet wird, wenn die Entfernung zum letzten Ort zu groß wird. Hierbei ist natürlich zu beachten, das diese Orte in der Lage sein müssen, diese Informationen zu hosten, es sich dabei also um stabile Regionen des Netzwerks handelt.

4.2.3 Mobiler Zustand – MobileState

Eine spezielle Variante der Nachrichten ist die Klasse der mobilen Zustandsobjekte. Prinzipiell gelten für diesen Objekttyp alle Eigenschaften und Aussagen, die auch für Nachrichten getroffen werden können. Der entscheidende Unterschied ist, dass es sich um aktive Applikationskomponenten handelt, die in der Lage sind, ihren Zustand auf andere mobile Geräte zu migrieren und dort ihre Aufgabe fortzuführen. Damit "wandern" sie wie Nachrichten durch das ad-hoc Netzwerk. Zustandsmigration beinhaltet nicht notwendigerweise Kodemigration. Auf Kodemigration kann verzichtet werden, wenn davon ausgegangen werden kann, dass der MobileState nur auf Geräten aktiv werden muss, die den Kode besitzen, was insbesondere beim Ziel der Kommunikation häufig der Fall ist. Wird in jedem Kommunikationsschritt der Kode mitübertragen bzw. überprüft, ob er übertragen werden muss, verschwimmen die Grenzen zwischen mobilem Kode und Zustandsmigration. Die Migration des aktuellen Zustands kann allein durch das Objekt selbst oder mit Unterstützung einer Middlewarekomponente erfolgen. Das Objekt kann besser entscheiden, was in Zukunft noch benötigt wird, und somit können kleinere Nachrichtenpakete erzeugt werden. Eine Middlewareunterstützung ist dennoch sinnvoll, ins-

besondere können aktive Timeouts, registrierte Ereignisbehandlungsroutinen oder sogar Kommunikationsverbindungen des Objekts automatisch verpackt und wiederhergestellt werden. In einem homogenen Applikationsumfeld besitzen alle beteiligten Geräte alle notwendigen Applikationskomponenten. In diesem Fall ist zustandsbasierte Migration ohne Kodemigration möglich und ist daher schwergewichtigeren Migrationsvarianten vorzuziehen. Der Vorteil gegenüber der Verwendung von reiner Nachrichtenkommunikation ist die einfachere Modellierbarkeit von aktiven mobilen Objekten, insbesondere dann, wenn eine Middlewarekomponente Teile der Zustandsmigration automatisiert. Somit muss beispielsweise nicht nach jeder Kommunikation ein und derselbe Timeout erneut gesetzt werden.

4.11 Ortsgebundene Dienste

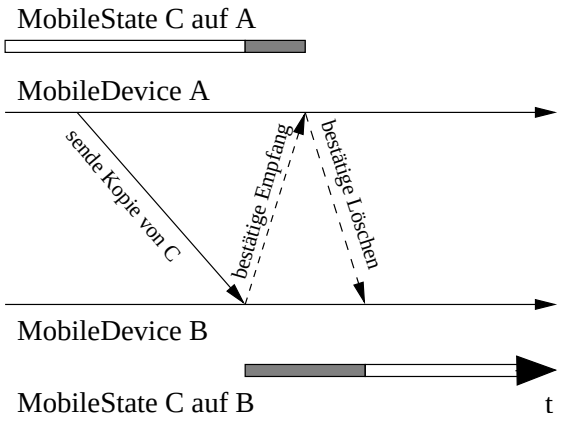
Ein Informationsdienst, der an einem bestimmten Ort ortsspezifische Informationen sammelt, verwaltet und anderen Objekten zur Verfügung stellt, kann als ortsgebundener Dienst realisiert werden. Dieser kann entweder fest an ein stationäres Gerät gebunden sein oder die aktuell an diesem Ort vorhandenen Geräte nutzen. Im letzteren Fall muss aktiv nach alternativen Geräten gesucht werden, wenn der aktuelle Wirt den Ort verlässt. Bei der Wahl eines alternativen Wirts können Kriterien verwendet werden, die zum einen die Ressourcen der Geräte, zum anderen aber auch die Stationarität, die Positionen vor Ort und Zahl der Nachbarn berücksichtigen. Es ist eine Frage der Realisierung, ob dabei Informationen verloren gehen, falls an diesem Ort keine alternativen Geräte vorhanden sind, oder ob die Informationen von den letzten Wirten behalten werden, um sie zu einem späteren Zeitpunkt an den Ort zurückzubringen oder dorthin zu senden. Häufig aber werden die Informationen nicht mehr benötigt, falls kein Gerät mehr vor Ort ist.

Wie bei Nachrichten ist auch hier Duplikation möglich, sollte aber in der Regel vermieden werden, wenn es nicht explizit gewünscht ist. Im Gegensatz zu Nachrichten ist dieser Objekttyp aktiv und kann darüber hinaus von sich aus den Wirt wechseln. So kann es ohne geeignete Maßnahmen zu Konflikten kommen, wenn mit anderen Objekten interagiert wird. Je nach Ziel ist es aber möglich, für gewisse Zeit Informationen darüber zu hinterlassen, dass ein bestimmtes Objekt an diesem Ziel war. Ist das Ziel eine Region, kann eine aktive Suche notwendig sein, um ein eventuell vorhandenes Duplikat zu finden oder zu erkennen, ob es dort war. Keine Kommunikationsstrategie ermöglicht es, in einem Fehlerfall während der Kommunikation festzustellen, ob ein Objekt tatsächlich vollständig migriert wurde oder nicht. Dies liegt darin begründet, dass es unter Umständen nicht mehr möglich ist, bei einem Verbindungsverlust mit einem anderen Host die Verbindung wieder herzustellen, um auf beiden Seiten konsistente Zustände bezüglich *erfolgreich Übertragen* oder *Verlust* herzustellen. Zumindest ist es möglich sicher zu sagen, dass potenziell eine

Duplikation stattgefunden hat, so dass nur dann Maßnahmen zur Duplikationsauflösung getroffen werden müssen.

4.12 Duplikationsabschätzung

Werden mobile Zustandsobjekte Hop-by-Hop übertragen, z.B. wenn eine Store-and-Forward Strategie verwendet wird, muss in jedem Hop ein Verlust des Objektes verhindert werden. Dazu wird das Objekt beim Sender erst dann gelöscht, wenn sichergestellt ist, dass es beim Empfänger angekommen ist. Dabei kann es aufgrund von Nachrichtenverlusten zu Duplikationen kommen, so dass weitere Maßnahmen getroffen werden müssen, damit eine mögliche Duplikation auch in allen Objektduplikaten vermerkt ist. Aus diesem Grund bestätigt der Sender das Löschen des Objektes nach Erhalt der Empfangsbestätigung, damit der Duplikationsvermerk beim Empfänger gelöscht werden kann. Wird die Objektnachricht oder die letzte Protokollnachricht verloren, kommt es zu Duplikationsvermerken, auch wenn kein Duplikat entstanden ist. Da die Duplikationsauflösung aufwendig ist, falls das Ziel der Nachrichten ein beliebiges Gerät einer Region ist (positionsbasierter Anycast), reicht es dann aber aus, nur Maßnahmen bei potenzieller Duplikation zu treffen. Eine detaillierte Beschreibung ist in [45] zu finden.



Ablauf des Protokolls zur Duplikationsabschätzung bei Übertragungsgarantie: Das zu übertragende Objekt C wird vor der Übertragung von Gerät A nach Gerät B als dupliziert (grau) markiert. Auf Gerät A wird das Objekt nach dem Empfang einer Bestätigung gelöscht. A bestätigt B das Löschen, so dass der Duplikationsvermerk von C entfernt werden kann.

Da dieser Objekttyp durch seine Aktivität in der Lage ist, selbst Migrationsentscheidungen zu treffen, kommt er dem Prinzip des aktiven Routings am nächsten. Das bedeutet, die Objekte können selbst entscheiden, welche Kommunikationsstrategie verwendet wird, um den nächsten Wirt zu errei-

chen. Dies kann soweit gehen, dass sie bei jedem Hop bis zum Ziel die Wahl des nächsten Hops beeinflussen und somit die Routingentscheidungen selbst treffen bzw. zumindest beeinflussen können. Somit ist es den Objekten auch möglich, bei einem Fehlschlag einer verwendeten Kommunikationsstrategie alternative Kommunikationsstrategien zu wählen und damit ein Verwerfen ihrer selbst zu verhindern. Das Vorgehen ist insbesondere dann sinnvoll, wenn das Objekt Zielalternativen besitzt oder in der Lage ist, selbst nach Zielalternativen zu suchen. Letzteres ist dann der Fall, wenn noch andere Wirte bekannt sind, die sich als Ziel einer Kommunikation eignen. Es ist also nicht notwendig, dass im Fehlerfall die Kommunikation dieses Objektes vom Ursprungsgerät neu gestartet werden muss und dabei unter Umständen ein Teil des Kommunikationspfades erneut belastet wird.

Insbesondere Zustandsobjekte, die an einen bestimmten Ort gebunden sind, benötigen ein Lokalitätsbewusstsein. Ausreichend ist meist die Unterscheidung, ob es sich an einem Ort befindet oder nicht. Stehen auch Positionen zur Verfügung, kann dies zur Migrationsentscheidung verwendet werden, z.B. um ortszentrale Wirte zu bevorzugen. Sollen Objekte mehrere Orte besuchen, sind gerade positionsbasierte Routingverfahren einsetzbar oder es können um adressierbare Orte erweiterte Linkbasierte Kommunikationsverfahren verwendet werden. Beim Einsatz von positionsbasierten Kommunikationsverfahren ist Lokalitätsbewusstsein notwendig, wenn ein Objekt die Kommunikationsstrategie beeinflussen kann. Wie bei anderen aktiven Objekten kann Lokalitätsbewusstsein auch indirekt durch die Verknüpfung mit einem Ortskontextobjekt vorhanden sein. Das Lokalitätsbewusstsein hängt auch, wie bei MobileMessages, von der zugrundeliegenden Kommunikationsstrategie ab.

Im Gegensatz zu Nachrichten ist ein MobileState nicht nur kommunizierbar sondern auch adressierbar. Die Adressierbarkeit kann aber gerade bei einem Objekt, welches sich sowohl in einem mobilen Netz befindet, als auch selbst mobil ist, problematisch sein, da es passieren kann, dass der momentane Wirt häufig gewechselt wird. Wenn das Objekt allerdings, zumindest für bekannte Zeiträume, ortsgebunden ist, ist es auch über größere Entfernungen im Vergleich zu beliebig mobilen Geräten gut erreichbar. Vor Ort sind die Objekte dann in der Regel über wenige Zwischengeräte erreichbar. Zu diesem Zweck benötigt man allerdings ein Kommunikationsprotokoll, welches die Objektmobilität berücksichtigt. Das bedeutet, dass im Fall von linkbasierten Kommunikationsverfahren der Endpunkt der Kommunikation nicht eine Geräteadresse ist sondern eine Objektadresse und die Routensuche und -reparatur den Wirtswechsel des Objektes berücksichtigen muss. Hierfür lassen sich aber gängige Verfahren wie DSR erweitern. Auch hier ist der Einsatz von Objektrepräsentanten sinnvoll, um die Kommunikationsvarianten zu kapseln bzw. die Abhängigkeit der Kommunikation von der Objektrealisierung zu modellieren. Wie Gerätерepräsentanten können auch diese Informationen über die Objektspur besitzen. Ist also das Objekt zu verschiedenen Zeiten an verschiedenen Orten oder sind Ortsalternativen bekannt, kann der Repräsen-

tant das Auffinden des Ortes übernehmen indem er eine Suchnachricht an alle Orts- oder Wirtsalternativen sendet, um im zweiten Schritt eine Kommunikation mit dem Objekt zu ermöglichen.

4.13 Objekt DSR

Wenn ein Objekt eindeutig adressierbar ist und keine ununterscheidbaren Duplikate existieren, kann das Kommunikationsverfahren DSR [56] nur durch geringe Modifikationen erweitert werden, indem diese Objekte genauso behandelt werden wie Geräte. Deshalb hat es für DSR den Anschein, dass diese "virtuellen" Gerät nur über die tatsächlichen Objektwirte erreichbar sind. Da nach einer Migration dem letzten Wirt meist der nächste bekannt ist und zumindest für gewisse Zeit die Route über ihn aufrechterhalten werden kann, besteht die Möglichkeit, das Verfahren zu verbessern. Existieren austauschbare Objektduplikate, die dieselbe Adresse besitzen, muss DSR mit Alternativpfaden eingesetzt werden. Die Wahl des Pfades kann durch DSR oder durch den Sender der Nachricht getroffen werden. Die Möglichkeit, mehrere Ziele parallel im Sinne eines Multicasts anzusprechen, erfordert aber umfangreichere Änderungen im eigentlichen Verfahren. Ununterscheidbare Ziele einzeln und transparent anzusprechen ist nur sinnvoll, wenn sehr einfache Interaktionen wie Informationsabfragen erfolgen. Aufwendigere Kommunikationsprotokolle setzen in der Regel voraus, dass immer dasselbe Ziel angesprochen wird. Somit muss sich das interagierende Objekt zu Beginn anhand der Hostadresse für ein Ziel entscheiden, und das Ziel darf während der Interaktion den Host nicht wechseln.

Für MobileStates gilt ebenso wie für Nachrichten, dass die Möglichkeit des Spurensammelns wegen des Kommunikationsaufwandes stark eingeschränkt ist. Aufgrund der Aktivität des Objektes ist aber eine im Vergleich zu Nachrichten einfachere Spurenverwaltung möglich. Die gesammelte Spur eines Objektes kann sehr umfangreich sein, solange der aktuelle Wirt nicht gewechselt wird, und erst im Falle der Migration muss eine Zusammenfassung oder Verwerfung der Information erfolgen. Wenn ein solches Objekt ortsgebunden ist, ist zumindest diese Information Teil der Spur. Auch Informationen über Objektduplikation können durch die Spur modelliert werden. Ist das Objekt mit einem bestimmten Gerät assoziiert, was bedeuten kann, dass es mit seinem Erzeuger kommunizieren muss, kann es auch zusätzlich Wissen über dessen Spur beinhalten. Aus den Spurinformationen lässt sich auf einem Host ein Gerätерepräsentant erzeugen, der aufgrund von Informationen über mögliche Aufenthaltsorte, Nachrichtenrelaystationen oder -relayorte diesem Gerät Nachrichten zukommen lassen kann.

Die Lebensdauer der meisten Nachrichten endet mit der Auslieferung am Ziel. Im Gegensatz dazu beginnt beim mobilen Zustand der eigentliche Zweck des Objektes erst bei Auslieferung an das Ziel. Dennoch sollten auch die-

se Objekte eine beschränkte Lebensdauer besitzen, so dass ein Wirt einen MobileState nach Ablauf bedenkenlos löschen kann. Bei langlebigen Objekten ist der Einsatz der propagierten Lebensdauer sinnvoll, so dass andere Objekte nach Ablauf der Lebensdauer Informationen über den MobileState löschen können, wenn keine aktuellere Lebensdauerinformation empfangen wurde. Damit kann auch ermöglicht werden, dass Objektrepräsentanten eines MobileState automatisch verschwinden, sobald längere Zeit kein Kontakt mit dem MobileState vorhanden war. Propagierte Lebensdauer ist insbesondere dann sinnvoll, wenn der MobileState als Dienst eingesetzt wird. Die Lebensdauerverlängerung kann implizit durch Erweiterung der propagierten Nachrichten um Lebensdauerinformationen erfolgen.

Die Migration eines MobileStates kann transparent erfolgen, d.h. eine Middlewarekomponente trifft Migrationsentscheidungen und bildet sie auf Standardkommunikationsverfahren ab. Wendet man aber das Prinzip des aktiven Routings an, übernimmt das Objekt selbst die Entscheidungen. Dies beinhaltet die Wahl des Migrationszeitpunkts, des nächsten Wirts und der Kommunikationsvariante um zu einem Wirt zu wechseln bzw. sich an einen anderen Ort zu begeben. Während die zweite Variante erheblich flexibler ist, vereinfacht die erste die Realisierung des mobilen Zustands. Da die Migrationsentscheidungslogik auch Teil des versendeten Kodes sein muss, falls auch Kodemigration betrieben wird, werden im ersten Fall die Objekte schlanker. Ein Mittelweg zwischen beiden Varianten ist die Realisierung von generischen Objekttypen die unterschiedliches Migrationsverhalten realisieren, oder die Möglichkeit, die Migration durch Zugriff auf Standardvarianten durchzuführen und nur die jeweils notwendigen Zustandsinformationen mitzuführen.

Gerade mobile Zustandsobjekte besitzen die Eigenschaft des selbstorganisierenden Entstehens. Dies wird inhärent bei der Migration eines Zustandes auf ein anderes Gerät betrieben. Mit MobileStates können auch Dienste modelliert werden, die nur bei Bedarf erzeugt werden oder wenn eines der möglichen Dienstreplikate weggefallen ist. Solche Dienste können daher auch inhärent die Eigenschaft haben, Wirte zu wechseln, falls der momentane Wirt nicht mehr geeignet ist oder zum Zwecke der Lastverteilung nach einer gewissen Zeit ein neuer Wirt gesucht werden soll.

4.2.4 Mobile Ortskontexte – MobileLocation

Wenn die Knoten des ad-hoc Netzwerks lokalitätsbewusst sind, ist es möglich, Orte zu erkennen und ihnen einen Kontext zu geben. Unter dieser Voraussetzung lassen sich Objekte modellieren, die den Ortskontext repräsentieren können. An diesen Ortskontext können sich weitere Objekte binden. Insbesondere kommen dafür Objekte in Frage, die einen MobileState repräsentieren, wie es z.B. bei ortsgebundenen mobilen Diensten oder marktplatzbasierter Kommunikation der Fall ist. Auch fest an Geräte gebundene Objekte können diesen Ortskontext nutzen, um ihren Zustand den Ortsgegebenheiten

4.14 Marktplatzaufsicht

Marktplätze können durch mobile Agenten repräsentiert werden, die das Wissen eines Markplatzes verwalten. Diese Marktplatzaufsicht wird dafür verwendet, um Duplikatserkennung auch nach Verlassen eines vermeintlich duplizierten Objektes zu gewährleisten. Die Objekte werden auch für aktive Dienste verwendet, z.B. zur Erzeugung marktplatzweit eindeutiger Tickets, zur zentralen Realisierung von wechselseitigem Ausschluss und vielem mehr. Eine besondere Aufgabe im Marktplatzkontext ist die Lastverteilung, die bei Bedarf Marktplätze aufteilt bzw. zusammenlegt. Die Aufgaben der Aufsicht ist in diesem Anwendungsfall das Auffinden eines geeigneten, geographisch möglichst nahen Ortes zur Aufteilung bzw. Zusammenlegung zu finden, registrierte Marktplatzdienste zu erzeugen bzw. zu beenden und die Durchführung der damit verbundenen Kommunikation, wie die Propagieren dieses Vorgangs. Marktplatzzentrale Informationsdienste ermöglichen das Auffinden von Objekten auf dem Marktplatz ohne das sonst notwendige Fluten. Je nach Wichtigkeit der Informationen der Marktplatzdienste ist es sinnvoll, diese durch mehrere gleichwertige Objekte auf unterschiedlichen Geräten zu realisieren. So kann der nicht unwahrscheinliche Verlust eines Objektes aufgrund der Duplikation aufgefangen werden. Dazu ist ein entsprechender Kommunikationsaufwand notwendig, um die Zustände konsistent zu halten. Generell sollte dennoch immer berücksichtigt werden, dass zumindest ein Teil der Information verloren gehen kann.

anzupassen, wie schon im genannten Beispiel der verteilten Skriptapplikation beschrieben.

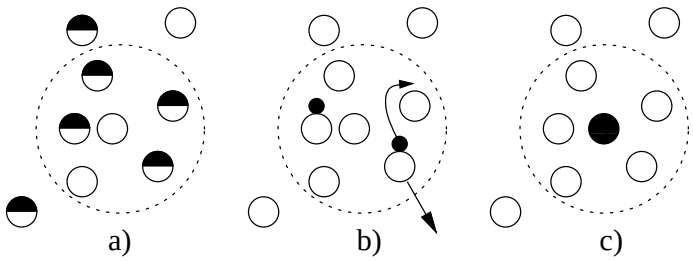


Abb. 4.1. Realisierungsvarianten eines Ortskontexts. a) Der Kontext eines Ortes wird allein durch Repräsentantenobjekte auf Geräten an diesem Ort erzeugt. Das Wissen ist also nur dann vor Ort, wenn mindestens eines dieser Geräte existiert, welches aktiv den Kontext nutzt. b) Der Kontext wird durch MobileState Objekte erzeugt, die sich für die Gültigkeitsdauer vor Ort aufhalten, solange sich Geräte am Ort befinden. c) Der Kontext wird durch ein, in der Regel fixes Gerät erzeugt. Dies kann b) entsprechen, macht aber eine Migration der Kontextobjekte überflüssig.

Ein Ort kann durch ein rein virtuelles Objekt repräsentiert sein. Das bedeutet, dass alle anderen beteiligten Objekte sich dieses Ortskontextes nur bewusst sind, ohne dass eine explizite zentrale Ortskontextobjektinstanz existieren muss. Dennoch ist es sinnvoll, zumindest lokal diesen Ort als Objekt zu realisieren. Dies kann durch Objektrepräsentanten erfolgen, wobei kein real existierendes Objekt vorhanden sein muss. Vielmehr umfasst das virtuelle Objekt alle Objekte vor Ort, die dieses virtuelle Objekt kennen. Damit sind zumindest die Geräte eingeschlossen, die einen Repräsentanten besitzen.

Wird sämtliches Objektwissen nur über Repräsentanten realisiert, ist es schwierig, das Wissen konsistent zu realisieren, wenn es sich nach der Erzeugung des Objektes ändern kann. Bleibt das Wissen hingegen nach dem Erzeugen des Kontext unverändert, entstehen keinerlei Konsistenzprobleme. Wenn nur der Erzeuger dieses Objektes dessen Zustand ändern darf, lässt sich zumindest im Laufe der Zeit ein konsistenter Zustand aller Repräsentanten, die sich an dem Ort befinden, erreichen. Ist es erlaubt, dass mehrere das Objektwissen ändern dürfen, ist dies zwar durch geeignete Maßnahmen durchführbar, der Kommunikationsaufwand ist aber entsprechend hoch. Eine mögliche Maßnahme ist ein verteilter wechselseitiger Ausschluss vor Ort, so dass die nächste Änderung erst dann erfolgt, wenn die letzte konsistent ist.

Um den Kommunikationsaufwand zwischen den Repräsentanten zu reduzieren, bietet es sich an, das Ortskontextobjekt durch ein oder mehrere mobile Zustandsobjekte zu realisieren, die das Objektwissen verwalten (Abbildung 4.1 b). Die Verwaltungsobjekte können aber nur solange vor Ort existieren, wie mögliche Wirte vor Ort sind. Befinden sich keine Geräte mehr vor Ort, sind die Objekte nicht notwendigerweise verloren bzw. müssen nicht immer gelöscht werden. Es ist möglich, diese zu ihrem Ort zurückzusenden, sobald wieder Geräte vor Ort sind und eine Verbindung möglich ist. Die Objekte sind dann aber nicht mehr für andere zugreifbar, da ihr Aufenthaltsort unbekannt ist. Eine mögliche Realisierung des Ortskontextes basiert also auf der Verwendung von MobileState Objekten, die selbstständig an dem Ort des Kontextes verbleiben und bei Bedarf selbstständig den Wirt wechseln.

Ein Ortskontext ist insbesondere dadurch mobil, dass er unter Umständen nur zu bestimmten Zeiten gültig bzw. aktiv ist und im Laufe der Zeit verschiedenen geographischen Orten zugeordnet sein kann. Dies ist beispielsweise der Fall, wenn mit dem Ortskontext eine universitäre Veranstaltung modelliert wird, die zu verschiedenen Zeiten an verschiedenen Orten stattfindet. Ein Kontext kann zu einem Zeitpunkt auch verschiedenen Orten zugeordnet sein. Aus Applikationssicht bedeutet das entweder eine Wahlalternative bzgl. Kommunikation oder die Notwendigkeit, mehrere Orte parallel anzusprechen. Diese Unterscheidung kann entweder in der Applikation selbst, oder in den lokal vorhandenen Repräsentanten des Objektes getroffen werden.

Die Adressierung von Nachrichten an einen Ortskontext kann ein Locationcast bedeuten, so dass alle dem Kontext zugeordneten Objekte vor Ort adressiert werden. Der Nachrichtentransport zum Ort basiert also auf ortsba-

4.15 SPBM als Ortsmulticast

Mehrere geographische Orte können mit einem Multicast gleichzeitig angesprochen werden. Der Multicast kann als eine Folge von positionsbasierten Anycasts an diese Orte realisiert sein. Da es aber möglich ist, dass die Nachrichten zum Teil auf gemeinsamen Pfaden ausgeliefert werden, sind nachrichteneffizientere Verfahren realisierbar. Eine Möglichkeit ist die Adaption des zur Geräteadressierung gedachten Multicasts SPBM [106]. Aufgrund der Ortsadressierung ist die Gruppenpropagierung von SPBM überflüssig und verursacht somit keine permanente Netzlast. Von SPBM wird die Zusammenfassung der Multicastpfade aufgrund einer geographisch hierarchischen Quadtreestruktur verwendet.

sierten Routingverfahren, in der Regel also auf positionsbasiertem Routing. Linkbasierte Verfahren können von einer Ortsadressierung profitieren, insbesondere wenn die adressierbaren Orte in der Zahl geringer und weniger mobil sind, als adressierbare mobile Geräte. Eine zweite Kommunikationsvariante, die vor allem für die Versendung von mobile Zustandsobjekte in Frage kommt, ist der ortsbasierte Anycast. Die Nachricht wird an ein beliebiges Gerät vor Ort ausgeliefert, wobei ähnliche Auswahlkriterien sinnvoll sind, wie bei der Wirtswahl für Zustandsobjekte beschrieben. Vor Ort kann in der Regel über wenige Hops mit anderen kommuniziert werden. Unter der Vorraussetzung, dass Ortskontexte nur für stabile Regionen des Netzwerks angewendet werden, kann dies auch mit vergleichsweise hoher Zuverlässigkeit erfolgen. Im universitären Umfeld sind z.B. Mensen, Cafeterien, Aufenthaltsbereiche und Veranstaltungsräume zu identifizieren, die zu bestimmten Zeiten stabile Regionen sein können. Eine höhere Gerätedichte impliziert aber auch eine höhere Konfliktwahrscheinlichkeit beim Zugriff auf das gemeinsame Medium. Daher müssen Anwendungen auch diesen Umstand berücksichtigen und aufwendige Kommunikation wie häufiges Fluten von Nachrichten vermeiden. Kommunikation kann beim Ortskontext auf den Ort eingeschränkt werden, so dass nur Objekte erreichbar sind, die sich auch im Kontext des Ortes befinden. Deshalb existiert auch für Unicastkommunikation eine bekannte Grenze, so dass nur innerhalb dieser der Kommunikationspartner lokalisiert werden muss. Für linkbasierte Kommunikation bedeutet das, dass Discoverynachrichten nur innerhalb des Ortes verbreitet werden müssen, für geographische Verfahren kann sich der notwendige Positionsinformationsdienst auf den Ortskontext beschränken. Der entscheidende Vorteil ist, dass Netzregionen außerhalb des Ortes nur im Randbereich – und da nur indirekt – durch die dortige Kommunikation betroffen sind.

Die Modellierung eines Repräsentanten zur Kommunikation kann soweit gehen, dass der Übergang zwischen Kommunikation vor Ort und Kommunikation von außerhalb zum Ort nahezu transparent geschieht. Verlässt ein

4.16 Intermarktplatzkommunikation

Für die marktplatzbasierte Kommunikation gibt es die Möglichkeit, dass Objekte von verschiedenen Marktplätzen aus miteinander kommunizieren. Da es sich um fixe und bekannte Orte handelt, kann hier über größere Entfernungen positionsbasiert kommuniziert werden. Vor Ort kann jeweils auf linkbasierte Kommunikation zurückgegriffen werden. Die Endpunkte der Kommunikation sind zwar mobil, dennoch kann aufgrund der fixen Orte, an denen sich die Endpunkte befinden, auch über größere Distanzen kommuniziert werden. Einsatz findet die Intermarktplatzkommunikation bei der Realisierung einer Marktplatzaufsicht zum Zwecke der Lastverteilung. Wenn Marktplätze zusammengelegt werden müssen, erfolgt hier eine Kommunikation zwischen den Aufsichtsobjekten der betroffenen Marktplätze um beispielsweise zu entscheiden, welcher der beiden Marktplätze erhalten bleibt.

kommunizierendes Gerät den Ort, kann dennoch die Kommunikation, zumindest für gewisse Zeit, bestehen bleiben. Nachrichten werden dann erst zum Ort gesendet und können danach innerorts weitergeleitet werden. Dies ist durch eine Verbindung zwischen geographischem Routing, um Nachrichten zum Ort, und linkbasiertem Routing, um Nachrichten innerorts zu transportieren, möglich. Werden Informationen über das Gerät außerhalb des Ortes in der Nachricht gespeichert, ist auch eine direkte Antwort von Objekten im Ortskontext möglich. Die Informationen können die Route der Anfrage oder die Position des Senders sein. Zusammengefasst kann festgehalten werden, dass auch Objekte dieses Typs durch Nachrichten adressierbar sind. In der Regel bedeutet dies aber eine Abbildung auf verschiedene Kommunikationsvarianten, die dann schließlich andere Objekte im Kontext dieses Ortsobjekts adressieren.

Die Spur des Ortskontextes muss zumindest Informationen über die Orte beinhalten, an denen der Kontext gültig ist. Darüber hinaus sind immer auch Zeitinformationen notwendig, um die Gültigkeit des Kontextes auch zeitlich beschreiben zu können. Das bedeutet, dass bei einer rein gerätelokalen Repräsentantenrealisierung alle Geräte initial diese Informationen besitzen müssen. Bei der Erzeugung des lokalen Repräsentanten müssen diese Informationen also vorhanden sein. Obwohl es einige Anwendungsfälle gibt, die mit solchen unveränderlichen Informationen auskommen (z.B. einmalige Veranstaltungen), ist es in der Regel notwendig, dass diese Informationen im Laufe der Objektgültigkeit veränderbar sind. Im einfachsten Fall kann dies durch den Erzeuger vor Ort erfolgen, z.B. den Leiter einer Veranstaltung, so dass eine Veränderung konfliktfrei geschehen kann. Für diese global gültigen Informationen können die Mechanismen angewendet werden, die schon bei der Repräsentantenrealisierung diskutiert wurden. Global gültige Spurinformationen sollten bei möglichst allen Objektrepräsentanten konsistent vorhanden

sein. Ortsglobale Informationen müssen nur bei Repräsentanten innerhalb des Ortes konsistent sein. Wenn von der Applikation benötigt, kann die ortsglobale Spur alle mit dem Ortskontext assoziierten Objekte vor Ort beinhalten. Ein weiteres Beispiel für ortsglobale Information ist die Duplikationsauflösung von mobilen Zustandsobjekten durch Marktplatzaufsichtsobjekte. Hier ist die ortsglobale Information nur in den Aufsichtsobjekten konsistent enthalten. Da allerdings ortsglobale Information ohne real existierende Ortskontextobjekte Kommunikationsaufwand bedeutet, um solche Informationen zu verbreiten und konsistent zu halten, sollte im Allgemeinen nur eine geringe Zahl von ortsglobal bekannten Informationen existieren.

Die Abschwächung des Konsistenzanspruches ermöglicht es, den Kommunikationsaufwand zu reduzieren. In diesem Fall reicht es in der Regel selbst an dynamischeren Orten aus, die Existenz wichtiger Objekte mit einer geringen Periode regelmäßig zu propagieren. Bei Orten mit geringer Dynamik können neu hinzugekommenen Geräten die Informationen mitgeteilt werden. In beiden Fällen wird keine Konsistenzgarantie gegeben. Reduziert man den Konsistenzanspruch weiter, reicht es aus, dass jeder Repräsentant lokal Spurinformationen sammelt, die unter Umständen nur für die lokal vorhandenen Objekte von Interesse sind. Für diese gerätelokalen Spurinformationen über einen Ortskontext muss keine Konsistenz mit Informationen auf anderen Geräten gelten. Somit kann ein Repräsentant auch vergangene Informationen sammeln, die in naher Zukunft von Interesse sein könnten. So können bei der marktplatzbasierten Kommunikation anhand des Wissens über empfangene Nachrichten in der näheren Vergangenheit neu entstandene mobile Zustandsobjekte potenzielle Kommunikationspartner finden, ohne explizit per Nachrichtenkommunikation nach ihnen suchen zu müssen.

Eine weitere Spurvariante, die kommunikationslokale Spur, sammelt Informationen, welche mit geringem Kommunikationsaufwand lokal propagiert werden. Darunter fällt das Sammeln von Informationen über direkt benachbarte Geräte, die Teil des Ortskontextes sind. Geräte können lokal vorhandene Informationen mittels Piggybacking an sowieso zu versendende Nachrichten anhängen, z.B. an periodische Beaconnachrichten. Periodicaststrategien, die ein Scheduling für die Wahl der anzuhängenden Information ermöglichen, erlauben es, auch größere Informationsmengen bei gleich bleibendem Kommunikationsaufwand an Nachbarn zu verbreiten. Allerdings führt eine zu umfangreiche Informationsmenge zu einer hohen Verbreitungslatenz. Daher sollte auch mit dieser Ressource immer sparsam umgegangen werden, um die Konkurrenz verschiedener Anwendungen gering zu halten.

Die Lebensdauer eines Ortskontextes ist durch die in der Spur enthaltenen Zeitinformationen beschränkt. Werden hiermit Veranstaltungen repräsentiert, entspricht dies dem Standardfall. Werden Ortskontexte aber selbstorganisierend an Orten erzeugt, von denen bekannt ist, dass sie zumindest noch in näherer Zukunft eine hinreichende Gerätedichte besitzen, kann die Lebensdauer nicht genau abgeschätzt werden. Hier ist die Verwendung der reaktivierba-

ren Lebensdauer sinnvoll, die verlängert wird, solange die Netzgegebenheiten für den Kontext vorhanden sind.

Ortskontexte können durch das Aufstellen ad-hoc fähiger stationärer Geräte oder durch das Einspeisung von Zusatzwissen im Netz administrativ erzeugt werden. Die Verbreitung von Veranstaltungsinformationen im Hintergrund ermöglicht das selbstorganisierende oder nutzergetriggerte Binden von Applikationen an diese besonderen Orte. Selbstorganisierende Bindung bedeutet, dass eine Applikation für eine gewisse Zeit eine stabilere Region im ad-hoc Netz benötigt und sich dafür selbst einen geeigneten Ort sucht. Verbreitet werden können auch Karten, die Regionen höherer Dichte sowie die Zeiten, wann dies zu erwarten ist, beinhalten. Zur Verbreitung dieser Informationen kann beispielsweise das Informationsradio [37] verwendet werden, wobei die Informationen von mobilen aber auch von fest installierten Geräten ins ad-hoc Netz gespeist werden können. Das Netzwerk kann auch die Existenz solcher Ortsinformationen aus Erfahrungen der einzelnen Geräte erlernen. Hierbei ist aber zu beachten, dass dieses Erlernen je nach Netzwerk sehr lange dauern kann und mit einer hohen Fehlerwahrscheinlichkeit behaftet ist. Darüber hinaus müssen Möglichkeiten zur persistenten Speicherung der Informationen über diese Orte im Netz gefunden werden - beispielsweise durch den Einsatz von stationären Geräten.

4.17 Ortskarten

Ortskarten wie Gebäudepläne, Wegenetze und Straßenkarten können als zusätzliche Informationsquelle verwendet werden, wenn Geräte in der Lage sind, sich aufgrund von Lokalitätsbewusstsein oder Positionierbarkeit in diesen Karten einzuordnen. Mit Hilfe von Karten lassen sich auch Orte ohne Positionsinformation in geographische Relation bringen, so dass Kommunikationsentscheidungen unterstützt werden. Karten lassen sich mit Zusatzwissen über Gerätehäufungen, ad-hoc Dienste oder Infrastrukturen anreichern. Informationen können dabei "administrativ" aufgrund externen Wissens vorgegeben und erweitert werden und vom Gerät selbst oder auch vom ad-hoc Netzwerk kollektiv erlernt werden. Geräte sind darüber hinaus in der Lage, aufgrund der Karten für sie wichtige Orte zu identifizieren, d.h. es lassen sich auch Aufenthaltsstatistiken einbringen. Schließlich können Karten auch zur Nutzernavigation verwendet werden, wenn dieser einen ihm unbekannten Ort erreichen will.

Somit kann eine auf selbstorganisierenden Ortskontexten basierende Anwendung bei Bedarf nach einem schon existierenden entsprechenden Ortskontext suchen und im Zweifel anhand der lokal oder in der Region vorhandenen Informationen selbstorganisierend einen neuen erzeugen. Das Auffinden existierender Ortskontextobjekte kann durch einen Pushansatz erfolgen, wobei die Existenz des Objektes durch sporadisches multihop Beaconing propa-

giert wird. Der Pullansatz kann auf Basis von Fluten wie bei einem Routediscovery bei linkbasierten Verfahren realisiert sein. Kombinierte Verfahren können ähnlich den Lokalisierungsdiensten des positionsbasiertem Routings realisiert sein. Hier kann die Existenz der Ortskontexte zu einem Teil der Geräte (z.B. Nord/Süd, Ost/West Achse [108]) gepusht werden, die dann von anderen Geräten abgefragt werden können. Dieser Verwendungszweck kann im Vergleich zur der Verwendung als Gerätelokalisierung erheblich effizienter genutzt werden, da diese Informationen erheblich weniger dynamisch sind und dies darüber hinaus auch im Verhältnis zur Gerätezahl eine geringere Informationsmenge zur Verfügung stellen muss. Außerdem ist es sinnvoll, Informationen über Ortskontexte regional zu beschränken. Da ein Routing über größere Entfernungen nur geringe Aussichten auf Erfolg hat, sind die Informationen ab einer gewissen geographischen Entfernung nicht mehr sinnvoll verwendbar.

4.18 Marktplatzsuche bei UbiBay

Die Applikation UbiBay setzt voraus, dass Informationen über Orte, die zu gewissen Zeiten eine hinreichende Gerätedichte besitzen, initial bekannt sind. Darüber hinaus werden Informationen über die Regionen, an denen bestimmte Auktionsklassen stattfinden, regional im Hintergrund verbreitet. Wird dann eine neue Auktion gestartet, ist die Information über den entsprechenden Marktplatz entweder lokal vorhanden oder die umliegenden geeigneten Orte werden angefragt, ob dort schon entsprechende Auktionen stattfinden. Wenn keine entsprechenden Auktionsregionen zu finden sind, wird in der nächsten geeigneten Region ein neuer Auktionsmarktplatz erzeugt und die Auktion dorthin gesandt. Ähnlich wird bei der Abfrage der momentan laufenden Auktionen vorgegangen. Hierbei kann die Abfrage auch bewusst dupliziert werden, wenn mehrere Zielorte in Frage kommen bzw. unterwegs weitere potenzielle Ziele gefunden werden.

Stehen keinerlei Ortsinformationen zur Verfügung, kann das Ortskontextobjekt auch zu bestimmten Zeiten an ein bestimmtes Gerät gebunden werden. Dieses Vorgehen ist vor allem dann einsetzbar, wenn kein Routing mittels Positionen benötigt wird, und ein bestimmtes Gerät immer im Kontext des Objektes sein muss. Bei Veranstaltungen, die immer von derselben Person gehalten werden, kann so ein Kontext erzeugt werden. Im Falle der verteilten Skriptapplikation kann hierauf zurückgegriffen werden, wenn weder globale Positionierung noch lokale Ortsidentifizierung möglich ist.

4.2.5 Mobile Gruppen – MobileGroup

Gerätegruppen besitzen in der Regel in sich schon eine hohe Dynamik. Geräte können spontan der Gruppe beitreten oder sie wieder verlassen. Befinden sich die Mitglieder einer Gruppe in einem ad-hoc Netzwerk und sind dort beliebig verstreut, erhält man dadurch einen weiteren Aspekt der Dynamik. In der Regel ist es hier nicht möglich, eine Gruppenstruktur aufrechtzuerhalten, mit der jedes Mitglied einer Gruppe erreichbar ist oder gar jedes Mitglied alle anderen der Gruppe kennt. Eine direkte multihop Multicastkommunikation zwischen allen Gruppenmitgliedern kann erfolgen, wenn das betrachtete ad-hoc Netzwerk nicht zu groß und darüber hinaus auch zusammenhängend ist. Wenn keine Annahmen bzgl. des Zusammenhangs getroffen werden können, wird häufig nur ein Teil der Geräte adressiert. Diese Teilgruppe ist durch die vom momentanen Sender erreichbaren Gruppen definiert. Für umfangreichere Gruppen in Netzen mit größerer Ausdehnung ist es nicht möglich, dass jedes Geräte alle Gruppenmitglieder kennt. Der Grund dafür liegt in der aufwendigeren Gruppenverwaltung, da jedes Gerät eine konsistente Sicht auf die Gruppenmitglieder benötigt. Je nach Anwendung ist dieses globale Gruppenwissen aber sinnvoll oder gar notwendig, beispielsweise wenn der Nutzer alle Gruppenmitglieder kennen muss. Schon in drahtgebundenen Netzen kann ein Multicast keine Auslieferung garantieren, wenn kein Wissen über die Gruppenmitglieder beim Sender vorhanden ist. In ad-hoc Netzwerken reduziert sich die Zuverlässigkeit der Auslieferung aufgrund der Unzuverlässigkeit des Netzes weiter, so dass in der Regel nur ein Teil der Mitglieder erreicht werden kann. Je nach Dynamik und Zusammenhang können nur die direkt benachbarten Mitglieder sinnvoll adressiert werden. Es ist aber häufig möglich, einen Zusammenhang der Gruppe über die Zeit zu definieren. Gerade im Falle von Endbenutzergeräten werden häufig Gruppen definiert, die realen Benutzergruppen entsprechen. Diese Nutzer treffen also auch physisch häufig aufeinander, so dass dies sich in den Verbindungen des ad-hoc Netzwerks widerspiegelt. Somit ist über die Zeit eine Nachrichtenverbreitung innerhalb der Gruppe möglich. Die En-Passant Kommunikation basiert auf diesen realen Benutzergruppen und kann daher Datenobjekte über die Zeit zumindest an einen großen Teil der Gruppenmitglieder verbreiten, ohne dass diese ein permanent zusammenhängendes Netz definieren und ohne dass alle Gruppenmitglieder bekannt sein müssen. Vorraussetzung ist, dass Mitglieder einer Gruppe sich auch geographisch häufig benachbart sind.

Gerade in ad-hoc Netzwerken sind neben der Realisierung der Gruppenkommunikation auch Gruppeninformationen von Interesse, die es ermöglichen, mit der Dynamik des Netzes besser umgehen zu können. Solche Informationen lassen sich wie auch bei anderen Objekten unter dem Bergriff der Spuren zusammenfassen. Allerdings ist es bei Gruppenobjekten im Allgemeinen nicht möglich, eine global eindeutige und veränderbare Spur zu realisieren. Globale Informationen beinhalten beispielsweise alle Mitglieder der Gruppe oder ausgezeichnete Dienste, die der Gruppe zugeordnet sind, aber nicht auf allen

Gruppenmitgliedern vorhanden sind. Zu lokalen Spurinformationen gehören direkt benachbarte Gruppenmitglieder, mit denen lokal interagiert werden kann, und mit der Gruppe assoziierte Objekte. Statistiken können bei fehlendem globalen Gruppenwissen das Wissen über die Gruppe verbessern. Eine Abschätzung über Größe und zeitlichen Zusammenhang kann bei einer Verbreitung von Informationen über die Zeit eine ungefähre Verbreitungsdauer liefern. Gruppen können aber auch definierte Zeiträume besitzen, für die ein Großteil der Gruppe sich an einem, unter Umständen bekannten, Ort aufhält. Darüber hinaus kann aufgrund von lokalen Statistiken abgeschätzt werden, bis wann eine gewisse Gruppenteilmenge in Zukunft getroffen worden ist, um beispielsweise eine Gruppenentscheidung auf Basis einer Gruppenteilmenge über die Zeit durchführen zu können.

Damit die Spurverwaltung für andere Repräsentanten vereinfacht wird, kann ein Gruppenbeitritt auf Zeit erfolgen. Dann muss ein Gruppenaufenthalt verlängert werden, wenn dieser über die aktuelle Beitrittszeit hinausgeht. Damit ist es möglich, dass andere Gruppenrepräsentanten Informationen über dieses Mitglied löschen können, falls längerfristig kein Kontakt mehr vorhanden ist. Dies setzt voraus, dass das Beitrittsende zumindest in einem Teil der propagierten Nachrichten enthalten ist. Ist das Beitrittsende erreicht und liegt keine Verlängerung mehr vor, können auch lokale Informationen und der Gruppenrepräsentant gelöscht werden. Wenn das sofortige Löschen nicht erwünscht ist, kann der Repräsentant zumindest deaktiviert werden, so dass ein späterer Wiedereintritt schneller möglich ist. Passive Gruppenrepräsentanten und daran assoziierte Objekte können auch auf potentere stationäre Geräte ausgelagert werden, um sie später zu reaktivieren.

Kommunikation und Spurverwaltung wird hier immer durch einen lokalen Objektrepräsentant realisiert, da dieser Objekttyp nur virtuell existiert, d.h. keine reale, eindeutige Instanz vorhanden ist. Dadurch kann auch erreicht werden, dass verschiedene Realisierungen des Typs unterschiedliche Kommunikationsstrategien verfolgen. In dünnen Netzen mit geringem Zusammenhang kann die Gruppenkommunikation auf die momentan benachbarten Mitglieder beschränkt werden. Erlaubt das Netz eine multihop Kommunikation, können zur Gruppenkommunikation auch multihop Multicasts verwendet werden. In beiden Fällen beschränkt sich die Spur meist auf die gesehenen Mitglieder und die der Gruppe zugeordneten Objekte. Sie beinhaltet dann also nur Informationen von direkt benachbarten und von aktiven, also nachrichtenversendenden Mitgliedern. Vor kurzem aktive Mitglieder können in der Regel für eine kurze Zeit auch mittels Unicastkommunikation erreichbar sein, so dass es möglich ist, auf eine Multicastnachricht per Unicast weitere Informationen anzufragen. Auch diese Kommunikation kann über den Objektrepräsentant erfolgen. Erlaubt dieser zusätzlich Antworten mittels multihop Unicast auf empfangene Multicasts, kann er entsprechende Vorkehrungen bei der Auslieferung einer Multicastnachricht treffen. Im Falle von linkbasierten Unicasts wird die Route zum Sender vorbereitet, bei positionsbasierten Unicasts wer-

den Positionsinformationen mitgeliefert. Die Informationen werden als Teil der lokalen Spur abgelegt. Unicastkommunikation im Kontext einer Gruppe kann auch zum Verwerfen von Unicast Nachrichten führen, falls der Empfänger kein oder nicht mehr Teil der Gruppe ist. Bei direkt benachbarten Geräten ist die Entscheidung über die Gruppenzugehörigkeit meist schon beim Sender möglich. Multihop Verfahren können diese Entscheidung allerdings erst beim Empfänger treffen.

4.19 Smallworld Multicast

Wird das ad-hoc Netzwerk aus Geräten gebildet, die permanent von ihren Benutzern getragen werden, besitzen die Geräte dieselben Nachbarbeziehungen wie die Benutzer. Somit besitzt das über die Zeit betrachtete ad-hoc Netzwerk Smallworldeigenschaften, wenn das soziale Netz des Nutzers diese auch besitzt. Entsprechen also Gerätegruppen auch Nutzergruppen, die sich durch geographische Nähe auszeichnen, sind diese Cluster im Smallworldnetz. Damit ist eine Gruppenkommunikation über die Zeit möglich, indem Nachrichten weitergegeben werden, sobald Gruppenmitglieder getroffen werden. Im Smallworldnetz lassen sich auch Hubs identifizieren, die zu den Mitgliedern einer Gruppe starke Verbindungen aufweisen und deshalb als Kommunikationshub für diese Gruppe dienen können. Hubs lassen sich aufgrund der Verbindungshäufigkeit zu einer bestimmten Gruppe klassifizieren. Somit kann die Gruppe auch erreicht werden, indem Nachrichten an Geräte weitergeleitet oder dupliziert werden, die solche Hubs darstellen und bessere Hubs sind als das momentane Gerät. Diese Art der Kommunikation basiert also auf den Eigenschaften des Smallworldnetzes und ist somit ein Smallworld Multicast. Eine Variante dieses Multicasts ist die altruistische Verbreitung der En-Passant Kommunikation. Hierbei werden Hubs aufgrund einer Statistik über die Nachbarschaft zu Gruppen identifiziert. Die Nachrichten werden aber von den Hubs “angezogen” und nicht von anderen zur Weiterleitung beauftragt. Die Berücksichtigung der Smallworldeigenschaft kann also verwendet werden, um die Zahl der unnötigen Objektduplikate zu reduzieren, indem nur Nachrichten für bessere Hubs dupliziert werden.

Gruppen müssen, wie alle Objekte, global eindeutig identifizierbar sein. Auch hier kann gelten, dass die Eindeutigkeit mittels der ID des Erzeugergerätes der Gruppe erreicht wird, oder allgemeiner durch die Eindeutigkeit eines dieser Gruppe eindeutig zugeordneten Objektes. Beispiele sind die Kennung eines Ortskontextobjekts, wenn auch die Gruppe derselben Veranstaltung zugeordnet ist, oder die eindeutige Kennung der Applikation, die diese Gruppe definiert hat. Bei Gruppendefinitionen kann auch eine eindeutige, hierarchische Struktur sinnvoll sein, die aber in der Regel global definiert, d.h. durch Administration, vorgegeben sein muss. Vorhandene Hierarchien können vorteilhaft sein, wenn die Gruppen beispielsweise zum automatischen Informationsabgleich verwendet werden, wie im Falle von UbiQuiz. In diesem Beispiel

sind Gruppenzugehörigkeiten nur lokal von Interesse und es wird keine aufwendige multihop Multicastkommunikation benötigt. Verwendung finden lokale Gruppen häufig in Zusammenhang mit kommunizierbaren Objekten, die Hop-by-Hop oder gar durch Store-and-Forward ausgetauscht werden. Hierbei lassen sich Migrationsentscheidungen anhand der Gruppenzugehörigkeit der Geräte frühzeitig vorfiltern und anstoßen.

4.20 UbiQuiz und Gruppenhierarchien

UbiQuiz gleicht Prüfungsfragen automatisch mit benachbarten Geräten ab. Dabei werden einfache Hierarchien verwendet, um festzustellen, ob ein Nachbargerät an denselben Fragen Interesse besitzt. Eine mögliche Hierarchie stellt die Untergliederung in Fach, Veranstaltung und Veranstaltungsthema dar. Diese Hierarchien werden entweder als gegeben vorausgesetzt, oder können bei der Erzeugung neuer Fragen angelegt werden. Letzteres kann aber dazu führen, dass eigentlich gleiche Kategorien unterschiedlich benannt und damit als verschieden behandelt werden.

4.2.6 Mobilitätsbewusste Applikationskomponenten

Alle Applikationskomponenten, die fix mit Geräten verbunden sind und nicht zu anderen Objekttypen gehören, fallen in die Klasse der mobilitätsbewussten Applikationskomponenten. Beispiele sind graphische Benutzerschnittstellen bei Endbenutzerapplikationen oder der Applikationskode, der das Zusammenspiel der anderen Objekttypen auf einem Gerät, welches Teil einer Applikation ist, regelt. In diese Klasse fallen auch Komponenten und Dienste einer Middleware, die auf allen Geräten vorhanden sein müssen, wie z.B. die Implementierungen der verwendeten Routingprotokolle. Geht man von einer Einzelapplikationsumgebung aus in der alle Geräte gemeinsam eine Applikation betreiben, kann davon ausgegangen werden, dass der Kode für diese fixen und auch für alle mobilen Komponenten auf allen Geräten vorhanden ist. Wenn mehrere Applikationen in einem Netz betrieben werden und nicht davon ausgegangen werden kann, dass der Kode aller Applikationskomponenten auf allen Geräten vorhanden ist, müssen Mechanismen existieren, die mit diesem Umstand umgehen können. Eine Möglichkeit ist, den Kode bei benachbarten Geräten oder aus zuverlässigen Quellen, wie fixe und gut erreichbare Geräte, dynamisch nachzuladen, wenn der Kode lokal benötigt wird. Ist die Interaktion auf ein Region beschränkt, wie es innerhalb eines Ortskontextes der Fall ist, kann der Kode nachgeladen werden, wenn eine Geräteteilmenge oder ein zuverlässiges stationäres Gerät vor Ort diese Daten besitzen. Generell verursacht mobiler Kode einen erheblich erhöhten Kommunikationsaufwand. Darüber hinaus ist hiermit auch ein erhöhtes Sicherheitsrisiko gegeben.

4.21 Mobiler Kode

Neben einfacher zustandsbasierter Migration von Objekten gibt es die Möglichkeit, Migration im Sinne von mobilen Agenten zu realisieren, wobei hier neben dem Objekt selbst und dem zugehörenden Zustand auch der Objektkode übertragen wird. Zum Zustand gehört bei schwacher Migration nur der initiale Objektzustand. Bei starker Migration gehört hierzu auch der letzte Ausführungszustand, der, je nach Realisierung, mehr oder weniger stark durch die Agentenplattform automatisiert erstellt wird. Im Gegensatz zum mobilen Zustand wird der Agentenkode immer mit übertragen. Zusammen mit umfangreicher starker Migration führt diese Art der Realisierung immer zu einer hohen Netzbelastung, da die notwendige Übertragen sehr umfangreich wird. Darüber hinaus sind gängige Agentenplattformen sehr schwergewichtig und für ressourcenschwächere Mobilgeräte ungeeignet. Echter mobiler Kode bedeutet auch immer ein Sicherheitsproblem, da es gerade in mobilen Netzen schwierig ist, festzustellen, ob der Kode aus einer zuverlässigen Quelle stammt. Dennoch ist ein Mittelweg wie die gleichzeitig durch die Middleware und das Objekt selbst gestützte, zustandsbasierte Migration mit einer potenzieller Kodeübertragung in Abhängigkeit der Netzumgebung sinnvoll einsetzbar. Der Migrationsaufwand bzgl. Kommunikation wird durch das eigenverantwortliche Erstellen der nächsten Objektinitialisierung reduziert. Die Migration besitzt aber dennoch eine größere Mächtigkeit als die schwache Migration.

In großen mobilen Netzen ist der Ansatz, Geräte transparent zu nutzen, falls sie kein Teil der Applikation sind, tragfähiger. Das heißt, sie dienen nur als transparente Router für kommunizierbare Objekte. Hierbei müssen aber gewisse Komponenten auf allen Geräten existieren, insbesondere müssen alle verwendeten Routingalgorithmen vorhanden sein. Soll eine Spurverwaltung der kommunizierbaren Objekte auch dort möglich sein, sind dafür generische Realisierungen notwendig. Eine teilweise Interpretierbarkeit von unbekannten Objekten erlaubt es dennoch, mit diesen Objekten umgehen zu können, sei es um diese altruistisch zu speichern und zur Verfügung zu stellen oder um diese für einen vermuteten späteren Applikationsbeitritt vorzuhalten. Geräte können auch unbekannte Applikationen erkennen und dem Nutzer Informationen zur Verfügung stellen und um einen möglichen Beitritt zur Applikation durch Sammeln von interpretierbaren Informationen vorzubereiten. Für Applikationen selbst kann eine teilautomatisierte Installation von benachbarten Geräten eingesetzt werden. Das hat zur Folge, dass ein Nutzer, der Interesse besitzt, an einer Applikation teilzunehmen, dies nur kann, wenn eine zuverlässige Quelle in der Nachbarschaft existiert.

Eine weitere Variante integriert nur Geräte, die auch Teil der Applikation sind und ignoriert die Möglichkeit, auch “fremde” Geräte zu nutzen. Ist eine Applikation sowieso nur auf lokale Interaktion mit anderen Geräten angewie-

sen, wie dies bei UbiQuiz der Fall ist, werden die Kommunikationsmöglichkeiten einer Applikation dadurch nicht eingeschränkt. Soll allerdings auch über mehrere Zwischenstationen interagiert werden, bei dieser Variante die Kommunikationsmöglichkeiten stark eingeschränkt, falls keine große Verbreitung der Applikation vorausgesetzt werden kann. In den bisherigen Untersuchungen wurde vorausgesetzt, dass alle Geräte den Kode der untersuchten Applikationen besitzen, wobei nur ein Teil der Geräte die Applikationskomponenten tatsächlich instanziiert.

Die wichtigste Eigenschaft gerätelokaler Komponenten ist, sich der Dynamik der Umgebung bewusst zu sein. Die Realisierung aktiver Objekttypen sollte somit stark ereignisgesteuert sein, da die Dynamik aller anderen Objekte dies erforderlich macht. Das bedeutet, dass der Umgang mit anderen Objekten häufig nur über Ereignisse und asynchrone Nachrichten erfolgen kann, die darüber hinaus, je nachdem wo sich das Objekt befindet, auch mit einer Unzuverlässigkeit der Interagierbarkeit einhergeht. Selbst bei lokal vorhandenen Objekten kann ohne Vorkehrungen nicht immer von einer zuverlässigen Interaktion ausgegangen werden, da beispielsweise MobileStates während einer Interaktion den Wirt wechseln können und Objekte aufgrund eine beschränkten Lebensdauer spontan verschwinden können. Selbst wenn Eigenschaften eines anderen Objektes bekannt sind, kann eine Interaktion häufig nur mit hoher Wahrscheinlichkeit erfolgreich sein. Aus diesem Grund muss immer ein potenzieller Fehlschlag berücksichtigt werden, damit die Applikation immer in einen konsistenten Zustand überführt werden kann. Daher ist eine Modellierung zumindest eines Teils des Kodes als Zustandsautomat sinnvoll, der alle möglichen Einflüsse der Dynamik berücksichtigen muss. Um diese Zustandsautomaten nicht zu komplex werden zu lassen, können als Lösung mehrere parallele und dafür kleinere Automaten realisiert werden. Dies kann soweit gehen, dass jede mögliche Interaktion mit anderen Objekten einen eigenen Zustandsautomaten erhalten kann, wobei alle Zustandsautomaten untereinander interagieren können. Ein einfaches Beispiel für einen Zustandsautomaten, der in einem mobilen Zustandsobjekt der Applikation UbiBay verwendet wird, ist in Abbildung 4.2 dargestellt.

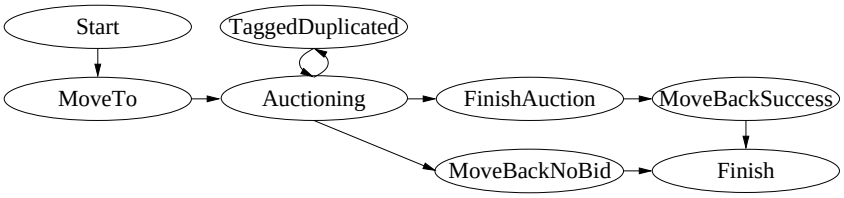


Abb. 4.2. Zustandsautomat eines Auktionsagenten der Applikation UbiBay.

Aufgrund der Dynamik sollte auch lokal ereignisbasiert interagiert werden, beispielsweise durch Signalisierung anderer Objekte oder durch das Erzeugen von Ereignissen. Dies führt dann bei anderen lokal aktiven Objekten zu Zustandsübergängen oder Änderungen in der objektlokalen Datenhaltung. Entfernte Interaktion erfolgt entweder durch die Verwendung von Objektrepräsentanten oder durch selbstständige Wahl und Adaption der Kommunikationsstrategie. Je nach momentanem Applikationszustand und momentaner Umgebung können hier unterschiedliche Strategien sinnvoll sein. Sollen fixe Applikationsobjekte auch über das Netzwerk ansprechbar sein, ist eine entfernte Repräsentierung sinnvoll.

Spurinformationen sind auch bei diesen Applikationskomponenten möglich. Da sie fix mit Geräten verbunden sind, bestehen keine extremen Ressourcenbeschränkungen wie bei kommunizierbaren Objekten. Spurinformationen werden aber in der Regel in den anderen genutzten mobilen Objekten selbst oder deren Repräsentanten gespeichert. Eine Aufgabe der Applikationskomponenten ist die Verwaltung sowie das Zusammenfassen und Löschen dieser Informationen, falls nicht aktive Objekte weiterversendet werden. Die gerätelokale Spurverwaltung kann aber auch immer Teil der Gerätespur selbst sein, so dass lokal gesammelte Spuren als Gerätespur modelliert sind und alle Objekte nur Referenzen auf Teile der Spur besitzen und lediglich einen Teil der Informationen, unter Umständen aufbereitet, mitnehmen. Der Einsatz einer zentralen Spurdatenbank kann sinnvoll sein, wenn viele verschiedene Objekte auch auf Spurinformationen anderer Applikationen zugreifen sollen.

4.3 Zusammenfassung

Die Einsetzbarkeit der einzelnen Objekttypen und ihrer Varianten hängen stark vom Einsatzzweck der Applikation und der Netzwerkumgebung ab, in der die Applikation eingesetzt werden soll. Ortskontexte stellen sowohl den Anspruch der Identifizierbarkeit von Orten, als auch ein gewisses Maß an Netzstabilität an diese Orte. Diese Ansprüche sind erfüllt, wenn sich dort entweder stationäre Geräte befinden, Geräte sich an einem Ort zu einem bestimmten Zweck (Veranstaltung) treffen und die Applikation Teil dessen ist oder wenn der Ort selbst eine hohe Gerätehäufung besitzt und dies der Applikation bekannt ist. Gruppenkontexte können in zwei Formen auftreten. Teile der Gruppe können explizit an einem Ort zusammenkommen und dabei die Applikation verwenden, so dass ein weniger dynamisches Umfeld vorgefunden werden kann. Eine zweite Gruppenvariante ist das genaue Gegenteil, bei dem es das Ziel der Kommunikation ist, möglichst viel Information in kurzer Zeit mit kurzzeitig benachbarten Gruppenmitgliedern auszutauschen (En-Passant). Ist hier die Gruppe stabiler, kann dies nachteilig sein, da diese Kommunikationsform zu einem (kontrollierten) Fluten degeneriert. In dieser Situation sind effizientere Kommunikationsverfahren wie multihop Multicasts einsetzbar. Kombiniert

man beide Konzepte, ist es notwendig, dass der Gruppenkontext oder die darauf basierende Applikation die unterschiedlichen Netzszenarien erkennt und somit zwischen hochdynamischem und geringdynamischem Zustand gewechselt werden kann. Mobile Zustände sind dann einsetzbar, wenn ihre Aufgabe nicht an ein bestimmtes Gerät gebunden ist, dass heißt, wenn sie eine Aufgabe für eine undefinierte Gerätemenge erfüllen. Diese Bindung ist einfach möglich, sobald die Geräte sich in einem Ortskontext befinden, der Ort also als Entscheidungskriterium zur Migration verwendet werden kann. Geräte in einem Gruppenkontext können zwar ein stabiles, nur leicht dynamisches Netz bilden, hier ist aber die selbstorganisierende Migration schwieriger. Es kann schwierig bis unmöglich sein, den MobileState selbstorganisierend in der Gruppe zu belassen, da Gruppenaustritte meist zu einem Kommunikationsverlust mit dem Rest der Gruppe führen. Hier ist also eine Objektduplikation wichtiger. Der Einsatzzweck für an Geräte gebundene MobileStates ist die unverbundene Operation und die Reduzierung des Kommunikationsaufwands. Ersteres wird dann eingesetzt, falls die Interaktion mit einem anderen Objekt erfolgen soll, wobei die Wahrscheinlichkeit sehr hoch ist, dass nur sporadisch eine Interaktion zwischen Gerät und Objekt möglich ist. Dann wird die Interaktionslogik zum Objekt verschickt. Der zweite Fall tritt ein, wenn der Kommunikationsaufwand zur Interaktion erheblich höher ist, als der Kommunikationsaufwand, um den MobileState zu versenden. In beiden Fällen ist die Rückkehr zum Gerät problematisch. Unterscheidbar sind dabei mehrere Varianten. (a) Der Verlust ist kompensierbar oder unkritisch. Die Applikation ist nicht von dem Ergebnis abhängig oder besitzt noch einen weiteren Kanal, um eine getroffene Vereinbarung zu bestätigen. Das Auffinden eines Angebots in einer Mitfahrzentrale wird beispielsweise eine zusätzliche telefonische Kontaktaufnahme zur Folge haben. (b) Es existiert genügend Wissen über das Gerät, so dass dennoch eine spätere Rückkehr ermöglicht werden kann, wenn keine direkte Kommunikation mehr möglich ist. Hier kann der MobileState, unter Umständen dupliziert, an möglichen zukünftigen Aufenthaltsorten warten, oder diese zu bekannten Aufenthaltszeiten ansprechen. (c) Ist eine Rückkehr nicht möglich, kann auch ein zuverlässiger Rückkanal eingesetzt werden, wie die Nutzung einer (kostenpflichtigen) Mobilfunkinfrastruktur[2]. Nachrichten lassen sich primär in zwei Einsatzklassen einteilen. Die Propagierung von Ereignissen basiert in der Regel auf kurzlebigen Objekten, während das Verteilen von Datenobjekten auf langlebigen Objekten basiert. Soll ein Ereignis genau ein bestimmtes Gerät adressieren, ist die multihop Weiterleitung nur unter bestimmten Umständen sinnvoll: Wenn das Netz dazwischen oder der Empfänger eine geringe Dynamik besitzt und mögliche Kommunikationswege[3] zum Empfänger bekannt oder mit geringem Aufwand auffindbar sind. Werden mehrere Geräte adressiert, kann die Auslieferung mit Verfahren erfolgen, die Nachrichtenduplikation verwenden. Ziel ist hierbei, die Zahl der Duplikatio-

[2] Beispielsweise UMTS oder GPRS. In diesem Fall müssen Mechanismen realisiert werden, so dass der Empfänger die entstandenen Kosten trägt.

[3] Route(n) oder geographische Informationen

nen gering zu halten bzw. die Duplikation möglichst spät durchzuführen. Eine höhere Duplikation erhöht aber auch die Auslieferungswahrscheinlichkeit, da unter Umständen redundante Alternativpfade verwendet werden können. Duplikation ist bei der Adressierung mehrerer Empfänger aber auch einfacher zu tolerieren als im Falle eines Unicasts. Unterscheidbar ist zum einen die sofortige Verbreitung einer Nachricht, wobei versucht wird, in möglichst kurzer Zeit möglichst viele Empfänger zu erreichen, die eine bestimmte Eigenschaft wie Gruppenzugehörigkeit oder Aufenthaltsort erfüllen. Dabei müssen häufig viele unbeteiligte Geräte involviert werden. Zum anderen kann die langsame Verbreitung eingesetzt werden, wobei der Aufwand entweder, wie im Falle des Informationsradios, im Hintergrund betrieben wird, oder fast ausschließlich Geräte involviert werden, die auch Interesse an den Informationen besitzen, wie im Falle der En-Passant Kommunikation. Die Darstellung der Geräte als Objekte ist unabhängig von der Netzsituation. Demgegenüber ist der Einsatz von Gerätерepräsentanten zur Kommunikation stark von der Netzsituation und den Ansprüchen der Anwendung abhängig. Eine Interaktion mit direkten Nachbarn in Kommunikationsreichweite ist nur abhängig von den beiden Interaktionspartnern und besitzt eine höherer Zuverlässigkeit als eine multihop Interaktion. Daher ist eine lokale Repräsentierung der benachbarten Geräte immer sinnvoll, da hierbei auch Information zur Verfügung gestellt werden können, die aus einem periodischen Beaconing stammen. Eine Verwendung des Repräsentanten zur multihop Kommunikation ist möglich, wenn in trägeren Netzen über wenige Hops kommuniziert wird, oder wenn das Aufbauen einer Kommunikationsroute aufgrund von Wissen über das Gerät mit geringem Aufwand möglich ist. Hier ist insbesondere das Wissen über tatsächliche und mögliche Aufenthaltsorte wichtig, was es ermöglicht, das Gerät im Netz, im Vergleich zu Fluten, mit geringem Aufwand wieder zu finden.

5

Objektspuren

Eine häufig anzutreffende Eigenschaft in ad-hoc Netzwerken ist, dass Nachrichten oder mobile Geräte zumindest bei Geräten temporäre Spuren hinterlassen. So werden in vielen Fällen ein- oder gar multihop Nachbarschaftslisten gepflegt, unter anderem, um lokal Routingentscheidungen treffen zu können. Nachrichtenidentifikationen werden zwischengespeichert, um Duplikation oder Wiederholung erkennen zu können, was insbesondere beim Fluten des Netzwerks mit einer Nachricht notwendig ist. Um den Informationsabgleich zwischen zwei Geräten zu automatisieren, werden feste Bindungen verwendet, d.h. beide Geräte müssen einander bekannt gemacht werden und diese Bindung speichern, um in Zukunft den Abgleich autonom durchführen zu können. Auf der anderen Seite steht das Vorgehen, mittels Nachrichten Teile der momentanen Struktur eines Netzwerks zu erlernen. So nutzen Linkbasierte Routingverfahren Routediscoverynachrichten, um eine Route vom Sender zum Empfänger zu finden. Bei positionsbasiertem Greedyrouting kann der Greedyroutingfehler umgangen werden, indem per Tiefen- oder Breitensuche nach näheren Geräten zum Ziel gesucht wird [102].

Wie schon im vorangegangenen Kapitel beschrieben wurde, können Spuren viel umfangreicher eingesetzt werden, um mit der Dynamik des ad-hoc Netzwerks umgehen zu können. In der Regel ist die Spurinformation stark applikationsspezifisch, so dass die konkrete Spurrealisierung häufig nur vom Anwendungsentwickler selbst durchgeführt werden kann. Der Nachteil ist dabei, dass häufig von unterschiedlichen Applikationen dieselben Informationen gesammelt und dadurch unnötig Ressourcen verbraucht werden. Darüber hinaus kann es passieren, dass beispielsweise in einer Nachricht dieselben Informationen doppelt vorhanden sind, falls auf der einen Seite das Routingverfahren Informationen über den Weg einer Nachricht sammelt und dieselbe Information auch von der kommunizierenden Applikation benötigt wird. Daher ist es notwendig, zumindest gewisse grundlegende Spurinformationen zu identifizieren und generische Mechanismen zu entwickeln, diese zu erstellen und zu

verwalten. Für rein anwendungsspezifische Spuren sind Entwurfsmuster notwendig, die dem Anwendungsentwickler die Spurrealisierung vereinfachen.

Eine Spur ist damit im ersten Schritt nur ein Entwurfsmuster, welches bei der Realisierung von mobilen Anwendungen eingesetzt werden kann. Anwendungskomponenten können diese nutzen, um Entscheidungen zur Interaktion mit anderen, zur Adaption der Kommunikation und zur Selbstorganisation zu treffen. Im Folgenden werden einige grundlegende Eigenschaften von Spurinformationen identifiziert, die bei der Realisierung eines Musters berücksichtigt werden sollten. Darüber hinaus werden verschiedene Ansätze, Spurinformationen im Netz zu verbreiten, vorgestellt. Abschließend werden verschiedene Darstellungs- und Verwaltungsarten für Spurinformationen diskutiert.

5.1 Grundlegende Eigenschaften

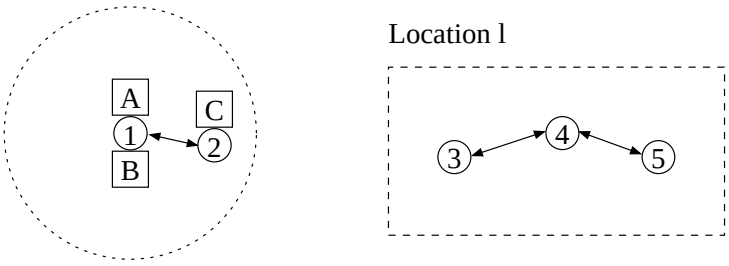


Abb. 5.1. Objektnachbarschaften definieren sich aufgrund von Gerätezugehörigkeit und der Gerätenachbarschaften. Objekte eines Gerätes sind immer benachbart (A, B und Geräteobjekt 1). Nachbarschaft aufgrund von Kommunikationsreichweite der Geräte 1 und 2 besteht z.B. zwischen den Objekten $\{A, B, 1\}$ und $\{C, 2\}$. Nachbarschaft aufgrund eines identifizierbaren Ortes l besteht zwischen den Geräten $3, 4$ und 5, wobei es in der Praxis wenig Sinn macht, die Ortsgröße im Vergleich zur Kommunikationsreichweite zu groß zu wählen, da der Kommunikationsaufwand mit der Entfernung steigt.

Ein wichtiger Aspekt bei der Betrachtung von Spuren ist der Begriff der Nachbarschaft. Objektnachbarschaft ist vom Prinzip die Verallgemeinerung der Gerätenachbarschaft, die in ad-hoc Netzwerken verwendet wird, um die Geräte zu speichern, mit denen lokal interagiert werden kann. Die Nachbarschaft eines Objektes beinhaltet alle Objekte, die momentan von Interesse sind. Dies können zum Beispiel die Assoziierung eines mobilen Zustands zu einem Gerät oder zu einem Ortskontext sein, die aktuell benachbarten Objekte desselben Typs oder Applikationen auf benachbarten Geräten, mit denen interagiert wird. Im Allgemeinen lässt sich über Nachbarn eines Objekts die

Aussage treffen, dass sie im Sinne des Kommunikationsaufwandes benachbart sind. Sie befinden sich also entweder auf demselben Gerät, auf direkt benachbarten Geräten oder nur wenige Hops entfernt. Eine wichtige Teilmenge der Nachbarschaft sind die Objekte, die als Grund für den aktuellen Aufenthaltsort angegeben werden können. Anders ausgedrückt kann jedes Objekt, welches als Teilziel eine Interaktion mit anderen Objekten besitzt, seine Interaktionsziele innerhalb der Spur modellieren.

5.1 UbiBay Bietagenten mit mehreren Zielen

Gebote werden bei der Auktionsapplikation UbiBay durch MobileStates, die Bietagenten, auf Marktplätze gesendet. Bietagenten können selbstständig auf Auktionen bieten. Sie können auch mehr als einer Auktion zugeordnet sein und parallel beide bearbeiten. Somit haben diese Objekte mehrere zukünftige Ziele, nämlich die Interaktion mit mehren Auktionsagenten. Migriert nun eines der Ziele auf einen zur Lastverteilung erzeugten Marktplatz an einen anderen Ort, wird der Bietagent dupliziert und die Aufgaben und Ziele werden aufgeteilt.

Im Allgemeinen werden Spuren so modelliert, dass sie Vergangenheit, Gegenwart und Zukunft beschreiben können. Diese zeitlichen Abschnitte müssen nicht notwendigerweise alle vorhanden sein. Es ist auch möglich, dass Objekte eine leere Spur besitzen. Elemente der Spur bewegen sich von der Zukunft über die Gegenwart in die Vergangenheit, wobei sich im Verlauf der Zeit der Informationsgehalt eines Elementes ändern kann, und zu jedem Zeitpunkt einzelne Elemente hinzukommen oder wegfallen können. Diese Vorgänge sind in der Regel applikationsspezifisch, d.h. die Applikationen müssen Mechanismen zur Verfügung stellen, um die Spur verwalten, erweitern bzw. verkürzen zu können. Insbesondere bei kommunizierbaren Objekten ist es wichtig, dass der Informationsgehalt stark beschränkt bleibt, da die Kommunikationsressourcen knapp sind. Bei multihop Kommunikation kann nicht davon ausgegangen werden, dass in jedem Zwischengerät zum Ziel der applikationsspezifische Kode zur Spurverwaltung vorhanden ist. Daher müssen generische Mechanismen vorhanden sein, um eine mögliche Lücke zu überbrücken.

Die Einordnung in die drei zeitlichen Abschnitte Vergangenheit, Gegenwart und Zukunft erlaubt einige grundlegende Interpretationsmöglichkeiten. Elemente der Gegenwart beschreiben in der Regel Fakten. Das bedeutet, dass im Spurelement aktuell zu beobachtende Informationen enthalten sind. Vergangene Elemente sind der Kondens der Elemente der Gegenwartsspur bzw. der dort enthaltenen Informationen. Zukünftige Elemente können nur sehr eingeschränkt Fakten beinhalten, da in einem stark dynamischen Umfeld das Eintreffen der Elementinformationen nicht oder nur eingeschränkt garantiert werden kann, da Nachrichten unter Umständen ihr Ziel nicht erreichen, Ver-

**5.2 Warum bin ich hier?
Und nicht woanders...**

Ein Aspekt der Objektnachbarschaft ist der Grund für die Erzeugung eines Spurelements. Ein mobiler Zustand kann damit ausdrücken, dass sein zukünftiges Ziel der Aufenthalt innerhalb eines Ortskontexts ist, um dort Aktionen durchzuführen. Damit kann das Ziel eines kommunizierbaren Objektes modelliert werden. Insbesondere Unicastnachrichten besitzen einen eindeutig identifizierbaren Empfänger. Für Geräte kann die Modellierung bedeuten, dass es sich zu einem bestimmten Zeitpunkt in einer Veranstaltung, modelliert durch einen Ortskontext, befindet. Hierbei können Elemente der Zukunftsspur meist aber nur eine Absichtserklärung bzw. eine Aufenthaltswahrscheinlichkeit geben.

anstaltungen ausfallen oder Nutzer sich nicht immer an einen geplanten Veranstaltungsbesuch halten. Darüber hinaus kann eine zukünftige Spur auch Wahlmöglichkeiten beinhalten, um beispielsweise Alternativen aufzuzeigen, falls ein zukünftiges Element nicht erfüllt werden kann. Bei Nachrichten sind alternative Ziele denkbar, bei Geräten alternative Aufenthaltsorte. Zusammen mit Wahrscheinlichkeiten lassen sich die Zukunftsalternativen in der Gegenwart besser interpretieren. So kann eine Applikation aus den Erfahrungen der Vergangenheit abschätzen, wie wahrscheinlich die Teilnahme an einer Veranstaltung in der Zukunft ist.

Spurelemente besitzen somit auch eine zeitliche Information, die als Zeitpunkt oder Zeitintervall, oder implizit durch eine zeitliche Ordnung innerhalb einer Spur, enthalten sein kann. Im einfachsten Fall besitzt man nur die Information darüber, ob es ein Element der Gegenwarts-, Vergangenheits-, oder Zukunftsspur ist.

Ein weiterer Aspekt der zeitlichen Information sind Gültigkeit und Lebensdauerinformationen von Objekten. Die Lebensdauer eines Objektes kann durch seine Spur selbst modelliert werden, d.h. nach Ablauf des letzten zukünftigen Spurelements ist in der Regel seine Aufgabe erfüllt. Auch Lebensdauerinformationen über benachbarte Objekte sind wichtig, da zum einen damit das Löschen der Nachbarschaftsinformation aus der Spur erfolgen kann, zum anderen aber auch in gewissem Maße abgeschätzt werden kann, wie lange mit einem Objekt interagiert werden kann. Darunter fallen insbesondere auch reaktivierbare Gültigkeiten, die Nachbarschaftsassoziierungen verlängern können. Somit kann z.B. ein Dienst periodisch seine Verfügbarkeit zu assoziierten Objekten propagieren. Dies ist prinzipiell eine Verallgemeinerung des Gerätebeaconings, da auch hier nur ein periodisches Propagieren der Existenz eine Interaktion in einem dynamischen Umfeld möglich macht und nur so unbekannte Geräte erkannt werden können.

5.3 Stundenpläne

Stehen Stundenplaninformationen bzw. Terminkalenderinformationen zur Verfügung, können in gewissem Maße Vorhersagen über zukünftige Aufenthaltsorte getroffen werden. Die unter Umständen recht wage Annahme, dass sich ein Benutzer an diese Vorgaben hält, kann durch Erfahrungen der Vergangenheit angereichert werden. Die jüngere Vergangenheit kann Aufschluss darüber geben, ob diese Vorgabe eintrifft oder nicht, wenn der Nutzer sich auf das Ziel zubewegt bzw. sich schon in der Nähe befindet. Längerfristige Vorhersagen sind nur bei wiederkehrenden Terminen, z.B. Universitätsveranstaltungen möglich, wenn zum einen allgemein Aussagen zur Regelmäßigkeit von Veranstaltungsbesuchen und zum anderen spezielle Aussagen zu bestimmten Veranstaltungen und Veranstaltungstypen getroffen werden können. Die Untersuchung der Vorhersagbarkeit von Benutzerverhalten wurde in dieser Arbeit nicht weiter untersucht.

Elemente einer Spur sind im Prinzip Beschreibungen von Zeit und Raum. Beschreibung im Raum bedeutet bei einer Spur die Objektumgebung zu einem Zeitpunkt. Im einfachsten Fall kann über den Raum auch keine weitere Aussage getroffen werden. Lokalitätsbewusstsein und globale Positionierbarkeit erlauben eine konkretere Beschreibung im Raum, die für ein Objekte selbst, aber auf für andere Objekte mehr Bedeutung erhält. Ortsinformationen ermöglichen es, dass für kommunizierbare Objekte auch Orte als Ziel ihrer Bewegung adressiert werden können. Hier kann die zukünftige Spur die zu besuchenden Orte in der Zukunft darstellen und damit nicht nur Routinginformation angeben, sondern auch Routingalternativen aufzeigen. Durch Ortsinformationen können auch Spurinformationen über wichtige Objekte insbesondere stationäre Dienste und Geräte gesammelt werden. Sind die Aufenthaltsorte bekannt und können Aussagen über die Stationarität oder über Verfügbarkeit eines Objektes an diesem Ort getroffen werden, kann mit dem Objekt auch zu einem späteren Zeitpunkt interagiert werden bzw. muss das Objekt vor Ort nicht explizit gesucht werden. Darüber hinaus kann ein altruistischer Austausch der Spurinformation auch anderen Geräten helfen, ohne umfangreiche Kommunikation bestimmte Objekte aufzufinden. Ein mögliches Szenario kann auch der Vorschlag an den Benutzer sein, sich für einen bestimmten Dienst an einen bestimmten Ort zu begeben.

Umgebungserfahrungen der Vergangenheit lassen sich mittels Statistiken aufbereiten und zusammenfassen und in eine Objektspur übernehmen. Die daraus gewonnenen Wahrscheinlichkeitswerte können dann für Entscheidungen in der Gegenwart verwendet werden. Beispiele sind Erfahrungen über Aufenthaltsorte eines Gerätes, häufig getroffene Geräte und darüber, welche Objekte sich an bestimmten, häufig besuchten Orten aufhalten. Hierbei dient die Statistik zum einen dazu, die Informationsmenge auf einem Gerät zu reduzieren, indem wiederholt vorkommende Informationen zusammengefasst wer-

5.4 Geh auf den Marktplatz!

Bei UbiBay kann auch an Auktionen teilgenommen werden, wenn sich das Gerät nicht in der Region befindet, in der die Auktionen stattfinden. Ein vom Benutzer dorthin geschickter Bietagent repräsentiert ihn dort und führt seine Bietaktionen durch. Dennoch erhält man vor Ort schneller Informationen und kann als Nutzer schneller auf andere Bieter reagieren und eingreifen. Somit kann die Applikation dem Nutzer vorschlagen, näher an oder gar auf den Marktplatz zu gehen, falls dieser bestimmte Funktionen anwählt, deren Kommunikation sehr lange dauert oder innerhalb eines Zeitfensters nicht möglich ist.

den. Zum anderen kann ein gewisses Maß an Vorhersage ermöglicht werden. So kann mittels Nachrichtenduplikation die Wahrscheinlichkeit erhöht werden, auch in unzusammenhängenden Netzen geographische Orte zu adressieren, indem eine Nachricht mehreren Geräten mitgegeben wird, die mit hoher Wahrscheinlichkeit diesen Ort in naher Zukunft besuchen.

5.5 Mögliche Elemente einer Objektspur

- Objektnachbarschaften und Objektinformationen
- Objektinteraktionen
- Zeitinformationen
- Gültigkeitsinformationen
- Ortsinformationen
- Statistiken und Wahrscheinlichkeiten
- Applikationsspezifische Informationen
- Generische, für Anwendungsklassen verwendbare Informationen

Neben den genannten allgemeingültigen Spurinformationen existiert eine Vielzahl applikationsspezifischer Spurinformationen, die sich nicht durch Objektnachbarschaften darstellen lassen. In gewissem Maße lassen sich dort dennoch Informationen identifizieren, die über eine Anwendung hinaus auch für andere Objekte von Interesse sind. Insbesondere der Objekttyp Gerät lässt eine Vielzahl von allgemein verwendbaren Informationen erkennen. Dies ist um so mehr der Fall, wenn es sich um Informationen über direkt benachbarte Geräte handelt. Informationen über verfügbare bzw. altruistisch zur Verfügung gestellte Ressourcen erlauben es, diese auch als Kriterien für eine Objektmigration oder -duplikation zu verwenden. Stehen Informationen über Mobilität bzw. Stationarität zur Verfügung, hilft dies bei Entscheidungen

darüber, ob ein Gerät eine ausgezeichnetere Rolle zuteil wird. Die Rolle kann das Zuweisen von Diensten bedeuten, oder auch eine ausgezeichnetere Rolle in einem selbstorganisierenden Protokoll. Auch Aussagen über Persistenz oder Qualität eines Kommunikationslinks bzw. über die mögliche Distanz eines Gerätes können Kommunikationsentscheidungen unterstützen. Sind beispielsweise mehrere äquivalente Kommunikationspartner vorhanden, kann aufgrund dieser Informationen der Partner gewählt werden, der für das lokale Gerät sowie für das umgebende Netz am besten geeignet ist. Letzteres ist dann der Fall, wenn zur Kommunikation das Gerät mit der vermutlich geringsten Distanz ausgewählt wird, so dass der Einfluss der Kommunikation auf das Restnetzwerk reduziert werden kann. Dadurch kann ein Gerät auch Energie sparen. Der energieeffizienteste Abstand ist aber aufgrund von Mindestenergieeinsatz und Störeinflüssen der Geräte untereinander nicht der geringste, sondern es existiert, je nach Technologie, ein energieoptimaler Abstand.

5.6 Mögliche Informationen über ein Nachbargerät

- Ressourceninformationen
- Stationarität/Mobilität
- Geschätzte Linkpersistenz
- Geschätzte Distanz
- Reputationswerte
- Positions- bzw. Ortsinformation, Bewegungsrichtung oder -ziele
- Wichtige Objekte (z.B. Dienste)
- Dessen Nachbargeräte (2-hop Information)
- Applikationsspezifische Informationen (z.B. Datenabgleichsprofile bei En-Passant)

5.2 Propagierung

Informationen über Objekte müssen zu anderen Objekten gelangen, damit diese dort gesammelt werden können. Hierbei können unterschiedliche Strategien zum Einsatz kommen, die in diesem Abschnitt vorgestellt werden. Unterschieden werden kann zwischen Push-, Pull- und Push/Pullansätzen. Pushansätze sind notwendig, wenn Informationen über ein Objekt gesammelt werden soll, das nicht notwendigerweise bekannt ist, bzw. das spontan erscheinen und wieder verschwinden kann. Geräte propagieren hierfür periodisch Beaconnachrichten, um von Nachbargeräten erkannt werden zu können. Die proaktive Verbreitung von Information dient auch als Auslöser für Interaktionen zwischen Objekten, wenn die Informationen bestimmten Kriterien entsprechen.

Der Pullansatz entspricht der expliziten Anforderung der Informationen. Er ist nur dann möglich, wenn das Objekt bekannt ist. Somit muss unter Umständen zuerst nach dem Objekt gesucht werden. Der Push/Pullansatz kombiniert beide Methoden, indem geringe Informationsmengen proaktiv verbreitet werden, die einen, unter Umständen mehrstufigen, Informationsabfrageprozess auslösen. Ein einfaches Beispiel hierfür ist, dass aufgrund eines neu empfangenen Gerätebeacons alle gerätespezifischen Informationen angefordert werden. Im Folgenden werden unterschiedliche Strategien vorgestellt.

- *Lokaler Zugriff und Propagierung*
  Befinden sich Objekte auf demselben Gerät, ist der Zugriff und das Sammeln von Spurinformationen einfach. Objekte können die Informationen sofort bei Erzeugung oder Änderung als Ereignis auf dem Gerät propagieren, ohne auf Kommunikationseffizienz Rücksicht nehmen zu müssen. Auch die Suche nach Objekten mit bestimmten Eigenschaften oder Spurinformationen ist lokal sehr einfach realisierbar und somit die Abfrage dieser Informationen einfach möglich. Um eine solche lokale Interaktion auch für entfernte Objekte zu ermöglichen, können Repräsentantenobjekte eingesetzt werden, die zumindest einen Teil der Information stellvertretend lokal zur Verfügung stellen.
- *Interaktionsinformationen*
  Wird mit einem anderen Objekt interagiert, können hierbei Informationen über das andere Objekt gesammelt werden, ohne zusätzlichen Kommunikationsaufwand zu betreiben. Aufgrund der Kommunikation und des Applikations- und Netzwerkkontext können neben dem eigentlichen Interaktionsinhalt weitere Informationen gesammelt werden, insbesondere wenn wiederholt mit demselben Objekt interagiert wird. Wird immer an demselben Ort interagiert, ist so zumindest ein wiederkehrender Aufenthaltsort des Kommunikationspartners bekannt. Die Häufigkeit der Interaktionsmöglichkeit können Annahmen über die Zuverlässigkeit treffen lassen. Aus Routinginformationen früherer Interaktionen können in Zukunft Kommunikationsparameter verbessert werden. Beispielsweise kann eine Anpassung des Routenfindungsprozesses erfolgen, indem die Hopdistanz oder die Region reduziert werden, wenn das Objekt das nächste Mal gesucht wird.
- *Interaktionserweiterung*
  An die Nachrichten, die während einer Objektinteraktion ausgetauscht werden, kann unabhängig vom Interaktionsinhalt Zusatzinformation angehängt werden. Dieses Piggybacking ermöglicht es, dass der Interaktionspartner weiteres Wissen über ein Objekt erhält. So kann ein Objekt zusätzlich mitteilen, dass sein Standort im Netz stabil ist oder dass es mit einem andern Objekt fest assoziiert ist. Die Assoziierung mit einem Ortskontext, der zu bekannten Zeiten aktiv ist, kann so eine zukünftige Interaktion ermöglichen. Wichtig ist hierbei, dass auf die Größe der

Nachrichten Rücksicht genommen wird, so dass bei einer möglichen Fragmentierung aufgrund der der Überschreitung der MTU kein zusätzliches Packet entsteht.

- *Objektbeaconing*
  Alle Objekte können, wie Geräte, periodisch ihre Existenz propagieren und dabei auch Zusatzinformationen verbreiten. Hierbei handelt es sich aber um Nachrichten mit geringem Informationsgehalt und damit einem hohem Kommunikationsoverhead. Daher ist es sinnvoll, das Versenden von Beaconnachrichten auf einem Gerät zentral durchzuführen. Somit können die Informationen zusammengefasst und im sowieso durchgeführten Gerätebeaconing propagiert werden. Dadurch kann an einer zentralen Stelle der notwendige Kommunikationsaufwand kontrolliert werden und die Häufigkeit der Propagierung an die momentane Netzsituation angepasst werden.

  Das Objektbeaconing wird nur von einem Teil der Objekte eingesetzt und ist nicht für alle permanent notwendig. Es wird auch selten dieselbe hohe Beaconingfrequenz benötigt, mit der Beispielsweise ein Gerätebeaconing erfolgt. Somit muss die Beaconnachricht nur sporadisch mit der Information eines Objektes erweitert werden. Je häufiger das Objektbeaconing erfolgen muss, desto weniger Informationen sollte das Objekt verbreiten. Seltenere Propagierung kann also einen höheren Informationsumfang, aber auch eine Erhöhung der Verbreitungsreichweite ermöglichen. Für eine höhere Verbreitungsreichweite kann multihop Beaconing eingesetzt werden oder eine Propagierung im Sinne des Informationsradios erfolgen. Dadurch kann in einer bestimmten Region Objektinformation (z.B. Veranstaltungsinformationen eines Ortskontextes) im Hintergrund verbreitet werden. Die Änderungsperiode und der Umfang der Information sind dabei sehr gering, so dass eine hohe Reichweite der Propagierung erreicht werden kann, ohne das Netz zu stark zu belasten. Die Information kann auch Objekte erreichen, die zwar in der Zielregion, aber in unterschiedlichen Netzpartitionen liegen.

  Das Objektbeaconing kann auch über eine direkte multihop Propagierung mit Multicast oder Publisher/Subscriber [16] Verfahren erfolgen. Diese Verfahren sind in der Lage, die Propagierung auf Geräte einzuschränken, die tatsächlich an den Informationen Interesse haben. Somit ist es möglich, nur einen Teil des Netzes mit der Propagierung zu belasten. Allerdings müssen hierfür Strukturen aufgebaut und aufrecht erhalten werden, die eine permanente Kommunikationslast verursachen. Wird aber die Multicaststruktur auch für weitere Kommunikation verwendet, ist dies zu vernachlässigen. Bei diesen Verfahren werden aber nur Geräte erreicht, die sich in derselben Netzpartition befinden.

- *Direkte Abfrage*
  Ist ein Objekt bekannt und besteht Interesse an zusätzlichen Spurinformationen, können diese singlehop wie multihop angefordert werden. Hier können erheblich mehr Informationen ausgetauscht werden, als es beim

proaktiven Beaconing der Fall ist. Es müssen aber die Existenz des anderen Objektes bekannt sein und häufig schon Informationen lokal vorhanden sein, die indizieren, dass eine Anfrage sinnvoll ist. Somit ist aufgrund der spontanen Objektnachbarschaft dennoch ein Objektbeaconing sinnvoll um eine Anfrage auszulösen. Es ist nicht sinnvoll, bei jedem Objekt eines bestimmten Typs diese Abfrage durchzuführen. So kann die Abfrage von Zusatzinformationen bei jedem Gerät, das in der Nachbarschaft erscheint, in sehr dynamischen Netzen zu einer sehr hohen Kommunikationslast führen, obwohl meist nur ein Bruchteil der Geräte von Interesse sind. Statt dessen können gewisse Geräteeigenschaften (z.B. Ressourceninformationen) indizieren, ob eine Anforderung notwendig ist. Diese Zusatzinformationen können zusätzlich auch eine Einschränkung der Abfrage auf bestimmte Informationen ermöglichen. Daher ist eine Kombination aus erweitertem Objektbeaconing und direkter Abfrage ideal. Im singlehop Fall entspricht dieses Vorgehen der En-Passant Kommunikation, wobei hier die propagierten Nachrichtenobjekte Spurinformationen enthalten. Prinzipiell ist dies auch im multihop Fall, allerdings mit geringerem Datenumfang möglich.

- *Mithören der Kommunikation*
  Linkbasierte Routingverfahren nutzen die Broadcasteigenschaft des Mediums, um auch Routen im Netz zu erhalten, die in der Nachbarschaft bekannt sind. Hierfür werden die Nachrichten der Nachbarn mitgehört oder auch die Routen von passierenden Nachrichten übernommen. Dieses Vorgehen kann in ähnlicher Weise für das Sammeln von Spurinformationen verwendet werden. Werden Interaktionsnachrichten erweitert oder Spurinformationen angefordert, sind von der Kommunikation betroffene Nachbargeräte in der Lage, Nutzen daraus zu ziehen und diese Informationen zu sammeln. Somit ist eine Weiterverarbeitung von Teilen der Nachrichten auch für das Auslesen von Spurinformationen sinnvoll.

Es stehen also unterschiedlichste Ansätze zur Propagierung von Spurinformationen zur Verfügung. Welches Verfahren im Einzelnen eingesetzt werden kann, hängt von den Bedürfnissen der Applikation aber auch von den Gegebenheiten des Netzes ab. Da ein Gerätebeaconig in einer dynamischen Umgebung notwendig ist, um auch neue Interaktionspartner frühzeitig erkennen zu können, ist das Objektbeaconing durch Erweiterung der Beaconnachricht mit geringem Aufwand realisierbar. Die Informationsmenge, die dabei verbreitet wird, muss aber beschränkt bleiben. Informationen, die lokal vorhanden sind, aufgrund der Broadcasteigenschaft des Mediums empfangen werden oder aufgrund von Interaktionserfahrungen gewonnen werden können, verursachen keinerlei zusätzlichen Kommunikationsaufwand, sind aber nicht immer einsetzbar. Es ist zwar beispielsweise möglich, aufgrund der von anderen Objekten durchgeführten Kommunikation von ihrer Existenz zu erfahren. Erfolgt aber keine regelmäßige Kommunikation von diesem Objekt, bleibt es verborgen. So bleiben auch passive Geräte ohne Beaconing unsichtbar.

5.3 Darstellung und Verwaltung der Objektspur

Die Gegenwartsspur besitzt für die meisten Objekttypen den größten Stellenwert. Dies liegt darin begründet, dass Spurinformationen der Vergangenheit hieraus erzeugt werden und daraus wiederum in manchen Fällen Informationen über die Zukunft gewonnen werden. Einige Objekt besitzen nur Gegenwartsinformationen und treffen ihre Einscheidungen nur auf Basis der aktuellen Umgebung. Dies ist insbesondere in der Nachrichtenkommunikation eine wünschenswerte Eigenschaft, da dies eine zustandsfreie Kommunikation bedeutet. Routingentscheidungen werden dabei nur auf Basis der aktuellen Umgebung getroffen. In der Nachricht muss außer dem Ziel der Nachricht keinerlei weitere Informationen gespeichert werden. Darüber hinaus wird auf jedem Gerät nur Information über die aktuelle nähere Geräteumgebung benötigt. Dieses Idealbild wird in der Theorie nur von positionsbasierten Routingverfahren erreicht, die in der Praxis aber dennoch Zustandsinformationen benötigen z.B. um die maximale Zahl der Nachrichtenübertragungen zu begrenzen.

Die Gegenwartsspur kann vom Prinzip her bei allen Objekttypen, auch bei kommunizierbaren, sehr umfangreich ausfallen. Informationen über die Gegenwart können nur auf einem Gerät erzeugt und von Objekten gesammelt werden. Sie sind in der Regel Informationen der Vergangenheit, wenn das sammelnde Objekt auf ein anderes Gerät übertragen wird und fallen somit weg oder werden zusammengefasst. Ausnahme bilden hier MobileStates, wenn deren Übertragung auf ein benachbartes Gerät keine Umgebungsänderung bedeutet. Dies gilt für MobileStates, die an einen Ort gebunden sind und das Wirtsgerät wechseln, um an diesem Ort zu verbleiben. Dabei ändert sich zwar ein Teil der gegenwärtigen Umgebung, nämlich der zugeordnete Wirt und die wirstslokale Umgebung. Der Ort selbst und bekannte, an diesen Ort gebundene Objekte ändern sich aber nicht. Somit bleibt hier ein Teil der Gegenwartsspur auch nach der Übertragung erhalten.

Die Sammlung von Informationen über andere Objekte kann die Kommunikationslast im Netz reduzieren, da wiederholtes Abfragen derselben Information von benachbarten Objekten vermieden werden kann. Ist das Wissen über die Existenz eines Objektes auf einem Gerät bekannt, muss es nicht im Netzwerk gesucht werden. Eine Dienstsuche und die Abfrage von Dienstinformationen kann so lokal ablaufen, falls dieser schon einmal lokal (von einem anderen Objekt) angefragt wurde. Informationen über direkt benachbarte Geräte und die sich darauf befindenden Objekte von allgemeinem Interesse gehören zu den wichtigsten Informationen der Gegenwartsspur. Daher ist es sinnvoll, die Verwaltung zentral durchzuführen und Informationen über Nachbargeräte applikationsübergreifend zu verwalten. Die Informationen können als Teil der Gegenwartsspur des lokalen Gerätes verwaltet werden, wie dies in Abbildung 5.2 a) dargestellt ist. Neben Informationen über Objekte, die sich lokal auf dem Gerät befinden und somit Nachbarn des Geräteobjektes sind, befinden sich auch Informationen über die benachbarten Geräte in der Gerätespur.

5.7 Die Spur mobiler Zustände auf Marktplätzen

Mobile Zustände werden bei der marktplatzbasierten Kommunikation dazu verwendet, (teil-) autonom mit anderen mobilen Zuständen innerhalb einer vorgegebenen geographischen Region zu interagieren. Dazu muss das momentane Wirtsgerät gewechselt werden, um in dieser Region zu verbleiben wenn sich das momentane Wirtsgerät wegbewegt. Zwangsläufig müssen dabei Assoziationen mit anderen Objekten des Marktplatzes und Informationen über den Marktplatz auf den neuen Wirt mitübertragen werden. Da ein Teil der Informationen wiederhergestellt werden kann, indem mit der Sammlung erneut begonnen wird bzw. der Wegfall bestimmter Informationen unkritisch ist, kann hier eine zweiphasige Übertragung realisiert werden. In der ersten Phase werden Objektdaten und notwendige Zustandsinformationen übertragen und in der zweiten Phase der unkritische Teil der Gegenwartsspur. Da ein Teil der Informationen schon auf dem Zielgerät vorhanden sein kann, kann in der zweiten Phase zusätzlich ein Informationsabgleich ähnlich der En-Passant Kommunikation durchgeführt werden.

Hierbei werden zu einer eindeutigen Objektidentifizierung zusätzliche Informationen abgelegt, die zu einem Objekt gesammelt wurden. Im Falle benachbarter Geräte beinhaltet diese Information auch Objektidentifizierungen der gehosteten Objekte und unter Umständen weitere Informationen über diese Objekte. Damit entsteht eine einfache Baumstruktur, die wie ein Verzeichnisbaum abgefragt werden kann. Die Struktur dient auch als Ausgangspunkt für die Verbreitung von Informationen an Nachbarn. Es existieren Bereiche, deren Informationen bis in eine bestimmte Tiefe den Nachbarn proaktiv oder reaktiv mitgeteilt werden, damit diese die Informationen zum Aufbau ihre Gegenwartsspur verwenden können. Hierzu zählen in der Regel aber nicht die Informationen über benachbarte Geräte, da dann in dichten Netzen die Beaconnachrichten zu groß werden könnten. Informationen aus diesem Baum werden entweder von anderen Objekten direkt verwendet, in ihre Spur "eingeblendet" oder kopiert. Werden Nachbargeräte durch Repräsentantenobjekte dargestellt, entspricht der dazugehörige Teilbaum der im Repräsentanten dargestellten Gegenwartsspur des Nachbargerätes. Änderungen und Abfragen sind also identisch, unabhängig davon, ob auf das Geräteobjekt des lokalen Gerätes oder den Repräsentanten zugegriffen wird. Weiter verallgemeinert kann die lokale Spur eines Gerätes in einem zentralen Verzeichnisbaum abgelegt sein, der Informationen über alle lokalen Objekte beinhaltet und darüber hinaus Teilbäume für benachbarte Geräte besitzt (Abbildung 5.2 b)). Werden Informationen über andere Geräte im Netz auch über die direkte Nachbarschaft hinaus gespeichert, können auch Teilbäume über nicht benachbarte Geräte enthalten sein. Diese Informationen sind aber nicht mehr Teil der Gegenwartsspur des lokalen Gerätes. Nachbargeräte werden in den Teilbaum des lokalen Gerätes eingeblendet, damit sie auch bei Abfragen über das lokale

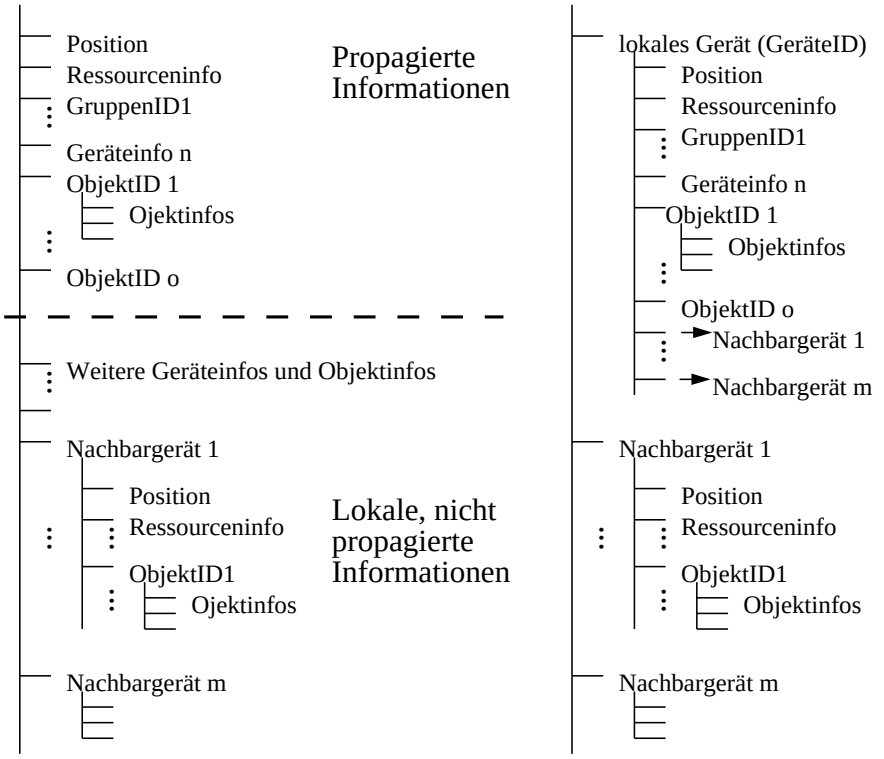


Abb. 5.2. (a) Die Gegenwartspur des lokalen Geräteobjekts. Sie beinhaltet gerätespezifische Informationen und Informationen über benachbarte Objekte, zu denen auch die momentan benachbarten Geräte zählen. Nur der obere Teil der Spur wird an Nachbargeräte propagiert, die diese Informationen in ihre Spur aufnehmen. (b) Darstellung der Gegenwartspur in einem zentralen Verzeichnis. Die Nachbargeräte erhalten eigene Teilbäume, das lokale Gerät besitzt aber Verweise, um die Darstellung (a) zu erhalten.

Geräte als benachbarte Objekte zugreifbar sind. Eine Möglichkeit, eine solche Darstellung zu realisieren ist ein Verzeichnisdienst. Mit ADS [54] entsteht zur Zeit ein solcher Dienst, der auch für die Propagierung der Informationen zuständig ist. Dieser soll auch einen automatischen Abgleich von Teilbäumen auf benachbarten Geräten ermöglichen.

Dem Vorteil der Wiederverwendbarkeit und Redundanzvermeidung bei zentraler Verwaltung steht ein hoher Verwaltungsaufwand gegenüber, da die Speicherung von nicht mehr benötigten Informationen vermieden werden muss. Problematisch sind insbesondere Informationen, die von mehreren Objekten eingetragen wurden und zwar eine beschränkte, aber nicht vorhersehba-

re Gültigkeit besitzen. Die Nutzung von Informationen durch mehrere Objekte ist aber gerade das Ziel der zentralen Speicherung. Referencecounting, also das Zählen der Informationsnutzer, löst das Problem zwar, erhöht aber auch den Aufwand des Zugriffs, da Objekte sich explizit registrieren und deregistrieren müssen. Darüber hinaus ist unklar, ob eine Information tatsächlich gelöscht werden soll, wenn ein Objekt einen Löschauftrag absetzt, die Information aber von mehreren Objekten erstellt wurde. Um insbesondere das Problem der unvorhersehbaren Informationsgültigkeit zu lösen, können Eintragungen immer mit einer fixen Gültigkeitsdauer erfolgen. Damit die Information nach Ablauf erhalten bleibt, muss die Gültigkeit vorher verlängert werden. Somit bleibt die Information so lange erhalten, wie diese noch von mindestens einem Objekt als gültig erachtet wird. Nachteilig bei diesem Verfahren ist der Aufwand der Gültigkeitsverlängerung, da hier Daten periodisch aktiviert werden müssen. Anwendung findet das Verfahren in ähnlicher Weise bei der Verwaltung der Nachbargeräte. Hier wird die Information darüber, ob ein Gerät vorhanden ist, durch die empfangenen Beaconnachrichten verlängert. Bleibt eine gewisse Zahl an Beaconnachrichten aus, kann sämtliche Information bzgl. dieses Gerätes gelöscht werden. Teilinformationen des Gerätes können aber eine andere Lebensdauer als das Gerät selbst besitzen. Daher müssen bei Bedarf für die Teilbäume eigene Gültigkeitsverlängerungen durchgeführt werden. Ein weiterer Nachteil der zentralen Speicherung im Vergleich zur dezentralen Speicherung in Objektrepräsentanten ist ein häufig höherer Speicheraufwand. Dieser liegt darin begründet, dass unter Umständen zusätzliche Informationen über den Spurdatenkontext abgelegt werden müssen, damit die Spurdaten nutzbar bzw. zuordenbar sind. So kann bei lokaler Speicherung implizit klar sein, von welchem anderen Objekt diese Informationen stammen oder was sie bedeuten.

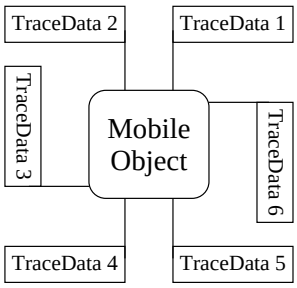


Abb. 5.3. MobileObjects können von anderen Objekten mit Spurinformationen "beklebt" werden. Diese Informationen sind nicht Teil des Objektes selbst, sondern werden in einem Tracepool verwaltet. Sie können beim Versenden des Objektes mitverpackt werden.

Eine einfache Variante der hierarchischen Speicherung ist die Spurmenge. Hierbei werden die Spurinformationen zu einem Objekt unstrukturiert abgelegt. Der Vorteil dieses Ansatzes ist seine Einfachheit. Allerdings ist die Suche nach Information unter Umständen aufwendiger. Eine mögliche Darstellung dieser Variante ist das "Bekleben" eines Objektes mit Zusatzinformationen, wie in Abbildung 5.3 gezeigt, da die Informationen auch von anderen Objekten hinzugefügt und von kommunizierbaren Objekten unter Umständen auch mitgenommen werden können. Die Darstellung motiviert sich aus der Tatsache, dass gewisse Informationen nicht als Teil des Objektes angesehen werden sollten, sondern nur als mögliche und nicht zwingend vorhandene Zusatzinformation. So ist die Zugehörigkeit zu einer oder mehreren Gruppen eine Zusatzinformation und nicht Teil des Objektes. Darstellbar ist dies auch mit Objektwrapping, wobei in einer Hülle das Objekt mit den Zusatzinformationen verbunden wird.

Werden Daten über ein entferntes Objekt geschlossen in einem Datenpool abgelegt, ist in gewissem Maße schon eine Objektrepräsentierung vorhanden. Die Repräsentierung ist allerdings passiv und muss von anderen Objekten verwaltet werden. Ein Beispiel sind Orts- und Zeitinformationen eines Ortskontextes, der nicht aktiv durch einen Repräsentanten dargestellt werden muss, da keine besonderen Kommunikationsmechanismen verwendet, keine lokalen oder ortsglobalen Ereignisse erzeugt und keine weiteren Spurinformationen verwaltet werden müssen.

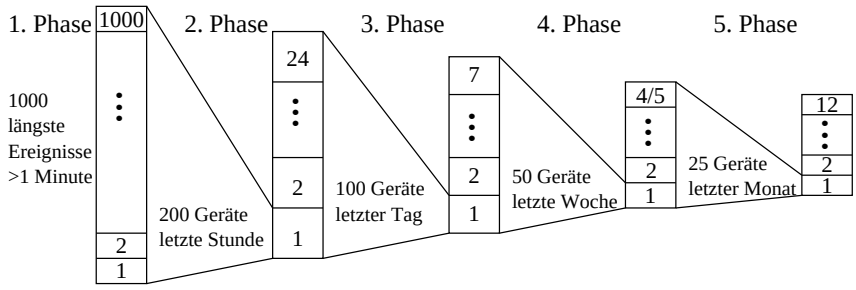
5.8 En-Passant Kommunikation und Profilcaching

Der effiziente Abgleich bei der En-Passant Kommunikation basiert auf so genannten Synchronisationsprofilen, die dem Nachbargerät Aufschluss darüber geben, welche Informationen fehlen und wie Informationen gefiltert werden sollen, um die Informationsmenge zu reduzieren oder den Interessenraum des anderen Gerätes zu beschreiben. Die Profile können auch die auf dem Nachbargerät vorhandenen passenden Objektkennungen speichern, so dass neu hinzugekommene passende Objekte an den Nachbarn versendet werden können. Die Profile werden für alle Nachbargeräte gespeichert, Änderungen an den Profileigenschaften eines Gerätes müssen somit propagiert werden.
Sobald ein Gerät die Nachbarschaft verlässt, können die Profile gecacht werden, um somit, wenn möglich, eine wiederholte Übertragung beim nächsten Treffen zu vermeiden. In diesem Fall ist ein vorgelagerter Schritt notwendig, der die Aktualität der Profile auf beiden Geräten überprüft und nur neuere überträgt. Ein Abgleich der Objekte findet aber dennoch statt, nur die Übertragung der Profile wird eingespart.

Informationen über die Vergangenheit entstehen schon allein durch die Tatsache, dass Informationen der Gegenwart über längere Zeit gespeichert

5.9 Phasenbasierte Nachbarstatistik für MobileObjects

Existieren MobileObject-Typen, für die bestimmte Instanzen häufiger getroffen werden als andere, können mittels einer Statistik der Begegnungen der Vergangenheit, Aussagen über künftige Begegnungen getroffen werden. Dies gilt beispielsweise für mobile Endbenutzergeräte. Dort existiert in der Regel eine Menge von Personen, die häufiger getroffen werden als andere. Ein weiteres Beispiel sind Geräte-(Personen-)gruppen – insbesondere wenn die Geräte der Gruppe selbst häufig eine geographische Nähe besitzen. Um über die Zeit diejenigen Objekte eines Typs zu bestimmen, die aktuell am häufigsten getroffen werden, wurde ein phasenbasiertes Verfahren entworfen. In der ersten Phase werden maximal n der längsten Begegnungsereignisse in einem festen Zeitfenster aufgezeichnet. Hierbei werden nur Ereignisse berücksichtigt, die eine definierte Mindestdauer besitzen. Die Mindestdauer kann so gewählt werden, dass zumindest theoretisch eine Interaktion mit dem Objekt möglich ist. Für Geräte kann das bedeuten, dass mindestens zwei Beaconnachrichten empfangen werden können. Ist das Zeitfenster der ersten Phase abgelaufen, werden die Ereignisse eines Objektes zusammengefasst und die m besten Objekte in die nächste Phase überführt. Die Phase $n+1$ besteht aus einer festen Menge der zusammengefassten Ereignisse aus Phase n und besitzt somit ein Zeitfenster, das einem Vielfachen der Phase n entspricht. Die Zahl der beobachteten Objekte sinkt, je länger die Phase dauert. Ein Beispiel sieht wie folgt aus:



Das Zeitgranulat und zeitlicher Umfang der Phasen ist für die erste Phase die letzten 1000 Ereignisse, für die 2. 24 der letzten vollen Stunden, die 3. 7 der letzten vollen Tage, die 4. 4-5 Wochen und die 5. 12 Monate. Somit lassen sich für einige wenige Objekte die in allen Phasen enthalten sind, sehr detaillierte Aussagen über ihre Vergangenheit treffen. Lässt das vergangene Verhalten Rückschlüsse auf die Zukunft zu, kann abgeschätzt werden, wie wahrscheinlich ein Treffen in einem gegebenen Zeitraum ist. Aber auch einfachere Aussagen wie "Diese k Objekte habe ich in einem bestimmten Zeitfenster am häufigsten getroffen" lassen sich hiermit herleiten. Diese Aussagen können sich von Applikationen verwendet werden, um häufig getroffene Objekte besonders zu behandeln. Auch eine Middleware selbst kann somit Spurinformationen von Objekten besser cachen.

– Fortsetzung Box 5.9 –
Aufgrund des abnehmenden Detailgrades der Informationen ist der Speicheraufwand gering. Im obigen Beispiel werden rund 7000 Einträge gespeichert, wobei nur im schlechtesten Fall auch 7000 verschiedene Objekte betrachtet werden.

werden und meist auch über einen längeren Zeitraum gültig sind. Sind diese mit Entstehungszeitpunkten versehen, besitzt man schon in der Gegenwartsspur detaillierte Informationen über die Vergangenheit. Wenn die Spurinformationen mit Gültigkeitsaktualisierung verwaltet werden, ist das Zeitfenster in der Regel größer, da dabei auch Informationen vorbehalten werden, die nicht mehr zur der aktuellen Umgebung gehören. Die einfachste Form der Vergangenheitsspur ist durch Caching zu erreichen. Caching wird meist nur zur schnelleren Wiederherstellung von Teilen der Gegenwartsspur verwendet. Beispielsweise ist es sinnvoll, Informationen über Nachbargeräte nicht sofort zu löschen, sondern zwischenzuspeichern und wiederherzustellen, falls nur ein kurzzeitiger Verbindungsabbruch vorlag. Caching wird auch zur Reduzierung des Kommunikationsaufwandes bei der En-Passant Kommunikation verwendet. Da nicht beliebig viele Elemente zwischengespeichert werden sollen, muss eine Verdrängungsstrategie verwendet werden. Wird der Zwischenspeicher nur zur Überbrückung von Kommunikationabbrüchen verwendet, genügt das einfache FIFO[1]-Verfahren. Hierbei wird ein Zeitfenster von nur einigen Minuten überbrückt, so dass die Informationen in der Regel noch aktuell sind. Werden die zwischengespeicherten Informationen beim nächsten Zusammentreffen abgeglichen (wie beim Profilcaching der En-Passant Kommunikation), sind beliebige Zeiträume zur Zwischenspeicherung denkbar. Hier wird die Verdrängungsstrategie wichtig, da bevorzugt Geräteinformationen vorgehalten werden sollen, die mit hoher Wahrscheinlichkeit auch wiederverwendet werden. Hierzu ist eine Statistik notwendig wie sie z.B. das Verfahren der *phasenbasierten Nachbarstatistik* liefert. Es handelt sich dabei um ein Verfahren, das die Vergangenheitsspur der benachbarten Geräte liefert und diese im Laufe der Zeit verblassen lässt, da der Detailgrad und die Zahl der betrachteten Geräte mit der Zeit abnehmen.

Ein ähnliches Vorgehen lässt sich auch für die Aufenthaltsorte eines Gerätes anwenden. Hierbei werden Informationen über von einem Gerät besuchte und identifizierbare Ort zwischengespeichert und auch in Phasen zusammengefasst. Darüber hinaus lassen sich Orte zusätzlich geographisch zusammenfassen, so dass zwar einzelne Orte aus der Statistik herausfallen können, diese aber dennoch in einer übergeordneten Region statistisch berücksichtigt werden (siehe Abbildung 5.4).

[1] First-In-First-Out – Der Eintrag, der sich am längsten im Cache befindet wird von einem neuen Eintrag verdrängt.

5.10 Smallworld Routing

Eine Anwendungsmöglichkeit für eine Nachbarschaftsstatistik ist das so genannte Smallworld Routing. Neben dem schon beschriebenen Multicast sind auch Unicastvarianten möglich. Im Falle eines Unicasts wird versucht, ein bekanntes Gerät zu erreichen, indem die Nachricht für dieses Gerät an Geräte weitergegeben wird, die dieses Zielgerät in der Vergangenheit häufig getroffen haben. Dabei wird angenommen, dass dies auch in der Zukunft zutreffen wird. Die Nachricht darf beschränkt dupliziert werden, wenn sie weitergegeben wird, bzw. darf sie wieder gelöscht werden, wenn sie auf eine bestimmte Zahl Geräte dupliziert worden ist, die bessere Statistikwerte für das Ziel besitzen. Es können aber keinerlei Garantien über Erfolg und Dauer einer Auslieferung getroffen werden.

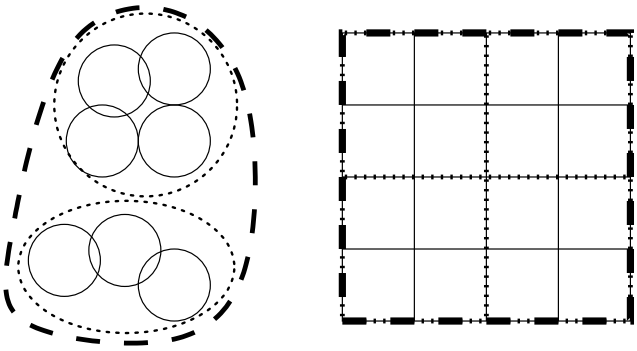


Abb. 5.4. Links können die Orte in den kleinen Kreisen durch fest installierte Geräte erkannt werden. Das Zusammenfassen dieser Orte benötigt zusätzliches Wissen (Räume gehören zu Gebäuden). Stehen nur Positionsinformationen zur Verfügung, kann die Ebene wie rechts dargestellt in eine Gitterstruktur unterteilt werden. Durch Zusammenfassen von 4 benachbarten Quadraten entsteht eine hierarchische Struktur (Quadtree).

Während die Nachbarschafts- und Ortsstatistiken immer auch Zeitinformationen verwenden müssen, um Informationen zusammenzufassen, lässt sich eine Zusammenfassung für bestimmte Informationen auch ohne Zeitinformationen bewerkstelligen. Werden in gleichmäßigen Abständen Informationen gesammelt, können diese auch ohne Zeitstempel zusammengefasst und ausgewertet werden. Die zeitliche Reihenfolge ergibt sich dann implizit aus der Reihenfolge der linearen Speicherung. Auch ohne Zusammenfassung finden häufig Listenstrukturen Anwendung. Kommunizierbare Objekte können so z.B. den zurückgelegten Pfad protokollieren, der alle oder einen Teil der besuchten Geräte oder Orte beinhalten kann. Das Kommunikationsverfahren DSR nutzt dies, um aus Routediscoverynachrichten den Rückweg zum Sender abzulei-

ten. DSR speichert prinzipbedingt auch in den Anwendungsnachrichten die zurückgelegte Route, da die Route zum Ziel immer in der Nachricht enthalten ist und besuchte Knoten nicht entfernt werden. In diesem Fall wird also auch die zukünftige Spur als lineare Liste dargestellt. Allgemein kann die Liste bei kommunizierbaren Objekten zu besuchende Orte oder Geräte beinhalten, die zur Unterstützung oder als Vorgaben von Routingentscheidungen verwendet werden. Für das positionsbasierte GSR werden geographische Zwischenstationen als Liste vorgegeben, die nacheinander abgearbeitet werden müssen, um das eigentliche Ziel zu erreichen.

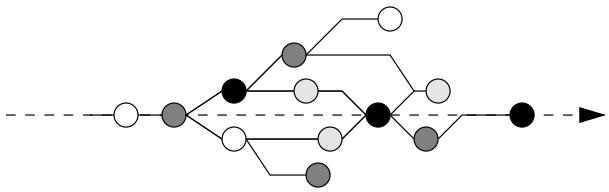


Abb. 5.5. Die Zukunftsspur, dargestellt als kreisfreier Graph über eine Zeitachse. Die Knoten können Ziele eines kommunizierbaren Objektes sein oder auch Objektzustände definieren.

Werden die zukünftigen Ziele mit Zeitinformationen versehen, kann dies ebenfalls für Routingentscheidungen genutzt werden. Wenn beispielsweise Fristen gesetzt werden, wann ein Ort spätestens erreicht sein soll, kann beim Überschreiten der Frist dieses Ziel übersprungen werden oder die Nachricht ganz verworfen werden. In der Listendarstellung können auch einfache Stundenpläne von Ortskontexten, z.B. Veranstaltungen, realisiert werden. Da Stundenpläne häufig verwendet werden, ist die Kalenderdarstellung erwähnenswert. Wenn es sich um sichere Termine eines Objekts handelt, können die Informationen direkt verwendet werden. Werden dagegen Wahrscheinlichkeiten verwendet, muss der Spurnutzer diese Information entsprechend relativieren. In der Regel werden im Zusammenhang mit Wahrscheinlichkeiten auch Wahlalternativen angeboten, so dass die Spur als kreisfreie Graphstruktur wie in Abbildung 5.5 dargestellt werden kann. Wahlalternativen mit Wahrscheinlichkeiten sind dann vorhanden, wenn z.B. zukünftige Aufenthaltsorte von Geräten abgeschätzt werden. Hierbei können Alternativpfade entstehen, wenn nicht nur Ziele sondern auch Zwischenstationen berücksichtigt werden. Eine solche Graphstruktur kann zu Routingzwecken eingesetzt werden. Hier können dem Routingverfahren Zielalternativen vorgegeben werden, so dass beispielsweise aufgrund von geographischer Distanz, Hopdistanz oder Uhrzeiten entschieden werden kann, welches der Ziele günstiger zu erreichen ist. Es ist aber auch möglich, Pfadalternativen zur Duplikation von kommunizierbaren Objekten einzusetzen. Hier steuert jedes Duplikat sein eigenes Ziel an, und unter Umständen können Duplikate später wieder zusammenge-

fasst werden. Zukunftsalternativen können auch bei Ortskontexten entstehen. Hier können aus Applikationssicht zwei Orte denselben Kontext besitzen und somit Kommunikationsalternativen bieten.

5.11 Marktplätze und Lastverteilung

Im Falle des marktplatzbasierten Kommunikationsmusters kann das Problem auftreten, dass zu viele MobileStates sich am selben Ort aufhalten wollen, es aber aufgrund von Ressourcenbeschränkungen der Geräte nicht können. Eine Lösungsmöglichkeit dafür ist, die Lastverteilung über verschiedene geographische Orte, so dass MobileStates, falls sie alle dem Markplatz zugeordneten Objekte erfassen wollen, verschiedene Orte besuchen müssen. Zu diesem Zweck muss an jedem Marktort die Information über Alternativorte vorhanden sein. MobileStates können in diesem Fall auch dupliziert werden, um parallel verschiedene Orte aufzusuchen. Somit können an verschiedenen Orten gleichzeitig nach Interaktionspartnern gesucht werden.

Insbesondere wenn aus der Vergangenheitsspur Informationen für zukünftige Spurelemente gewonnen werden, müssen die Elemente der Zukunftsspur nicht tatsächlich in einer Datenstruktur abgelegt werden. Es ist auch legitim, dass dieser Teil der Zukunftsspur rein virtuell ist. Das bedeutet, dass die Informationen dynamisch bei der Abfrage aus der Vergangenheitsspur gewonnen werden. Beispielsweise kann bei der Bestimmung des nächsten Aufenthaltsortes aus früheren Erfahrungen, zusammen mit Terminkalenderinformationen und den aktuellsten Gerätebewegungen, eine gute Vorhersage getroffen werden, die bei jedem Zugriff aktuell erstellt wird.

6

Objektrepräsentanten

Objektrepräsentierung von MobileObjects, die sich nicht lokal auf einem Gerät befinden, motiviert sich aus mehreren Gründen. Werden Informationen über ein entferntes Objekt von verschiedenen Objekten auf demselben Gerät gesammelt, können Redundanzen entstehen, die durch Speicherung in einem zentralen Objekt vermieden werden können. Existiert ein Objekt nur als Konzept, wie Orts- oder Gruppenkontexte, gibt es unter Umständen keine zentrale Objektinstanz, um die Funktionalität des Objektes zu realisieren und Informationen zu speichern und zu verwalten. Diese Aufgabe wird dann von einem Teil oder von allen Geräten übernommen, die Teil des Konzeptes sind. Um eine Trennung von anderen Objekten und von Funktionalitäten zu erhalten, ist auch hier die Kapselung in einem eigenständigen Repräsentantenobjekt sinnvoll.

Kommunikation mit einem entfernten Objekt hängt häufig von den Eigenschaften des Objektes selbst ab. So ist es hilfreich zu wissen, ob ein Objekt an einen bestimmten Ort oder ein bestimmtes Gerät gebunden ist. Somit kann eine objektspezifische Kommunikationsanpassung durch den Sender sinnvoll sein. Kommunikation mit einem virtuellen Objekt entspricht einer Abbildung auf Kommunikationsmechanismen, die je nach Zustand des lokalen Gerätes und des virtuellen Objektes variieren können. Adressiert werden MobileObjects, die mit dem virtuellen Objekt assoziiert sind. Bei der Verwendung von Repräsentanten übernehmen diese die Aufgabe der Kommunikationsanpassung und nach dem Empfang die Weitergabe von Nachrichten an assoziierte Objekte. Kommunikationsanpassungen können durch Verwendung lokal vorhandener Spurinformationen durchgeführt werden. Darüber hinaus kann aufgrund von Spurinformationen der Erfolg einer Kommunikation abgeschätzt bzw. die Adressierbarkeit des Objektes festgestellt werden. Somit ist es sinnvoll, Spurverwaltung und Kommunikationsabstraktion in ein und demselben Repräsentanten zu realisieren.

6.1 Veranstaltungskontext und Kommunikation

Veranstaltungskontexte sind durch Raum und Zeit bestimmt. Somit kann lokal entschieden werden, ob sich ein Gerät in diesem Kontext befindet oder nicht. Ist ein Gerät räumlich nicht im Kontext, kann es aber dennoch mit Geräten im Kontext kommunizieren. Hierbei wird aber mit anderen Mechanismen kommuniziert, als innerhalb des Kontextes. Nachrichten werden erst zum Ortskontext und dann erst zu den adressierten Zielen ausgeliefert. Die zeitliche Komponente bestimmt die Adressierbarkeit. Außerhalb der Veranstaltungszeiten ist es nicht zu erwarten, über den Ortskontext Objekte adressieren zu können.

6.1 Spurrepräsentierung

Wie schon im letzten Kapitel deutlich wurde, ist die Speicherung und die Darstellung des aktuellen Geräteumfelds wichtig, da hieraus die Informationen für aktuelle Entscheidungen gewonnen werden und hier sich der Ausgangspunkt für Informationen der Spurvergangenheit befinden. Nachbargeräte können durch lokale Repräsentanten dargestellt werden, um die zwischengespeicherten und gesammelten Informationen innerhalb eines Objektes darzustellen. Diese Repräsentanten entstehen und verschwinden dynamisch, wenn Nachbargeräte in Kommunikationsreichweite kommen oder diese wieder verlassen. Häufig bleiben die Repräsentanten für kurze Zeit erhalten, auch wenn ein Gerät nicht benachbart ist, um kurzfristige Linkabbrüche zu kompensieren. Repräsentierung von Geräten kann auch längerfristig Sinn machen. Will eine Applikation eine langfristige Bindung mit einem bestimmten Gerät eingehen, kann der Gerätерepräsentant solange auf dem anderen Gerät verbleiben, bis die Bindung gelöst wird oder nach einem vorgegebenen Zeitraum ausläuft, ohne verlängert zu werden. In beiden Fällen ist in den Objekten vermerkt, dass diese nicht direkt benachbart sind, eine Kommunikation also lokal nicht möglich ist.

Im Falle der Repräsentierung von virtuellen Objekten kann zwischen zwei Spurinformationstypen unterschieden werden. Der erste Typ entspricht Informationen, die lokal über das Objekt gesammelt werden, ohne dass dies Informationen der Spur des repräsentierten Objektes sein müssen. Hierunter fallen lokal geführte Statistiken über das repräsentierte Objekt. Diese Informationen müssen also nicht mit Informationen auf anderen Geräten konsistent sein. Im zweiten Fall handelt es sich um lokal zwischengespeicherte Informationen der im eigentlichen MobileObject vorhandenen Spur. Hier kann es also zu Inkonsistenzen kommen, wenn im eigentlichen MobileObject Informationen geändert werden. Ist eine Kommunikation zwischen repräsentiertem MobileObject und Repräsentant möglich, können Änderungen sofort bzw. in fest vorgegeben Abständen mittels einer Gruppenkommunikation an Repräsentan-

ten verbreitet werden. Existiert das repräsentierte Objekt nicht als Instanz, bedeutet dies, dass die Spurinformationen auf allen Geräten vorhanden sind bzw. auf einer Mindestteilmenge vorhanden sein müssen. Um Konsistenz zu erreichen, muss jeder Repräsentant mit jedem anderen Repräsentanten kommunizieren. Da aber Änderungen nicht von einem, sondern von beliebigen Geräten erfolgen können, ist hier die Konsistenzproblematik größer. Im Allgemeinen ist nur eine eingeschränkte Konsistenz erreichbar. Repräsentanten von Ortskontexten können die Konsistenz auf den Ort beschränken, so dass in der Regel eine Kommunikation der Geräte untereinander möglich ist und Inkonsistenzen beispielsweise durch ein wechselseitiges Ausschlussverfahren vermieden werden kann.

6.2 Veranstaltungskontext und Spuren

Der Veranstaltungskontextrepräsentant in Kapitel 9 unterscheidet vier Spurinformationstypen: globale, ortsglobale, ortslokale und gerätelokale Informationen. Globale Informationen sind zukünftige Veranstaltungsorte und -zeiten, die nur von einem ausgezeichneten Gerät geändert werden können. Ortsglobale Information ist innerhalb des Veranstaltungskontexts konsistent, d.h. sie wird zu jedem Zeitpunkt von genau einem Objekt erzeugt und geändert. Gerätelokale Information wird nur auf den Geräten selbst gesammelt und ortslokale Information wird auch direkten Nachbargeräten mitgeteilt.

6.2 Kommunikation

Die Kommunikation mit anderen Objekten ist geprägt von der Eigenschaft, dass jederzeit mit einem spontanen Kommunikationsabbruch zu rechnen ist, Fehlerbehandlung sollte also die Regel und nicht die Ausnahme sein. Darüber hinaus sollten Anwendungen die Eigenschaften und Eigenarten der verwendeten Kommunikationsmechanismen kennen und deren spezifische Informationen interpretieren können. Wird ein Objekt mit einem auf einen Ort beschränkten Kommunikationsverfahren nicht erreicht, kann dennoch mit einem anderen Kommunikationsverfahren ein Weg zu dem Objekt gefunden werden. Ein und dieselbe Nachricht an ein und dasselbe Ziel kann je nach Applikationszustand und momentaner Umgebung unterschiedlich kommuniziert werden. So kann es notwendig sein, die Nachrichten auch dann auszuliefern, wenn das Ziel die direkte Kommunikationsreichweite verlässt d.h. es sollte versucht werden, über multihop Kommunikation die Nachrichtenauslieferung doch noch zu ermöglichen. Es kann auch ausreichen, die Auslieferung nur bei tatsächlicher

Objektnachbarschaft durchzuführen. Das bedeutet aber auch, dass die Anwendung mit jeder Nachricht in der Lage sein muss zu entscheiden, wie der tatsächliche Transport erfolgen soll. Die Entscheidungen können meist nur von der Anwendung selbst getroffen werden - eine darunter liegende Systemkomponente kann dieses Wissen nicht besitzen und unter Umständen einen stark erhöhten Protokollaufwand produzieren, ohne dass eine Notwendigkeit dafür existiert. Selbst die Absicherung eines Nachrichtentransports kann für jede Nachricht unterschiedliche Anforderungen besitzen. Wird eine Information periodisch mittels eines Locationcasts an eine Region gesandt und ist ein sporadischer Verlust akzeptabel, kann auf eine aufwendige Transportabsicherung verzichtet werden. Muss die Applikation aber über einen potenziellen Verlust informiert werden, kann eine Absicherung zumindest des Anycastwegs zur Region erfolgen. Auch die Übertragung einer einzelnen Nachricht kann je nach Anspruch der Applikation und je nach vorhandenem Netzwerkumfeld unterschiedlich sein. Ob die Nachricht Hop-by-Hop übertragen werden soll oder erst am Ziel wieder zusammengesetzt wird, sollte auch von der Applikation beeinflusst werden können. In vielen Fällen kann auch eine nahezu beliebige Kombination von Routingverfahren sinnvoll sein. So können Routediscovery Nachrichten von linkbasierten Kommunikationsverfahren mittels geographisch beschränkten Broadcastverfahren versendet werden, wie es bei der Kommunikation innerhalb von Ortskontexten eingesetzt wird. Auch eine Hintereinanderschaltung von Verfahren kann benötigt werden, beispielsweise die Verknüpfung von positionsbasierten mit linkbasierten Verfahren [8] oder die Abarbeitung mehrerer unterschiedlicher Kommunikationsziele. Über die verwendeten Verknüpfungen und Adaptionen kann meist nur der Applikationsentwickler entscheiden. Daher sollten Routingverfahren komponentenbasiert realisiert werden und diese für jede kommunizierte Nachricht wählbar, austauschbar und konfigurierbar sein. Darüber hinaus ist es notwendig, verschiedene Sicherungs- und Transportstrategien auswählen zu können, die dynamisch hinzu geschaltet oder abgeschaltet werden können.

Insgesamt wird das traditionelle Schichtenmodell aufgeweicht - in adhoc Netzwerken ist häufig mehr Flexibilität notwendig als es mit Erweiterungen wie der schichtenübergreifenden Kommunikation möglich ist (z.B.: [72, 48, 98, 99]), die bisher auch meist auf die unteren Schichten beschränkt bleiben. Die Aufweichung bedeutet aber auch einen enormen Transparenzverlust und damit eine Zunahme der Komplexität für Applikationsentwickler. Die hohe Adaptierbarkeit und Konfigurierbarkeit wird aber nicht immer benötigt wird. Daher setzt hier die Repräsentantenkommunikation an, die applikationsspezifisch und auch applikationsübergreifend einen Teil der hinzugekommenen Komplexität kapselt. In der Regel ist es möglich, für eine bestimmte Realisierung eines Objekttyps die Kommunikation für ein gegebenes Netzwerk zu modellieren. Je nach aktueller Netzwerkumgebung bzw. je nach Wissen über das Ziel und das Netzwerk, kann hierbei dennoch zwischen verschiedenen Strategien gewählt werden. Im Beispiel der Ortskontextkommunikation ist die Entscheidung einfach, ob der Sender vor Ort ist oder nicht.

Häufig kann aber lokal vorhandenes Wissen nur durch Heuristiken interpretiert werden, da nur Vermutungen über die Erreichbarkeit des Ziels und die möglichen Wege dorthin getroffen werden können. In diesem Fall muss sich die Applikation zum einen der Tatsache einer nicht garantierten Auslieferung bewusst sein und zum anderen auch die Kommunikationsstrategie beeinflussen können. So kann mit der Duplikation von Nachrichten eine höhere Auslieferungswahrscheinlichkeit erreicht werden. Das Wissen, auf dem Heuristiken basieren können, gehört zu den Spurinformationen über das Zielobjekt. Somit ist es sinnvoll, Kommunikation und Spurverwaltung innerhalb eines Objektes zu vereinen.

Die einfachste Variante der Repräsentantenkommunikation ist die Kommunikation mit Objekten in direkter Kommunikationsreichweite. Auch hier hängt die Art und Weise der Kommunikation von dem momentanen Umfeld des Netzwerks ab. Handelt es sich um einen Orts- oder Gruppenkontext, zu dem die Geräte explizit zusammengekommen sind, kann mit hoher Wahrscheinlichkeit zuverlässig kommuniziert werden. Handelt es sich dagegen um ein beliebiges benachbartes Gerät und kann keine Einschränkung über dessen Verhalten getroffen werden, kann eine gestartete Interaktion früh abbrechen und unter Umständen nie beendet werden. Lokale Interaktion kann aber auch die Hinzunahme von wenigen Kommunikationshops bedeuten, wenn von einem stabileren Umfeld ausgegangen werden kann. Gruppen- und Ortskontexte können sich so über wenige Hops erstrecken. Lokale Interaktion mit Nachbargeräten lässt sich auch zur multihop Kommunikation ausweiten. So kann beispielsweise versucht werden, eine lokal begonnene Interaktion über weitere Nachbargeräte noch abzuschließen, auch wenn die lokale Kommunikation abgebrochen ist. (siehe Abbildung 6.1)

Soll über größere Entfernung kommuniziert werden, ist dies meist nur möglich, wenn der Empfänger gewisse Eigenschaften besitzt. Existieren nur sehr wenige Kommunikationsziele im Vergleich zur Gesamtzahl aller Ziele, können hier auch linkbasierte Verfahren eingesetzt werden. Das liegt darin begründet, dass in den beschränkten Routingtabellen bzw. -caches diese Ziele mit hoher Wahrscheinlichkeit enthalten sind. Auch geographisch stationäre Objekte lassen sich über größere Entfernungen leichter adressieren, wenn die Geräte des Netzwerks Positionsinformationen besitzen. Insbesondere geographisches Wissen über ein entferntes Objekt kann helfen, es mit geringerem Aufwand zu adressieren. Müssen hingegen Zielobjekte zuerst aufwendig im Netzwerk gesucht werden, ist eine Kommunikation über größere Entfernungen nur bedingt realisierbar. Insbesondere dann, wenn viele oder gar alle Geräte im Netz nach nahezu beliebigen, im Netz verstreuten Zielen suchen. Die einzige brauchbare Lösung, um ein beliebiges Objekt in einem dynamischen Umfeld zu adressieren, ist das Wissen über seinen geographischen Ort. Darüber hinaus müssen Geräte, die auf einem Weg zu diesem Ort liegen, in der Lage sein, Nachrichten dem Ziel mittels lokalem Wissen näher zu bringen. Beispiele für solche Kommunikationsverfahren sind positionsbasierte Verfahren und

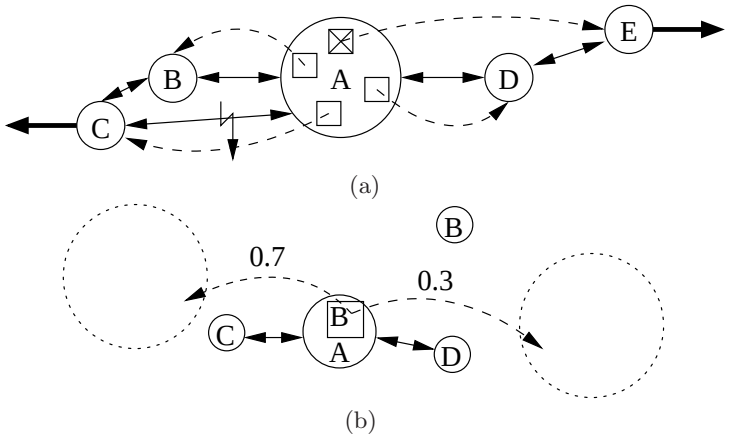


Abb. 6.1. Beispiele für den Einsatz lokaler Gerätерepräsentanten. In (a) werden alle Nachbarn lokal repräsentiert. Interaktionen mit dem markierten Gerät E werden über D fortgesetzt auch wenn es nicht mehr benachbart ist. Interaktionen mit Gerät C werden abgebrochen, selbst wenn eine Interaktion über B möglich ist. In (b) ist lokal das Wissen vorhanden, dass sich Gerät B sehr wahrscheinlich in einer der markierten Regionen aufhält. A kann in diesem Fall über beide Nachbarn in den Regionen nach B suchen. A kann aber auch direkt eine Nachricht an B in beide oder nur in Richtung der wahrscheinlicheren Region senden. Es ist möglich, dass B sich nicht in den Regionen aufhält. Selbst wenn B erreichbar ist, kann A dann das Ziel so nicht finden.

Clusteringverfahren in Netzen, die in bestimmten Netzregionen zu quasipermanenten Gerätehäufungen neigen und diese durch Cluster abdecken können.

Wenn das Ziel selbst ein Ortskontext, ein an einen Ort gebundener Dienst, oder ein stationäres Gerät mit bekannter Position ist, kann das Wissen über den Ort vorhanden sein, falls schon einmal mit diesem Objekt interagiert worden ist. Somit ist mit diesen Objekten nach einer lokalen Interaktion häufig auch eine entfernte Interaktion möglich. Die Aufenthaltsorte können sich auch über die Zeit ändern, und zukünftige Orte unter Umständen sogar im Vorfeld bekannt sein. Veranstaltungen aber auch Marktplätze können diese Eigenschaften besitzen. Das Wissen über die zukünftige Spur des Objektes kann lokal in den Repräsentanten abgelegt werden. So kann, je nach Uhrzeit, ein anderer Zielort angesprochen werden. Objekte wie z.B. Marktplätze können auch an mehreren Orten zur selben Zeit existieren. Applikationen sind dabei in der Lage, eine Auswahl zu treffen, mit welchen Zielen interagiert wird. Zur Auswahl der Ziele kann die Zuverlässigkeit, die Distanz, die Zahl der zu erwartenden Interaktionspartner oder andere Kriterien dienen. Darüber hinaus kann es auch sinnvoll sein, mehrere der zur Auswahl stehenden Ziele anzusprechen. Die Auswahl des besten Ziels zur Interaktion kann auch innerhalb des lokalen Objektrepräsentanten getroffen werden. Da die Applikation

berücksichtigen muss, dass eine Zielauswahl erfolgen kann, kann sie aber nicht gänzlich transparent erfolgen. Aufeinanderfolgende abhängige Interaktionen können bei völliger Transparenz zu unerwarteten Effekten führen, wenn das Interaktionsziel gewechselt wird. Redundanz, also das gleichzeitige Ansprechen von mehreren Zielen, ist auch nur dann möglich, wenn die Applikation mit mehreren, unter Umständen verschiedenen Antworten rechnet. Redundanz kann bei der Wahl des Ziels sinnvoll sein, um festzustellen, welches der Auswahlziele am besten bzw. überhaupt erreichbar ist. Diese initiale Zielfindung ist aufgrund der verwendeten Vorgaben erheblich effizienter und erzeugt erheblich weniger Netzwerklast als auf Fluten basierende Verfahren.

Die oben beschriebene Zielfindung kann auch für MobileObjects verwendet werden, deren Aufenthaltsorte nicht fest vorgegeben sind. Vorraussetzung ist hier, dass Abschätzungen existieren, an welchen Orten sich das Objekt zu einem bestimmten Zeitpunkt aufhalten kann. Eine solche zukünftige Spur kann für Geräte erstellt werden, die sich nahe an feste Vorgaben (z.B. Stundenpläne oder Terminkalender) halten, oder ein wiederkehrendes Verhalten zeigen. Stehen für einen bestimmten Zeitpunkt mehrere Aufenthaltsorte zur Auswahl, können diese auch hier parallel zur Zielfindung abgefragt werden, um herauszufinden, an welchem Ort sich das Objekt tatsächlich aufhält.

6.3 Gerätерepräsentanten und Stundenpläne

Stehen Stundenpläne eines Nutzers zur Verfügung, wie es bei Studenten, die regelmäßige Veranstaltungen besuchen, der Fall ist, kann aufgrund vergangener Erfahrungen mit gewisser Wahrscheinlichkeit ein Aufenthaltsort oder eine Region vorhergesagt werden. U.U. stehen auch Alternativorte zur Auswahl. Diese Informationen können anderen Objekten mitgegeben werden, so dass diese zu einem späteren Zeitpunkt mit dem Gerät interagieren können. Als Beispielszenario kann UbiBay dienen, bei dem ein Student mit einer Auktion über zwei Sunden teilnimmt. Der Beispielstudent hält sich für diese Zeit mit einer gewissen Wahrscheinlichkeit in der Vorlesung auf oder befindet sich in der Cafeteria. Beide Orte können angesprochen werden, so dass der Student mit einer hohen Wahrscheinlichkeit erreicht wird.

Neben der Zielfindung können Spurinformationen auch weitere Unterstützung zum Treffen von Kommunikationsentscheidungen bieten. So kann z.B. die Wahl des eigentlichen Kommunikationsverfahrens vom Zustand des lokalen Gerätes, von der Umgebung und vom Ziel abhängig sein. Der Wechsel zwischen linkbasierter und geographischer Zielfindung kann durchgeführt werden, wenn das Ziel sich am selben oder an einem anderen, aber bekannten, geographischen Ort aufhält. Im ersten Fall kann das Ziel mit wenigen Hops oder gar singlehop erreicht werden – hier sprechen Positionsungenauigkeiten und das benötigte Gerätepositionsverzeichnis gegen positionsbasierte Verfah-

ren. Im zweiten Fall kann zuerst die Aufenthaltsregion des Ziels adressiert werden, und dann vor Ort linkbasierte Zielfindung verwendet werden. Das Wissen darüber, dass ein Gerät sich zumindest in derselben Region befindet, kann für linkbasierte Verfahren auch zum Einschränken einer Routenfindung verwendet werden; außerhalb der Region muss das Gerät also nicht gesucht werden.

Objektrepräsentanten von virtuellen Objekten können auch Kommunikationsmuster repräsentieren. Ein lokaler Marktplatzrepräsentant realisiert somit einen Teil der marktplatzbasierten Kommunikation. Aufträge, die als MobileState realisiert sind, werden dem lokalen Markplatzrepräsentanten übergeben. Dieser kümmert sich um die Zielfindung des Objektes. Hierfür muss zuerst ein geeigneter Zielmarktplatz in der näheren Umgebung existieren und gefunden werden. Wenn dies nicht der Fall ist, wird ein neuer Marktplatz erzeugt. Die Ortsdaten werden dem MobileState übergeben, der nun per geographischem Store-and-Forward dorthin gesendet werden kann.

6.4 Geräterepräsentanten auf Fixpunkten

Objektrepräsentanten auf stationären Gräten können ausgezeichnete Rollen zugewiesen bekommen. So können auf diesen Geräten die aktuelle Position eines Gerätes hinterlassen werden, und Nachrichten von anderen Objekten somit an den momentanen Ort des repräsentierten Gerätes weitergeleitet werden. Dies kann bei der marktplatzbasierten Kommunikation verwendet werden, wenn ein Auftrag längere Zeit in Anspruch nimmt. Hierbei verwaltet das auftraggebende Objekt eine Folge von Orten oder stationären Geräten, die eine positionsbasierte Route zum Gerät ermöglichen. Das Auftragsobjekt kennt nur die Position der ersten Station der Folge. Die Folge wird verändert, sobald der Auftraggeber seinen Aufenthaltsort stark verändert. Dabei können Stationen hinzukommen oder aufgrund von Optimierungen wieder wegfallen.

6.3 Repräsentantenverwaltung

Aktive Repräsentantenobjekte können auf zwei Arten entstehen. Sie können zum einen explizit durch eine Applikation erzeugt werden. Diese besitzt hierbei Wissen über das zu repräsentierende Objekt, um die minimal nötigen Spurinformationen zur Erzeugung zur Verfügung zu stellen. Zur Kommunikation mit dem repräsentierten Objekt kann beispielsweise der Objektaufenthaltsort notwendig sein, also eine Geräteadresse oder ein identifizierbarer Ort. Zum anderen können Repräsentantenobjekte aber auch selbstorganisierend bzw. teilautomatisch entstehen. Hierfür müssen lokal Mechanismen vorhanden sein,

die für verschiedene MobileObjects desselben Typs zuständig sind und Repräsentantenobjekte für diese erzeugen. Die einfachste Realisierung lässt sich generisch anbieten, wobei das Repräsentantenobjekt einen Konstruktor anbietet, um aus einer gegebenen Klasse, Konfigurations- und Spurdaten ein Repräsentantenobjekt zu erzeugen. Die dafür benötigten Daten müssen vom repräsentierten Objekt selbst oder einem anderen Repräsentanten abfragbar sein, wobei diese sich in direkter Kommunikationsreichweite befinden. Somit kann ein solcher Dienst mit lokalen Unicasts, unter Angabe eines Nachbargerätes und einer Objektkennung, die benötigten Daten anfragen und lokal einen Repräsentanten[1] erzeugen und aktivieren. Der Erzeugungsauftrag stammt dabei von einem anderen, lokal aktiven Objekt. Der Dienst entscheidet also nicht über das Erzeugen von lokalen Objektrepräsentanten. Andere Objekte können beispielsweise anhand der Gerätespur des Nachbargerätes ein Objekt erkennen, welches automatisch erzeugt werden kann und darauf aufbauend die Entscheidung zur Erzeugung des Repräsentanten treffen. Ein Sonderfall sind benachbarte Geräte. Hier werden die Repräsentantenobjekte automatisch vom MobileDevice-Objekt des lokalen Gerätes erzeugt.

Repräsentantenobjekte können die Eigenschaft besitzen, sich automatisch zu beenden und zu löschen, sobald das repräsentierte Objekt sich nicht mehr in der Nachbarschaft befindet. Außerdem können Repräsentantenobjekte eine Lebensdauer besitzen, die kleiner ist, als die des zu repräsentierenden Objektes. In beiden Fällen kann es notwendig sein, das Löschen des Repräsentanten zu verhindern oder zu verschieben, damit die Objektinformationen nicht verloren gehen oder um weiterhin mit dem Objekt auch über größere Entfernung zu kommunizieren. Daher können Repräsentanten *stay* und *leave* Direktiven anbieten, die sie beispielsweise an eine von zwei generische Realisierungen delegieren können. Die erste Realisierung arbeitet mit Referencecounting, d.h. alle Objekte, die *stay* signalisiert haben oder den Repräsentanten erzeugt haben, müssen auch *leave* signalisieren. In der zweiten Variante setzt die *stay* Signalisierung die Restlebenszeit auf den übergebenen Wert, falls dieser größer ist. Hierbei können sich auch Objekte registrieren, die im Falle des Ablaufs der Lebenszeit ein “Veto” einlegen, also die Lebenszeit verlängern können.

[1] In der jetzigen Realisierung muss der Repräsentantencode lokal vorhanden sein. Dieser kann aber bei Bedarf einfach mit versendet werden.

7

Simulation mobiler ad-hoc Netzwerke

Die Realisierung und das Testen von Anwendungen, Middlewarekomponenten oder Protokollen, die für den Einsatz in mobilen ad-hoc Netzwerken gedacht sind, ist ungemein schwieriger, als es für stationäre Netzwerke der Fall ist. Eine direkte Realisierung auf Basis von Endgeräten inklusive umfangreicher Tests gestaltet sich schon allein deshalb als schwierig, weil eine große Zahl mobiler Geräte notwendig ist. Diese müssen darüber hinaus in einer Vielzahl von Testkonstellationen aufgestellt werden, um möglichst viele der im realen Einsatz möglichen Netzwerkszenarien durchspielen zu können. Darüber hinaus ist im Falle der Betrachtung mobiler Szenarien auch eine Vielzahl an Testpersonen notwendig, die Gerätemobilität nachbilden. Testpersonen sind insbesondere dann notwendig, wenn Applikationen auch Benutzerinteraktion benötigen. Bei der Durchführung von Feldversuchen ist es häufig schwierig, aufgetretene Fehler zur genaueren Untersuchung zu reproduzieren, da exakte Testwiederholungen nahezu unmöglich sind. Aus diesem Grund ist es üblich, insbesondere Protokolle für mobile ad-hoc Netzwerke simulativ zu untersuchen und zu testen. In diesem Bereich haben sich einige Simulatoren wie z.B. ns-2 [23], OMNeT++ [107] oder GloMoSim [114] etabliert, die insbesondere für die Realisierung, das Testen und die Evaluation von Routingprotokollen für multihop ad-hoc Netzwerke eingesetzt werden. Daher wird hier sehr viel Wert auf eine möglichst realitätsnahe Simulation der Protokollschichten und des drahtlosen Mediums gelegt. Der dabei entstehende Simulationsaufwand schränkt die Größe des simulierten Netzwerks erheblich ein; üblich sind hier wenige hundert simulierte Geräte. Zur Simulation von Anwendungen oder Middlewarekomponenten sind diese Simulatoren aber nicht vorgesehen. Die Anwendungsschicht besteht bei diesen Simulatoren meist aus simulierten Ende-zu-Ende Datenströmen, die etablierte Internetanwendungsprotokolle wie FTP oder HTTP über TCP nachbilden. Bei Anwendungen für mobile ad-hoc Netzwerke steht weniger eine detailgetreue Netzwerksimulation, sondern stärker die Simulation der Netzwerkstrukturen im Vordergrund. Die Netzwerkstrukturen entstehen auf Basis von Gerätemobilitätsmodellen, die Gerätebewegungen simulieren.

Die ereignisbasierte Simulation ist der verbreitetste Realisierungsansatz bei Simulatoren für ad-hoc Netzwerke. Zentrales Element ist der Ereigniskalender, in den zukünftige Ereignisse eingetragen werden, um an dem ihnen zugeordneten Zeitpunkt ausgeführt zu werden. Die Zeit schreitet hierbei aufgrund der eingetragenen Ereignisse voran. Das bedeutet, dass der Zeitpunkt des jüngsten Ereignisses den aktuellen Simulationszeitpunkt bestimmt. Ereignisse werden aus Sicht des Kalenders in Nullzeit ausgeführt, die Zeit schreitet also während der Ereignisroutine nicht voran. Ereignisse können neue Ereignisse erzeugen, die aber in der Gegenwart oder in der Zukunft liegen müssen. Die ereignisbasierte Betrachtung ist gerade für mobile ad-hoc Netzwerke ideal, da diese Umgebung aufgrund des dynamischen Umfelds stark durch Ereignisse geprägt ist. Applikationen sollten das dynamische Umfeld ebenfalls ereignisorientiert betrachten, um somit jederzeit auf Änderungen reagieren zu können.

Routingprotokolle können mit einfachen Verfahren wie der Erzeugung einer konstanten Bitrate, zufällig erzeugtem Nachrichtenaufkommen oder dem Nachbilden von Protokollabläufen höherer Schichten (z.B. FTP) durch zufällig generierte Anfragen getestet werden. Das Testen gestaltet sich aber bei Applikationen, die speziell für ad-hoc Netzwerke entworfen wurden und somit nicht das Verhalten von Applikationen in Infrastrukturnetzen besitzen erheblich schwieriger. Das gilt insbesondere dann, wenn Applikationen untersucht werden, die auch Benutzerinteraktion berücksichtigen müssen. Benutzerverhalten zu simulieren ist zwar möglich, aber gestaltet sich häufig relativ schwierig, da auch unerwartetes Benutzerverhalten modelliert und getestet werden muss. Hierfür bieten sich hybride Ansätze an, wobei Applikationen in einer simulierten Umgebung von Benutzern getestet werden können, indem Teile der Applikation, insbesondere die Benutzerschnittstelle, auf externe aber an die Simulation angebundene Geräte ausgelagert werden. Verbreiteter als der hybride Ansatz ist die Netzwerkemulation. Hierbei werden in drahtgebundenen Netzen mobile Netze emuliert, indem die Kommunikation auf einen Simulationsserver umgeleitet wird, der Mobilität und Netzwerk simuliert und die Nachrichten an die Zielgeräte im drahtgebundenen Netzwerk weiterleitet [25, 15]. Die Emulation erfolgt für die darauf basierenden Anwendungen vollkommen transparent, d.h. sie sind nicht in der Lage, auf tiefere Schichten im Protokollstapel zuzugreifen oder diese zu adaptieren. Darüber hinaus wird ein Großteil des Protokollstapels des Betriebssystems verwendet, der nicht in der Lage ist, mit der Dynamik eines ad-hoc Netzwerks umzugehen. Ein ähnlicher Ansatz verfolgt MarNet[21]. Dabei werden allerdings mehrere Betriebsysteminstanzen auf einem Gerät in der Virtualisierungsumgebung ausgeführt und die Netzwerksimulation erfolgt auf dem selben Gerät. Eine Adaption ist bei MarNet möglich, indem zur Laufzeit die Kernelmodule von Routingverfahren ausgetauscht werden.

Insbesondere für die Realisierungen von Anwendungen, die auf ad-hoc Netzwerken mit einer größeren Zahl mobiler Geräte basieren, bietet sich ein

dreistufiger Entwicklungsprozess wie in [66] beschrieben an. In einem ersten Schritt werden dabei Anwendungskomponenten und Interaktionsprotokolle in einer simulativen Umgebung realisiert und untersucht. Wenn möglich, werden diese Anwendungsteile mit simuliertem Benutzerverhalten getestet. In einer simulativen Umgebung ist es möglich, sehr umfangreiche statistische Untersuchungen durchzuführen und den implementierten Code unter einer Vielzahl von Szenarien zu testen. Aufgrund des deterministischen Verhaltens eines Simulators lassen sich Fehlerfälle leicht reproduzieren und unterschiedliche Applikationsparameter hinsichtlich ihrem Einfluss auf das Applikationsverhalten testen. Wenn möglich, lässt sich die Applikation mit simulierten Benutzerverhalten testen, so dass schon hier eine Abschätzung über das Verhalten in der Realität getroffen werden kann. Im nächsten Schritt wird der verbleibende Rest der Applikation, insbesondere die graphische Benutzeroberfläche implementiert. Die Gesamtapplikation kann nun in einer hybriden Umgebung mit simulierten mobilen Geräten und realem Benutzerverhalten getestet werden, wobei reale Geräte an die Simulation angebunden werden. Die Reproduktion von Fehlern ist hier allerdings schwieriger, als in einer rein simulierten Umgebung, da aufgrund der asynchronen Benutzerinteraktion kein deterministisches Simulationsverhalten mehr möglich ist. Dennoch ermöglicht die hybride Umgebung eine schnellere Reproduktion von Fehlerfällen als ein Feldversuch und vor allem aber eine reproduzierbare Gerätemobilität. Eine Variante dieses Vorgehens erlaubt auch die Beeinflussung der Gerätemobilität durch die Testpersonen. Im Idealfall entspricht die Durchführung eines Feldversuchs einem Proof-of-Concept, bei dem nur noch leichte Änderungen zur Anpassung an die reale Umgebung notwendig sind. Feldversuche sind erforderlich, da die simulierte Umgebung nur eine Annäherung an die reale Umgebung sein kann, und es immer zu Abweichungen, auch in den Tendenzen, der Messergebnisse zwischen der simulierten Umgebung der Realität kommen kann. Es kann sogar passieren, dass die Applikation ein stark verändertes Verhalten aufweisen kann. Dann ist es notwendig, dass die Simulationsparameter an die Erfahrungen aus den Feldversuchen angepasst werden und die simulative Untersuchung der Applikation wiederholt werden muss. Somit muss auch eine einfache Reproduzierbarkeit von Feldversuchen in der Simulation möglich sein.

Wird dieses Vorgehen angewendet, ist es wichtig, dass die Arbeitsumgebung, primär die Simulationsumgebung, die notwendige Unterstützung für alle diese drei Entwicklungsphasen bietet. Insbesondere sollte der Aufwand, alle Phasen durchzuführen, minimal sein und es nicht notwendig machen, für alle drei Modi Applikationskode neu zu implementieren. In den Arbeiten [67, 33], wird eine Simulationsumgebung auf Java-Basis vorgestellt, die diesen Anforderungen entspricht. Daher wurde diese Umgebung als Ausgangsbasis verwendet, und um die in Abschnitt 8.1 beschriebenen Mechanismen des Middlewarekerns erweitert.

7.1 Mobilitätsmodelle

Für Anwendungen in großen mobilen ad-hoc Netzwerken, die Annahmen an das sie umgebende Netzwerk und insbesondere an die Benutzermobilität stellen, muss sehr viel Wert auf das verwendete Gerätemobilitätsmodell gelegt werden, damit das simulierte Netz auch diesen Eigenschaften genügt. Zur Simulation von Routingverfahren werden stationäre und sehr einfache mobile Netzwerkszenarien verwendet. Insbesondere bei zustandslosen Verfahren wird aufgrund der Annahme, dass die Paketauslieferung im Vergleich zur Gerätemobilität um ein Vielfaches schneller abläuft, häufig mit unterschiedlichen stationären Netzwerkszenarien simuliert. Hierbei werden die mobilen Geräte unter gewissen einschränkenden Parametern auf eine vorgegebene Fläche verteilt. Wichtigste Einschränkung ist, dass zumindest die betrachteten Sender und Empfänger durch mindestens einen möglichen Kommunikationspfad verbunden sind. Zu nennen sind auch Parameter wie Dichte, mittlerer Nachbarschaftsgrad der Knoten und Häufungs- und Leerbereiche in der Netzabdeckung der Fläche (siehe z.B. [31]). Als mobiles Netzwerkszenario wird meist eine Variante des *Random-Waypoint* [56] Modells verwendet. Bei dem Verfahren wird ausgehend von der letzten Geräteposition eine zufällige nächste Position aus der betrachteten Ebene gewählt und das Gerät linear mit einer vorgegebenen oder auch zufällig gewählten Geschwindigkeit dorthin bewegt. Nach einer in der Regel exponenzialverteilten Wartezeit wird der Vorgang wiederholt. Auf Basis dieser einfachen Variante wurden, insbesondere zur Untersuchung von Applikationen, komplexere Modelle entworfen. Die Hinzunahme von Hindernissen [53] schließt Regionen als Zielwahl aus und macht eine nichtlineare Bewegung zum Ziel notwendig, wenn der direkte Weg zum Ziel versperrt ist. Dadurch erhält man auch in mobilen Szenarien zuverlässig Leerbereiche in der Netzwerkabdeckung der Fläche. Mehr oder weniger ausgeprägte Häufungsregionen können durch die Überlagerung von mehreren Random-Waypoint Modellen [45] realisiert werden. Im einfachsten Fall wird zur nächsten Zielpunktwahl zufällig ein Random-Waypoint Modell bestimmt und dieses zur eigentlichen Zielpunktwahl verwendet. Die Geräte bewegen sich, wie im Ursprungsverfahren von ihrer momentanen Position linear zum so bestimmten, nächsten Ziel. Somit lassen sich anonyme Gerätehäufungen realisieren. Es ist aber nur gewährleistet, dass Geräte sich mit höherer Wahrscheinlichkeit an bestimmten Orten aufhalten, aber es ist nicht möglich, bestimmte Geräte oder Gerätegruppen an bestimmte Orte zu binden.

Personen bewegen sich meist auf fest vorgegebenen Wegen wie Straßen oder Gebäudefluren. Solche Wegnetze sind die Grundlage für Pfadnetzbewegungsmodelle. Die einfachste Variante ist das *Restricted Random-Waypoint* Modell des Terminodes Projekts [8]. Hier werden Städte als Rechtecke modelliert, in denen sich Geräte nach dem ursprünglichen Random-Waypoint Modell bewegen. Soll ein Gerät die Stadt wechseln, wird zufällig eine benachbarte Stadt ausgewählt, dort der nächste Zielpunkt bestimmt und das Gerät linear dorthin bewegt. Somit entstehen zwischen den Städten Korridore, auf

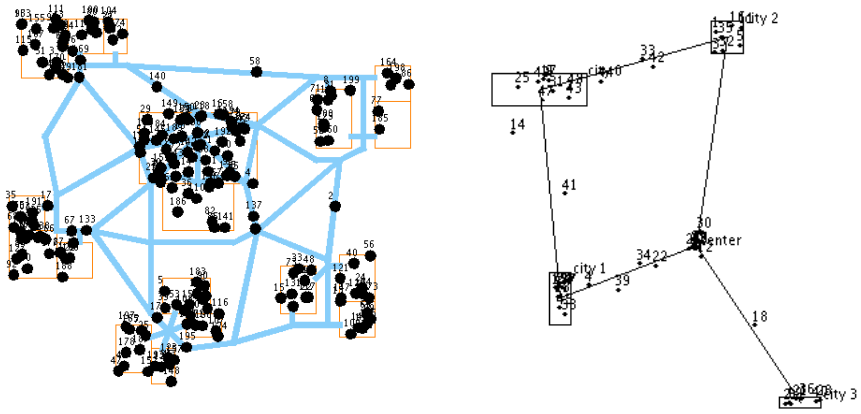


Abb. 7.1. Links eine Visualisierung eines Pfadnetzmodells, rechts eine Restricted Random-Waypoint Variante.

denen sich Geräte bewegen. Da hier das Pfadnetzwerk nur zur Städtewahl verwendet wird, ist eine Modellierung von echten Kreuzungspunkten nur über Städte möglich und eine Beschränkung der Wegbreite nur sehr schwierig realisierbar. Dennoch lassen sich hieraus Szenarien entwerfen, die kleinere Netze realisieren, in denen die Städte Gebäuden entsprechen, beispielsweise eines Universitätscampus. Echte Pfadnetzmodelle wie [105, 35] verwenden für die Zielregionen häufig das Random-Waypoint Modell, das Pfadnetz wird aber direkt zur Bewegung der Geräte verwendet. Die Kanten eines Pfadnetzes entsprechen den Wegen des Modells mit einer vorgegebenen Breite. Knoten sind hierbei Wegkreuzungen oder Zielregionen. Wird von einem Gerät eine neue Zielregion gewählt, wird zuerst ein Weg zum Ziel im Pfadnetz gesucht. Dabei wird entweder ein Kürzeste-Wege-Algorithmus verwendet (z.B. Dijkstra oder für Pfadalternativen k-Shortest-Paths [22]) oder auf vorgegebenen Routingentscheidungen, basierend auf Wahrscheinlichkeiten in den Knoten, zurückgegriffen. Die Bewegung des Gerätes wird dann entlang des gefundenen Pfades, unter Umständen mit einer Variierung in der Wegbreite, durchgeführt. Mit diesen Modellen lassen sich reale Umgebungen detailgetreu nachbilden, der Aufwand hierfür ist allerdings sehr hoch. Neben Random-Waypoint lassen sich für die Zielregionen beliebige Zielwahlverfahren einsetzen. In [47] wird ein Hörsaal oder ein Seminarraum modelliert, indem sich die Geräte zufällig auf vorgegebene Plätze begeben. Wenn, wie in den Pfadnetzmodellen, identifizierbare Regionen zur Verfügung stehen, lassen sich den Geräten auch Verhaltensmuster zuordnen. Im universitären Umfeld lassen sich Stundenplanmodelle [42] einsetzen, um Geräte auf einem Pfadnetz nach gegebenen Veranstaltungsplänen zu bewegen. Damit selbst bei gleichen Stundenplänen dennoch unterschiedliches Verhalten der Geräte auftritt, werden Start- und Endzeiten der Veranstaltungen, der tatsächliche Besuch einer Veranstaltung

und die Wegewahl zwischen Veranstaltungen randomisiert. Auch hier ist eine detaillierte Modellierung möglich, aber mit hohem Aufwand verbunden.

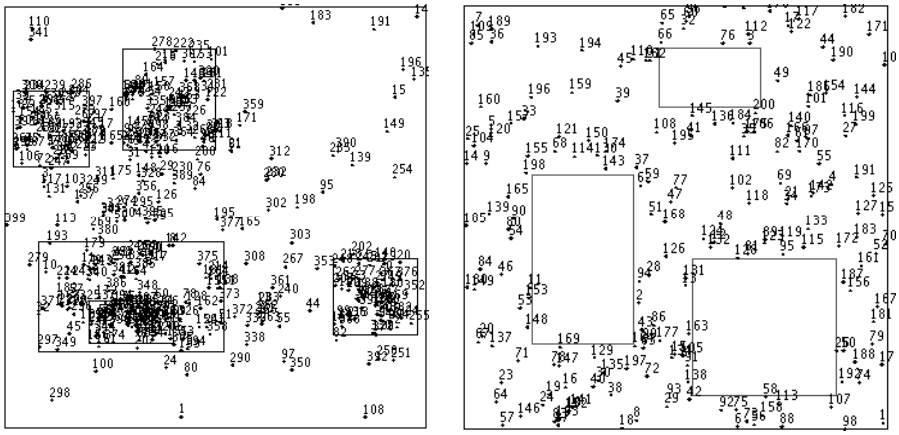


Abb. 7.2. Links eine Visualisierung eines Modelle aus mehreren Random-Waypoint Modellen, rechts werden die Rechtecke bei der Punktewahl ausgelassen.

Stehen reale Bewegungsdaten zur Verfügung, können diese meist direkt in Simulatoren verwendet werden. Solche, in der Regel sehr speziellen, Daten stehen z.B. für auf Interfahrzeugkommunikation basierende Netze zur Verfügung [27]. Bewegungsdaten lassen sich natürlich auch durch das Erstellen von deterministischen Bewegungsskripten für einzelne Geräte erzeugen. Größere Szenerien auf diese Weise zu modellieren, ist aber aufgrund des Aufwands der Skripterstellung nicht vertretbar. Einsatz finden Bewegungsskripte meist bei der Untersuchung kleiner spezieller Szenerien, um bewusst Extremfälle zu modellieren. In Kombination mit Feldversuchen, die mit Bewegungsplänen [6] durchgeführt werden, kann hiermit versucht werden, ein Bewegungsszenario in der Simulation zu reproduzieren. Interaktive Bewegungsszenarien, bei denen die Geräte zur Laufzeit der Simulation durch Benutzerinteraktion gesteuert werden, erlauben es schließlich, on-the-fly Bewegungsmuster zu erzeugen. Einsatz findet dies zu schnellen Tests mit wenigen Geräten, aber auch bei interaktiven Demonstrationen [42].

7.2 Netzwerkmodelle

Im Idealfall bildet das simulierte Netzwerk das reale Netzwerk möglichst detailgetreu nach. Das bedeutet neben einer korrekten Implementierung der unteren Protokollschichten, insbesondere der MAC-Schicht, auch eine Nachbildung der physikalischen Eigenschaften des Funknetzwerks. In [68] wird eine

solche detaillierte Netzwerkrealisierung auf Basis des verwendeten Simulationskerns vorgestellt. Es handelt sich um eine Implementierung des IEEE802.11 MAC Protokolls, wobei der Schwerpunkt der Implementierung auf dem ad-hoc Modus des Standards beruht. Die MAC Schicht basiert auf einer simulierten physikalischen Schicht, die wiederum auf einem simulierten Medium realisiert ist. Das Medium unterstützt beliebige Signalausbreitungsmodelle, die aufgrund der Schichtung einfach ausgetauscht werden können. Standardmäßig wird das *Two-Ray-Ground Modell* [95] verwendet, welches neben der Abschwächung des Signals über die Distanz auch die Bodenreflexion des Signals berücksichtigt. Für jede versendete Nachricht muss überprüft werden, welche Geräte sich im Empfangs- oder zumindest im Störbereich des Senders befinden um somit Kollisionen und Störungen beim Empfang bestimmen zu können. Je nach Detailgrad der Simulation werden die Bereichsbestimmungen für jede Nachricht mehrfach wiederholt. Bei diesem Modell wird die Signalstärke des Senders, der Antennengewinn, ein randomisierbares Hintergrundrauschen und der Signal-Rauschabstand beim Empfänger berücksichtigt. Die physikalische Schicht entscheidet in dieser Realisierung, ob Nachrichtenpakete empfangen werden oder ob der Störeinfluss anderer Geräte zu groß ist. Hierbei wird, ähnlich der ns-2 [23] Realisierung, das stärkste Signal empfangen, sobald es über einem Empfangsschwellenwert liegt (capture effect). Die MAC Schicht realisiert das IEEE802.11 Protokoll und bildet es auf Ein- und Ausschalten des Transmitters ab. Eine Bitkodierung der zu übertragenden Daten findet nicht statt.

Diese detaillierte Simulation des Netzwerks führt zu einem hohen Berechnungsaufwand, dem in manchen Simulationsszenarien ein geringer Nutzen gegenüber steht. Während bei der Simulation von Kommunikationsverfahren eine detaillierte Netzwerksimulation im Vordergrund steht, um z.B. Kommunikationsoverhead, Verzögerung, Datendurchsatz oder Energieverbrauch bestimmen und vergleichen zu können, treten diese Eigenschaften auf Anwendungsebene eher in den Hintergrund. Hier ist es meist wichtiger, die Anwendung unter verschiedenen Mobilitätsszenarien zu betrachten und dabei eine hohe Gerätezahl zu erreichen. Dennoch sollen in gewissem Maße die Eigenschaften eines drahtlosen Netzes erhalten bleiben. Daher können in diesem Fall statistische Netze verwendet werden, wobei Nachrichtenverluste, Datendurchsatz oder Protokolleigenschaften nicht detailliert simuliert werden. Stattdessen werden auch hier, ähnlich der Ausbreitungsmodelle, Zufallsverteilungen eingesetzt. Eine weitere Möglichkeit ist die vereinfachte Modellierung von Protokollschichten, bei der zu Gunsten des Berechnungsaufwandes versucht wird, das Protokollverhalten nachzubilden, und nicht die tatsächlichen Algorithmen zu implementieren. Die verwendete Simulationsumgebung bietet ein Netzwerkmodell (`SharedNetwork`), das den IEEE802.11 Standard mit RTS/CTS nur nachbildet. Hierbei wird des geteilte Medium mittels globalem Wissen realisiert. Sender und Empfänger reservieren das sie umgebende Medium zuverlässig, so dass kein benachbartes Gerät die Kommunikation stört. Kollisionen treten nur aufgrund der Mobilität auf. Bei Broadcastkommuni-

kation reserviert nur der Sender, so dass das Hidden-Terminal-Problem, und somit Nachrichtenkollisionen, auftreten können. Der Datendurchsatz ist idealisiert, so dass eine Übertragung im Senderadius immer mit voller Bandbreite erfolgt. Ähnlich den statistischen Netzen lassen sich auch Verfahren einsetzen, die die Bandbreite mit der Distanz reduzieren (z.B. basierend auf dem Two-Ray-Ground Modell).

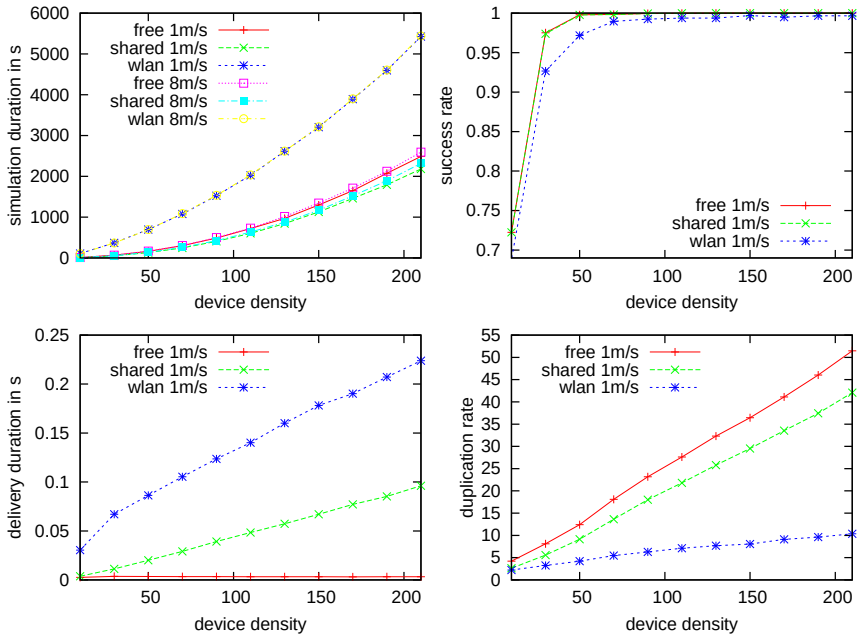
Den geringsten Berechnungsaufwand besitzen idealisierte Netzwerke wie das `CollisionFreeNetwork`. In diesem Netzwerk ist nur die Sendereichweite der Geräte beschränkt, innerhalb dieser die Geräte kollisions- und verlustfrei kommunizieren können. Somit treten Paketverluste nur aufgrund von Mobilität auf. Dennoch kann dieses Netzwerk sinnvoll verwendet werden, insbesondere wenn primär oder ausschließlich der Netzwerkgraph oder das Mobilitätsmodell im Vordergrund steht. Wird beispielsweise im Fall geographischer Unicastverfahren nur die Mindestzahl der Kommunikationsereignisse zwischen Sender und Empfänger bestimmt, ist die exakte Simulation dieser Kommunikationsereignisse überflüssig. Auch bei der Untersuchung des Verhaltens von Protokollen aufgrund von Änderungen der vorhandenen Kommunikationsverbindungen ist die detaillierte Simulation der einzelnen Übertragung häufig nebensächlich. Dies ist insbesondere dann der Fall, wenn die übertragene Datenmenge zwischen Verbindungsaufbau und -abbau ab einem gewissen Schwellenwert (i.d.R. der Protokollaufwand) irrelevant ist. Diese idealisierten Modelle lassen sich auch mit randomisierten Paketverlusten kombinieren, die in Abhängigkeit der Distanz zum Sender Pakete verwerfen.

7.3 Workbenchansatz

Zu dem in [66] vorgestellten Workbenchansatz gehört neben der Simulation auch ein Hybridmodus und ein Plattformmodus. Diese beiden Modi werden im Folgenden kurz skizziert.

Konzeptionell bedeutet der hybride Modus im verwendeten Simulationskern, dass ein externes Gerät mit einem simulierten Gerät verbunden wird und eine Einheit bilden. Applikations- und Middlewarekomponenten, die sich auf dem simulierten Gerät und auf dem dazugehörigen externen Gerät befinden, erkennen im Idealfall diesen Unterschied nicht. Um diese Transparenz zu gewährleisten, wird eine Verbindung zwischen den beiden Ausführungsumgebungen hergestellt und die externen Komponenten in die simulierte Umgebung eingeblendet. Faktisch wird somit auch ein Großteil der Aktivität auf dem Gerät über den Simulationskern abgehandelt. Um aber eine Trennung zwischen Simulationskern und externen Geräten zu erhalten, insbesondere um eine Entkopplung der Ereignisbehandlung zu erhalten, besitzen externe Geräte eine eigene Ereignisliste, die durch einen eigenen Thread abgearbeitet wird. Die Einbindung externer Geräte bedeutet aber auch einen Verlust der exakten Reproduzierbarkeit der Simulationsläufe. Diese Einbindung ist

7.1 Netzwerkmodelle



Um die verschiedenen Netzwerkimplementierungen zu vergleichen, wurden auf einer Fläche von $200m$ x $200m$ 10 - 210 Geräte mit einem Senderadius von $100m$ mit einer Geschwindigkeit von $1m/s$ - $10m/s$ bewegt. Das Netzwerk wird jede Sekunde über 40 Simulationsminuten mit einem 1 KByte großen Datenpaket geflutet. Die Datenrate beträgt in den Netzvarianten 802.11MAC (`wlan`), CollisionFreeNetwork (`free`) und SharedNetwork (`shared`) je $1MBit/s$. Gemessen wurde neben der Dauer der Simulation die Auslieferungsrate, die Maximaldauer des Flutens und die Zahl der zuviel empfangenen Duplikate je Gerät.

Die Simulationszeit von `wlan` ändert sich mit steigender Geschwindigkeit nicht, da sie nur von der absoluten Gerätezahl abhängig ist. Die Simulationszeit von `free` und `shared` nur marginal, da die Zahl der Verbindungsereignisse dabei nur gering steigt. Für alle Varianten steigt die Ausführungsdauer bei steigender Knotenzahl quadratisch, da in allen Implementierungen für jedes Gerät wiederholt die Distanz zu allen anderen bestimmt werden muss. Interessant ist, dass die Simulationszeit von `shared` etwas geringer ist als von `free`, obwohl die Implementierung aufwendiger ist. Dies liegt vermutlich in der Reduzierung und der gleichmäßigeren Verteilung der Nachrichtenereignisse begründet. In diesem Szenario erfolgt bei `free` die Nachrichtenverbreitung unabhängig von der Netzdichte in nur 0.0032 Simulationssekunden. Die Erfolgsrate von `free` und `shared` ist nahezu identisch und höher als von `wlan`. `wlan` kann aufgrund des verwendeten Signalausbreitungsmodells nicht immer alle Geräte erreichen, da mit der Distanz die Auslieferungswahrscheinlichkeit aufgrund von anderen Sendeereignissen sinkt.

– Fortsetzung Box 7.1 –

`shared` ist, wie in den Abbildungen zu sehen, im Vergleich zu `wlan` zu optimistisch. `shared` addiert zur Zeit noch keinen Overhead der MAC-Schicht zu der Nachrichtenlängensimulation hinzu. Darüber hinaus können bei `shared` auch noch distanzabhängige zufällige Nachrichtenverluste hinzugenommen werden. Hier kann also noch eine bessere Anpassung erfolgen, so dass `shared` und `wlan` ähnlichere Ergebnisse liefern.

aber zur Integration realen Benutzerverhaltens gedacht, welches sowieso nur schwer exakt zu reproduzieren ist. Technisch wird die Einbindung mittels Java RMI realisiert. Hierbei werden nur Middlewareaufrufe synchron übertragen. Die Ereignisse der Simulation und der externen Geräte, die die Abarbeitung von von Ereignisbehandlungsroutinen zur Folge haben, werden asynchron zwischen Simulation und externen Geräten übertragen. Somit ist eine Entkopplung möglich, die eine atomare Behandlung von Ereignisroutinen erlaubt und eine Verklemmung der beteiligten Kontrollflüsse verhindert.

Die transparente Realisierung auf realen Geräten benötigt im einfachsten Fall nur eine Ereigniswarteschlange, die gleichzeitige Ereignisse nacheinander ausführt. Im Falle der Untersuchung einer einzelnen Applikation reicht der Singlethreadedansatz aus. Werden mehrere Anwendungen betrieben, kann der Ereigniskern auch multithreaded realisiert werden. Hier ist aber zu beachten, dass die Ereignisbehandlungsroutinen weiterhin atomar ausgeführt werden müssen. Synchrone Aufrufe zwischen verschiedenen Komponenten können Inkonsistenzen oder Verklemmungen verursachen, falls diese von verschiedenen Kontrollflüssen ausgeführt werden. Werden synchrone Aufrufe erlaubt, ist entweder die Ausführung in einem Kontrollfluss notwendig, oder es müssen zusätzliche Synchronisationsmaßnahmen getroffen werden, die somit Portierungsaufwand bedeuten. Im Idealfall ist eine Portierung von Komponenten aus der simulierten in eine hybride oder eine reale Umgebung ohne zusätzlichen Aufwand möglich. Dieses Ziel ist in dem dreistufigen Workbenchansatz nahezu erreicht. Aufgrund der Idealisierung der simulierten Umgebung sind allerdings beim Übergang in die reale Umgebung häufig dennoch Adaptionen von Parametern notwendig.

8

Middleware Prototyp

Der Middlewareprototyp wurde auf Basis der Workbench für mobile Anwendungen [66] realisiert und sukzessive erweitert [33, 46, 103, 42, 43]. Daraus wurden insbesondere der Simulationskern und die Simulation der Gerätemobilität übernommen. Der Middlewareprototyp steht aufgrund der Workbenchrealisierung in den drei Modi Simulation, hybride Umgebung und Ausführungsplattform zur Verfügung. Er wurde sukzessive erweitert und an die Anforderungen der untersuchten Applikationsprototypen angepasst. Abbildung 8.1 gibt einen vereinfachten Überblick der Middlewarearchitektur. Im Middlewarekern wurden nur die notwendigsten Funktionalitäten realisiert. Middlewaredienste und Applikationskomponenten basieren auf dem Kern. Hierzu gehören insbesondere auch Hardwareabstraktionen. So ist die drahtlose Netzwerkkommunikation (*Singlehop Communication*) und die Positionierung des Gerätes beispielsweise mittels eines GPS Empfängers (*Location Manager*) jeweils durch eine eigene Komponente abstrahiert. Kommunikation mit anderen Geräten erfolgt somit immer über eine Interaktion mit einer anderen Komponente. Komponenten in der Middleware sind MobileObjects[1]. Sind diese aktiv und können mit anderen Objekten interagieren, handelt es sich um ActiveObjects.

Im Folgenden werden die einzelnen Komponenten des Middlewareprototyps näher erläutert, beginnend mit dem Middlewarekern. Anschließend werden die wichtigsten Middlewaredienste skizziert. Hier steht das konfigurierbare Kommunikationsframework im Vordergrund, das auch die Kombination von Kommunikationsmechanismen erlaubt. Es lässt sich in drei Bereiche unterteilen: (1) singlehop Kommunikation, (2) direkte multihop Kommunikation und (3) Store-and-Forward und En-Passant Kommunikation. Zur Spurpropagierung und zum Objektbeaconing wurde ein erweiterbarer Neighbordis-

[1] MobileObjects sind aus Designsicht Komponenten und können aus mehreren Objekten (z.B. Java Objekten) bestehen. Dennoch werden diese Begriffe Komponente, (mobiles) Objekt und MobileObject im Folgenden synonym behandelt und auf die Unterschied nur bei Bedarf eingegangen.

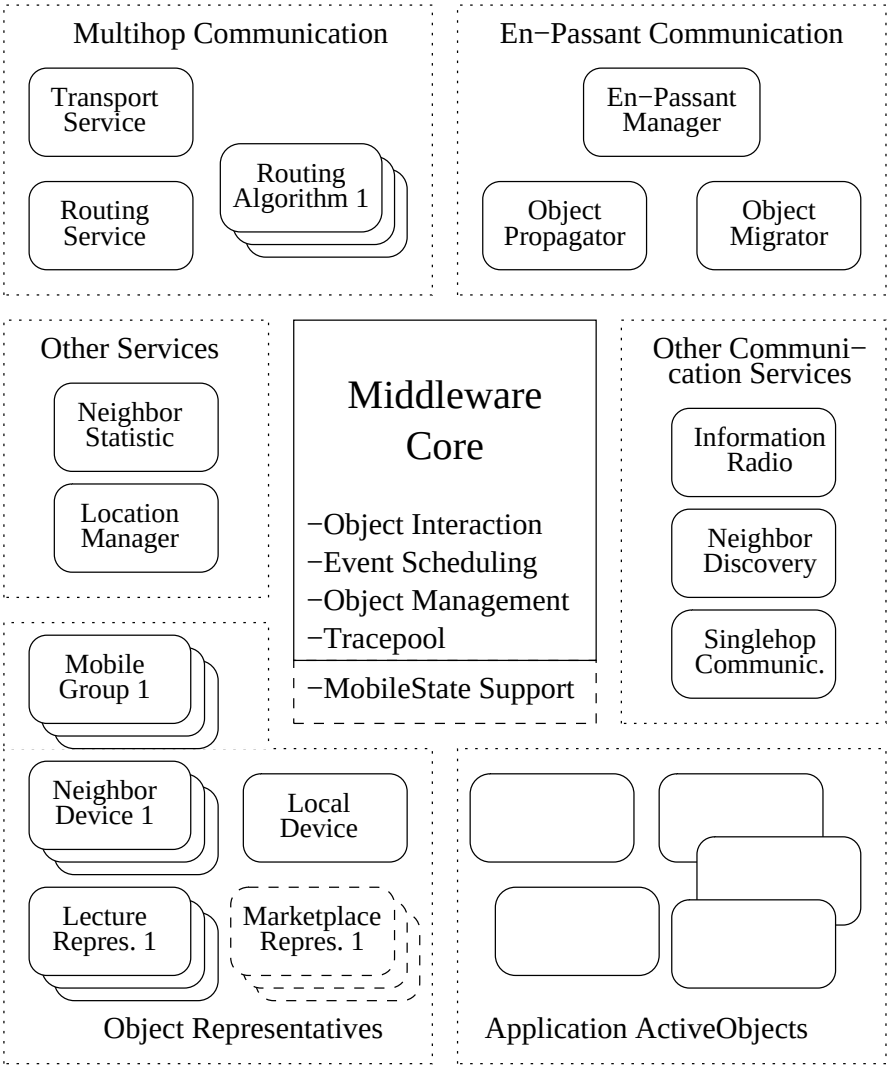


Abb. 8.1. Architekturübersicht der Middleware. Im Zentrum befindet sich der Kern der Middleware, auf dem die darum gruppierten Objekte basieren. Gestrichelte Komponenten sind noch nicht in die aktuell realisierte Version eingeflossen.

coverydienst realisiert, der das prioritätsbasierte Piggybacking von Zusatzinformation ermöglicht und ein Informationsradio integriert, das eine regional eingeschränkte Hintergrundverbreitung zur Verfügung stellt. Das lokale Gerät und seine Nachbargeräte werden in der Middleware durch eigene aktive Objektrepräsentanten dargestellt. Diese und weitere realisierte Repräsentanten

werden kurz skizziert. Die Unterstützung für MobileState Objekte ist nicht in den Prototypen integriert. Diese wurde noch nicht von dem Vorgänger der Middleware SELMA [46] portiert. Eine mögliche Integration wird abschließend diskutiert.

8.1 Middlewarekern

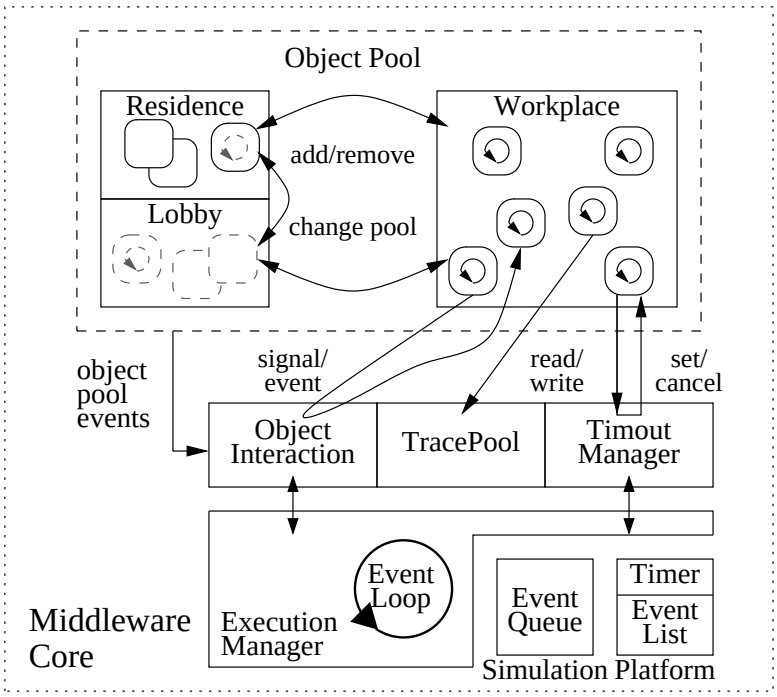


Abb. 8.2. Die Architektur des Middlewarekerns.

Die Aufgaben des Middlewarekerns (Abbildung 8.2) beschränken sich auf die Realisierung der lokalen Objektinteraktion, die Objektverwaltung und die Ereignisverwaltung. Jegliche weitere Funktionalität muss also auf diesem Minimalkern realisiert sein. Kommunikation mit anderen Geräten erfolgt somit auf Basis einer lokalen Interaktion mit lokalen Objekten. Jegliche Aktivität der Middleware ist ereignisbasiert und wird über den `ExecutionManager` verwaltet. In der simulierten Umgebung findet hier die Abbildung auf den Ereigniskalender der Simulation statt. In der Ausführungsplattform findet an dieser Stelle eine Abbildung auf Timer und eine zentrale Ereignisliste statt. In der Liste befinden sich alle Ereignisse, die sofort ausgeführt werden sollen. Die Abarbeitung der Ereignisse wird in beiden Fällen von einem einzelnen Thread

übernommen. In der Ausführungsplattform ließe sich aber eine Variante auf Grundlage eines Threadpools realisieren. Die aktuell abzuarbeitenden Ereignisse werden dabei an eine feste Menge Threads delegiert und können somit parallel ausgeführt werden. Hierbei muss gewährleistet sein, das immer nur ein Ereignis pro Komponente ausgeführt wird, sich also nicht zwei Kontrollflüsse in einer Komponete befinden.

Die Middleware unterscheidet zwischen aktiven, passiven und passiven-temporären MobileObjects, die jeweils in den Objektpools *Workplace*, *Residence* und *Lobby* verwaltet werden. Im Workplace befinden sich nur aktive Objekte (ActiveObjects), die Ereignisse auslösen und auf Ereignisse reagieren können. ActiveObjects sind rein ereignisbasiert realisiert und stellen Ereignisbehandlungsroutinen zur Verfügung, die von der Middleware durch Timeouts und durch andere Objekte ausgelöst werden können. Sie können andere Objekte zwischen den Pools verschieben, daraus löschen oder neu hinzufügen. Der direkte Zugriff ist aber nur auf passive Objekte in Lobby oder Residence erlaubt. ActiveObjects interagieren untereinander über eine ereignisbasierte Objektinteraktion. Passive Objekte können nicht mit anderen Objekten interagieren und sind unter Umständen auch nicht aktivierbar. MobileObjects in der Lobby befinden sich nur temporär auf einem Gerät und besitzen immer eine Restlebensdauer, nach der sie aus dem Objektpool automatisch gelöscht werden. Jedem dieser Objekte lassen sich Zusatzinformationen mittels des `TracePool`s zuordnen. Diese Informationen sind mit dem Objekt verknüpft und werden automatisch gelöscht, sobald die zugehörigen Objekte gelöscht werden.

8.1.1 ActiveObjects

ActiveObjects sind mobile Objekte, die Ereignisbehandlungsroutinen zur Verfügung stellen, die von der Middleware oder anderen ActiveObjects ausgelöst werden können. Sie sind darüber hinaus in der Lage, selbst Ereignisse zu erzeugen. ActiveObjects stellen zumindest eine `start` und eine `finish` Methode zur Verfügung (siehe Listing 8.1[2]). Wenn ein ActiveObject im Workplace abgelegt wird, wird ein Ereignis erzeugt, welches die `start` Methode aufruft. Die übergebene Middlewarereferenz ermöglicht es dem ActiveObject, weitere Ereignisse zu erzeugen. Dies kann eine Interaktion mit anderen ActiveObjects sein, das Durchführen von Operationen auf den Objektpools oder dem `TracePool` oder das Setzen von Timeouts, um zukünftige Ereignisse zu erzeugen.

Ein ActiveObject kann aus mehreren Objekten bestehen. Das Hauptobjekt implementiert das Interface `ActiveMobileObject`. Zusätzlich zu den dort definierten Methoden können weitere Methoden durch das Exportieren von Interfaces zum Ereignisempfang registriert werden. Zum Ereignisempfang

[2] Alle Programmlisten sind an die Programmiersprache Java angelehnt.

können auch weitere Objekte im Middlewarekern registriert werden. Die dort aufgerufenen Methoden werden im Kontext des Hauptobjektes ausgeführt. Kontext bedeutet hier, dass die dabei ausgelösten Ereignisse der Kennung des Hauptobjektes zugeordnet werden. Um eventuelle Kollisionen bei gleicher Methodensignatur zu vermeiden, werden zusätzlich exportierte Objekte eindeutig gekennzeichnet (siehe auch Abschnitt 8.1.4).

Listing 8.1 Ein MobileObject beinhaltet mindestens eine Objektkennung, die zumindest gerätelokal eindeutig sein muss. Zusätzlich besitzen ActiveObjects Ereignisbehandlungsroutinen, die beim Starten und beim Beenden des Objektes ausgeführt werden.

```
1 interface MobileObject{
2     public ObjectID getObjectID();
3 }

5 interface ActiveMobileObject extends MobileObject{
6     public void start (Middleware middleware);
7     public void finish();
8 }
```

Alle auszuführenden Ereignisse auf einem Gerät werden vom `ExecutionManager` der Middleware verwaltet. Dieser kann ein beliebiges Ereignisscheduling für gleichzeitige Ereignisse verwenden. Eingesetzt wird FCFS[3], wobei dies in der simulierten Umgebung auf den Ereigniskalender der Simulation abgebildet wird. Die Ausführungsplattform verwendet einen singlethreaded `ExecutionManager`. Der Einsatz mehrerer Threads an dieser Stelle ist möglich, ist aber aufgrund der Einzelapplikationsprototypen, die bisher untersucht wurden, nicht notwendig. Wird Multithreading eingesetzt, ist der Einsatz des synchronen Zugriffs auf andere MobileObjects (siehe Abschnitt 8.1.4) problematisch, da es ohne Vorkehrungen zu Inkonsistenzen kommen kann. Somit muss darauf entweder verzichtet werden oder es müssen bei Bedarf zusätzliche Synchronisationsmaßnahmen getroffen werden, die so in der Simulation nicht notwendig sind. Jedes ausgeführte Ereignis ist mit der Kennung des auslösenden Objekts markiert. Somit kann in einer Ereignisbehandlungsroutine immer die Kennung des Ereignisauslösers abgefragt werden. Es kann also festgestellt werden, welches Objekt das Aktivieren eines Objektes ausgelöst hat oder an welches Objekt eine etwaige Antwort erfolgen soll.

[3] "First come first served", d.h. es wird eine Warteschlange eingesetzt. Neue Ereignisse werden am Ende angehangen und das nächste, dass verarbeitet werden soll, von vorne entfernt.

8.1.2 Lokale Objektverwaltung

Die MobileObjects werden in drei Objektpools verwaltet: Lobby, Residence und Workplace (Abbildung 8.2). Ein Objekt ist immer genau einem dieser drei Pools zugeordnet. Je nach Pool besitzt ein Objekt einen unterschiedlichen Status. Objekte im Pool Lobby sind mit einer Lebensdauer assoziiert, so dass sie gelöscht werden können, wenn die Zeit abgelaufen ist. Der Name Lobby entstammt dem Haupteinsatzfeld dieses Pools. Empfangene Objekte, die nicht sofort weiterbearbeitet werden, können dort abgelegt werden und auf eine Weitervermittlung "warten". Die Wartestellung darf nicht beliebig lange andauern, da sich die Lobby sonst mit Objekten füllt, die nicht weiterverarbeitet wurden. Weitervermittlung bedeutet, dass ein Objekt in einen der anderen Pools verschoben, von ActiveObjects empfangen oder an ein benachbartes Gerät weitergeleitet wird. Der letzte Anwendungsfall wird von Kommunikationsverfahren verwendet, die auf Basis von Store-and-Forward arbeiten. Sobald eine sofortige Weiterleitung nicht mehr möglich ist, wird das Nachrichtenobjekt in der Lobby hinterlegt, bis entweder die maximale Aufenthaltsdauer verstrichen ist oder ein adäquates Nachbargerät gefunden wurde.

8.1 Einsatzbeispiele Lobby

Caching von Broadcastdatenobjekten:
Empfangene Datenobjekte können für Geräte von Interesse sein, ohne dass eine entsprechende Anwendung aktiv oder installiert ist. Wird beispielsweise im Rahmen einer Veranstaltung Veranstaltungsmaterial innerhalb des Veranstaltungsortes verteilt, wird dieses bei Ausnutzung der Broadcasteigenschaft des Mediums auch von Geräten empfangen, die unter Umständen erst später die entsprechende Anwendung aktivieren. Werden die empfangenen Datenobjekte für eine bestimmte Zeit, z.B. für die Dauer der Veranstaltung, in der Lobby abgelegt, müssen die Informationen nicht über das Netzwerk angefordert werden, wenn die Anwendung aktiviert wird.

Altruismus bei En-Passant Kommunikation:
Datenobjekte, die im altruistischen Modus der En-Passant Kommunikation [44] vorgehalten werden, können für eine beschränkte Zeit in der Lobby abgelegt werden. Eine Weiterverteilung der Datenobjekte kann dann zu einer Verlängerung der Lebensdauer führen, und somit bleiben häufig geforderte Objekte länger erhalten. Werden keine altruistisch gespeicherten Objekte angefragt, verschwinden diese mit der Zeit automatisch.

Im Residence Pool befinden sich alle passiven Objekte, die permanent auf dem Gerät verbleiben und für alle anderen Objekte zugreifbar sein sollen. Objekte in diesem Pool müssen also explizit gelöscht werden, wenn sie nicht mehr benötigt werden.

Innerhalb des Workplace befinden sich alle momentan aktiven Objekte. Damit ein Objekt aktiv werden kann, muss es neu erzeugt im Workplace abgelegt oder von einem der anderen Pools in den Workplace verschoben werden. Dies ist nur für Objekte möglich, die das Interface `ActiveMobileObject` implementieren und somit die Ereignisbehandlungsroutinen für den Start und das Beenden der Aktivität realisieren.

Die Objekte werden in den Pools über ihre Objektkennungen abgelegt, die zumindest lokal auf dem Gerät eindeutig sein müssen. Kommunizierbare Objekte müssen global eindeutig gekennzeichnet sein. Wird keine Kennung vom Objekt vorgegeben, wird diese global eindeutige Kennzeichnung beim Erzeugen durch eine lokal eindeutige Kennzeichnung und die Hinzunahme der Gerätekennung erreicht. Hierbei wird die Existenz einer global eindeutigen Gerätekennzeichnung vorausgesetzt[4]. Wird vom Objekt eine eigene Kennzeichnung vorgeben, wird nur die lokale Eindeutigkeit überprüft, die globale Eindeutigkeit liegt in der Verantwortung des Objekterzeugers. Neben der eindeutigen Identifizierung kann die Objektkennzeichnung auch der Objektbeschreibung dienen, so dass ohne zusätzliche Abfragen Rückschlüsse auf das identifizierte Objekt möglich sind. Die jetzige Realisierung verwendet die Klassenhierarchie des Objektes, um eine einfache Klassifizierung zu ermöglichen. Werden solche Objekte kommuniziert, ist dieses Vorgehen in einer realen Umgebung nicht tragfähig, da vorausgesetzt wird, dass Klasseninformationen beim Empfänger vorhanden sind oder zusätzlichen Kommunikationsaufwand verursachen, um diese zu übertragen. Hierarchische Strukturen lassen sich allerdings leicht realisieren, falls sie benötigt werden - aufwändig ist hier die zusätzlich notwendige explizite Definition. Applikationsspezifische Zusatzinformation in einer Objektkennung kann durch selbst definierte Objektkennungen erfolgen. So lassen sich beispielsweise wichtige Objekteigenschaften schon anhand der Kennung feststellen.

Die Pools selbst sind nur logischer Natur, d.h. die Realisierung verwendet einen einzigen Objektpool, der die unterschiedlichen Pooleigenschaften abbildet. Die Zugriffsmethoden sind in einem Interface zusammengefasst, wie vereinfacht in Listing 8.2 dargestellt. Auf Objekte kann durch Angabe der Objektkennung zugegriffen werden. ActiveObjects im Workplace sind allerdings nicht direkt zugreifbar. Hier liefert der `get()` Aufruf einen Kommunikationsproxy (siehe 8.1.4), über den mit dem eigentlichen Objekt interagiert werden kann. Auch für die Abfrage von Objekten in den Pools werden Klassenhierarchien verwendet. Somit lassen sich alle Objekte auflisten, die ein bestimmtes Interface oder eine bestimmte Klasse implementieren.

Für Lobby und Residence sind auch Zugriffsmethoden des *Query-By-Example* sinnvoll, aber noch nicht realisiert. Hierbei lassen sich Objekttemplates angeben, und alle Objekte werden zurückgegeben, die dem Template

[4] In der Praxis bietet die WLAN MAC-Adresse eine nahezu eindeutige Kennzeichnung. Werden weitere Geräteinformationen zur Erzeugung verwendet, ist die Wahrscheinlichkeit für eine Kollision gering.

in den gesetzten Attributen entsprechen[5]. Die Lobby benötigt im Falle der Methode `add()` zusätzlich ein Zeitdelta der Lebensdauerinformation, das so genannte *Lease*. Der Aufruf mit einem schon vorhandenen Objekt setzt die Restlebenszeit auf das übergebene Zeitdelta, falls dieses größer als die aktuelle Restlebenszeit ist. Nach Ablauf des Leases wird das Objekt automatisch gelöscht.

Listing 8.2 Ein Auszug der Methoden des Objektpoolinterfaces.

```
1  public interface ObjectPool {

3      public ObjectID addLobby(MobileObject mob, double lease);
4      public ObjectID addResidence(MobileObject mobileObject);
5      public ObjectID addWorkplace(ActiveObject activeObject);

7      public boolean moveToLobby(ObjectID objectID, double lease);
8      public boolean moveToResidence(ObjectID objectID);
9      public boolean moveToWorkplace(ObjectID objectID);

11     public Collection<ObjectID> getObjectIDs(Class implementedClass);

13     public MobileObject get(ObjectID objectID);
14     public MobileObject getFromWorkplace(ObjectID objectID,
15         Class implementedClass);
16     public MobileObject remove(ObjectID objectID);

18    (...)
19 }
```

Der Objektpool erzeugt Ereignisse, sobald sich der Zustand eines Objektes in einem der logischen Pools ändert. Mögliche Objektzustandsänderungen sind (1) erstes Hinzufügen eines Objektes zu einem der Pools d.h. das Objekt "betritt" das Gerät (`EnterDevice`), (2) komplettes Entfernen aus den Pools, d.h. das Objekt verlässt das Gerät (`LeaveDevice`), (3) Wechseln eines Objektes zwischen zwei Pools, bestehend aus Betreten eines der Pools und Verlassen des anderen (`EnterPool` und `LeavePool`), und (4) Verlängerung der Lebensdauer in der Lobby `ChangeLease`. `EnterPool` und `LeavePool` werden zusätzlich auch beim Betreten beziehungsweise beim Verlassen des Gerätes erzeugt. Neben der eindeutigen Objektkennung beinhalten die Ereignisse auch die Objektklasse, den Pooltyp und die Kennung sowie Klasse des ActiveObject, welches die Operation durchgeführt hat. Wird die Lobby betreten, beinhaltet das Ereignis zusätzlich die Leaseinformation.

[5] Zu Query-By-Example siehe auch Abschnitt 8.1.4. Query-By-Example wird auch in objektorientierten Datenbanken eingesetzt. Eine leichtgewichtige, auf Java basierende und für mobile Geräte geeignete Objektdatenbank ist z.B. DB4O [19]. JavaSpaces [75] ermöglicht ebenfalls diese Zugriffsmechanismen und insbesondere auch die Lebensdauereigenschaft der Lobby. Der Query-By-Example Zugriff liefert nur den ersten Treffer. Beide Lösungen sind für eine Realisierung innerhalb der Simulationsumgebung zu schwergewichtig, da hier hunderte von Geräten möglichst in Echtzeit simuliert werden sollen.

8.1.3 Lokale Spurverwaltung

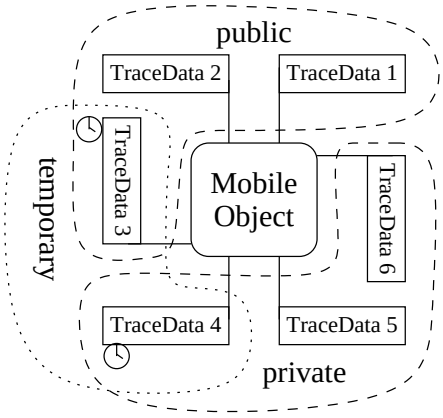


Abb. 8.3. MobileObjects können von ActiveObjects mit Spurinformationen "beklebt" werden. Die Informationen sind nicht Teil des Objektes selbst, sondern werden im `TracePool` verwaltet. Mögliche Eigenschaften dieser Informationen sind entweder privat oder öffentlich und entweder temporär oder permanent.

Objekte in den Pools können über einen zentralen `TracePool` mit Spurinformationen versehen werden, die von anderen Objekten über den `TracePool` zugreifbar sind. Somit können ActiveObjects ihre Spurinformationen veröffentlichen und einen direkten Zugriff auf das ActiveObject selbst überflüssig machen. Passive MobileObjects können von ActiveObjects mit Spurinformationen versehen werden. Dadurch können beispielsweise Objektnachbarschaften durch Hinzufügen einer Objektkennung ausgedrückt werden. Das Hinzufügen einer Gruppenobjektkennung kann somit Objekte dieser Gruppe zuordnen. Neben der Zuordnung von Objektkennungen können auch umfangreichere Spurdatenobjekte als Schlüssel/Wert-Paar zugeordnet werden. Der zugeordnete Schlüssel muss zumindest für das MobileObject eindeutig sein.

Die zentrale Ablage von Spurinformationen in einem Pool ermöglicht es, nach Objektmengen zu suchen, denen bestimmte Spurinformationen zugeordnet sind. So kann nach allen Objekten gesucht werden, die einer bestimmten Gruppe zugeordnet sind. Zu diesem Zweck bietet der `TracePool` einige Suchmethoden an, welche aber in ihrer Mächtigkeit beschränkt sind. Wie im Falle des Objektpools kann auch hier eine Query-By-Example-Abfrage den Suchkomfort erhöhen und den Suchaufwand reduzieren. Zusätzlich lässt sich die Suche auf Objekte eines bestimmten Typs beschränken, so dass beispielsweise nur MobileDevice Objekte gesucht werden. Diese Typbeschränkung basiert

Listing 8.3 Auszug aus dem Interface des `TracePool`s.

```
1  public interface TracePool{
2      public void addTraceInformation(ObjectID objectID, TraceData info,
3          boolean isPublic, double lease);
4      public void addNeighborInformation(ObjectID objectID,
5          ObjectID neighborObject, boolean isPublic, double lease);

7      public void removeTraceInformation(ObjectID objectID,
8          DataID infoID);


11     public Collection<TraceData> getAllTraceInfo(ObjectID objectID,
12         boolean isPublic);
13     (...)
14     public Collection<ObjectID> findObjects(DataID dataID);
15     public Collection<ObjectID> findObjects(Class tracedataClass);
16     public Collection<ObjectID> findObjects(Class objectClass,
17         DataID dataID);
18     (...)

20 }
```

auf der Java Klassenhierarchie. Ein Auszug aus dem Interface des `TracePool`s ist in Listing 8.3 dargestellt.

Informationen im `TracePool` können die Eigenschaften temporär/permanent und privat/öffentlich besitzen. Sind sie temporär, ist ihnen eine verlängerbare Gültigkeit (Lease) zugeordnet. Nach Ablauf des Leases wird die Information vom Objekt entfernt. Permanente Informationen besitzen keine eigene Gültigkeitsdauer. Sie müssen entweder explizit gelöscht werden oder werden mit Löschen des zugehörigen Objektes automatisch gelöscht. Die Interpretation von öffentlich und privat ist unabhängig vom Spurinformationspool. In der Regel bedeutet öffentlich, dass die Information an andere Geräte verbreitet wird oder dass sie beim Versenden des MobileObjects automatisch verpackt und mitgenommen wird. Privat bedeutet in diesem Fall, dass diese Information nicht mitgenommen wird. Falls das MobileObject nicht dupliziert sondern auf das Nachbargerät migriert wird, wird die private Information gelöscht.

Das Hinzufügen, Entfernen und Verändern von Spurinformationen erzeugt Ereignisse (`NewTraceInformation`, `RemoveTraceInformation` und `ChangeTraceInformation`), die von ActiveObjects empfangen werden können. Die Ereignisse beinhalten die Objektkennung und Klasse des Objektes, zu der die Information gehört, und den Schlüssel. Besteht die Information nicht nur aus dem Schlüssel, ist auch das zugeordnete Spurinformationsobjekt, welches umfangreichere Daten beinhaltet, und dessen Klasse enthalten.

8.2 `TracePool` und MobileDevice Objekte

Ein Beispiel für die Verwendung des `TracePool`s sind benachbarte Geräteobjekte, deren lokale Repräsentanten die Informationen im `TracePool` ablegen. In diesem Fall wird die in der Beaconnachricht empfangene Positionsinformation an den lokalen Repräsentanten geklebt, und so kann ein ActiveObject sich auf den Empfang von Änderungsereignisse bzgl. der Positionsinformation registrieren oder im Pool nach allen Objekten einer Klasse mit diesen Informationen suchen.

```
1  middleware.registerEventListener(
2      new TracePoolEvent(MobileDevice.class, PositioningData.
           DATA_ID, ...),
3      new TracePoolEventHandler(){
4      void newTraceInformation(TraceInformation information
           , ...){
5          PositioningData position=(PositioningData)information
               ;
6          //Nachbar mit Positionsinformationen gefunden
7      }
8      void changeTraceInformation(TraceInformation information
           , ...){
9          //Nachbar hat neue Positionsinformationen
10     }
11     void removeTraceInformation(){
12         //Nachbargerät hat keine Positionsinformationen mehr
13         //oder ist nicht mehr benachbart
14     }
15 });

17 //Finde alle Geräte mit Positionsinformationen
18 Collection<ObjectID> devicesWidthPosition=
19     middleware.objectTracePool.findObjects(
20     MobileDevice.class,
21     PositioningData.DATA_ID);
```

8.1.4 Lokale Objektinteraktion

ActiveObjects können auf Basis von drei verschiedenen Kommunikationsdirektiven miteinander kommunizieren: Signale, Events und synchroner Zugriff. Während die ersten beiden strikt asynchron arbeiten, erlaubt die dritte Variante den direkten Zugriff auf andere Komponenten. Der direkte Zugriff ist primär aus "Komfortgründen" erlaubt. Er soll die Zustandsabfragen an anderer Komponenten vereinfachen und auch nur dafür eingesetzt werden und keine zustandsverändernden Operationen ermöglichen. Die primäre Interaktion erfolgt mit asynchroner Kommunikation. Begründet liegt dies zum einen in der Tatsache, dass die Komponenten ereignisbasiert auf ihre Umgebung reagieren müssen, da diese in mobilen ad-hoc Netzwerken stark dynamisch ist. Zum anderen führen zustandsverändernde Aufrufe zu Problemen, wie in Abbildung 8.4 dargestellt. So kann es passieren, dass Komponenten während der Abarbeitung eigentlich atomarer Ereignisbehandlungsroutinen

von anderen Komponenten aufgerufen werden, und es dadurch zu inkonsistenten Zuständen der Komponenten kommen kann. Im Folgenden werden die drei Kommunikationsdirektiven vorgestellt.

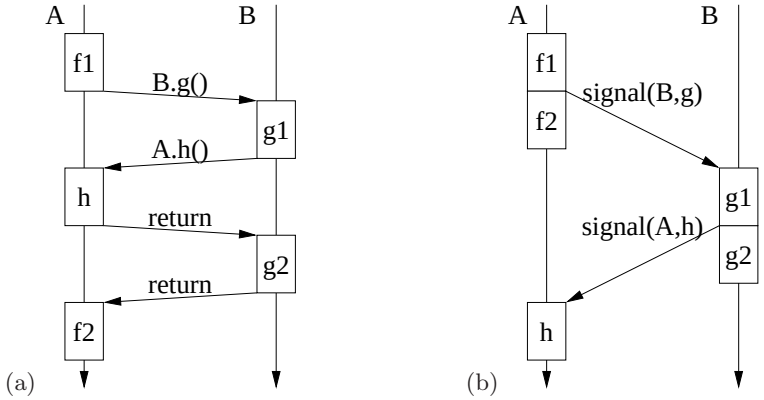


Abb. 8.4. (a) Wird eine Ereignisroutine einer anderen Komponente direkt aufgerufen, kann es passieren, dass diese sich bereits in der Abarbeitung einer anderen Routine befindet. (b) Wird der Zugriff auf andere Dienste asynchron durchgeführt, kann eine Ereignisbehandlungsroutine vollständig abgearbeitet werden, bevor der Zugriff auf eine andere Komponente erfolgt.

Eventkommunikation

Alle ActiveObjects im Workplace sind in der Lage Ereignisse zu erzeugen, um in anderen Objekten Ereignisbehandlungsroutinen auszulösen. Die Abarbeitung der Ereignisse erfolgt nach dem Beenden der aktuellen Ereignisbehandlungsroutine des Senders. Ausgeliefert werden die Ereignisse an registrierte ActiveObjects. Die Registrierung zum Empfang bestimmter Ereignisse erfolgt mittels *Event-By-Example*. Zu diesem Zweck wird eine Objektinstanz des Ereignisses als Vorlage angegeben und gesendete Ereignisse mit den im Templatemanager registrierten Vorlagen verglichen (Abbildung 8.5). Dabei wird auch die Klassenhierarchie der Ereignisse betrachtet, so dass durch Vorgabe einer Superklasse als Vorlage alle abgeleiteten Ereignisse beim Vergleich berücksichtigt werden. Die Attribute der Objektvorlage werden mit den Attributen des Senders verglichen. Stimmen die Attributwerte überein oder enthält die Vorlage Nullwerte, wird das Ereignis an den registrierten Empfänger ausgeliefert. Ein Sonderfall sind Attribute vom Typ `Class`[6]. Bei diesem wird überprüft, ob die Klasse, die das Vorlageattribut beschreibt, eine Superklasse der im gesendeten Ereignisattribut beschriebenen Klasse ist.

[6] Java Klassenobjekt. Dieses Objekt repräsentiert eine Javaklasse.

Listing 8.4 Die Superklasse aller Objektereignisse. Alle Ereignisse beinhalten die Kennung und Klasse des Ereigniserzeugers, die von der Middleware gesetzt werden. Der Objektkonstruktor wird für die Erzeugung von Objektvorlagen verwendet. Durch Überschreiben der Methoden `handle` und `getEventReceiverClass` können eigene Ereignisbehandlungsmethoden angegeben werden. Die Methoden der Superklasse bleiben gültig und werden in Abhängigkeit des registrierten Objektes ausgeführt. Die `copy()` Methode wird vor der Auslieferung ausgeführt und sollte bei veränderlichen Objekten implementiert sein.

```
1  public interface ObjectEventListener{
2      public void handle(ObjectEvent event);
3  }

5  public class ObjectEvent{
6      private ObjectID senderID;
7      private Class senderClass;
8      // Templatekonstruktor
9      public ObjectEvent(ObjektID senderID, Class senderClass){
10         this.senderID=senderID;
11         this.senderClass=senderClass;
12     }
13     // Ereigniskonstruktor
14     public ObjectEvent(){}
15     //Ereignisobjekt ist unveränderbar
16     public Event copy(){
17         return this;
18     }
19     // Aufruf der Standardereignisbehandlungsroutine
20     public void handle(EventListener listener){
21         ((ObjectEventListener)listener).handle(this);
22     }
23     //Der Empfänger muss ObjectEventListener implementieren,
24     //wenn die handle Methode aufgerufen werden soll.
25     public Class getEventReceiverClass(){
26         return ObjectEventListener.class;
27     }
28 }

30 public class MyEvent extends ObjectEvent{
31     private String val;
32     // Ereigniskonstruktor
33     public MyEvent(String val){
34         this.val=val;
35     }
36     // Aufruf der wigenen Ereignisbehandlungsroutine
37     public void handle(EventListener listener){
38         ((MyEventListener)listener).handleMyEvent(senderID, val);
39     }
40     //Der Empfänger muss MyEventListener implementieren, damit obige
41     //handle Methode aufgerufen werden kann
42     public Class getEventReceiverClass(){
43         return MyEventListener.class;
44     }
45 }
```

So kann der Ereignisempfang beispielsweise auf eine bestimmte Senderklasse eingeschränkt werden. Alle Ereignisse beinhalten die Kennung und die Klasse des Ereigniserzeugers. Diese Informationen werden automatisch von der

Middleware gesetzt. Somit ist eine Einschränkung auf diese Attribute immer möglich.

8.3 Event-By-Example

Sender / *Empfänger*	*X(1,A)*	*Y(1,A)*	*Z(1,A)*	*X(2,A)*	*X(1,B)*	*X(1,C)*
X(1,-)	*X*	*X*	*X*		*X*	*X*
Y(1,-)		*X*				
Z(1,-)			*X*			
X(-,-)	*X*	*X*	*X*	*X*	*X*	*X*
X(2,-)				*X*		
X(-,A)	*X*	*X*	*X*	*X*	*X*	*X*
X(-,B)					*X*	
X(-,C)						*X*

Beispiele für Event-By-Example. $E(l, m)$ bezeichnet den Ereigniskonstruktor, wobei E die Ereignisklasse angibt. Y und Z sind Subklassen von X. Somit können auch Ereignisse vom Typ Y oder Z empfangen werden, wenn als Registrierung eine Instanz von X angegeben wurde. l ist ein Integerattribut, m ein Klassenattribut. A ist Superklasse von B und C. Daher passen auch B und C, wenn A angegebenen wird. $-$ bedeutet ein nicht initialisierter Wert und passt immer. In der Tabelle sind einige Beispielkombinationen angegeben. Die Zeilen geben Empfängerregistrierung an, Spalten die gesendeten Ereignisse. Ein X steht an den Stellen, an denen ein Ereignis empfangen wird.

Ereignisempfänger können eine Standardereignisbehandlungsroutine registrieren, die für alle Ereignisse anwendbar ist. Hierbei wird das Ereignis bei der Auslieferung übergeben und der Empfänger muss den tatsächlichen Ereignistyp kennen, um auf typspezifische Methoden zugreifen zu können. Somit können Ereignisse empfangen werden, wenn nur die Ereignissuperklasse bekannt ist. Ereignisse können aber auch Ereignisbehandlungsroutinen vorgeben, die beim Empfang aufgerufen werden. Hierfür gibt das Ereignis das Interface an, welches der Empfänger implementieren soll. Das Ereignis kann nun eine Methode des Interfaces aufrufen und die notwendigen Attribute übergeben. Dazu muss der Empfänger den Typ des Ereignisses nicht bei der Ereignisbehandlung kennen. Stattdessen werden jegliche Typkonversionen vom Ereignis selbst durchgeführt. Da für jedes Ereignis eine eigene Behandlungsroutine angegeben werden kann, kann ein Ereignisbehandlungsobjekt verschiedene Ereignisse empfangen, ohne selbst Fallunterscheidungen treffen zu müssen.

Da Ereignisse von mehreren Objekten empfangen werden und darüber hinaus auch Referenzen auf andere Objekte beinhalten können, sollten Ereignisse bzw. die darin enthaltenen Informationen unveränderlich sein. Das bedeutet,

dass weder der Sender nach der Ereigniserzeugung noch einer der Empfänger das Ereignis verändern darf, da dies sonst zu Seiteneffekten führen kann. In der Regel besitzen Ereignisse nur Methoden zum lesenden, aber nicht zum verändernden Zugriff. Ist dies nicht der Fall, muss das Ereignis die `copy()` Methode implementieren. Diese Methode wird vor der Ereignisauslieferung aufgerufen und die zurückgegebene Ereigniskopie ausgeliefert. So können die im Ereignis enthaltenen Informationen kopiert werden, indem neue Objektinstanzen erzeugt werden.

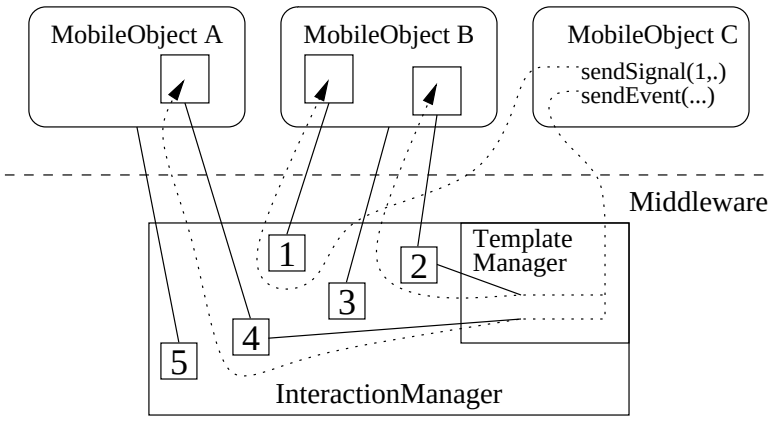


Abb. 8.5. Objekte, die Ereignisbehandlungsroutinen zur Verfügung stellen, müssen vom MobileObject in der Middleware registriert werden, um Ereignisse oder Signale empfangen zu können. Sie besitzen immer eine eindeutige Kennung, so dass sie adressiert werden können (Signal an Kennung 1 für Objekt in B). Der Templatemanager verwaltet Ereignistemplates und speichert die Kennung der registrierten Ereignisbehandlungsroutine. Ein Ereignis kann mehrere Handlerkennungen in unterschiedlichen MobileObjects adressieren (2 in B und 4 in A). Alle ActiveObjects sind automatisch zum Signalempfang über ihre Objektkennung registriert, wie für Objekt A und B dargestellt.

Signalkommunikation

ActiveObjects können Signale an andere ActiveObjects versenden. Im Gegensatz zur Ereigniskommunikation werden bei der Signalkommunikation andere Objekte direkt adressiert. Versendete Signale rufen vorgegebene Methoden des adressierten Objekts auf. Die Interfacevorgabe erfolgt hier genauso wie im Falle eines Objektereignisses (siehe Listing 8.5). ActiveObjects sind in der Lage, weitere Objekte zu registrieren, die Signale empfangen können. Die Signalempfangsobjekte (Interface `SignalListener`) werden eindeutig gekennzeichnet und können somit von anderen ActiveObjects

adressiert werden. Durch Vorgabe einer Kennung bei der Registrierung eines `SignalListener` können wohlbekannte Interaktionspunkte geschaffen werden, ähnlich eines Ports, nur dass hier die Interaktionsmöglichkeit durch das Objektinterface bestimmt wird.

Listing 8.5 Das `Signal` Interface. `getSignalReceiverClass()` liefert die Klasse der Objekte, die in `handle()` übergeben werden dürfen. Somit kann dort sicher die richtige Methode aufgerufen werden.

```
1 public interface Signal{
2     public void handle(SignalListener receiver);
3     public Signal copy();
4     public Class getSignalReceiverClass();
5 }
```

Wenn umfangreichere Interaktionen mit einem anderen Objekt stattfinden soll, ist die Realisierung mittels Signalen sehr aufwändig, da viele verschiedene Ereignisroutinen, die von anderen Objekten mit einzelnen Signalen angesprochen werden müssen, existieren. Aus diesem Grund gibt es einen Mechanismus, der aus einem vorgegeben Interface zur Laufzeit Signalproxies erstellt. Diese generieren beim Aufruf einer Interfacemethode automatisch das entsprechende Signal. ActiveObjects können durch Angabe der Signalempfängerkennung und eines vom Empfänger implementierten Interface einen Signalproxy erstellen lassen. Das erzeugte Objekt genügt dem erwarteten Interface, wenn auch der Signalempfänger diesem Interface genügt. Werden Methoden des erzeugten Objektes aufgerufen, wird ein Signal erzeugt und asynchron an das eigentliche Ziel ausgeliefert. Dort wird die entsprechende Methode des Ziels aufgerufen (siehe Abbildung 8.6).

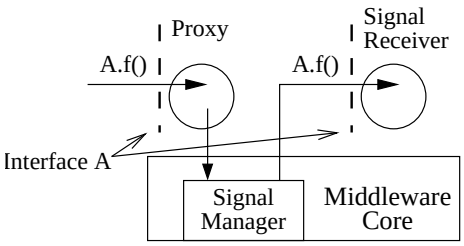


Abb. 8.6. Der Signalproxy entspricht demselben Interface wie das zu adressierende Objekt. Der Sender kann also durch Aufruf von Objektmethoden Signale erzeugen, die asynchron dieselbe Methode beim Empfänger aufrufen.

Signalproxies werden mittels des Java Reflection Mechanismus erstellt. Methodenaufrufe des Proxyobjektes werden alle an eine `invoke()` Methode

delegiert, die als Parameter ein Methodenbeschreibungsobjekt und die übergebenen Argumente erhält. Diese Methode erstellt ein Signal für den Methodenaufruf und übergibt es dem `InteractionManager` der Middleware, der die Auslieferung des Signals im Ereigniskalender einplant. Wird das generische Signal ausgeliefert, wird dem Signal eine Referenz auf das Signalempfängerobjekt übergeben, und kann dort mittels Reflection die richtige Methode mit den Methodenparametern aufrufen (siehe Listing 8.6). Da Java bei Methodenaufrufen Objektreferenzen übergibt, führen Änderungen der übergebenen Objekte durch Sender oder Empfänger zu ungewollten Seiteneffekten. Es sollten also nur Basistypen, unveränderliche oder neu erstellte Objekte als Argumente übergeben werden. Technisch würde hier ein automatisiertes Kopieren des Objektes helfen, ähnlich der Objektserialisierung durch Implementierung des `Serializeable` Interface. Da dabei immer ein automatisiertes Kopieren des Objektes durchgeführt wird, die auch alle im Objekt enthaltenen Attribute kopiert, ist der Aufwand hierfür sehr groß. Daher wurde das automatisierte Kopieren aus Performancegründen, insbesondere hinsichtlich der simulierten Umgebung, nicht implementiert.

Listing 8.6 Vereinfachter Programmkode des InvocationHandlers der Signalproxies und des generischen Proxysignals.

```
1  public class SignalProxyInvocationHandler implements InvocationHandler
2  {
3      (...)
4      public Object invoke(Object proxy, Method method, Object[] args)
5              throws Throwable{
6          (...)
7          interactionManager.sendSignal(
8              senderContext, receiverContext,listenerID,
9              new ProxySignal(signalListenerClass,method,args, ...)
10         );
11         return null; //Signale haben keinen Rückgabewert!
12         }
13 }

15 public class ProxySignal implements Signal{
16     private Method method;
17     private Object[] args;
18     private Class signalListenerClass;
19     (...)
20     public void handle(SignalListener listener){
21         (...)
22         method.invoke(listener, args);
23     }
24     public Class getSignalListenerClass(){
25         return signalListenerClass;
26     }
27 }
```

Da es sich um asynchrone Methodenaufrufe handelt, können nur Methoden verwendet werden, die keinen Rückgabewert besitzen. Daher wird auch in der Methode `invoke()` immer `null` zurückgegeben. Werden komplexere Interak-

tionen durchgeführt, ist die Realisierung selbst mit Signalproxies aufwändig. Ein Beispiel ist eine Auftragsdurchführung, bei der der Auftragsempfänger dem Sender bestimmte Auftragszustände signalisieren soll. Zu diesem Zweck können dem Signalproxy auch Signalempfängerobjekte übergeben werden. Der Signalproxy erstellt für Objekte, die das Interface `SignalListener` implementieren, automatisch einen Signalproxy der beim Signalempfang dem Empfänger übergeben wird. Der Auftragsempfänger kann nun seinerseits durch Methodenaufrufe Auftragszustände signalisieren. Diese Art der Signalproxies lässt sich auch delegieren. Somit kann ein anderes ActiveObject einen Auftrag weiterverarbeiten und die Auftragszustände signalisieren.

Listing 8.7 Ein einfaches Codebeispiel zur proxybasierten Signalkommunikation zwischen zwei ActiveObjects

```
1  interface SignalReceivingArgument extends SignalListener{
2      public void reply();
3  }

5  interface SignalReceivingObject extends SignalListener{
6      public void method1();
7      public void method2(SignalReceivingArgument arg);
8  }

10 class MobileObject1 implements ActiveMobileObject,...{
11     (...)
12     public void handleSomething(...){
13         (...)
14 //Für das übergebene Interface generiert die Middleware einen
       Signalproxy
15         otherObject=(SignalReceivingObject) middleware.
16             getSignalProxy(IDOfOtherObject, SignalReceivingObject.
                   class);
17 //Dieser Aufruf erzeugt ein Signal, dass erst nach der Abarbeitung
18 //der Ereignisbehandlungsroutine handleSomething ausgeliefert wird
19         otherObject.method1();

21         SignalReceivingArgument arg=new SignalReceivingArgument(){
22             public void reply(){
23                 //asynchrone Antwort des anderen Objekts
24             }
25         }
26 //Dieser Aufruf erzeugt ein Signal und registriert automatisch
27 //das übergebene Objekt zum Signalempfang.
28         otherObject.method2(arg);
29     }
30 }

32 class MobileObject2 implements SignalReceivingObject,...{
33     public method1(){
34         //Empfang eines Signals
35     }
36     public void method2(SignalReceivingArgument arg){
37         // Empfang eines Signals
38         // Das Argument arg ist ein automatisch erzeugter Proxy,
39         // dessen Methodenaufrufe Antwortsignale generiert
40         arg.reply();
41     }
42 }
```

Ein einfaches Beispiel für die Signalproxykommunikation ist in Listing 8.7 dargestellt. Hier interagieren zwei ActiveObjects, wobei `MobileObject1` die Interaktion beginnt. `MobileObject1` kennt die eindeutige Objektkennung von `MobileObject2` und das von ihm implementierte Interface `SignalReceivingObject`. Mit diesem Wissen kann es sich von der Middleware ein Signalproxy erzeugen lassen, welches diesem Interface entspricht. Der Aufruf der Methoden erzeugt ein Signal, welches frühestens nach Abarbeitung der aktuellen Ereignisroutine behandelt wird. Der Methode `method2()` wird ein Objekt zum Signalempfang übergeben, woraus wiederum ein Proxy erzeugt wird, so dass `MobileObject2` asynchron auf den Methodenaufruf antworten kann.

Synchrone Kommunikation

Synchrone Kommunikation ähnelt der Realisierung der Signalkommunikation, nur dass die Auslieferung sofort erfolgt und Rückgabewerte beinhalten kann. Die Middleware schränkt diesen Interaktionstyp ein, um Konflikte zu vermeiden. So dürfen in der Aufrufhierarchie keine Kreise entstehen, damit ein und dasselbe Objekt nicht ein zweites Mal besucht wird und somit eine nicht beendete Ereignisroutine vorfindet. Allgemein ist die synchrone Kommunikation primär zu komfortableren Zustandsabfrage anderer ActiveObjects gedacht. Zustandsverändernde Aufrufe sollten hiermit nicht durchgeführt werden. Die Abstraktion der Kommunikationskomplexität wird wie bei der Signalkommunikation durch die automatische Generierung von Kommunikationsproxies reduziert.

8.1.5 Workbenchintegration

Die Middleware wurde auf Basis der Workbench für Mobile Anwendungen (siehe Kapitel 7) realisiert. Im Simulationsmodus ist es daher möglich, auf Wissen zuzugreifen, das in der Realität nicht vorhanden ist. Somit lassen sich vereinfachte Dienste realisieren, die komplexe Aufgaben durch globales Wissen effizienter lösen können, um den Rechenaufwand zu reduzieren. Ein Beispiel hierfür ist ein Positionsverzeichnisdienst für geographisches Routing, dessen Kommunikationsaufwand in einer Routinguntersuchung die nur die Hopzahl bestimmt nicht berücksichtigt werden braucht, und somit erheblicher Simulationsaufwand eingespart wird. Es existieren spezielle ActiveObjects, die mit Simulationswissen arbeiten können. Simulationswissen bedeutet zum einen den Zugriff auf globales Wissen wie z.B. exakte Gerätepositionen oder exakte Simulationszeit, aber auch Simulationsparameter. Diese Objekte sind darüber hinaus in der Lage, mit ActiveObjects auf anderen Geräten mit den oben vorgestellten Mechanismen zu interagieren. Somit lassen sich auf Basis der Middleware auch Objekte realisieren, die die Kommunikation mit benachbarten Geräten simulieren. Ein Sonderfall dieser ActiveObjects sind

globale Objekte. Hier wird ein und dieselbe Instanz in die Workplacepools aller Geräte eingeblendet. Aus diesem Grund lassen sich simulationsspezifische Aufgaben einfacher global lösen. Ein Beispiel ist die Realisierung des Mediumzugriffs der IEEE802.11 Netzwerkimplementierung.

Eine weitere Simulationsbesonderheit betrifft alle aktiven und alle kommunizierbaren MobileObjects. Jedes dieser Objekt kann angeben, wie es visualisiert werden soll und sich somit auf einer graphischen Simulationsoberfläche darstellen. Hierbei kann ein Objekt geometrische Formen, Texte und Bilder angeben. Diese werden relativ zum Aufenthaltsort des Objektes gezeichnet, also relativ zum Gerät oder bei der Kommunikation des Objektes zwischen zwei Geräten. Diese können aber auch mit absoluten Koordinaten dargestellt werden. Somit lassen sich Applikations- und Protokollzustände visuell aufbereiten, so dass sich auch ohne das Lesen von Logdateien Eigenschaften aber auch Fehler beobachten lassen.

Auch der Hybridmodus wird unterstützt. Bei diesem lässt sich ein externes Gerät an die Simulation der Middleware anbinden und mit einem simulierten Gerät verbinden. Hierbei können ActiveObjects auf dem externen Gerät gestartet werden. Die Objekte werden in den Workplace des simulierten Gerätes eingeblendet. Dazu werden allerdings Objektproxies benötigt, die Ereignisroutinenraufrufe über RMI an das externe Objekt weiterleiten, ähnlich der Signalproxies. Middlewareaufrufe des externen Objektes werden an die Middleware des simulierten Gerätes weitergereicht. Somit hat das externe Objekt dieselbe Sicht als würde es sich auf dem simulierten Gerät befinden und die Objekte dort können keinen Unterschied zu lokalen Objekten sehen. Dadurch können z.B. GUI Komponenten auf einem realen Gerät getestet werden. Im Idealfall befinden sich nur noch auf Simulationswissen basierende Objekte in der Simulation, alle anderen Objekte sind auf das externen Gerät ausgelagert. Es ist aber auch ein gemischter Ansatz möglich, bei dem sich auch simulierte Geräte ohne externes Pendant in der Simulation befinden, eine Applikation also gleichzeitig mit rein simulierten und mit hybriden Geräten getestet werden kann. Die Anbindung externer Geräte kann zu einem beliebigen Zeitpunkt erfolgen. Ist also die Applikation auch ohne graphische Benutzeroberfläche im Netzwerk voll funktionstüchtig, kann diese GUI-Komponente auch erst später bei Bedarf gestartet werden.

8.2 Kommunikation

Jegliche Interaktion mit Objekten auf anderen Geräten sowie die Verbreitung von Datenobjekten erfolgt über ein ActiveObject, welches das drahtlose Medium repräsentiert[7]. Diese Objekt realisiert nur die Kommunikation mit

[7] Hierbei ist es möglich, mehrere Medien durch verschiedene ActiveObjects zu repräsentieren.

direkten Nachbarn, also singlehop. ActiveObjects, die aufwändigere Kommunikationsmechanismen realisieren, um auch entferntere Objekte als die direkt benachbarten zu adressieren, basieren auf der lokalen Interaktion mit diesem Netzwerkobjekt. Kommunikation mit entfernteren Objekten kann beispielsweise mit multihop Routingverfahren erfolgen, die die Nachrichten sofort ausliefern. Hierfür stellt die Middleware ein Routingframework zur Verfügung, das dem Sender die Wahl und Adaption des Routingverfahrens und darüber hinaus eine Kombination von Routingverfahren ermöglicht. In dünnen Netzen ist häufig eine Store-and-Forward Kommunikation notwendig, wobei Objekte weitergeleitet werden, sobald sich der Zustand der direkten Gerätenachbarschaft entsprechend ändert. Zu diesem Zweck stellt die Middleware Mechanismen zur Verfügung, die Objektmigrationen auslösen können. Dabei kann zusätzlich auf Routingalgorithmen des Routingframeworks zugegriffen werden - entweder um Routingentscheidungen treffen zu lassen oder um zwischen Store-and-Forward und direkter Kommunikation zu wechseln zu.

8.2.1 Singlehop Kommunikation

Kommunikation mit direkten Nachbargeräten erfolgt auf Basis des Data-Link-Layers der drahtlosen Kommunikation und wird für jedes vorhandene Netzwerkgerät in der Middleware als ActiveObject repräsentiert. Dieses Objekt (`LinkLayer`) unterstützt folgende singlehop Kommunikationsprimitiven:

- *Unicast*
- *Broadcast*
- *adressierter Broadcast*
- *adressierter Multicast*

Die beiden letztgenannten Kommunikationsprimitiven können durch Versenden der Nachricht per Broadcast mit zusätzlichen Unicastbestätigungen realisiert werden, falls die Sicherungsschicht sie nicht zur Verfügung stellt. Da hierbei keine Fragmentierung und Einzelbestätigung der Fragmente erfolgt, ist die Effizienz der Übertragung erheblich geringer als bei einer Realisierung in der Sicherungsschicht.

Den Primitiven kann zusätzlich ein Konfigurationsobjekt übergeben werden, um die Sicherungsschicht auf Nachrichtenebene anpassen zu können. Als Parameter stehen in den jetzigen Realisierungen[8] die maximale Zahl der Nachrichtenwiederholungen, das Zeitdelta für Nachrichtenwiederholungen und die Reduzierung der Sendestärke zur Verfügung.

Die Kommunikationsaufträge erfolgen asynchron. Daher kann der Auftraggeber ein Rückrufobjekt übergeben, welches das Netzwerk zur Signalisierung des Zustands des Kommunikationsauftrags verwendet. Alle Primitiven

[8] Vollständig realisiert in der Implementierung des simulierten IEEE802.11 MAC.

8.4 IEEE802.11 und der adressierte Broadcast/Multicast

Der adressierte Broadcast mit nur einem Empfänger kann sehr einfach im 802.11 MAC Protokoll realisiert werden. Dazu wird der MAC-Header eines Unicasts um ein Flag erweitert, das einen impliziten promiscuous Modus* bedeutet. Somit empfangen und bearbeiten alle Geräte im Sendebereich diese Nachricht, wenn dieses Flag gesetzt wird, auch wenn das Gerät nicht adressiert wurde. Der Unicastempfänger bestätigt die Nachricht wie im IEEE802.11 Protokoll vorgesehen. Darauf aufbauend lassen sich die adressierten Varianten des Broadcasts und Multicast aufbauen, indem weitere Adressen in einem angehangenen Header mitgesendet werden.
Eine Untersuchung der vollständigen Integrierbarkeit dieser Primitiven in das IEEE802.11 MAC-Protokoll und insbesondere die Anpassung der Distributed Coordination Function (DCF) steht noch aus. Eine vollständige Integrierung ermöglicht eine höhere Zuverlässigkeit und Effizienz dieser Primitiven.

* Befindet sich ein Netzwerkinterface im promiscuous Modus, werden auch Nachrichten verarbeitet, die nicht an das Interface adressiert sind. "Implizit" bedeutet dabei, dass nicht der Empfänger über die Weiterverarbeitung der Nachricht entscheidet, sondern der Sender durch das Setzen eines Flags im Header.

signalisieren den Abschluss des Auftrags und den lokal erkennbaren Fehlerfall. Soll exakt ein Empfänger die Nachricht bestätigen, wird zusätzlich der Erfolg oder das Ablaufen des Kommunikationstimeouts signalisiert. Wenn mehrere Empfänger bestätigen müssen, wird der Erfolg der einzelnen Empfänger signalisiert. Abschließend wird entweder der Erfolg aller Bestätigungen oder bei Ablauf des Timeouts der Gesamtzustand der Bestätigungen – Erfolg, Fehler oder Timeout – signalisiert.

8.2.2 Routingframework

Routingalgorithmen im Routingframework sind ActiveObjects, die für das Treffen von Routingentscheidungen zuständig sind. Der Algorithmus erhält nicht die weiterzuleitende Nachricht, sondern nur die für das Treffen von Routingentscheidungen notwendigen Informationen. Somit erhält ein Routingverfahren einen Routingauftrag mit einer für den Routingalgorithmus spezifischen Konfiguration und signalisiert dem Auftraggeber die darauf basierende Routingentscheidung. Die Signalisierung der Entscheidung kann mit erheblicher Verzögerung erfolgen, da dazu unter Umständen aufwändige Aktionen notwendig sind. Beispielsweise kann im Falle von linkbasierten Verfahren eine Routediscoveryphase das Treffen einer Routingentscheidung stark verzögern. Die möglichen Routingentscheidungen (Interface `RoutingTaskHandler`)sind:

- `forwardAsUnicast`:
  Leite die Nachricht an ein Nachbargerät weiter.
- `forwardAsAddressedMulticast`:
  Leite die Nachricht an mehrere Nachbargeräte weiter.
- `forwardAsBroadcast`:
  Leite die Nachricht an alle Nachbargeräte weiter.
- `delegate`:
  Delegiere den Auftrag an ein anderes Routingverfahren.
- `drop`:
  Verwerfe die Nachricht, z.B. aufgrund eines Routingfehlers.
- `ignore`:
  Ignoriere die Nachricht, z.B. aufgrund eines wiederholten Empfangs beim Fluten.
- `deliver`:
  Nachricht ist am Ziel.

8.5 Headerinformationen

- Kennung des Routingalgorithmus
- Nachrichtenkennzeichnung
- Senderkennung
- Empfängerkennung
- Senderposition
- Empfängerposition oder Region
- Aktueller und maximaler Hopcount
- Nachrichtenroute
- Delegationsdaten
- Flags: `isPromiscuousMessage`, `isPromiscuousHeader`
- Referenz auf den letzten Linklayerheader
- Konfigurationsdaten des Routingalgorithmus
- Routingalgorithmenspezifische Headerdaten

Der Routingalgorithmus kann jeder Entscheidung Informationen in einem selbstdefinierten Header mitgeben, die zum Treffen von Routingentscheidungen auf dem nächsten Gerät benötigt werden. Die Header erweitern einen Standardheader, der Basisinformationen und möglicherweise Zusatzinformationen beinhaltet. In jedem Header muss als Basisinformation eine eindeutige Nachrichtenkennung und die Objektkennung des zugehörigen Routingalgorithmus enthalten sein. Alle weiteren, in Box 8.5 angegebenen Daten sind

optional. Je nach Routingalgorithmus müssen oder können diese Informationen im Routingauftrag enthalten sein.

Listing 8.8 Handlermethoden eines Routingalgorithmus im Routingframework

```
1  public interface RoutingAlgorithm extends ActiveMobileObject{
2  //Intialer Routingauftrag
3      public void startRouting(RoutingTaskHandler handler,
4          RoutingHeader routingHeader);

6  //Fortsetzung eines Routingauftrags
7      public void continueRouting(RoutingTaskHandler handler,
8          RoutingHeader routingHeader);

10 //Der Routingauftrag wurde delegiert
11     public void delegateRouting(RoutingTaskHandler handler,
12         RoutingHeader routingHeader);

14 //Empfang des Headers einer Nachricht die nicht an dieses Gerät
15 // adressiert ist, aber das isPromiscuousHeader Flag gesetzt hat
16     public void handlePromiscousHeader(RoutingHeader routingHeader);

18 //Unicastforwarding ist fehlgeschlagen
19     public void unicastError(RoutingTaskHandler handler,
20         RoutingHeader header, Address failedReceiver);

22 //Multicastforwarding ist fehlgeschlagen
23     public void addressedMulticastError(RoutingTaskHandler handler,
24         RoutingHeader header, Address[] successReceivers,
25         Address[] failedReceivers, Address[] timeoutReceivers);
26 }
```

Unicastverfahren benötigen beispielsweise immer die Empfängeradresse. Sie kann aber bei allen Verfahren spätestens bei vor Auslieferung zu den Empfängerobjekten im Header korrekt gesetzt werden. Im Falle von positionsbasierten Unicastverfahren kann eine Vorgabe der Empfängerposition eine unter Umständen aufwändige Abfrage der Empfängerposition in einem Verzeichnisdienst überflüssig machen. Zusatzinformationen können im Header enthalten sein, auch wenn das Routingverfahren diese nicht verwendet bzw. setzt. Absenderinformationen wie Adresse und Position werden von vielen Verfahren nicht notwendigerweise benötigt. Auch Informationen über die zurückgelegte oder noch zurückzulegende Route wird bisher nur von dem Verfahren DSR benötigt, dennoch kann es für den Nachrichten versendenden Dienst Sinn machen, die zurückgelegte Route zu protokollieren. Beispielsweise kann so eine Unicastantwort mit dem Verfahren DSR erfolgen. Das Flag `isPromiscuousHeader` erlaubt Routingalgorithmen, auch Headerinformationen von Übertragungen in der Nachbarschaft zu empfangen. Somit können beispielsweise linkbasierte Verfahren Routinginformationen durch das Mithören von Übertragungen sammeln. Das Flag `isPromiscuousMessage` erlaubt dem nachrichtenversendenden Dienst die Auslieferung der Nachricht an alle Geräte im Routingkorridor der Nachricht, auch wenn diese aufgrund

des Routingverfahrens eigentlich keine Empfänger sind. Beide Flags lösen also einen impliziten Promiscuousmode bei den Nachbargeräten aus, die diese Nachricht daher (zumindest zum Teil) verarbeiten, falls sie korrekt empfangen wurde. Delegationsdaten werden benötigt, wenn Routingalgorithmen von sich aus Nachrichten an andere Verfahren delegieren wollen. Ist eine Rückkehr zum Ursprungsverfahren vorgesehen, kann es notwendig sein, dass Routinginformationen des ursprünglichen Verfahrens erhalten bleiben. Verwendet wird die Delegation beispielsweise bei der Kombination von positionsbasierten Greedyverfahren mit einem Faceroutingverfahren als Rückfallstrategie für den Greedyroutingfehler. Im Fehlerfall delegiert Greedy die Nachricht an das Faceroutingverfahren, welches unter bestimmten Voraussetzungen wieder auf das Greedyverfahren zurückfallen kann. Somit lassen sich auch neue Verfahren aus bestehenden Implementierungen aufbauen. GSR [71] kann beispielsweise als eine Folge von geographischen Anycasts implementiert werden, welche die Nachrichten an die vorgegeben Zwischenstationen ausliefern.

Listing 8.9 Dieser dynamische Routingheader verknüpft positionsbasiertes mit linkbasiertem Routing. Greedy liefert die Nachricht in die Zielregion `geoLocation`, DSR findet dort das eigentliche Gerät `deviceAddress`. DSR ist hierbei bei der Suche auf die Zielregion beschränkt.

```
1  public class GeoLinkUnicast implements MultipleRoutingHeader {

3      private  GeographicLocation geoLocation;
4      private Address deviceAddress;
5      (...)
6      public DefaultRoutingHeader getFirst() {
7          return new GreedyRouting.
8              AnycastGreedyRoutingHeader(geoLocation);
9      }

11     public MultipleRoutingHeader copy() {
12         return new GeoLinkUnicast(deviceAddress,geoLocation);
13     }

15     public DefaultRoutingHeader getNext(
16             DefaultRoutingHeader lastRoutingHeader){
17         hasNextHeader=false;
18         // die Standardinformationen des alten RoutingHeaders
19         // werden übernommen
20         return DSRRouting.LocationDSRRoutingHeader(deviceAddress,
21             geoLocation,lastRoutingHeader);
22     }

24     public boolean hasNext() {
25         return hasNextHeader;
26     }

28 }
```

Da Routingalgorithmen nicht selbst Nachrichten weiterleiten, muss eine steuernde Komponente existieren. Um Nachrichten direkt multihop auszuliefern, existiert ein generischer Routingdienst. Dieser verwaltet die Routingauf-

träge, leitet Nachrichten an Nachbargeräte weiter und liefert die Nachricht lokal aus. Der Routingdienst verwaltet auch die Standardinformationen im Routingheader. Deshalb ist es anderen Diensten möglich, Headerinformationen zu verwenden und Kommunikationskonfigurationen zu verändern, auch wenn der verwendete Routingalgorithmus diese nicht unterstützt. Neben der Delegation von Routingaufträgen durch die Routingalgorithmen selbst ist auch eine Delegationsfolge vom Sendeauftraggeber vordefinierbar. Im einfachsten Fall wird der Nachricht die Folge der Routingheader mitgegeben und diese abgearbeitet. Nachrichteneffizienter ist die dynamische Headererzeugung (siehe Listing 8.9). Hierbei wird nach Abschluss eines Routingverfahrens der Header für das nächste Routingverfahren erzeugt. Dies setzt voraus, dass neben den erzeugten Headern auch der Kode zur Headererzeugung auf allen Geräten vorhanden ist oder mitversendet wird.

Die steuernde Komponente kann, neben dem Routingdienst, aber auch ein beliebiges anderes ActiveObject sein. Auch Routingalgorithmen können auf andere Verfahren direkt zugreifen, um dadurch einen Teil der Routingentscheidungen von dem anderen Verfahren treffen zu lassen. Beispielsweise nutzt die in diesem Routingframework realisierte Variante des Multicastalgorithmus SPBM [106] positionsbasierte Anycastroutingalgorithmen, um die Geräte zu bestimmen, die den Multicast weiterleiten sollen. Diese Entscheidungen werden von SPBM zusammengefasst und die Nachricht mit einem adressierten Broadcast versendet. Routingalgorithmen können auch zur Store-and-Forward Kommunikation eingesetzt werden (siehe Abschnitt 8.2.4). Hierfür sind aber nicht alle Routingverfahren geeignet. Falls während der Kommunikation Änderungen in den Links der besuchten Knoten erfolgen, können beispielsweise Faceroutingverfahren, die direkt auf den Kommunikationslinks arbeiten, keine Kreisfreiheit mehr garantieren.

8.6 Routingverfahren

- Greedyroutingvarianten
- Faceroutingvarianten
- CFR (Clusterbasiertes Facerouting)
- GSR (Geographic Source Routing)
- SPBM (Scalable Positionbased Multicast)
- Broadcastingvarianten (Regionsbeschränkt/Hopbeschränkt)
- DSR (Dynamic Source Routing)
- AODV (Ad-hoc On demand Distance Vector routing)

In der jetzigen Version des Routingframeworks existiert eine Vielzahl von Implementierungen gängiger Routingalgorithmen. Die positionsbasierten Un-

icastverfahren sind jeweils auch als positionsbasierter Anycast realisiert. Die linkbasierten Verfahren können ortsbasiert beschränkt werden, d.h. die Routenfindung findet nur innerhalb einer vorgegebenen Region statt. Bisher existieren für die in Box 8.6 gegeben Verfahren Implementierungen.

8.2.3 Sicherungsframework

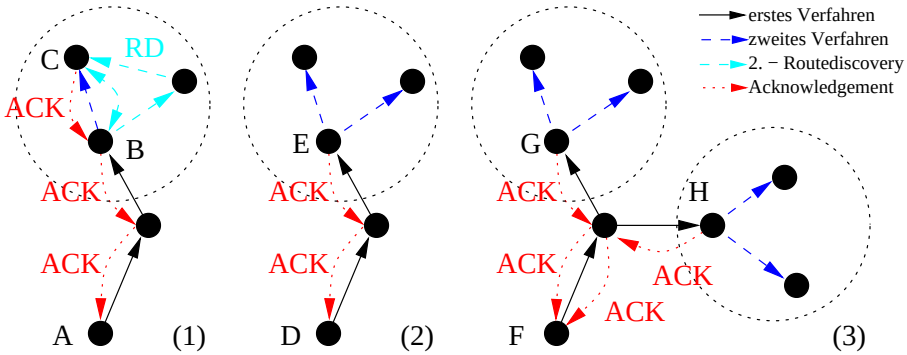


Abb. 8.7. Beispiele für Sicherungsvarianten multimodaler Kommunikationsverfahren. (1) Gerät A sendet einen Unicast nach C und verwendet bis Ankunft in der Zielregion bei Gerät B einen ortsbasierten Anycast. Nach einem Routediscovery in der Zielregion wird schließlich C mit einem linkbasiertem Verfahren erreicht und die Nachricht bestätigt. (2) Eine Nachrichtenbestätigung muss nicht am Ende der Kommunikation erfolgen. Bei dieser Locationcastvariante sendet E vor dem Fluten der Zielregion eine Bestätigung an D. (3) Mehrere Enden/Zwischenenden zur Bestätigung sind möglich (G und H), die Zahl der zu erwartenden Bestätigungsnachrichten muss F aber bekannt sein.

In der Middleware steht eine einfache Transportsicherung zur Verfügung, die eine Absicherung für direkte multihop Kommunikation durchführt. Hierbei muss das Sicherungsende nicht notwendigerweise ein vorgegebenes Gerät oder das endgültige Ziel sein. Beispielsweise kann das Zielgerät im Falle eines ortsbasierten Anycast erst bei Empfang in der Zielregion feststehen. Aufgrund der Möglichkeit der Hintereinanderschaltung von Kommunikationsverfahren ist es möglich, einen Wechsel des Verfahrens als Sicherungsendpunkt zu definieren. Beispielsweise kann ein Locationcast als ortsbasierter Anycast gefolgt von einem ortsbasierten Fluten realisiert werden. Somit kann die Ankunft einer Nachricht in der Zielregion des Locationcasts durch die Absicherung der Anycastkommunikation bestätigt werden. Eine Verallgemeinerung, d.h. die Wahl von mehreren aufeinander folgenden Zwischenpunkten, die bestätigt werden, ist auch denkbar. Eine Ende-zu-Multiende Absicherung benötigt das Wissen

über die Zahl der erwarteten Bestätigungen. Werden, wie in Abbildung 8.7(3), eine bekannte Zahl von Zielregionen adressiert, ist dem Sender auch die zu erwartende Zahl der Bestätigungen bekannt. Bei Multicastkommunikation ist im Allgemeinen dieses Wissen nicht vorhanden. So kann es zu so genannten *acknowledgement storms* kommen, wenn eine sehr große Empfängerzahl einen Multicast bestätigt. Somit ist die Ende-zu-Multiende Absicherung nur sinnvoll einsetzbar, wenn das Kommunikationsverfahren bzw. das multimodale Kommunikationsverfahren dazu geeignet ist. Möglich sind dann die multihop Varianten des Manycasts und des adressierten Multicasts. Im Falle des Manycasts wird die Nachricht durch eine undefinierte Empfängerteilmenge fester Größe bestätigt, der adressierte Multicast gibt die Empfängerteilmenge vor.

Die Sicherungsschicht sendet eine positive Bestätigung, wenn der vorgegebene Endpunkt bzw. Zwischenpunkt erreicht ist, und eine negative Bestätigung, wenn die Nachricht im Routingframework verworfen wurde. Die Bestätigung kann mit dem selben Kommunikationsverfahren, über das die Nachricht lokal empfangen wurde, zurückgesendet werden. Es ist auch möglich, das Antwortverfahren durch den Sender vorgeben zu lassen oder ein Verfahren zu verwenden, das als Standard eingerichtet wurde. Die Vorgabe des Verfahrens ist notwendig, wenn zur Antwort nicht dasselbe Verfahren verwendet werden kann, wie es beispielsweise bei einem Anycastverfahren der Fall ist.

Konfigurierbar sind in dieser einfachen Realisierung das Bestätigungsintervall und die Zahl der Nachrichtenwiederholungen in diesem Intervall. Im Bestätigungsintervall wartet der Sender auf positive oder negative Bestätigungen und wiederholt die Nachricht falls, diese ausbleiben. Sämtliche Konfigurationen können durch Standardwerte vorgegeben oder für jede Nachricht konfiguriert werden. Somit werden im Nachrichtenheader zusätzliche Informationen benötigt. Damit der Sender Nachrichten wiederholen kann, muss die Nachricht bei allen Empfängern vorgehalten werden, in Abhängigkeit vom gewählten Timeout. Deshalb muss dieses Delta im Header enthalten sein, wenn es vom Standardwert abweicht. Wird ein alternatives Antwortverfahren verwendet, wird auch dies bei Bedarf im Header codiert.

8.2.4 En-Passant Kommunikation

Die En-Passant Kommunikation realisiert die Store-and-Forward Kommunikation in der Middleware. Hiermit kann zum einen eine Objektpropagierung erreicht werden, wie sie in [44] beschrieben wird. Dabei werden Objekte dupliziert, sobald Geräte in der Nachbarschaft daran Interesse besitzen. Die Entscheidungen werden anhand von Informationen im Beaconing und unter Umständen durch Austausch von Abgleichprofilen getroffen. Zum anderen kann eine Store-and-Forward Kommunikation wie in [45] realisiert werden, die einzelne Objekte auf Nachbargeräte migriert, sobald die Nachbargeräte bestimmte Eigenschaften, z.B. Position und Bewegungsrichtung, besitzen. Die

Objekte sind also nach einer erfolgreichen Übertragung nur auf einem Gerät vorhanden.

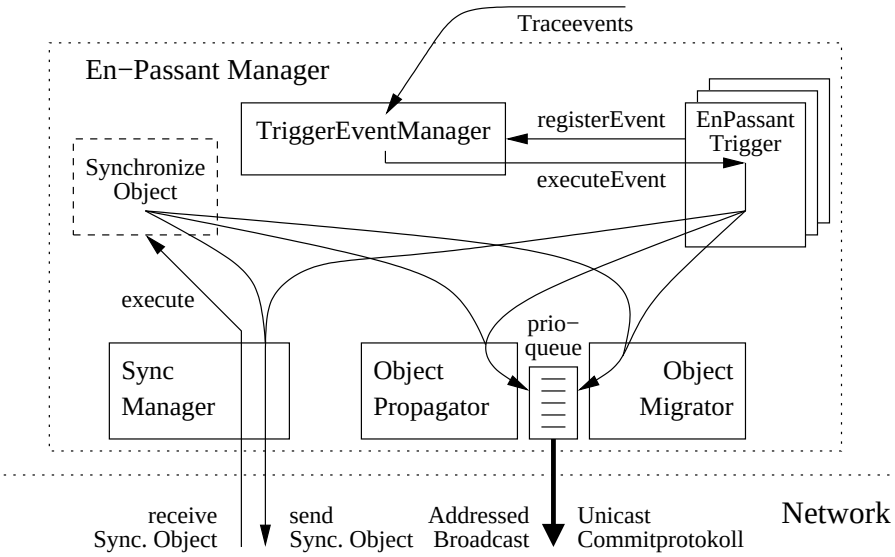


Abb. 8.8. Übersichtsbild des En-Passant Managers.

Die Unterstützung der En-Passant Kommunikation basiert somit auf Ereignissen, die von Änderungen in der Gerätenachbarschaft ausgelöst werden. Dazu zählen das Hinzukommen und Wegfallen von Nachbargeräten und die Änderung der Gerätespuren sowie Änderungen der Spuren der sich auf dem Gerät befindlichen Objekte. Reagiert wird also auf das Hinzukommen und Wegfallen von Gerätereprésentanten im lokalen Objektpool und auch auf Änderungen in den zu den lokalen Repräsentanten assoziierten Spurinformationen. Diese Informationen werden entweder vom Neighbordiscoverydienst (Abschnitt 8.3) über periodische Beaconnachrichten verbreitet oder aufgrund anderer Geräteinteraktionen vom lokalen Repräsentanten gesammelt.

In einem En-Passant Manager können ActiveObjects `EnPassantTrigger` registrieren, die aufgrund bestimmter Spurereignisse aktiviert werden sollen. Hierfür stellen sie eine Menge von Ereignistemplates zur Verfügung, die sich auf die Änderung in den Spuren der Gerätenachbarschaft oder des lokalen Gerätes beziehen. `EnPassantTrigger` werden auf drei Arten ausgelöst. (1) bei der Initialisierung, um die aktuelle Nachbarschaft zu überprüfen, (2) bei jedem Ereignis, wobei die dem Ereignis entsprechende Spurinformation des Objektes übergeben wird und (3) nach einer vorgegebenen Zeit, wobei alle gesammelten Ereignisse übergeben

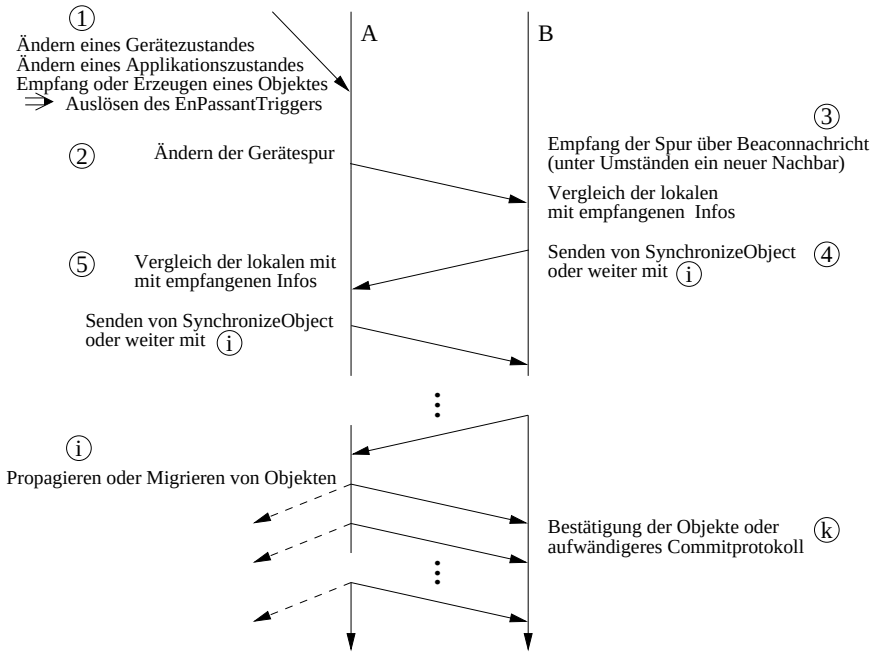


Abb. 8.9. Ablauf eines En-Passant Kommunikationsprotokolls. (1) Aufgrund von lokalen Zustandsänderungen ändert sich die Gerätespur (2) eines Gerätes. Diese wird über das Beaconing propagiert und von anderen empfangen. Neue oder geänderte Spurinformationen lösen En-Passant Trigger aus, die diese mit den lokalen Objektinformationen oder Migrationszielen vergleichen (3). Dies kann eine Folge von Übertragungen von `SynchronizeObject`s zwischen A und B auslösen ((4), (5), ...), bis die zu migrierenden/propagierenden Objekte bestimmt sind. An jeder Stelle nach (3) kann die Migration/Propagierung von Objekten erfolgen (i), auch von Gerät B aus. Die Propagierung erfolgt mit dem adressierten Broadcast auch an andere Geräte: zuverlässig, falls die Objekte auch angefordert wurden, oder unbestätigt zur altruistischen Speicherung. Der Empfänger bestätigt den Empfang (k) bzw. führt im Falle der Migration ein umfangreicheres Commitprotokoll (z.B. aus Box 4.12) durch.

werden. Im 3. Fall kann das Auslösen des Ereignisses noch weiter verzögert werden, um das Beenden von Objektpropagierungen in der Nachbarschaft abzuwarten. Dies ist dann sinnvoll, wenn zu erwarten ist, dass die Nachbarschaft stabiler ist und über längere Zeit besteht oder eine Spuränderung durch eine momentane Objektverbreitung ausgelöst worden ist und diese Objekte noch empfangen werden. Ein `EnPassantTrigger` besitzt Zugriff auf den `ObjectPool` und den `TracePool` der Middleware, um die Objekte zu bestimmen, die propagiert bzw. migriert werden sollen. Deren Kennungen können dem En-Passant Manager mitgeteilt werden, um die Kommunikation anzustoßen. `EnPassantTrigger` können diese Entscheidung auch an ein Nachbargerät delegieren, welches mit Hilfe zusätzlich kommunizierter

Informationen seinerseits Objekte bestimmen kann. Hierfür wird ein Abgleichobjekt (`SynchronizeObject`) erzeugt, welches Zusatzinformationen und eine Abgleichfunktion beinhaltet. Dies kann, je nach Abgleich, beliebig oft erfolgen (siehe Abbildung 8.9). Ein `EnPassantTrigger` kann somit nach dem Auslösen (1) ein `SynchronizeObject` an einen oder mehrere Nachbarn versenden, (2) eine Menge von Objektkennungen angeben, die entweder zu einen Nachbarn propagiert oder migriert werden sollen oder (3) die Auslösung verschieben oder aussetzen.

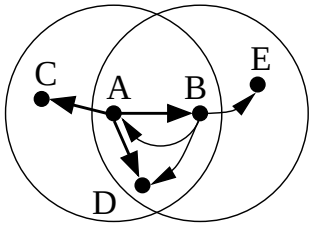
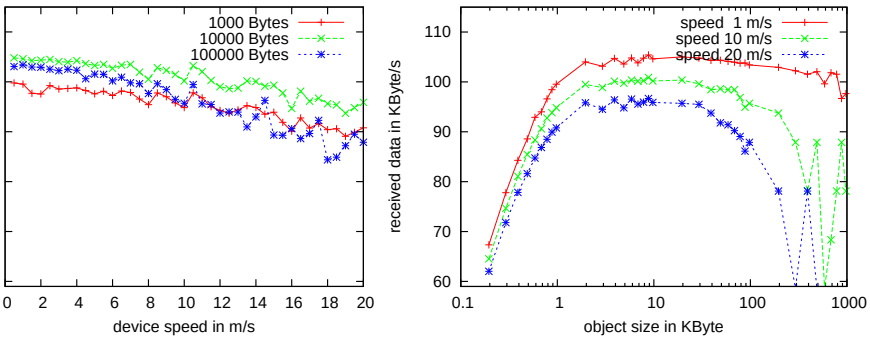


Abb. 8.10. Die Objektpropagierung. Gerät A propagiert Objekte an B. Die Objekte können von allen Geräten im Sendebereich von A empfangen werden. B bestätigt den Objektempfang und somit können E und D eventuelle Propagierungen des Objektes an B einsparen.

Die Objektpropagierung erfolgt mit dem `ObjectPropagator`. Dieser sammelt Propagierungsentscheidungen und fasst sie, wenn möglich, zusammen. Die Objekte werden prioritätsbasiert mit einem adressierten Broadcast an die Nachbargeräte verbreitet. Die Empfangsbestätigung erfolgt von allen Geräten, die für einen Empfang vorgesehen sind. Alle anderen dürfen die Objekte empfangen, um sie altruistisch zu speichern oder um einen überflüssigen Abgleich zu vermeiden. Propagierungsaufträge werden entfernt, wenn Nachbarn verschwinden oder eine Bestätigungsnachricht für dasselbe Objekt von einem Gerät empfangen wird. Letzteres ist möglich, wenn gleichzeitig ein Abgleich mit einem anderen Gerät erfolgt oder das Objekt über einen anderen Mechanismus propagiert wurde. Auf diese Art propagierte Objekte gelangen immer zuerst in die Lobby der Middleware und müssen deshalb explizit (z.B. durch eine Anwendung) in den Residencepool verschoben werden, um längerfristig auf diesem Gerät verbleiben zu können. Die Lebensdauer und die Prioritäten der Propagierung werden vom `EnPassantTrigger` bzw. vom `SynchronizeObject` festgelegt. Die Propagierung der Objekte erfolgt einzeln, da es jederzeit zu einem Abbruch der Verbindung kommen kann. Somit wird nur das Objekt nicht empfangen, das während dem Abbruch versendet wird und nur dieses Objekt wurde unnötig versendet. Um den Kommunikationsoverhead zu minimieren, können sehr kleine Objekte zu größeren zusammengefasst und in einer Nachricht propagiert werden. In diesem Fall sind aber bei einem Verbin-

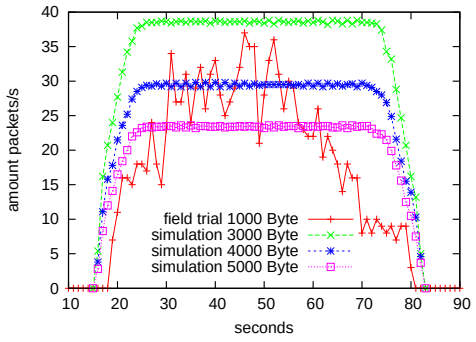
8.7 En-Passant



Zur Untersuchung der En-Passant Kommunikation werden zwei Geräte aneinander vorbei bewegt, wobei Gerät 1 versucht, möglichst viele Objekte an Gerät 2 zu übertragen. Hierbei wird die Gerätegeschwindigkeit zwischen 0.5 und $20m/s$ variiert. Die Sendereichweite des simulierten IEEE802.11 Netzwerks beträgt $100m$, so dass in Abhängigkeit der Geschwindigkeit der Datenaustausch zwischen 100 und 2.5 Sekunden andauern kann. Um das andere Gerät zu erkennen, werden im Sekundentakt, randomisiert mit einem Jitter von 0.5 Sekunden, Beaconnachrichten versendet. Die Objektgröße variiert zwischen 100 Bytes und 10^6 Bytes.
Links ist die Datenrate in Abhängigkeit der Gerätegeschwindigkeit für die Objektgrößen 1000, 10000 und 100000 Bytes abgetragen. Die Datenrate sinkt, weil der Anteil der genutzten Begegnungszeit immer geringer wird, da das andere Gerät erst erkannt werden muss. Ab einer Geschwindigkeit von $50m/s$ ist zu erwarten, dass keine Objekte mehr übertragen werden können, da die Begegnungszeit nur noch zum Erkennen des anderen Gerätes ausreicht. Rechts ist die Datenrate in Abhängigkeit der Objektgröße für die Geschwindigkeiten $1, 10$ und $20m/s$ abgetragen. Hier ist zu erkennen, dass die optimale Objektgröße zwischen 4 und 11 KByte liegt. Je höher die Geschwindigkeit, desto wichtiger ist diese optimale Größe. Das Optimum liegt darin begründet, dass für kleine Pakete der Overhead der Übertragung groß ist, bei großen Paketen aber die Wahrscheinlichkeit für einen Verlust aufgrund von Störungen steigt. Darüber hinaus geht die letzte Nachricht immer verloren, weil die Kommunikation abbricht. Je länger die Nachricht ist, desto mehr Daten wurden unnötig verbreitet.
Dieser Versuch wurde in [44] mit realen Geräten und WLAN durchgeführt. Hierbei wurden bei Schrittgeschwindigkeit (ca. $1.5-2.0m/s$) und einer Reichweite von etwa $100m$ insgesamt 1.4MB Daten mit einer Objektgröße von 1000KB übertragen. In der Simulation liegt dieser Wert bei 6.5MB ($1.5m/s$) bzw. 4.9MB ($2.0m/s$). Die Paketverlustrate war im Feldversuch erheblich höher - hier waren auch Störeffekte der Antennen bei der Begegnung und die Abschirmung der Geräte durch die Versuchspersonen nach der Begegnung in den Verlustraten zu beobachten.

– Fortsetzung Box 8.7 –

Unten sind die Ergebnisse des Feldversuchs mit denen der Simulation (bei $1.5m/s$) verglichen. Dargestellt sind die empfangenen Pakete je Sekunde. Sind die Pakete um den Faktor 3 größer, wird die reale Messung gut angenähert. Wird das Netzwerkrauschen mit einer höheren Varianz randomisiert, werden vermutlich auch die Schwankungen stärker ausgeprägt sein.



dungsabbruch alle Objekte der aktuell propagierten Nachricht nicht empfangen worden.

Der `ObjectMigrator` migriert Objekte zu Nachbargeräten. Das bedeutet, das Objekt wird nur an genau ein Gerät versandt und nach erfolgreichem Empfang beim Sender gelöscht. Das Kommunikationsprotokoll nutzt das in Box 4.12 dargestellte Agentenübertragungsverfahren aus [45]. Hierbei wird der Objektverlust verhindert, unter Umständen aber eine Duplikation verursacht. Ist möglicherweise eine Duplikation entstanden, wird diese Information als Spur an das Objekt gehangen. Die Information beinhaltet die Geräteadresse und einen Zeitstempel des Gerätes, auf dem das Protokoll fehlgeschlagen ist.

8.2.5 Kommunizierbare Objekte

Die Middleware unterscheidet zwei Typen kommunizierbarer Objekte. Diese unterscheiden sich in der Art der Auslieferung an den Empfänger. Ein RemoteEvent entspricht der Auslieferung der lokalen Ereignisse, d.h. ein Empfänger kann sich genauso für den Empfang einer Nachricht registrieren, wie für den Empfang eines Ereignisses. Die Ereignisse werden meist zur direkten Interaktion zwischen ActiveObjects auf unterschiedlichen Geräten eingesetzt oder um auf anderen Geräten Ereignisse auszulösen. In der Regel werden diese Objekte nicht beim Empfänger gespeichert. Der zweite Objekttyp MessageObject wird nach dem Empfang in der Objektlobby abgelegt. Dieser Typ wird zum Verbreiten von Datenobjekten eingesetzt, die nach dem

Empfang erhalten bleiben sollen. Ein Beispiel ist das Veranstaltungsmaterial einer Vorlesung wie im Falle der Applikation DistScript (Kapitel 9), das empfangen, gespeichert und weiterverbreitet wird. Datenobjekte werden meist längerfristig gespeichert, daher müssen Anwendungsobjekte bei Interesse an bestimmten Objekttypen diese in den Residencepool verschieben.

Kommunizierbare Objekte können, unabhängig von ihrem Typ, auf zwei Arten versendet werden. Die direkte Kommunikation (Abschnitte 8.2.1 - 8.2.3) liefert ein Objekt sofort zu den Zielen aus und verwirft es, falls eine sofortige Auslieferung nicht möglich ist. Um auf diesem Wege ein Ziel erreichen zu können, ist ein zusammenhängendes Netzwerk notwendig. Im Falle der Store-and-Forward Kommunikation hängt die Auslieferung der Nachricht von der Realisierung der `EnPassantTrigger` ab. Prinzipiell ist auch hier eine direkte Auslieferung möglich, in der Regel wird die Weiterleitung aber ereignisgetriggert durchgeführt, d.h. erst Änderungen im Geräteumfeld führen zu einer Objektweiterleitung. Somit kann die Store-and-Forward Kommunikation auch als Rückfallstrategie für die direkte Kommunikation verwendet werden, wie es beispielsweise im Falle der marktplatzbasierten Kommunikation eingesetzt wird.

Kommunizierbare Objekte besitzen immer eine beschränkte Lebensdauer. Werden RemoteEvents direkt kommuniziert, ist diese nicht zeitlich sondern in der Zahl der Hops beschränkt, d.h. die Objekte werden nach einer festen Zahl besuchter Geräte verworfen. Eine zeitlich beschränkte Lebensdauer besitzen alle anderen Kombinationsvarianten. MessageObjects benötigen eine Zeitinformation für die Leasezeit zur Speicherung in der Lobby, damit sie nach Ablauf des Leases gelöscht werden können, falls sie nicht weiterverarbeitet oder weiterversendet werden. Die Store-and-Forward Kommunikation verwendet die Lobby auch zur Speicherung auf Zwischenstationen. Die beschränkte Lebensdauer ist hier notwendig, falls eine Weiterverbreitung von Objekten nicht mehr möglich ist. Die Lebensdauer in der Lobby kann allerdings verlängert werden, wenn ein lokales ActiveObject dies durchführt.

Alle beim Empfang lokal verfügbaren Informationen über die erfolgte Kommunikation werden mitgeliefert. Im Falle eines RemoteEvents wird eine Referenz auf diese Informationen mitgegeben, für ein MessageObjects wird diese im Spurpool mit dem Objekt als privates Spurelement assoziiert. Die Information beinhaltet immer die Informationen des Linklayers über den letzten Sender, im Falle der direkten multihop Kommunikation den Routingheader und unter Umständen auch vorhandene Sicherungsinformationen. Der Empfänger kann die Informationen verwenden, um Wissen über den Sender bzw. das Netzwerk zu sammeln oder auch um dies für Entscheidungen zur Kommunikationsadaption für eine Antwort zu nutzen.

Die Vorgehensweise zur Versendung eines kommunizierbaren Objektes ist stark von den verwendeten Kommunikationsmechanismen abhängig. Direkte Kommunikation ist auftragsorientiert, d.h. es wird ein Kommunikationsauftrag an einen der Kommunikationsdienste abgesetzt. Aufträge können mit

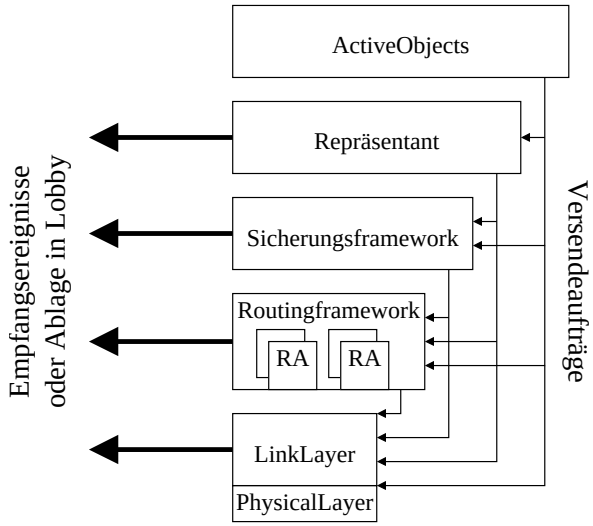


Abb. 8.11. Mögliche Nachrichtenwege über Kommunikationsdienste der Middleware im Falle der direkten Kommunikation. Eine traditionelle Schichtung ist möglich, aber nicht zwingend. Empfangene Objekte werden je nach Typ als Ereignis propagiert oder in der Lobby abgelegt.

Rückrufobjekten versehen werden, so dass der Absender über den Zustand seines Versendeauftrags informiert wird. Signalisiert werden kann immer der lokale Zustand der Nachricht bis zum Verlassen des Gerätes. Wird das nächste Gerät mit einer gesicherten Kommunikation angesprochen, kann auch das erfolgreiche Weiterleiten zum nächsten Gerät signalisiert werden, z.B. wenn eine Unicastnachricht mit direkter singlehop Kommunikation an einen Nachbarn gesendet wird. Wird zusätzlich das Sicherungsframework verwendet, kann auch der Zustand der multihop Kommunikation signalisiert werden. Alle Kommunikationsdienste sind für jede einzelne Nachricht konfigurierbar. Das bedeutet, dass mit jeder versendeten Nachricht unterschiedliche Strategien nutzbar sind. Somit ist aber keinerlei Transparenz mehr gegeben. Andere Objekte müssen deshalb die verwendeten Kommunikationsdienste und ihre Möglichkeiten kennen. Der Transparenzverlust kann in gewissem Maße durch den Einsatz von Objektrepräsentanten (siehe Kapitel 6) ausgeglichen werden, die die Kommunikation mit anderen Objekten zum Teil kapseln. Diese können auch umfangreichere Kommunikationsstrategien und -muster darstellen. Durch Konfiguration und die Beauftragung eines "höheren" Kommunikationsdienstes entspricht die Anordnung der Dienste im Falle der direkten Kommunikation der traditionellen Schichtung entsprechend dem ISO/OSI Referenzmodell, wie in Abbildung 8.11 dargestellt. Diese Anordnung ist aber nur eine der möglichen. ActiveObjects sind in der Lage, auf jedem dieser Dienste direkt zu basieren und die höheren Schichten zu umgehen. Prinzipiell

könnte das Sicherungsframework auch zur Sicherung der singlehop und auch der (Unicast) Store-and-Forward Kommunikation verwendet werden – dies ist aber nicht realisiert. Möglich ist auch ein direkter Zugriff auf Routingalgorithmen, um Routingentscheidungen treffen zu lassen und selbst die Nachrichten an das nächste Gerät zu übertragen.

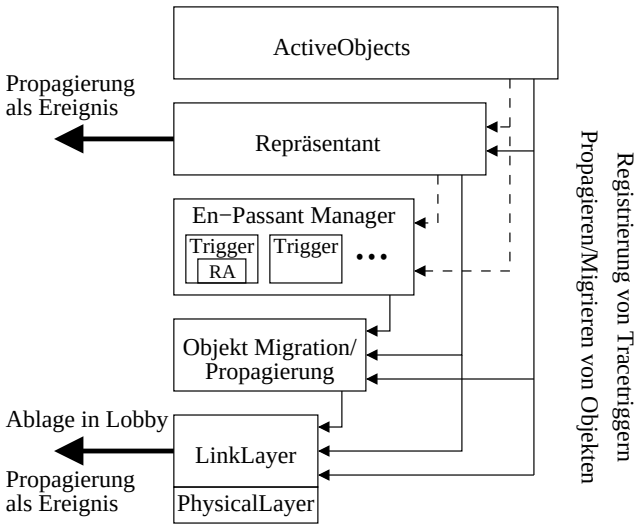


Abb. 8.12. Schichtung der Store-and-Forward Kommunikation.

Die Store-and-Forward Kommunikation ist implizit. Das bedeutet, es werden keine Kommunikationsaufträge für einzelne Objekte gegeben. `EnPassantTrigger` zur Objektpropagierung werden stattdessen für Objektklassen oder Objektgruppen registriert und können somit zur Weiterverbreitung führen, sobald ein neues Objekt entsteht oder empfangen wird. Bei einer Objektpropagierung besteht somit auch keine Notwendigkeit, über den Erfolg einer Weiterpropagierung zu informieren, da diese nicht explizit beauftragt wird. Werden `EnPassantTrigger` zur Objektmigration eingesetzt, ist die Registrierung des Triggers einem Auftrag gleichzusetzen. Den Auftrag führt der Trigger durch, der über den Erfolg einer Migration informiert wird. Ähnlich der direkten Kommunikation lässt sich auch für diese Kommunikationsart eine Schichtung angeben (siehe Abbildung 8.12). Diese besitzt aber eine andere Bedeutung, da der Zugriff auf den En-Passant Manager der Registrierung/-Deregistrierung von `EnPassantTrigger` entspricht. ActiveObjects können dies direkt durchführen oder indirekt durch einen Zugriff auf ein Repräsentantenobjekt auslösen. Tracetrigger können, wie hier dargestellt, auch auf Routingalgorithmen oder andere Dienste zugreifen. Kommunizierbare Objekte können zur Store-and-Forward Kommunikation präpariert werden, wenn sie

durch Zugriff auf einen Repräsentanten mit diesem assoziiert werden. So kann Gruppeninformation zur Objektspur hinzugefügt werden, so dass eine Propagierung im Rahmen eines Gruppentracetriggers erfolgen kann. Die Migrations- bzw. Progagierungskomponente kann auch direkt verwendet werden und auftragsorientiert können Objekte migriert oder propagiert werden. Schließlich kann die Store-and-Forward Kommunikation durch direkten Zugriff auf den Linklayer von einem ActiveObject selbst realisiert werden.

8.3 Beaconing und Neighbordiscovery

Das Periodische Versenden von Nachrichten, so genannte *Hello-* oder *Beaconnachrichten*, ist in mobilen ad-hoc Netzwerken ein vielfach eingesetzter Mechanismus, um zumindest die Geräte in der direkten Umgebung zu kennen und Änderungen des direkten Umfelds zu erkennen. Beaconing ist häufig ein notwendiger Teil des Media Access Control (MAC), wie z.B. bei IEEE802.11 (WLAN) oder 802.15.1 (Bluetooth). Neben der Absenderadresse sind darin noch weitere, zur Netzverwaltung und Synchronisation notwendige Informationen enthalten. WLAN Beaconnachrichten beinhalten beispielsweise einen Zeitstempel, das momentane Beaconingintervall und Netzwerkinformationen wie z.B. die ID des Servicesets (SSID) in der sich das Gerät befindet. Kommunikationsverfahren benötigen häufig Beaconnachrichten zum Treffen von lokalen Routingentscheidungen oder um Routingtabelleneinträge zu invalidieren, wenn ein Nachbar wegfällt. Auch hier werden häufig zusätzliche Informationen verbreitet, bei positionsbasierten Verfahren beispielsweise die Geräteposition. Schließlich werden Beaconnachrichten auch von MobileObjects benötigt, um Spurinformationen zu sammeln und auch zu verbreiten (siehe Kapitel 5) oder um überhaupt von der Existenz anderer MobileObjects zu erfahren.

Um zu verhindern, dass sehr viele verschiedene Nachrichten periodisch versandt werden, die darüber hinaus auch redundante Informationen beinhalten (z.B. Geräteadresse und Position), ist es sinnvoll, periodische Nachrichtenverbreitung in einem zentralen Dienst zu realisieren. Dieser Dienst versendet periodisch Nachrichten, die zumindest Basisinformationen des Gerätes beinhalten. Andere ActiveObjects sind in der Lage, Daten im Beaconingdienst zu registrieren, die an diese periodische Nachricht angehangen werden (so genanntes Piggybacking). Die Daten besitzen zur Unterscheidbarkeit eine eindeutige Kennung und entsprechen den Kennungen im `TracePool`. Notwendig ist dies auch zur Interpretierbarkeit der Daten beim Empfänger.

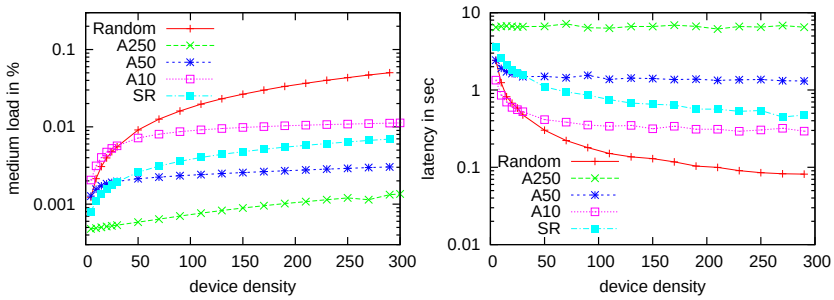
Die Periode, mit der die Beaconnachricht versendet wird, ist in der Regel nicht statisch. Eine feste Periode kann zu längerfristigen Kollisionen aufgrund des Hidden-Terminal-Problems kommen. Hierbei können aufgrund wiederholt gleichzeitiger Sendeereignisse zweier Nachbarn diese über längere Zeit nicht sichtbar sein. Eine randomisierte Periode kann die Wahrscheinlichkeit und die Dauer dieses Phänomens reduzieren. Steigt die Zahl der Geräte, nimmt aber

auch die Zahl der möglichen Kollisionen zu, bis hin dazu, dass die gesamte Bandbreite aufgrund von Beaconnachrichten verbraucht wird. Eine mögliche Lösung hierfür ist die Anpassung der Beaconingperiode an die Zahl der sichtbaren Nachbarn. Die Nachrichten werden also seltener versendet, je mehr Beaconnachrichten empfangen werden. Dadurch reduziert sich natürlich auch die Aktualität der propagierten Daten. Der `Beaconscheduler` des Neighbormanagers (siehe Abbildung 8.13) existiert in drei Realisierungen: statisch, randomisiert und adaptiv randomisiert. Eine weitere Möglichkeit, Kollisionen zu verhindern und die Medienbelastung zu reduzieren, ist die Reduzierung der Sendestärke. Dies kann erfolgen, wenn die Zahl der Nachbarn einen Schwellenwert überschreitet. Damit werden nur kommunikationstechnisch günstige, also meist geographisch nahe, Nachbarschaften aufgebaut. In [59, 104] wird bei einer gleichverteilten Knotenmenge ein Knotengrad von sechs bzw. acht als optimal beschrieben.

In vielen Anwendungen des Beaconingmechanismus werden Ereignisse benötigt, wenn ein Gerät die Nachbarschaft betritt oder verlässt. Somit müssen Nachbarschaftsinformationen zwischengespeichert und wieder gelöscht werden, um dadurch eine Nachbarschaft als beendet erklären zu können. Diese Invalidierungsperiode entspricht einem Vielfachen der Beaconingperiode, um das Ausbleiben von mehreren aufeinander folgenden Beaconnachrichten tolerieren zu können. Da die Beaconingperiode der einzelnen Nachbarn unterschiedlich sein kann, muss die aktuelle Periodendauer Teil der Beaconnachricht sein.

Das Piggybacking kann nicht beliebig dynamisch sein, da eine Beaconnachricht nicht beliebig groß werden darf. Je größer eine Nachricht ist, die ungesichert übertragen wird, desto größer ist auch die Verlustwahrscheinlichkeit aufgrund von Kollisionen oder Störungen im drahtlosen Medium. Darüber hinaus sind Broadcastnachrichten auf eine feste MTU beschränkt – eine Fragmentierung ist meist nicht vorgesehen. Um dennoch dynamisch Daten anhängen zu können, ist es also notwendig, mit jeder Beaconnachricht eine Auswahl zu treffen, welche Daten versendet werden sollen. Eine Möglichkeit ist ein prioritätsbasiertes Datenscheduling, wie es z.B. beim Verfahren Periodicast [35] eingesetzt wird. Hier werden aus Listen mit unterschiedlichen Prioritäten Daten ausgewählt und in die Beaconnachricht geschrieben bis diese gefüllt ist. Eine Priorisierung der Daten ist notwendig, da z.B. Daten von Basisdiensten eine höhere Wichtigkeit besitzen als Daten von Objekten, die nur eine einzige Anwendung betreffen. Priorisierung kann auch aufgrund der Änderungshäufigkeit von Daten erfolgen. Ändern sich Daten seltener, müssen sie nicht unbedingt in jeder Beaconnachricht enthalten sein. Die Middleware bietet hierfür zwei Realisierungen an. Der `SimpleDataScheduler` versendet mit jedem Beaconing alle registrierten Daten. Diese einfache Realisierung kann eingesetzt werden, wenn der Datenumfang von vorneherein bekannt ist. Der `PriorityDataScheduler` verwendet Prioritäten und versendet die Daten mit einem Vielfachen der Basisperiode, wobei sich das Vielfache aus der Priorität

8.8 Beaconing – Random vs. Adaptive



Es wurden Simulationen durchgeführt, um den Aufwand des Beaconings zu bestimmen und den Unterschied zwischen randomisierten und adaptiven Verfahren zu zeigen. Auf einer Fläche von $200m$ x $200m$ sind Geräte mit einer Sendereichweite von $50m$ (802.11MAC; 11Mbit) gleichmäßig verteilt und die Zahl der Geräte ist so gewählt, dass die durchschnittliche Zahl der Nachbarn zwischen 5 und 300 variiert. Das randomisierte Beaconing versendet jede Sekunde eine Nachricht, die Varianten des adaptiven Verfahrens variieren bei einer beobachteten Nachbarzahl von 0 bis 500 die Beaconingperiode linear zwischen 0.5 und 10 (`A10`), 50 (`A50`) bzw. 250 (`A250`) Sekunden und zusätzlich wird die Anpassung $0.5+\sqrt{\#Nachbarn}/2$ (`SR`) untersucht. Alle Verfahren verwenden einen Abweichung von 10% ihrer aktuellen Periode.
Auf der linken Seite ist die lokal beobachtete Medienauslastung gemittelt über alle simulierten Geräte und Simulationsläufe über den Knotengrad abgetragen. Die Last ist dabei über eine logarithmische Skalierung abgetragen. Während die Medienlast bei linear steigenden Knotengrad beim randomisierten Beaconing linear steigt, steigt die Last bei linear adaptiven Verfahren nahezu marginal. Bei `SR` steigt die Last zwar auch linear aber erheblich schwächer.
Um die Auswirkung des verlangsamten Beaconing zu überprüfen wird über das Beaconing eine kurze Information in Form eines Zeitstempels im Netz propagiert. Diese Information wird auf einem Gerät auf der einen Seite des Netzes periodisch inkrementiert und alle anderen führen eine Maximumsbildung durch. Die rechte Seite stellt die Dauer der Propagierung dar, beginnend mit dem ersten Versenden bis zum ersten Empfang eines Zeitstempels auf einem bestimmten Gerät auf der gegenüberliegenden Netzseite. Theoretisch ist zu erwarten, dass alle Verfahren gegen das Latenzoptimum der sofortigen Weiterverbreitung der Information konvergieren, da mit zunehmender Dichte die Wahrscheinlichkeit für eine sofortige Weiterleitung bei einem der Geräte in Richtung des Ziels steigt. Die einzelnen Verfahren unterscheiden sich in der Konvergenzgeschwindigkeit. Während beim randomisierten Verfahren die Latenz am stärksten fällt, ist das Gefälle um so schwächer, je stärker die Adaption ausfällt. `SR` besitzt das stärkste Gefälle der Adaptiven Verfahren und ist aber erst bei einer durchschnittlichen Gerätezahl von 290 gleichauf mit dem linearen Verfahren `A10`. Im Fall von `A250`, ist die Latenz nahezu konstant bei 7.5 Sekunden. Der Abstand zwischen Random und `A10` ist minimal, nimmt aber im gemessenen Bereich zu.

– Fortsetzung Box 8.8 –

Es ist festzuhalten, dass die lineare Adaption in dünneren Netzen (< 50 Nachbarn) stärker ausfallen muss, als in Netzen mit sehr hoher Dichte (ab ca. 50 Nachbarn). Somit ist eine Kombination aus `A10` und `A250` ideal, wobei ersteres für Dichten bis zu 50, das zweite für höhere Dichten verwendet wird. Auch das Verfahren `SR` ist ein guter Kompromiss zwischen Netzlast und einer nur schwachen Adaption des Beaconings. Allerdings ist hier die Adaption unter 20 Nachbarn zu stark.

bestimmt. Hierbei ist die Beaconnachrichtengröße immer noch unbeschränkt. Wird die Beaconnachrichtengröße beschränkt, kann bei großem Datenumfang eine direkte Abbildung von Priorität auf Periode nicht mehr erfolgen und somit werden bei Überlast Daten seltener propagiert. Daten der höchsten Priorität (z.B. für Routingverfahren), bei denen also eine Verletzung der Priorität kritischer ist, sollten demnach von bekannt und in der Größe nicht variabel sein, um ein Propagieren mit der Basisperiode gewährleisten zu können.

Die Medienlast kann reduziert werden, wenn Daten nur bei Bedarf gesendet werden, dass heißt, entweder wenn Daten geändert wurden oder die Nachbarschaft sich geändert hat. Eine Propagierung von Daten sollte aber weiterhin periodisch erfolgen, um Kommunikationsoverhead bei der Propagierung kleiner Datenmengen zu vermeiden. Um aber Nachbarschaftsänderungen zu erkennen, müssen weiterhin Beaconnachrichten versendet werden, ohne dabei Zusatzdaten anzuhängen. Problematisch ist dann der Verlust der Nachrichten mit Zusatzinformationen, da keine implizite Redundanz durch Nachrichtenwiederholungen vorhanden ist. Der `AdaptiveDataScheduler` (eine Erweiterung des `PriorityDataScheduler`) sendet die notwendigen Nachrichten mehrmals auf Basis der Beaconingperiode und wiederholt sie mit einem konfigurierbaren Basisperiodenvielfachen, um Nachrichtenverluste zu kompensieren. Darüber hinaus wird ein Hashwert über alle zu versendenden Daten gebildet der in jeder Beaconnachricht ohne Daten enthalten ist. Somit kann fehlende Information erkannt werden und explizit oder implizit, durch Erweiterung der eigenen Beaconnachricht, angefordert werden.

Unterschiedliche Sendereichweiten können dazu führen, dass aber nur eine unidirektionale Verbindung zwischen benachbarten Geräten besteht. Daher ist eine direkte Interaktion dieser Geräte nicht möglich. Dies kann bei Fehlschlag einer direkten Interaktion erkannt und entsprechend vermerkt werden. Die Bidirektionalität einer Verbindung ist somit allgemein eine wichtige Eigenschaft, da ein Fehlschlag von Unicastkommunikationsversuchen hohe Kommunikationslast verursachen kann. Die Verwendung von 2-hop Beaconnachrichten, wobei zumindest Adressen der Nachbargeräte zur Nachricht hinzugefügt werden, kann Bidirektionale Verbindungen erkennen. Hierbei ist aber die Beaconnachrichtengröße abhängig von der Netzwerkdichte und ist ab einer bestimmten

Netzwerkdichte nicht mehr tragbar. Der `AdaptiveDataScheduler` hängt nur eine feste Zahl der Nachbarn an, wobei die vorhandenen Nachbarn über die Beaconnachrichten verteilt werden. Ein aufwändigeres Scheduling ist auch hier denkbar. Je nach Anwendungsanforderung, können Adressen von Nachbarn die vermutlich weit weg[9] oder sehr nahe liegen bevorzugt angehangen werden. Zusammen mit der Berücksichtigung der empfangenen 2-hop Nachbarinformationen, kann die erste Variante die kritischen Fälle überprüfen, die wahrscheinlich nur unidirektional verbunden sind. Über einem Distanzschwellenwert werden Nachbarn also erst dann als bidirektional verbunden angesehen, wenn die eigene Adresse vom Nachbarn empfangen wurde. Darunter gelten alle Nachbarn als bidirektional verbunden. Die zweite Variante reduziert die Nachbarschaft auf die sicheren Nachbarn, d.h. Anwendungen interagieren dann nur mit dieser Teilmenge. Hier gelten nur die Nachbarn als bidirektional verbunden, wenn diese die Adresse des Empfängers mitgesendet haben. Schließlich ist auch eine Sendestärkenreduzierung des Beaconsignals möglich, um die Zahl der Nachbarn zu reduzieren. Der `BeaconPropagator` kann dies adaptiv anhand der Zahl der lokal beobachteten Nachbarn ermöglichen, wenn die Linklayerkomponente dies erlaubt. Somit wird auch die Wahrscheinlichkeit für bidirektionale Verbindungen im Neigbhormanagement reduziert.

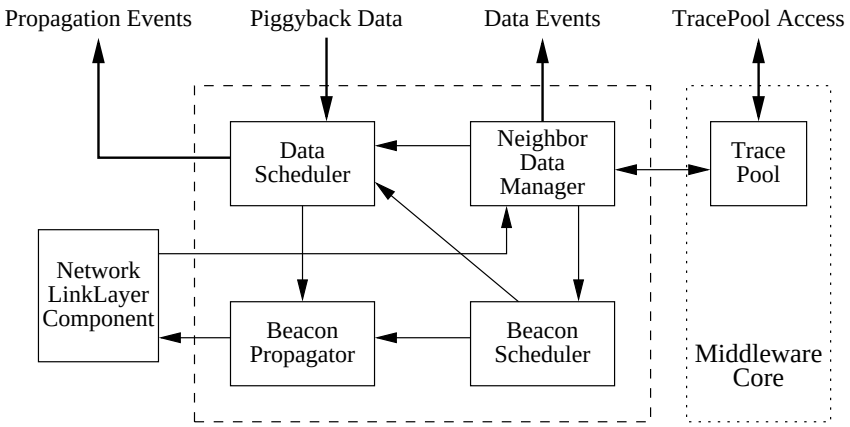


Abb. 8.13. Der `DataScheduler` verwaltet die an die Beaconnachricht anzuhängenden Daten. Der `NeighborDataManager` verwaltet die Nachbarschaftsinformationen und stellt sie in die Spurdatenbank. Der `BeaconScheduler` bestimmt die aktuelle Basisperiode des Beaconings, der `BeaconPropagator` ist für die eigentliche Übertragung zuständig.

Die Nachbarschaftsverwaltung ist somit nicht nur für das Versenden von Beaconnachrichten zuständig, sondern entscheidet auch darüber, was versen-

[9] Z.B. auf Basis der beobachten Signalstärken der Beaconsignale oder Positionsinformationen.

det wird und verwaltet die empfangenen Daten. Diese Daten werden vom `NeighborDataManager` im `TracePool` mit dem entsprechenden MobileDevice Objekt assoziiert. Die Speicherungsdauer der Informationen hängt von der Beaconingperiode des Gerätes und dem verwendeten DataScheduler ab. Somit muss der NeighborDataManager zum verwendeten DataScheduler passen, da dieser auch Daten in einer Periodenvielfachen versenden, bzw. Daten zeitweise auslassen kann. Konzeptionell ist der DataScheduler und der NeighborDataScheduler Teil des lokalen MobileDevice bzw. der Objektrepräsentanten der Nachbarn, da es sich hier um deren Spurverwaltung handelt. Realisiert sind diese Komponenten aber innerhalb eines zentralen Diensts.

8.4 Hintergrundverbreitung

Um Informationen von großem Interesse aber geringer Änderungshäufigkeit in vorgegebenen Netzregionen zu verbreiten, stellt die Middleware einen Hindergrundverbreitungsdienst zur Verfügung. Dieser basiert auf dem in [37] realisierten Verfahren des Informationsradios. Hierbei werden den Nachrichtenobjekten Gültigkeitsraum und -zeit zugeordnet. Die Objekte werden periodisch verbreitet solange deren Gültigkeiten erfüllt sind. Unabhängig von der Dichte des Netzwerks wird versucht, eine konstante Mediumsbelastung zu erreichen. Auf Basis einer adaptiven Periode wird eine beschränkte Datenmenge propagiert. Diese Basisperiode wird aufgrund des empfangenen Datenaufkommens von Nachbargeräten variiert, so dass eine maximale Mediumsnutzung nicht überschritten wird. Die Verbreitung der Datenobjekte erfolgt in zwei Phasen. Wird eine Nachricht erzeugt oder zum ersten Mal empfangen, wird sie mit der nächsten Basisperiode weiterversendet, wenn sie nicht zu häufig empfangen worden ist (So genanntes Counterbased Flooding [110]). Es wird also versucht, eine möglichst weite Verbreitung zu erreichen. In der zweiten Phase wird jede Nachricht periodisch wiederholt, wobei die Periode ausgesetzt wird, wenn die Nachricht mehrmals in der Nachbarschaft propagiert wurde. Mit dieser zweiten Phase werden zum einen Netzpartitionen überwunden, zum anderen aber auch Geräte erreicht, die eine Nachrichtenverbreitungsregion betreten. Die Verbreitung wird eingestellt oder ausgesetzt, wenn die Gültigkeitszeit abgelaufen ist, respektive der Gültigkeitsraum verlassen wird.

Empfangene Objekte werden in der Lobby allen anderen Komponenten zur Verfügung gestellt. Eine in der Praxis notwendige Einschränkung der publizierten Objekte und der Objekte bzw. Geräte, die Nachrichten erzeugen dürfen, ist nicht realisiert. Es ist aber möglich, die Verbreitung auf Informationen zu beschränken, die von bestimmten Geräten signiert wurden. Die öffentlichen Schlüssel der ausgezeichneten Geräte können mit einer allen bekannten “Master”signatur auch über das Radio verbreitet werden. Darüber hinaus ist auch eine Integration mit dem Beaconingdienst sinnvoll aber noch nicht realisiert.

8.5 Gerätерepräsentant

Jedes Gerät besitzt im lokalen Workplacepool einen Repräsentanten seiner selbst. Dieser stellt, neben lokalen Spurinformationen, auch Kommunikationsdirektiven für die Kommunikation mit der direkten Nachbarschaft zur Verfügung. Hierbei handelt es sich um eine andere Darstellung der lokalen Kommunikationsdirektiven, die vom `LinkLayer`-Objekt zur Verfügung gestellt werden. Somit repräsentiert das lokale MobileDevice Objekt auch die direkte singlehop Gerätenachbarschaft. Dieses Objekt ist auch dafür zuständig, im Workplace Objektrepräsentanten der Nachbargeräte zu erzeugen. Dies geschieht aufgrund von empfangenen Nachrichten, insbesondere der Beaconnachrichten. Repräsentanten von Nachargeräten sammeln Wissen über die repräsentierten Geräte und stellen es anderen Objekten zur Verfügung. Sie können auch zur Kommunikation verwendet werden, wobei aber nur das repräsentierte Gerät und die darauf enthaltenen Objekte adressierbar sind.

Das lokale Gerät besitzt zwei Spurtypen. Öffentliche Spurinformation werden mittels ausgehender Nachrichten verbreitet, private stehen nur Objekten auf demselben Gerät zur Verfügung. Eine Dritte Variante, die Spurinformationen auf Anfrage zur Verfügung stellt, ist denkbar, aber nicht realisiert. Öffentliche Spurinformationen werden mit Prioritäten assoziiert (siehe auch Abschnitt 8.3), die einem Vielfachen einer Propagierungsbasisperiode entsprechen. Dadurch können die Repräsentanten den empfangenen Spurinformationen Gültigkeiten zuweisen. Gerätерepräsentanten besitzen also eine Menge temporärer Informationen, die vom eigentlichen Gerät selbst verlängert werden können. Empfangene Informationen können aber auch als *permanent* markiert sein, d.h. sie sind unveränderlich und besitzen dieselbe Gültigkeit wie das Gerät. Als permanent markiert können Geräteparameter sein, die die Ressourcenmächtigkeit eines Gerätes beschreiben, aber auch Ortsinformationen, falls das Gerät stationär ist. Der Repräsentant erstellt auch selbst Informationen aus lokal beobachtetem Wissen. Somit können Distanzwerte aus Positionen und auch aus Signalstärken erstellt werden. Linkzuverlässigkeiten und Dauer der bisherigen Nachbarschaft zählen ebenfalls dazu. Repräsentanten besitzen aber auch rein lokale Spurinformationen, die von anderen lokalen Objekten zugeordnet werden. Beispielhaft seien hier die bisher empfangenen Abgleichsprofile der En-Passant Kommunikation genannt, die aufgrund der Zuordnung zum Repräsentant mit diesem automatisch gelöscht werden können.

Bleiben nach einem Vielfachen der Beaconingperiode jegliche Nachrichten aus, wird ein Repräsentantenobjekt zuerst deaktiviert und nach einem weiteren Vielfachen der Beaconingperiode gelöscht. Somit werden kurzfristige Linkabbrüche z.B. aufgrund von hoher Netzlast kompensiert. Das Löschen und das Deaktivieren können von anderen ActiveObjects verhindert werden, indem dem Repräsentanten `stay` signalisiert wird. Somit ist es möglich mit einzelnen Geräten längerfristige Bindungen einzugehen. Dies kann rein zur Informationserhaltung der Spurinformationen erfolgen oder um eine längerfristi-

ge Kommunikation zu ermöglichen. Wenn eine Bindung mit einem Gerät nicht mehr benötigt wird, kann diese durch eine `leave` Signalisierung gelöst werden. Um mehreren Objekten eine Bindung mit demselben Gerät zu ermöglichen, wird Referencecounting verwendet.

Die Kommunikationsprimitiven der Repräsentanten von Nachbargeräten dienen der Adressierung der jeweiligen Geräte bzw. darauf befindlicher Objekte. Ohne weitere Vorkehrungen ist nur eine singlehop Kommunikation möglich, d.h. eine Kommunikation schlägt fehl, wenn ein Gerät nicht in Sendereichweite ist. Somit muss für Repräsentanten, die mittels `stay` auf einem Gerät verbleiben, die Kommunikationsfähigkeit erst hergestellt werden. Eine Möglichkeit ist die Angabe einer Konfiguration für die direkte multihop Kommunikation, die eine Sicherungs- und eine Konfiguration des Routingframework enthält. Es lassen sich aber auch Kommunikationsplugins registrieren, die verwendet werden, um mit dem Gerät zu kommunizieren (Listing 8.10). Diese können auf Spurinformationen des Gerätes zugreifen und setzen bestimmte Informationstypen voraus, damit eine Kommunikation ermöglicht werden kann. Sind Plugins registriert, ist es möglich, vor der eigentlichen Kommunikation eine direkte Erreichbarkeit zu überprüfen. Hierbei wird ein einfaches “Ping” verwendet, d.h. eine Nachricht wird versandt und vom Gerät beantwortet.

Die bisher beschriebenen Mechanismen betreffen nur das aktuelle Umfeld, also die Gegenwartsspur eines Gerätes. Vergangene Informationen über Geräte können aber aus der Nachbarschaftsstatistik (siehe Abschnitt 8.8) zur Verfügung gestellt werden. Hier ist jedoch noch keine Darstellung der Informationen im Repräsentanten realisiert. Es ist aber möglich, die Statistikinformationen als Teil der Gegenwartsspur zu modellieren. Das bedeutet, dass Spurinformationen hinzugefügt werden, die auch Informationen der Vergangenheit beinhalten. Dies gilt auch für Informationen über die Zukunft wie Stundenpläne oder Abschätzungen aufgrund der Nachbarschaftsstatistik. Dabei geht allerdings eine zeitlich korrekte Darstellung verloren bzw. die zeitliche Darstellung ist im Objekt selbst und nicht in der gesamten Spurrepräsentierung enthalten.

8.6 Repräsentant einer lokalen Gruppe

Der Repräsentant für lokale Gruppen ermöglicht nur die Interaktion zwischen Gruppenmitgliedern, die sich in direkter singlehop Reichweite befinden. Ein Broadcast an die Gruppe entspricht also einem singlehop Multicast auf Netzwerkebene. Ein Unicast an Geräte wiederum adressiert nur Objekte, die Teil der Gruppe sind. Verlässt also ein Gerät die Gruppe, aber nicht den Sendebereich, schlägt die Kommunikation über den Repräsentanten fehl. Über die Zeit kann dennoch eine Store-and-Forward Kommunikation zwischen allen Gruppenmitgliedern stattfinden wie in Abbildung 8.14 dargestellt. Gruppenobjekte erweitern die öffentliche Gerätespur um die Gruppenkennung, so dass eine

Listing 8.10 In diesem Kodebeispiel wird jedem Gerät eine `HelloMessage` zugeschickt, sobald es benachbart ist. Bei allen Geräten, die angeben, stationär zu sein, soll der Repräsentant lokal verbleiben und die Kommunikation mittels positionsbasiertem Routing ermöglicht werden, sobald das Gerät die direkte Nachbarschaft verlässt. Das Plugin greift auf die Spurinformation des Gerätes zu, benötigt hierbei die `PositioningData` Information und versucht, Nachrichten an diese Positionen zu senden.

```
1  //Registriere einen Ereignishandler. Dieser wird aufgerufen, wenn
2  //das Ereignis DeviceRepresentativeEventNeighboring von einem
3  //beliebigen (null) Gerätepräsentanten gesendet wird, das Gerät sich
4  //also in direkter Nachbarschaft befindet.
5  mw.addEventListener(new DeviceRepresentativeEventNeighboring(null),
6      new DeviceRepresantativeEventNeighboringHandler(){

8      //Ein neues Gerät ist in der Nachbarschaft.
9      void handleNeighboring(MobileObjectID objectID){
10         //Ein Signalproxy des Gerätepräsentanten wird
11         //von der Middleware erzeugt
12         DeviceRepresentative neighbor=(DeviceRepresentative)
13             mw.getSignalStub(objectID, DeviceRepresentative.class);

15         //Sende eine Nachricht an das Nachbargerät
16         neighbor.receiveOne(new HelloMessage("Say Hello"),
17                             /*StatusHandling*/);

19         //Die Positionsinformation des Gerätes wird aus dem
20         //Spurpool der Middleware abgerufen
21         PositioningData data=(PositioningData)
22             mw.getTracePool().getTraceInfo(
23                 objectID,
24                 PositioningData.DATA_ID);

26         //Ist die Spurinformation Position unveränderlich,
27         //kann der Repräsentant lokal verbleiben und
28         //an die Geräteposition Nachrichten gesendet werden
29         if (dara.isFixedPosition()){
30                 neighbor.stay();
31                 neighbor.addRemoteCommunicationPlugin(
32                 new PostionbasedRoutingPlugin());
33         }
34     }
35 });
```

frühzeitige Erkennung von anderen Geräten der selben Gruppe möglich ist. Lokal werden die sichtbaren Gruppenmitglieder verwaltet und eine Historie der zuletzt getroffenen Mitglieder geführt.

Auf Basis lokaler Gruppen lässt sich die En-Passant Gruppenkommunikation realisieren, wie in [44] dargestellt. Dem Repräsentanten der Gruppe können spezielle En-Passant Synchronisationsobjekte übergeben werden, die den Austausch von Objekten, die der Gruppe zugehörig sind, realisieren. Das auslösende Ereignis bei dieser spezielleren En-Passant Variante ist immer das Erscheinen eines Gerätes der Gruppe in der direkten Nachbarschaft. Empfangene Synchronisationsobjekte werden gespeichert, auch über die Nachbarschaft hinaus, solange die Nachbarschaftsstatistik das Gerät beinhaltet.

Listing 8.11 Auszug aus den Methoden des Gerätерepräsentanten. Hierbei sind Methoden mit Rückgabewerten synchron, alle anderen asynchron. In der Implementierung sind synchrone und asynchrone Methoden in je einem Interfaces definiert.

```
4  //Repräsentant bleibt erhalten.
5  public void stay();
6  //Repräsentant darf automatisch verschwinden
7  public void leave();


10 //Sende ein Objekt an das Gerät
11 public void receiveOne(SendableObject so);
12 public void receiveOne(SendableObject so, ReceiveHandler handler);
13 //Verwende die Kommunikationskonfiguration,
14 // falls das Gerät nicht lokal erreichbar ist.
15 public void receiveOne(SendableObject so,
16     RoutingConfiguration conf, ReceiveHandler handler);


19 public void addRemoteCommunicationPlugin(
20     RemoteCommunicationObject rco);
21 public boolean isAddressable();
22 public boolean isSinglehopNeighbor();

24 //Zugriff auf Spurdaten
25 public boolean hasData(DataID id);
26 public TraceData getData(DataID id);
27 public boolean hasData(Class dataClass);
28 public DataID[] getData(Class dataClass);
29 public boolean isFixedTraceData(DataID id);
30 (...)

32 //lokale Spurdaten
33 public void addLocalData(TraceData data);
34 //properties: validity, public/private, fixed
35 public void addLocalData(TraceData data, DataProperties properties);
36 public void addRemoteData(TraceData data, SuccessHandler handler);
37 (...)
```

Ein registriertes Synchronisationsobjekt ist für einen Objekttyp zuständig (in der Realisierung durch Angabe der Javaklasse). Alle berücksichtigten Objekttypen einer Gruppe werden in einem speziellen altruistischen Gruppenrepräsentanten registriert. Dieser führt einen altruistischen En-Passant Abgleich durch. Dabei werden nur Objekte berücksichtigt, für die lokal kein Gruppenrepräsentant existiert und kein spezielles Synchronisationsobjekt registriert ist. Die betrachteten Objekte müssen dabei aus einer Gruppe stammen, aus der häufig Geräte getroffen werden. Diese Information wird aus der Nachbarschaftsstatistik gewonnen, wobei hier nicht die Geräte-, sondern die Gruppenkennungen betrachtet werden. Der altruistische Gruppenrepräsentant verwaltet eine feste Anzahl von Objekten in der Lobby, die von ihm periodisch verlängert werden. Dabei wird versucht, zum einen möglichst aktuelle Objekte altruistisch vorzuhalten, zum anderen die Objektzahl nach der Häufigkeit

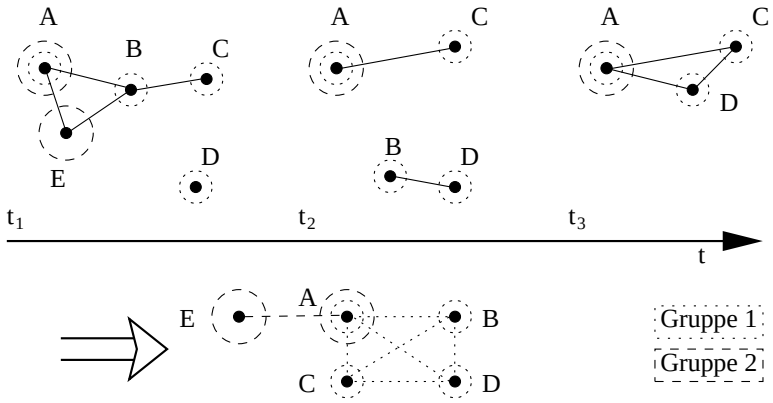


Abb. 8.14. Lokale Gruppen erlauben nur eine singlehop Interaktion. Zum Zeitpunkt t_1 ist auf Basis der Gruppe 1 keine direkte Interaktion zwischen A und C möglich, und D ist vom Rest der Gruppe 1 nicht erreichbar. Über die Zeit (t_1, t_2, t_3) kann ein Gruppencluster entstehen (unten), ohne das die Gruppe zu einem Zeitpunkt zusammenhängend gewesen sein muss. So kann innerhalb der Gruppe Nachrichten per Store-and-Forward verbreitet werden. Die En-Passant Kommunikation auf Basis von Gruppen setzt hier an. Nur Gerät A kann mittels des lokalen Repräsentanten der Gruppe 2 mit Gerät E interagieren, für Gruppe 1 ist E nicht sichtbar. Dadurch entsteht über die Zeit ein Overlaynetz, wobei die Kanten auf Basis der Gruppenzugehörigkeit und tatsächlichen Begegnungen definiert sind. Gerät A ist hierbei ein Hub zwischen Gruppe 1 und Gruppe 2.

der Gruppenbegegnungen zu relativieren. Somit sind für die Gruppe, die am wahrscheinlichsten getroffen wird, die meisten Objekte lokal vorhanden. Hierbei werden auch Objekte berücksichtigt, die nicht explizit, sondern implizit aufgrund der Broadcasteigenschaft des Mediums empfangen wurden. Datenobjekte müssen mit einer passenden Gruppenkennung in ihrer Spur versehen sein, um berücksichtigt zu werden.

8.7 Positionierungs- und Lokalisierungsdienste

Ortsinformation kann in der Middleware auf zwei Arten zur Verfügung stehen: zum einen als Position in einem globalen Koordinatensystem, zum anderen als abstrakter Ort, wobei nur entscheidbar ist, ob ein Gerät dort ist oder nicht. Da diese Informationen meist Gerätehardwareabhängig sind, existieren unterschiedliche Realisierungen, die dieselbe Funktionalität liefern. Als Positionsquelle auf einem Gerät existieren simulationsspezifische Implementierungen, die exakte Simulationspositionen liefern oder randomisiert die exakte Simulationsposition verfälschen. Für die Middleware auf realen Geräten existiert eine GPS-Anbindung. Allen Positionsdiensten ist gemein, dass sie entweder

periodisch Positionsereignisse erzeugen oder nur Ereignisse generieren, wenn die Position sich in einem gewissen Maße verändert hat. Im Falle der GPS-Implementierung ist die Periode durch die Hardware mit einer Sekunde vorgegeben.

Der abstrakte Ort kann als Fläche, gegeben durch Positionen im globalen Koordinatensystem, oder auf Basis einer Ortskennung definiert sein. Diese Kennung wird auf Basis eines speziellen Lokationssystems wie z.B. Funk- oder Infrarotbeacons realisiert. Beacons sind in diesem Kontext Geräte, die fest installiert sind und mit Hilfe einer Kommunikationstechnik, die unter Umständen eine andere ist als die des verwendeten Netzwerks, periodisch eine global eindeutige Kennung aussenden. Somit kann ein Ort als Kommunikationsreichweite eines oder mehrerer solcher Beacons erkannt werden. Ein durch Beacons bestimmter Ort kann durch Kommunikationsaufwand vergrößert werden, so dass die Reichweite des Beacons erhöht wird, z.B. kann die Existenz eines Beacons über eine feste Anzahl von Knoten im Netz propagiert werden. Informationen über abstrakte Orte werden über einen Lokalisierungsdienst propagiert und können dort abgefragt werden. Andere Objekte können Ereignisroutinen für Orte registrieren und somit über das Betreten und Verlassen eines Ortes informiert werden. Je nach Realisierung ist es möglich, dass immer Ereignisse erzeugt werden, wenn identifizierbare Orte betreten oder verlassen werden. Werden Positionsinformationen als Ortsbasis verwendet, müssen dem Dienst die Orte bekannt sein, um Ereignisse zu erzeugen. Hierfür können Ortslisten vorgegeben werden, die zur Laufzeit veränderbar sind, oder es kann ein Raster vorgegeben werden, mit dem die Ebene in Quadrate unterteilt wird.

8.8 Objektnachbarschaftsstatistik

Diese Komponente verwaltet die Statistik über vergangene Begegnungen mit anderen Objekten. Dazu können die Nachbarschaftsereignisse beliebiger Objekte mitgeteilt werden, wobei nur die Objektkennungen und keine konkreten Objekttypen berücksichtigt werden. Somit muss für jeden Objekttyp (Gerät, Gruppe, Ortskontext,...) eine eigene Komponente erzeugt werden. Die Komponente fasst die Begegnungsereignisse der am häufigsten auftretenden Objektkennungen zusammen (siehe Box 5.9). Konfiguriert werden kann die Dauer der einzelnen Phasen, die Zahl der Phasen und die Zahl der Kennungen pro Phase. Abgefragt werden kann dann für vorgegebene Zeiträume die Liste der gespeicherten Kennungen samt der Statistik. Da hierbei die Zeiten unterschiedliche Auflösungen besitzen können (Stunden, Tage, Wochen,...), kann der geforderte Zeitraum nicht exakt geliefert werden. Es kann nur eine Annäherung gegeben werden, die auf dem internen Zeitgranulat basiert. Eine weiterführende Aufbereitung der Statistik, insbesondere zur Begegnungsvorhersage, ist möglich, wurde bisher aber nicht untersucht.

8.9 Unterstützung für MobileState

Die vorgestellte Middleware bietet zurzeit keine Unterstützung für MobileState Objekte. Die Grundlagen zur Unterstützung von MobileStates wurden in der auf die marktplatzbasierte Kommunikation spezialisierte Middleware SELMA realisiert und in [46] vorgestellt. Ein Aspekt dieser Middleware ist die Migrationsunterstützung für MobileStates. Hierbei werden in der Middleware registrierte Ereignisbehandlungsroutinen automatisch migriert, so dass der Migrationsaufwand auf Anwendungsebene reduziert wird. Die Migrationsunterstützung kann in dieser Form auch in der in dieser Arbeit vorgestellten Middleware realisiert werden. Allerdings sind die Möglichkeiten der Registrierung von Ereignisbehandlungsroutinen in dieser Middleware umfangreicher und somit ist die Implementierung der Migration aufwändiger. Darüber hinaus müssen auch Signalproxyobjekte migriert werden. Dies ist möglich, wenn die Kennung des zugehörigen Objektes auf allen Migrationszielen gleich ist. Diese speziellen, von der Middleware generierten Objekte könnten vor der Objektserialisierung automatisch aus dem Objekt entfernt werden und auf dem Zielgerät wieder erzeugt werden, bevor das Objekt reaktiviert wird. Anstelle des Signalproxies wird in der Serialisierung nur die Kennung des zu adressierenden Objektes gespeichert. Die Migration con MobileStates muss also Teil des Middlewarekerns sein und kann nicht von einem Zusatzdienst durchgeführt werden.

Listing 8.12 Interface eines MobileStates. Zusätzlich zum ActiveObject besitzt ein MobileState die Methoden `handleSuspend` und `handleResume` um die Migration zu signalisieren. Die übergebene Middleware muss ein angepasstes Interface besitzen, um nur serialisierbare Ereignisroutinen registrieren zu können. `handlePossibleDuplication` ist die Methode, die von der Objektmigrierungskomponente der En-Passant Kommunikation aufgerufen wird.

```
2  public void handleStart(MSMiddleware ms);

4  public void handleSuspend();

6  public void handleResume(MSMiddleware ms);

8  public void handleFinish();

10 public void handlePossibleDuplication(DuplicationHint hint);
```

SELMA integriert Migrationsinformationen in ein generisches Basisobjekt, das auch für die positionsbasierte Bewegung und für das Migrationsprotokoll des MobileStates zuständig ist. Eine Trennung wäre an dieser Stelle besser, d.h. die Kommunikationslogik sollte nicht notwendigerweise enthalten sein. So kann diese für unterschiedliche Objekte auch unterschiedlich realisiert sein. Eine Realisierung der marktplatzbasierten Kommunikation würde

somit auf En-Passant Triggern, der Objektmigration von MobileStates und einem positionsbasierten Greedyroutingalgorithmus basieren. Eine Erweiterung des ursprünglichen reinem Store-and-Forward Konzeptes könnte zuerst direkte positionsbasierte Kommunikation verwenden, um erst im Fehlerfall auf Store-and-Forward Kommunikation zurückzufallen. Die Kommunikationsrealisierung würde auf dem Kommunikationsframework der Middleware basieren und nicht wie bei SELMA Teil des Middlewarekerns sein.

9

Applikationsstudie

Auf Basis des Middlewareprototyps wurde eine Applikation für das universitäre Umfeld entworfen. Diese Applikation nutzt den Kontext einer Vorlesung, um eine höhere Kommunikationszuverlässigkeit zu ermöglichen. Die meisten Teilnehmer befinden sich hierbei an einem Ort und können untereinander mit geringem Aufwand interagieren, da die Distanz der Teilnehmer nicht groß ist. Entfernte Teilnehmer können diesen bekannten, geographischen Ort mit einem ortsbasierten Anycast erreichen, um dann in einem zweiten Schritt dieselben Verfahren einzusetzen wie vor Ort. In der Applikation werden dafür Positionsinformationen vorausgesetzt. Prinzipiell ist innerhalb eines Veranstaltungsortes, also eines Hörsaals oder eines Seminarraums, ein vollständig verbundenes Netz zu erwarten. Allerdings steigt mit der Zahl der kommunizierenden Geräte auch der Koordinationsaufwand und die Wahrscheinlichkeit für Kollisionen. Darüber hinaus sinkt die effektive Datenrate, die für jedes Gerät zur Verfügung steht, da sich alle Geräte das drahtlose Medium teilen müssen. Bei einer gleichverteilten Gerätemenge beschreiben [59, 104] einen Knotengrad von sechs bzw. acht als optimal. Somit kann, je nach Anwendung und ihrem Kommunikationsverhalten, eine Reduzierung der Sendestärke und somit auch innerhalb von Räumen multihop Kommunikation sinnvoll sein. Während einer Veranstaltung ist ein dichtes Netz zu erwarten, so dass die Broadcasteigenschaft des Mediums ausgenutzt werden sollte, um die Netzwerklast stark zu reduzieren. Auch die auf einer Veranstaltung definierten Applikationen sollten dies berücksichtigen. Somit ist es auch in diesem, eigentlich idealem, Szenario notwendig, dass die Applikation die Kommunikationsmechanismen der Middleware adaptieren kann. Das Wissen über den Ort ist also notwendig, um das Kommunikationsverhalten entsprechend anpassen zu können. Darüber hinaus ist eine Applikation in der Lage, ihr Verhalten aufgrund des Wissens um diesen speziellen Ort anzupassen. So können vor Ort andere Funktionalitäten zur Verfügung stehen als außerhalb. Die Modellierung der Veranstaltung als Ortskontext (Abbildung 9.1) ist eine Möglichkeit, dies zu realisieren.

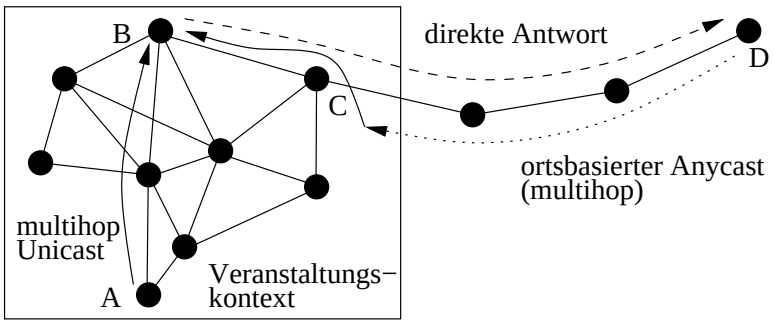


Abb. 9.1. Der Ortskontext einer Veranstaltung. Die meisten Geräte befinden sich an einem Ort. Kommunikation mit Gerät B ist vor Ort mit ein oder zwei Hops möglich (z.B. von A). Geräte außerhalb (D) können den stationären Ort adressieren und dann vor Ort ein beliebiges (z.B. C), alle, oder ein bestimmtes Gerät (B) adressieren. Hierbei ist, zumindest für kurze Zeit, eine direkte Antwort möglich.

Die Applikationsidee von Distributed Script (DistScript) wird in Abschnitt 9.1 skizziert und die von der Applikation verwendeten und verbreiteten Datentypen in Abschnitt 9.2 vorgestellt. Die Applikation basiert auf einem Ortskontext, der durch die Orte und Zeiten einer Veranstaltung definiert ist. Der Repräsentant und die Aufgaben des Veranstaltungskontextes wird in Abschnitt 9.3 beschrieben. Dieser bietet direkte Kommunikationsmechanismen (Abschnitt 9.4) und Store-and-Forward Kommunikation (Abschnitt 9.5) an, mit der die Applikation Veranstaltungsmaterial in einem ersten Schritt sofort an alle erreichbaren Teilnehmer propagiert und zusätzlich verpasstes Material mit direkten Nachbarn austauscht. Um die Überarbeitung von Veranstaltungsmaterial zu ermöglichen, wird ein Tokendienst eingesetzt, der in Abschnitt 9.6 beschrieben wird. Abschließend wird das Applikations-GUI (Abschnitt 9.7) und die prototypische Realisierung (Abschnitt 9.8) vorgestellt.

9.1 Anwendungsbeschreibung

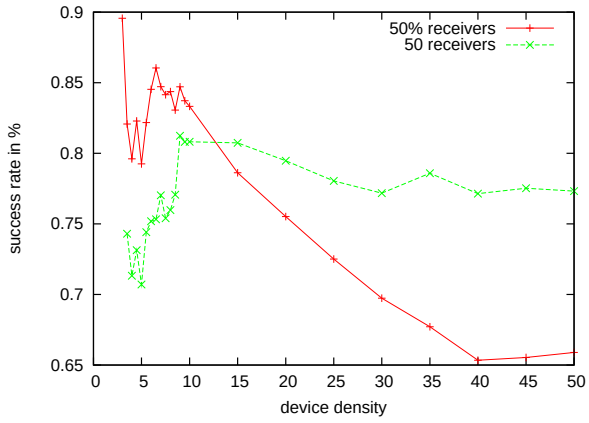
Ziel der Applikation *DistScript* ist es, dass Studierende im Kontext einer Veranstaltung gemeinschaftlich ein Skript zu dieser Veranstaltung erstellen. Dazu werden vom Dozenten der Veranstaltung die Folien zu den einzelnen Vorlesungen am Anfang der Veranstaltung zur Verfügung gestellt. Somit ist es möglich, auf Basis des Foliengranulats individuelle Anmerkungen zu einzelnen Folien zu erstellen. Mit Folien können auch Fragen assoziiert werden, die von den anderen Studierenden beantwortet werden können. Diese, mit den Folien assoziierten Informationen können überarbeitet oder in einem zweiten Schritt zusammengefasst werden, so dass mit der Zeit ein Skript zur Vorlesung entstehen kann. Das Überarbeiten verursacht Konsistenzprobleme, die

aber im Kontext einer Vorlesung gelöst werden können. Gemeinschaftlich kann eine “offizielle” Version einer Information erzeugt werden, da ein Großteil der Studierenden sich an einem Ort befindet und ihre Geräte mit hoher Zuverlässigkeit kommunizieren können. Ein Überarbeiten ist auch außerhalb des Vorlesungskontextes möglich. Hier kann es ebenfalls zu Inkonsistenzen kommen, die aber vom Nutzer selbst aufgelöst werden müssen. Konsistente Versionen können nur im Kontext der Veranstaltung erzeugt werden. Neu erzeugte konsistente Informationen und Folien werden im Kontext der Veranstaltung verbreitet, um möglichst viele Studierende zu erreichen. Dennoch ist die Menge des Veranstaltungsmaterials eines Studierenden häufig nicht vollständig. Aufgrund der Unzuverlässigkeit des drahtlosen Mediums ist nur mit hohem Aufwand eine zuverlässige Informationsverbreitung an eine große Gerätemenge möglich. Darüber hinaus sind aufgrund von Netzpartitionen nicht immer alle Geräte erreichbar, selbst wenn diese sich in der Nähe der Veranstaltung befinden. Schließlich werden Informationsverbreitungen verpasst, wenn Studierende verspätet erscheinen, der Veranstaltung neu beitreten oder das Gerät zu einem späteren Zeitpunkt einschalten. Somit muss ein Mechanismus existieren, um fehlendes Material zu erhalten – sowohl während als auch außerhalb einer Veranstaltung. Dieser Austausch geschieht auf Basis der En-Passant Kommunikation.

Obwohl es möglich ist, ist eine gegenseitige Erreichbarkeit aller Geräte vor Ort nicht immer erwünscht. So kann eine Beschränkung der Sendestärke bei der Kommunikation mit sehr nahen Nachbarn die Zahl der dadurch gestörten Geräte stark reduzieren. Anders ausgedrückt berücksichtigt die Applikation dies bei der Wahl des Kommunikationspartners und die Sendestärke kann entsprechend reduziert werden. Fehlendes Material wird daher bei dem kommunikationstechnisch günstigsten Nachbarn angefordert. Verbreitung neuen Materials aber kann mit höherer Sendestärke und unter Nutzung der Broadcasteigenschaft des Mediums erfolgen, so dass möglichst viele Geräte mit einem Kommunikationsschritt erreicht werden. Dennoch ist hier der Einsatz eines multihop Verfahrens sinnvoll, da nicht generell eine gegenseitige Erreichbarkeit möglich ist und auch Geräte außerhalb der Veranstaltung erreicht werden können. Darüber hinaus ermöglicht eine Sendestärkenreduzierung eine Verringerung des Energiebedarfs der Geräte. Damit eine konsistente Sicht auf erstelltes oder überarbeitetes Material erreicht werden kann, erhält in diesem Szenario ein Gerät eine ausgezeichnete Rolle um einen wechselseitigen Ausschluss zu ermöglichen. Hier bietet sich die Nutzung des Dozentengerätes an. Dieses Gerät ist in jeder Veranstaltung anwesend und muss initial die Veranstaltungsfolien verteilen. Dadurch ist es den anderen Geräten bekannt und besitzt alle Vortragsfolien. Mit diesem Gerät wird als einziges per multihop Unicast innerhalb des gesamten Ortskontexts kommuniziert. Prinzipiell kann diese Rolle aber von allen Geräten übernommen werden, wenn das lokal vorhandene Veranstaltungsmaterial vollständig ist.

9.1 Zuverlässigkeit von SPBM

SPBM nutz positionsbasierte Anycastverfahren und “erbt” somit deren Probleme. Da Greedy auch unter idealisierten Bedingungen keine Auslieferung garantieren kann, muss auch SPBM Planargraphenroutingverfahren als Rückfallstrategie einsetzen. So kann SPBM die Auslieferung an alle Geräte einer Gruppe in einer zusammenhängenden Netzpartition unter idealisierten Bedingungen garantieren (idealisierte MAC-Schicht, Unitdiscgraph, exakte Positionen, keine Mobilität). Wird keine idealisierte MAC-Schicht verwendet, kann es aufgrund von Kollisionen zu kurzzeitigen Linkabbrüchen kommen, so dass ein Teil des Multicastbaums unterbrochen wird. Ist der abgebrochene Link die einzige mögliche Verbindung eines Teilbaums, führt dies dazu, dass alle Geräte, die hinter diesem Link liegen, nicht erreicht werden. Um dies simulativ zu zeigen, wurden mehrere stationäre Netze der Fläche $500m$ x $500m$ mit steigender Netzdichte (im Schnitt 3 – 50 Nachbarknoten; 32 – 410 Knoten) untersucht. Die Geräte verfügen über einen simulierten 802.11MAC mit 11MBit und einer Reichweite von ca. $100m$. Aus den Geräten treten zufällig 50% (16 – 205) bzw. 50 Geräte (bei Dichte < 5.5 alle Geräte) der Multicastgruppe bei und eines dieser Geräte sendet in einem Simulationslauf 20 Nachrichten an die Gruppe. Jede Parameterbelegung wurde 40 mal wiederholt. Die Auslieferungsraten für beide Varianten sind in den Graphiken über der Gerätedichte abgetragen. Mit zunehmender Dichte nimmt die Erfolgsrate in beiden Varianten ab, da die Wahrscheinlichkeit für Nachrichtenkollisionen steigt. Bei steigender Empfängerzahl ist die Abnahme deutlicher, da auch die Zahl der auszuliefernden Nachrichten steigt. Die Einbrüche bei geringer Dichte sind durch die geringere Zahl der Alternativpfade zu erklären. Hier führt der Wegfall eines Pfades aufgrund wiederholter Kollision häufiger zu einer Partition des Netzes. Darüber hinaus werden bei einer Dichte unter 5.5 bei konstanter Empfängerzahl alle Geräte adressiert. Insgesamt liegt die Erfolgsrate deutlich unter 90%, so dass es notwendig wird, verbreitete Informationen zusätzlich auch über Alternativverfahren zu verbreiten.



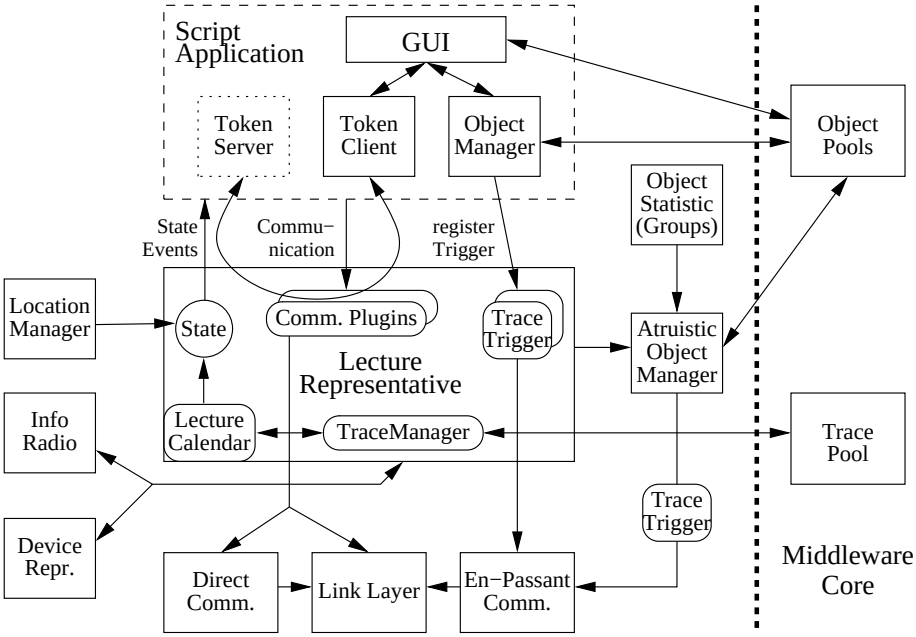


Abb. 9.2. Architekturbild der Scriptapplikation. Zur Applikation selbst gehören nur die GUI, die Tokendienste und das Objektmanagement. Der `ObjektManager` registriert `TraceTrigger` und verschiebt empfangenes Veranstaltungsmaterial in den Residence Pool. Die GUI bezieht das Material direkt aus den Pools. Die direkte Applikationskommunikation erfolgt über den Veranstaltungsrepräsentanten, der hierfür je nach Kommunikationsparadigma Plugins besitzt. Die Plugins verwenden direkte singlehop und multihop Kommunikation. Der aktuelle Zustand des Repräsentanten wird durch Ereignisse eines Kalenders und des `LocationManager`s bestimmt. Spurinformationen werden über das Informationsradio, den lokalen Gerätepräsentanten und den lokalen Spurpool verbreitet bzw. empfangen. Das altruistische Objektmanagement tauscht unabhängig von der Applikation Objekte aus.

Abbildung 9.2 stellt das Architekturbild der Applikation, des Repräsentanten des Veranstaltungskontexts und die von ihr verwendeten Middlewarekomponenten dar.

9.2 Informationsorganisation

Datenobjekte der Applikation werden als MessageObjects im Residencepool der Middleware abgelegt. Folien (`ImageSlide`) stellen einzelne Datenobjekte dar, so dass der Umfang eines einzelnen Objektes gering bleibt. Der

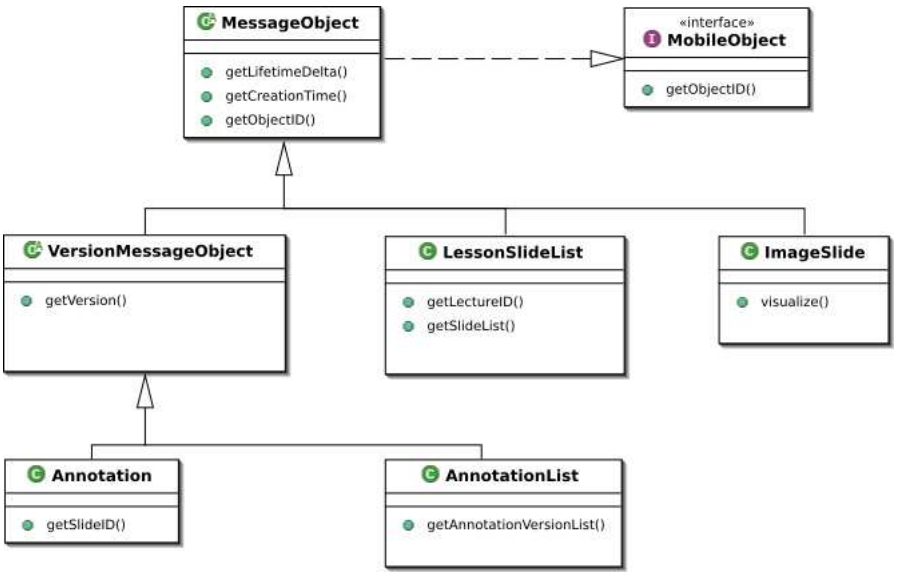


Abb. 9.3. Distributed Script Datenobjekte

Aufbau der Folienserie einer einzelnen Vorlesung ist in dem gesonderten Datenobjekt `LessonSlideList` abgelegt. Somit ist auch lokal feststellbar, ob Folien einer Serie fehlen. Zu einer Folie können eine oder mehrere Annotationen (`Annotation`) existieren. Jede Annotation ist durch die Folienkennung und eine laufende Nummer eindeutig gekennzeichnet und besitzt zusätzlich eine Versionsnummer. Gültige Annotationsversionen zu einer Folie sind in der Liste `AnnotationList` gespeichert, so dass auch ein Löschen einer Annotation (z.B. durch Zusammenfassen) möglich ist. Diese Liste ist auch versioniert. Alle Objekte können einer oder mehreren Veranstaltungen bzw. Gruppen zugeordnet werden. Für diese Zuordnung wird die Veranstaltungskennung zur öffentlichen Objektspur hinzugefügt und mit übertragen. Aufgrund der Gruppenkennungen kann ein altruistischer Austausch der Datenobjekte erfolgen, ohne dass der Inhalt des Objektes bekannt sein muss.

Eine eigene Objektmanagerkomponente der Applikation kümmert sich um die der Applikation zugeordneten Veranstaltungsmaterialien. Sie verschiebt empfangene Objekte der Applikation von der Lobby in den Residencepool. Für alle Objekttypen werden Ereignisroutinen für das Erscheinen in der Lobby registriert. Es werden aber nur Objekte verschoben, die in ihrer Spur auch die entsprechende Veranstaltungskennung besitzen. Die Objektmanagerkomponente registriert auch die zu den Objekten gehörenden Tracetrigger (siehe Abschnitt 9.5).

9.3 Veranstaltungskontext

Für die Applikation ist es wichtig zu wissen, ob ein Gerät sich im Kontext einer Veranstaltung befindet. Erst dann ist es möglich, konsistentes Vorlesungsmaterial zu erzeugen und mit geringem Aufwand viele Geräte zu erreichen. Das Wissen um Orte und Zeiten der Veranstaltung vorausgesetzt kann ein Gerät erkennen, ob es sich an einem Ort aufhält, an dem eine Vorlesung der Veranstaltung stattfindet, oder nicht. Verbreitet wird dieses Wissen mittels des Informationsradios (Abschnitt 8.4) in einer beschränkten Region, beispielsweise das Fachbereichsgebäude, in dem die Veranstaltung stattfindet. Interessierte Studierende können aus diesen Informationen einen lokalen *Veranstaltungsrepräsentanten* erzeugen und der Veranstaltung beitreten. Eine zweite Beitrittsvariante ist das Erzeugen von direkt benachbarten Geräten. Geräte propagieren die Kennungen der Veranstaltungskontexte in denen sie sich befinden und können dadurch bei Nachbargeräten einen Beitritt auslösen. Die zum Erzeugen des Repräsentanten fehlende Veranstaltungsinformation wird in diesem Fall vom benachbarten Gerät angefordert. Der Beitritt erfolgt automatisch, falls der Studierende schon vor dem Treffen seinen Beitrittswillen zu einer Veranstaltung mitgeteilt hat. Ein Studierender kann aber auch die Veranstaltungsinformationen seiner Nachbarn sichten und beitreten, solange das entsprechende Geräte benachbart ist. Der Veranstaltungsrepräsentant kann unter Angabe eines Nachbargerätes, das im Besitzt von Veranstaltungsinformationen ist, sich selbst aufgrund eines Factorymethodenaufrufs instanziieren. Das Sichten von Veranstaltungen und das Beauftragen eines Beitritts sind nicht Teil des Repräsentanten. Diese Funktionalität ist in einer applikationsunabhängigen Komponente realisiert, da der Veranstaltungskontext auch für weitere Applikationen eingesetzt werden kann.

Die Aufgaben des Veranstaltungsrepräsentanten sind die Kommunikation mit anderen Geräten und die Verwaltung von veranstaltungsspezifischen Spurinformationen. Der Repräsentant unterscheidet drei Gerätezustände, die als Ereignisse an andere Objekte auf einem Gerät propagiert werden:

1. INSIDELESSON: Das Gerät befindet sich zur Veranstaltungszeit an einem gültigen Veranstaltungsort.
2. OUTSIDELESSON: Das Gerät befindet sich zur Veranstaltungszeit nicht an einem gültigen Veranstaltungsort.
3. NOLESSON: Zurzeit findet keine Veranstaltung statt.

Abbildung 9.4 stellt den entsprechenden Zustandsautomaten dar. Die Zustände werden zur Kommunikationsadaption verwendet. Prinzipiell kann mit dem Wissen des Veranstaltungskontextes und abhängig von dem jeweiligen Zustand unterschiedlich kommuniziert werden:

1. INSIDELESSON: Andere Geräte können in diesem Zustand mit hoher Zuverlässigkeit erreicht werden.

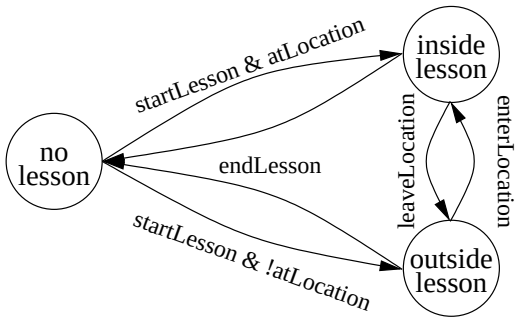


Abb. 9.4. Der Zustandsautomat des Veranstaltungsrepräsentanten zur Spurverwaltung und Kommunikationsanpassung. Nach jedem Zustandsübergang wird der Zustand als Ereignis propagiert. Die Übergänge erfolgen aufgrund von Ereignissen des Locationmanagers oder des Kalenders der Veranstaltung.

2. OUTSIDELESSON: Geräte im Zustand INSIDELESSON können aufgrund des Wissens um ihren Aufenthaltsort erreicht werden.
3. NOLESSON: Nur mit direkt benachbarten Geräten des Veranstaltungskontextes kann interagiert werden.

Zusätzlich ist es möglich, dass ein Gerät A im Zustand INSIDELESSON ein Gerät B im Zustand OUTSIDELESSON aufgrund einer vorherigen Kommunikation von B nach A adressieren kann, da hierbei ein Kommunikationspfad A nach B etabliert werden kann (siehe auch Abbildung 9.1)

Die Spurverwaltung des Repräsentanten unterscheidet vier Spurinformationsklassen, die auf unterschiedlichen Wegen verbreitet und empfangen werden.

1. *Globale Spurinformation.* Diese sind in allen Repräsentanten enthalten und entstammen dem Informationsradio. Somit ist ein Empfang neuer Information nur in der geographischen Reichweite der Information möglich. Nur ausgezeichnete Geräte dürfen eine globale Spurinformation erzeugen. Eine Weiterverbreitung über den Verbreitungsbereich hinaus findet nur zwischen Geräten der Veranstaltung statt. Somit hängt die Globalität der Information vom zeitlichen Zusammenhang des Netzes innerhalb des Radiobereichs und vom zeitlichen Zusammenhang der Teilnehmer ab. Veranstaltungsorte und Zeiten sind globale Spurinformation.
2. *Ortsglobale Spurinformation.* Diese sind in allen Repräsentanten im Zustand INSIDELESSON enthalten und nur zur Veranstaltungszeit gültig. Die initiale Verbreitung erfolgt an alle Geräte im Zustand INSIDELESSON und wird vom Urheber periodisch erneuert oder an veranstaltungsweite Nachrichten angehangen. Die ortsglobalen Spurinformationen können auch auf Geräten im Zustand OUTSIDELESSON vorhanden sein, falls Nachrichten

von innerhalb der Veranstaltung dorthin propagiert wurden. Die Kennung des Dozentengerätes sind ortsglobale Spurinformationen.

3. *Kommunikationslokale Spurinformation.* Informationen sind in Geräten enthalten, die mit dem Urheber direkt benachbart sind. Initial wird diese Information an alle Nachbarn verbreitet und vom Urheber periodisch erneuert. Die Verbreitung und Erneuerung erfolgt auf Basis des Neighbordiscovery (Abschnitt 8.3). Somit werden die Informationen je nach Priorität an ausgehende Nachrichten angehangen. Diese Informationsklasse wird in weitere Teilklassen unterteilt: Informationen, die (1) nur im Zustand INSIDELESSON, (2) nur außerhalb des Ortes in den Zuständen OUTSIDELESSON und NOLESSON oder (3) in allen Zuständen verbreitet werden. Distributed Script verwendet diese Informationen um Hinweise zum En-Passant Abgleich des Veranstaltungsmaterials zu geben.
4. *Lokale Spurinformation.* Diese Informationen stehen nur den Objekten auf einem Gerät zur Verfügung. Sie werden also nicht aktiv an andere Objekte propagiert. Erkannte, direkt benachbarte Geräte derselben Veranstaltung sind beispielsweise lokale Spurinformation.

Der Repräsentant selbst stellt die Veranstaltungsinformationen, also die Orte, Zeiten und Themen der einzelnen Vorlesungen und den Veranstaltungsnamen zur Verfügung. Orte und Zeiten sind in einem Kalender eingetragen. Ist eine Vorlesung aktiv, ist die Information der laufenden Veranstaltung im Spurpool zugreifbar. Zur Verfügung gestellt werden auch die im Veranstaltungskontext benachbarten Geräte und deren ortslokale Spurinformation. Die ortslokale Spur ist über den Gerätepräsentanten zugreifbar. Neben den benachbarten Geräten werden weiter entfernte Geräte temporär gespeichert, sobald Nachrichten von diesen empfangen wurden. Hierbei werden auch der Zustand und die Adressierbarkeit des Gerätes vermerkt. Eine Abschätzung der Teilnehmerzahl zu vergangenen Veranstaltungen ist möglich, wird aber von der betrachteten Applikation nicht benötigt.

9.4 Direkte Kommunikation

Zur initialen Verbreitung der Vorlesungsmaterialien und zum Erstellen konsistenter Versionen benötigt die Applikation direkte multihop Kommunikation. Hierbei sind mehrere Kommunikationsprimitiven realisierbar. Der Unicast muss nur die Adressierbarkeit von Geräten innerhalb des Veranstaltungskontextes ermöglichen. Daher kann bei einem linkbasierten Verfahren auf die Routensuche außerhalb des Ortes verzichtet werden. Ein positionsbasiertes Verfahren benötigt nur die Positionen der Geräte im Kontext. Alle Geräte im Kontext können mit einem auf den Ort beschränkten multihop Broadcast erreicht werden, multihop Multicastverfahren können zusätzlich auch einen Teil der näheren Geräte außerhalb des Ortes einbinden. Multihop kann zu

Veranstaltungszeiten auch von außerhalb durch die Adressierung des Veranstaltungsortes kommuniziert werden. Das bedeutet, dass den veranstaltungsinternen Verfahren ein ortsbasierter Anycast vorgeschaltet wird. Um auf diese Kommunikation antworten zu können, müssen für das Antwortverfahren Zusatzinformationen mitgeliefert oder eine Kommunikationsroute vorbereitet werden. In folgender Tabelle sind die gewählten Kommunikationsverfahren und die durchgeführten Anpassungen aufgeführt:

Zweck	Verfahren	Konfiguration
Unicast vor Ort	DSR [56]	Beschränkung des Route-Discovery auf den Ort
Anycast von außerhalb	Greedy [24] mit GCR [30] Recovery	Anycastvariante
Unicast von außerhalb	Greedy mit GCR verknüpft mit DSR	Anycast Verknüpfung der Verfahren
Unicast nach außerhalb	Greedy mit GCR	Anycast liefert Senderposition für Antwort
Multicast an Teilnehmer	SPBM [106] auf Basis Greedy mit GCR	Kann bei allen Geräten der Route empfangen werden

Die Unicastverfahren werden nur verwendet, um einen Dienst zum wechselseitigen Ausschluss (*Tokendienst*) anzusprechen. Ein solcher Dienst wird benötigt, um konsistente Annotationsversionen zu erzeugen. Deshalb muss nur ein Gerät bekannt sein und dessen Kennung kann Teil der ortsglobalen Spur sein. Da im Falle dieser Applikation das Dozentengerät als Host des Tokendienstes periodisch weitere Nachrichten verbreitet (siehe Abschnitt 9.7), ist hierfür nur ein geringer Datenaufwand nötig. Im Applikationsszenario können auch Geräte außerhalb des Vorlesungsortes integriert werden, da der Dienst auf Anfragen direkt antwortet. Zu diesem Zweck wird bei der Anfrage die Absenderposition mitgesendet, obwohl das Anycastgreedyverfahren diese nicht benötigt. Dadurch wird eine direkte Antwort auf Basis eines positionsbasierten Unicasts ermöglicht.

Zur Verbreitung von neuerstelltem oder überarbeitetem Material wird SPBM eingesetzt, um auch Geräte außerhalb des Veranstaltungsortes erreichen zu können. SPBM verursacht allerdings einen permanenten Kommunikationsaufwand, da in den Regionen mit Teilnehmern Gruppeninformationen geflutet werden. In der Praxis ist es sinnvoll, diese Propagierung nur innerhalb einer bestimmten räumlichen Nähe zur Veranstaltung durchzuführen, da eine direkte Verbreitung von Material auch positionsbasiert ab einer gewissen Entfernung nicht mehr sinnvoll ist. Denkbar ist hier eine Beschränkung auf das Gebäude oder den umgebenden Universitätscampus, ähnlich dem Informationsradius für die Veranstaltungsinformationen.

Die Informationsverbreitung über den Multicast erreicht auch Geräte, die nicht Teilnehmer der Veranstaltung sind. Multihop Kommunikation im draht-

losen Medium beschreibt einen Broadcastkorridor, in dem die Nachricht empfangen werden kann. Wenn es sich dabei um Vorlesungsmaterial handelt (Verbreitung als MessageObjects), wird es auch von unbeteiligten Geräten weiterverarbeitet und in der Lobby der Middleware abgelegt. Diese Geräte speichern die Objekte altruistisch und stellen sie anderen Geräten zur Verfügung. Voraussetzung ist, dass sie diese Veranstaltungsgruppe kennen, hierfür Ressourcen zur Verfügung stellen und noch Ressourcen vorhanden sind. Das zu verbreitende Material erhält ein Flag im Kommunikationsheader, das angibt, dass die Informationen zur altruistischen Speicherung geeignet sind. Andere Nutzungen des Multicasts, insbesondere die Ereigniserzeugung (Senden eines RemoteEvents) bei anderen Teilnehmern, besitzen diese Kennung nicht. Somit gibt es zwei Multicastvarianten im Repräsentanten. Diese zwei Auslieferungsvarianten existieren für alle Kommunikationsvarianten, wobei die Applikation nicht alle Kombinationen nutzt. Zusammengefasst bietet der Veranstaltungsrepräsentant folgende multihop Kommunikationsdirektiven an:

- *receiveEventMany*: Erzeuge ein Ereignis auf allen Geräten im Zustand insideLesson und outsideLesson[1].
- *receiveDataMany*: Verbreite Material an alle Geräte mit dem Zustand insideLesson oder outsideLesson sowie im Broadcastkorridor der Kommunikation.
- *receiveEventOne*: Sende ein Ereignis an ein Gerät mit dem Zustand insideLesson.
- *receiveDataOne*: Sende Material an ein Gerät mit dem Zustand insideLesson und an Geräte im Broadcastkorridor der Kommunikation.
- *receiveEventAny*: Sende ein Ereignis an ein beliebiges Gerät mit dem Zustand insideLesson.
- *receiveDataAny*: Sende Material an ein beliebiges Gerät mit dem Zustand insideLesson und an Geräte im Broadcastkorridor der Kommunikation.

RemoteEvents im Kontext des Repräsentanten beinhalten neben den Absenderinformationen immer auch die Kennung der Veranstaltung. Somit können gleiche Ereignisse von Veranstaltungsrepräsentanten verschiedener Veranstaltungen verwendet werden. Eine Anwendung für RemoteEvents ist das Propagieren der aktuellen Folie im Netz. Somit kann die Applikation bei den Studierenden die Folien automatisch weiterblättern, wenn dies gewünscht ist.

Kommunikation kann auch explizit auf Geräte in Kommunikationsreichweite beschränkt werden. Hierbei tritt kein Unterschied zwischen den Repräsentantenzuständen auf. Im Zustand noLesson ist immer nur die singlehop Kommunikation möglich. Die Propagierung eines RemoteEvents ent-

[1] SPBM ermöglicht auch die Adressierung eines Teils der Geräte außerhalb der Veranstaltung.

spricht einem singlehop Multicast, die Propagierung eines MessageObject entspricht einem singlehop Broadcast. Lokale Kommunikation wird in drei Varianten angeboten:

1. Als ungesicherter Multicast/Broadcast.
2. Als adressierter Multicast/Broadcast.
   (Der Auftraggeber bestimmt die Adressen)
3. Als impliziter adressierter Multicast/Broadcast.
   (Adressen sind alle lokal bekannten Nachbarn des Kontexts)

Die Applikation kann also wählen, ob sie nur lokal oder auch multihop im Veranstaltungskontext kommunizieren will. Wird der Zustand des eigenen Gerätes im Veranstaltungskontext (insideLesson oder outsideLesson) abgefragt, kann auch auf den transparenten Übergang zwischen Kommunikation innerhalb und von außerhalb verzichtet werden. Schließlich ist es der Applikationen bei allen Kommunikationsvarianten möglich, zwischen dem Versenden von RemoteEvents und MessageObjects zu unterscheiden.

9.5 En-Passant Kommunikation

Fehlendes Vorlesungsmaterial kann von einem Gerät zur Vorlesungszeit beim Dozentengerät angefordert werden. Das Dozentengerät besitzt neben dem vollständigen Foliensatz auch alles weitere Material, da es aufgrund des Hostens des Tokendienstes auf dem aktuellen Stand ist (siehe Abschnitt 9.6). Dadurch wird aber in vielen Fällen das gesamte Netz der Veranstaltung belastet. Aus diesem Grund wird versucht, fehlendes Material bei direkten Nachbarn anzufordern, so dass möglichst wenige Geräte von der Kommunikation betroffen sind (siehe Abbildung 9.5). Ist keiner der Nachbarn geeignet, kann immer noch auf ein entfernteres Gerät zurückgegriffen werden.

Außerhalb der Veranstaltungen wird Veranstaltungsmaterial nur auf Basis der En-Passant Kommunikation mit anderen abgeglichen. Die Interaktion findet also immer mit direkten Nachbarn statt. In beiden Fällen müssen die Geräte das auszutauschende Material bestimmen. Dieser Abgleich wird aufgrund der Mobilität der Geräte ausgelöst (En-Passant) oder im stationären Fall auf Basis von Spuränderungen der Geräte. Letzteres propagiert Hinweise, die den aktuellen Stand des Gerätes beinhalten. Aufgrund dieser Hinweise kann dann ein benachbartes Gerät erkennen, ob lokal Informationen fehlen. Innerhalb wie außerhalb des Veranstaltungsortes wird die Beaconnachricht des Geräts um Zusatzinformationen erweitert. Der Unterschied besteht in der Art und dem Umfang der Information. Innerhalb des Ortes kann davon ausgegangen werden, dass eine Nachbarschaft lange Bestand hat. Somit können erheblich seltener, aber dafür mehr Informationen angehangen werden. Insbesonde-

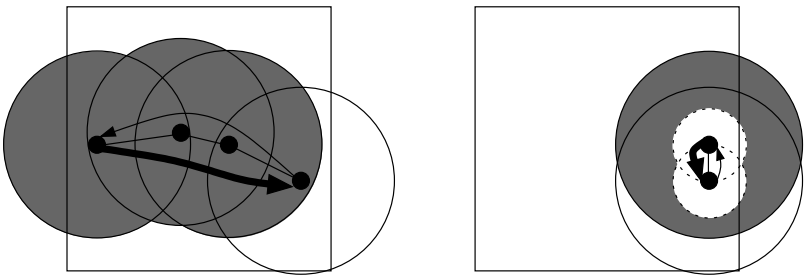


Abb. 9.5. Wird fehlendes Material immer von einem bestimmten Gerät angefordert (links), wird im schlechtesten Fall der gesamte Veranstaltungsraum durch die Kommunikation gestört. Rechts wird ein direkt benachbartes Gerät angefragt. Eine Anpassung der Sendestärke reduziert die Belastung der Umgebung noch weiter.

re ist auch die Auswahl der möglichen Nachbarn größer, so dass hier Datenumfang, Aktualität und auch die zu erwartende Kommunikationslast für das Gesamtnetz[2] als Auswahlkriterien verwendet werden. Es werden also durch umfangreichere Information unnötige Anfragen vermieden. Auf Basis der in Tabelle 9.1 dargestellten Informationen registriert die Anwendung En-Passant Trigger zur Objektpropagierung. Ändern sich diese Informationen bei einem Nachbargerät, wird die registrierte Methode aufgerufen. Das Auslösen der Trigger wird zeitlich und solange Objekte lokal empfangen werden verzögert, um überflüssige bzw. parallele Abgleiche zu verhindern. Dies kann passieren, wenn dieselben Objekte auch bei anderen benachbarten Geräten fehlen. Somit kann ein schon ausgelöster Abgleich die fehlenden Objekte mittels des adressierten Broadcasts ausliefern.

Datentyp	Spurinformation
Folie	Kennung der letzten vorhandenen Folie der Folienreihe
Folienliste	Flag “vorhanden für aktuelle Vorlesung”
Annotationsliste	Summe aller Versionen
Annotation	Flag vollständig
Material der letzten Vorlesung	Hashwert

Tabelle 9.1. Für jeden Datentyp der Applikation wird innerhalb der Vorlesung die kommunikationslokale Spur um Informationen erweitert, die sporadisch über die Beaconnachricht des Geräts verbreitet werden.

Für jeden Materialtyp wird eine Abgleichsmethode registriert, die aus den aktuellen Nachbarn einen auswählt, der fehlendes Material liefern soll. Im Falle der Folien- und Annotationsobjekte liefert der Trigger dem Nachbar-

[2] Ist die Sendestärke adaptierbar ist hier in der Regel das geographisch nächste bzw. das Gerät mit der größten Signalstärke am günstigsten.

gerät direkt die Kennungsliste der fehlenden Objekte, für Folienlistenobjekte wird der Trigger ausgelöst, wenn ein Nachbar diese besitzt. Als detaillierteres Beispiel ist in Abbildung 9.6 der Annotationslistenabgleich dargestellt.

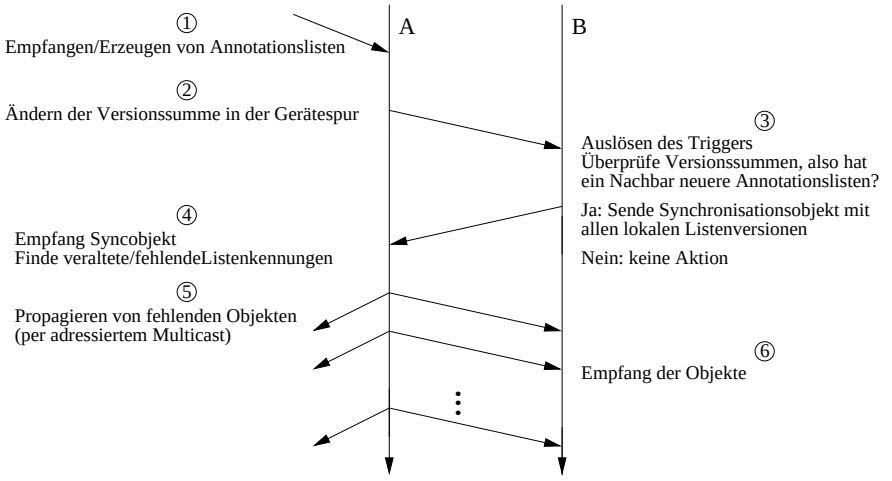


Abb. 9.6. Der Materialabgleich am Beispiel der Annotationslisten. Gerät A ändert seine propagierte Spurinformation, sobald sich lokal eine Versionsnummer der Annotationslisten ändert. Dies löst verzögert bei Nachbargeräten (B) den registrierten En-Passant Trigger aus. Ist das Nachbargerät auf einem aktuelleren Stand, werden die lokalen Listenkennungen und deren Versionsnummern an den besten Nachbarn verschickt. Dieser kann die fehlenden und veralteten Listen bestimmen. Verbreitet werden diese an alle Nachbarn, wobei diejenigen, die die Daten explizit angefordert haben, den Empfang bestätigen (adressierter Broadcast).

Außerhalb des Veranstaltungsortes sind die Kontakte mit anderen Geräten der Veranstaltung häufig sehr kurzfristig, so dass ein Abgleich so früh wie möglich beginnen muss. Aus diesem Grund wird häufiger eine Information angehangen, um möglichst frühzeitig die Notwendigkeit zum Abgleich zu erkennen. Um die periodischen Nachrichten klein zu halten muss aufgrund der häufigen Erweiterung der Beaconnachricht die angehangene Informationsmenge klein bleiben. Konkret wird die Gruppenkennung der Veranstaltung und ein Hashwert über das lokal vorhandene Material hinzugefügt. Dadurch werden unnötige Abgleichsversuche bei Gleichheit des Materials mit hoher Wahrscheinlichkeit vermieden. Der Abgleichsprozess selbst dauert aber länger, da nun in einem vorgelagerten Schritt die Informationen versendet werden müssen, die innerhalb der Veranstaltung schon in den Beaconnachrichten verbreitet werden.

Stehen die Objekte fest, die an einen Nachbarn propagiert werden sollen, werden sie vom Objektpropagator mit einem adressierten Broadcast verbrei-

tet. Somit werden die Objekte auch von Geräten empfangen, die diese nicht explizit angefordert haben. Empfangene Objekte werden in der Lobby abgelegt und müssen vom Objektmanager zur längerfristigen Speicherung in den Residencepool verschoben werden. Für Geräte, die diese Applikation nicht nutzen, werden die Objekte von dem altruistischen Objektmanager verwaltet. Er verwaltet, d.h. verlängert und propagiert, nur eine feste Menge von aktuellen Objekten, die zu bestimmten Gruppen gehören, ähnlich der altruistischen Objektverwaltung der lokalen Gruppe (Abschnitt 8.6). Zur altruistischen Speicherung wird ein Zeitdelta benötigt, das angibt, wie lange Objekte mindestens auf anderen Geräten verbleiben sollten, wenn die Ressourcen dies ermöglichen. Hierbei wird die Dauer bis zur nächsten Vorlesung angegeben[3].

9.6 Wechselseitiger Ausschluss

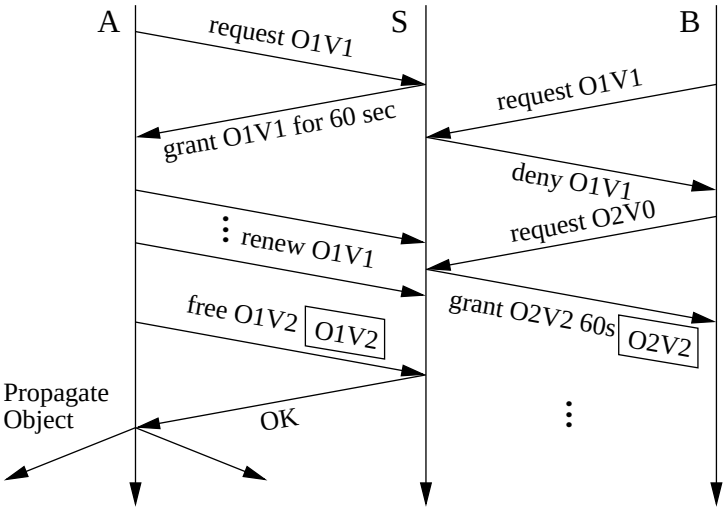


Abb. 9.7. Kommunikationsbeispiel zwischen Tokenclients (A und B) und Tokenserver (S)

Während einer Veranstaltung befinden sich die meisten Geräte an einem geographischen Ort. Aufgrund der geographischen Nähe ist es möglich, Inkonsistenzen beim gleichzeitigen Überarbeiten von Veranstaltungsmaterial durch

[3] Die Wahl dieses Wertes simulativ zu evaluieren ist aber mangels realem Benutzerverhaltens nicht sinnvoll durchführbar. Die Überlegung ist hierbei, dass in der nächsten Veranstaltung neues Material erstellt und verbreitet wird, das die bisherigen altruistisch gespeicherten Objekte verdrängen wird.

Kommunikation mit anderen Geräten zu vermeiden. Wird eine Annotation neu erzeugt, muss die Annotationsliste der zugehörigen Folie überarbeitet werden. Damit die Liste konsistent bleibt, sollte dies zu einem Zeitpunkt nur ein Gerät tun, so dass Inkonsistenzen verhindert werden. Wird eine Annotation überarbeitet, werden die Annotation und die entsprechende Annotationsliste verändert. Auch hier sollten beide Objekte zu einem Zeitpunkt nur von einem Gerät verändert werden. Deshalb muss vor dem Überarbeiten eines Objektes angefragt werden, ob es von keinem anderen Gerät bearbeitet wird. Dieser Ausschluss ist zentral realisiert und erfolgt auf dem Dozentengerät. Dadurch kann mit nur zwei Unicastnachrichten überprüft werden, ob ein bestimmtes Objekt bearbeitet werden kann. Abbildung 9.7 stellt den Ablauf der Kommunikation mit dem Tokenserver dar, Abbildung 9.8 den Zustandsautomaten des Tokenclients. Gerät A fordert das Token für Objekt $O1$ an. $O1$ liegt lokal in der Version $V1$ vor. Da dieses Objekt nicht gesperrt ist, kann der Tokenserver das Token für dieses Objekt an A herausgeben. Ein Token wird nur für eine feste Zeit vergeben. Gerät A muss also periodisch das Token erneuern. Somit wird der Verlust eines Tokens durch den Ausfall eines Gerätes (z.B. durch Verlust der Verbindung zur Veranstaltung) erkannt. In diesem Fall wird das Token wieder freigegeben. Mögliche Inkonsistenzen aufgrund erfolgter Änderungen muss das Gerät auflösen, welches das Token verloren hat. Hier muss im Zweifelsfall der Benutzer aus der konsistenten offiziellen und der lokalen Version eine neue Version erzeugen. Ist das Bearbeiten abgeschlossen, wird das Token von Gerät A freigegeben und die neue Version mitgesendet. Der Tokenserver besitzt also immer einen konsistenten Informationspool. Danach kann eine Verbreitung der Information erfolgen. Während Gerät A Objekt $O1$ editiert, kann kein anderes Gerät ein Token für $O1$ erhalten. Für andere Objektkennungen werden Tokens herausgegeben und somit ist ein Editieren von $O2$ möglich. Gerät B besitzt in diesem Beispiel lokal nicht die aktuelle Version $V2$ sondern $V0$. Deshalb wird die aktuelle Version vom Tokenserver mitgeliefert.

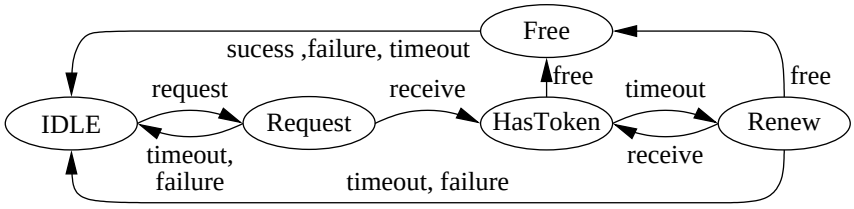


Abb. 9.8. Zustandsautomat des `TokenClients`.

Der Tokenserver kann prinzipiell auf jedem Gerät gestartet werden, das einen vollständigen Informationspool besitzt. Somit ist eine Migration bzw. Neuinstanziierung des Tokenservers möglich, falls der Host die Veranstaltung verlässt oder ausfällt. Diese Realisierung ist nicht implementiert, aber möglich,

wenn in der ortsglobalen Spur eine Gesamtversionsnummer verbreitet wird. Nun kann für diese Aufgabe eines der Geräte ausgewählt werden, die dieser Versionsnummer genügen. Die Herausgabe von Tokens muss bei einer Neuinstantiierung aber um die Erneuerungszeit eines Tokens verzögert werden, um herausgegebene Tokens zu an den Erneuerungsnachtrichten zu erkennen und nicht explizit nach vergebenen Tokens suchen zu müssen.

9.7 Benutzeroberfläche

Die Benutzerinterfacekomponente hat neben der Darstellung der aktuellen Folie und den dazugehörigen Anmerkungen noch einige weitere Aufgaben: Sie beantragt die zur Annotationsbearbeitung notwendigen Tokens im lokalen `TokenClient` und gibt die Verbreitung der erzeugten bzw. veränderten Datenobjekte in Auftrag.

Damit immer die korrekten Objekte dargestellt werden, registriert die GUI Ereignisroutinen, die aufgerufen werden, wenn neue Objekte im Residencepool erscheinen. Während einer Vorlesung ist die GUI auf das Erscheinen des Folienlistenobjektes und auf die aktuell darzustellende Folie registriert, falls diese lokal nicht vorhanden sind. Daher werden sie sofort dargestellt, sobald sie sich im Objektpool befinden. Zur aktuell dargestellten Folie registriert sich die GUI zusätzlich auf die dazugehörigen Annotationsobjekte, so dass empfangene Änderungen sofort dargestellt werden. Die Zustandsereignisse des Veranstaltungskontextes werden verwendet, um den Nutzer über Zustandsänderungen informieren zu können. So kann der Nutzer bei Verlassen der Veranstaltung darauf hingewiesen werden, wenn lokale Änderungen noch nicht verbreitet wurden.

Die GUI des Dozenten erzeugt zusätzlich Folienwechselereignisse, die über den Veranstaltungskontext verbreitet werden und auf die sich die Studierenden registrieren können. Somit werden in der GUI der Studierenden die Folien automatisch umgeblättert. Periodisch wird die aktuelle Folienkennung über den Veranstaltungskontext propagiert. Hierbei werden auch die ortsglobalen Informationen des Dozenten verbreitet.

In Zukunft soll die Dozenten-GUI auch das Verwalten der Veranstaltung ermöglichen. Dazu gehört das Zuweisen von neuen Datenobjekten wie Folienobjekten, zur Veranstaltung. Auch das Verwalten der Veranstaltungszeiten und -orte ist noch nicht über die GUI realisiert. Im Prototypen sind diese Aufgaben noch fest kodiert. Die Folien werden also zu Beginn der Veranstaltung automatisch eingelesen und mit der Veranstaltung assoziiert. Die initiale Folienverbreitung erfolgt mit dem ersten Betreten der Veranstaltung – diese Funktionalität ist zurzeit ebenfalls in der Dozenten-GUI realisiert.

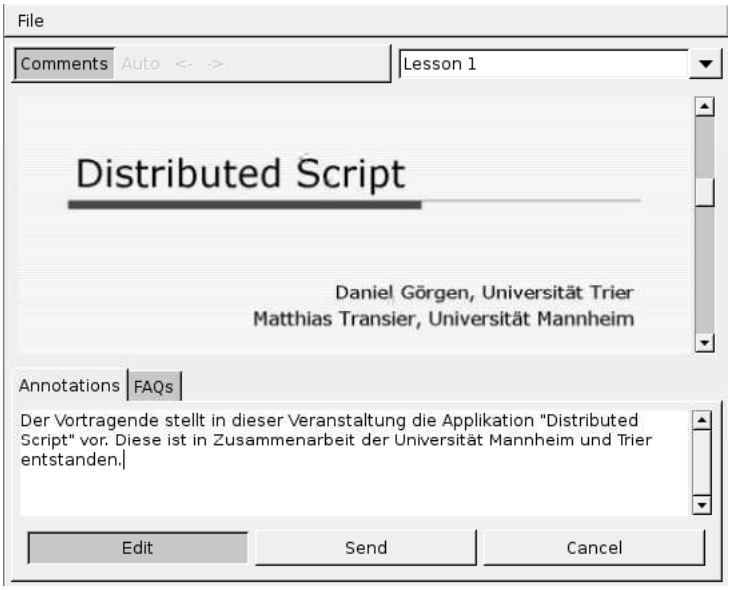


Abb. 9.9. Screenshot der Benutzeroberfläche von DistScript.

9.8 Prototyp

Realisiert wurde die Applikation unter Nutzung des dreistufigen Entwicklungsprozesses für mobile Anwendungen [66]. Im ersten Schritt wurden die Anwendungskomponenten in einer Umgebung aus simuliertem Netzwerk und simulierter Gerätemobilität realisiert und getestet. Die Testszenarien wurden mit simuliertem Benutzerverhalten durchgeführt. Im zweiten Schritt wurde die graphische Benutzeroberfläche implementiert und die simulierte Anwendung mit realer Benutzerinteraktion getestet. Schließlich wurden kleinere Tests mit realen Geräten durchgeführt. Hierbei konnten die Applikationskomponenten ohne Änderungen auf die Ausführungsplattform der Middleware portiert werden. Zur effizienten Kodierung der Beaconnachrichten wurden Kodierfunktionen registriert, die statt der Standard Java Objektserialisierung eine effizientere Bitcodierung der Beaconnachrichten durchführen. In Abbildung 9.10 ist die Applikation in der zweiten Phase, dem Hybridmodus, dargestellt. Unten sind zwei graphische Benutzeroberflächen dargestellt, die mit der Applikation auf zwei Geräten (15 und 2) in der simulierten Umgebung oberhalb verbunden sind. Mit der linken GUI wird momentan eine Anmerkung bearbeitet, der rechten ist somit der ändernde Zugriff darauf verwehrt. Die obere GUI zeigt die simulierte Umgebung. Hierbei handelt es sich um ein interaktives Bewegungsmodell, wobei die Geräte zwischen Hörsälen bewegt werden und auf die angedeuteten Sitzplätze gesetzt werden können. Geräte mit mindestens einem Kreis besitzen einen Veranstaltungsrepräsentanten, Geräte mit meh-

reren Kreisen sind im Kontext der Veranstaltung im oberen linken Hörsaal. Die restlichen Kreise visualisieren die Zustände der Applikationskomponenten. Hellgraue Verbindungslinien visualisieren potenzielle Netzwerkverbindungen zwischen Geräten. Das Netzwerk ist hier so angelegt, dass der Hörsaal in drei Hops durchquert werden kann. Schwarze Linien zeigen den Multicastbaum der letzten versendeten Nachricht an, in diesem Falle ausgehend vom Dozentengerät links.

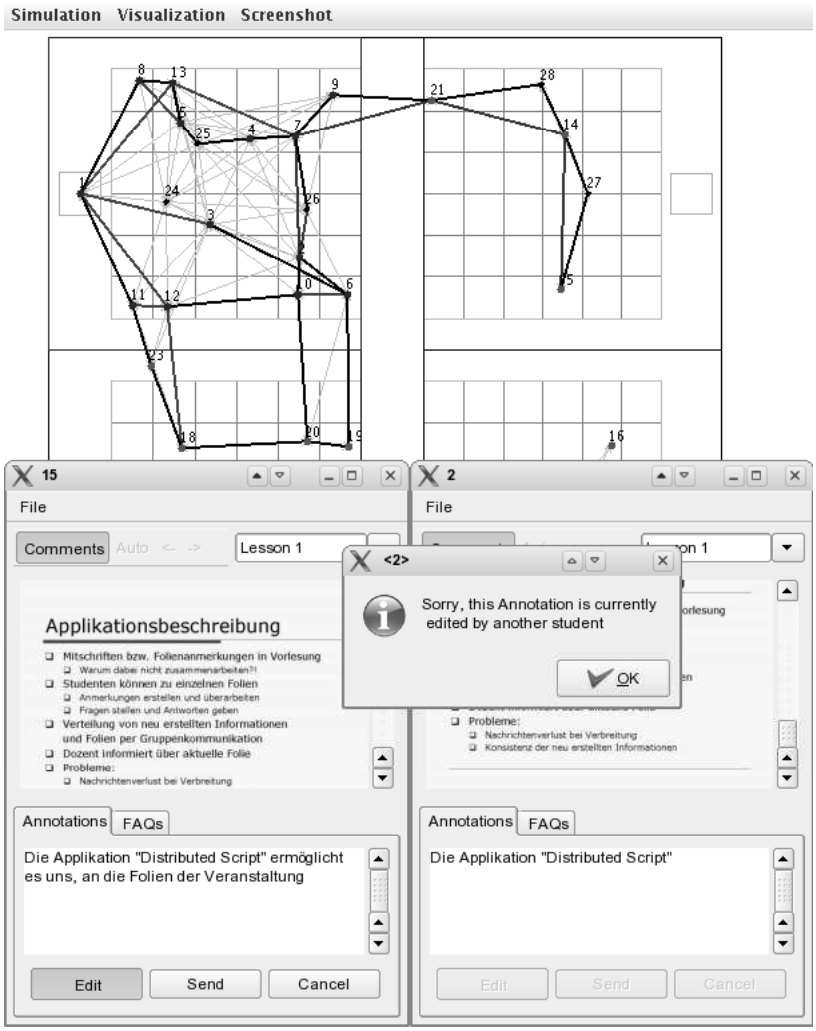


Abb. 9.10. Die Applikation DistScript im Hybridmodus der Middleware.

10

Verwandte Arbeiten

Traditionelle Middlewaresysteme wie CORBA [90] oder Microsofts DCOM [11] setzen eine hohe Kommunikationszuverlässigkeit und zentrale Infrastrukturen wie Namensdienste voraus. Aus diesem Grund sind sie nur bedingt in mobilen ad-hoc Netzwerken einsetzbar. Dienen ad-hoc Netzwerke als letzter Hop in einem traditionellem Infrastrukturnetzwerk oder ist das multihop Netzwerk stark zusammenhängend und nur von geringer Dynamik, sind die genannten Middlewaresysteme dennoch einsetzbar. Sie benötigen in der Regel aber auch über größere Entfernungen eine Netzwerkbandbreite, die in multihop ad-hoc Netzen nicht zur Verfügung steht. Darüber hinaus sind diese Systeme ohne Anpassungen zu schwergewichtig für die in ad-hoc Netzen eingesetzten, meist leistungsschwachen Geräte. Auch eine dynamische Anpassung zur Laufzeit an die sich permanent ändernden Netzwerkbedingungen ist nicht vorgesehen. Um eine Adaptierbarkeit zur Laufzeit zu ermöglichen, wurden adaptive Systeme entworfen, welche die Transparenz der Middlewareschichten aufheben. OpenCORBA [65] und NextGenerationMiddleware [7] erweitern CORBA um eine Adaptierbarkeit auf Basis von Reflection. Somit sind diese Systeme besser für dynamische Netze geeignet, sie sind aber primär für den Einsatz in Infrastrukturnetzen gedacht. Auch hier gilt, dass die benötigten zentralen Infrastrukturen, erforderliche Netzbandbreite, und ein hoher Ressourcenbedarf den Einsatz in mobilen ad-hoc Netzwerken nahezu ausschließen.

In den vergangenen Jahren wurde eine Vielzahl von Middlewareansätzen für mobile singlehop und multihop ad-hoc Netzwerke entworfen. Im folgenden werden einige dieser Systeme detaillierter vorgestellt.

Proem

Proem [62] ist eine Peer-to-Peer Plattform für mobile ad-hoc Netzwerke, bestehend aus Endbenutzergeräten. Kommuniziert wird mit allen erreichbaren Knoten im Netzwerk, wobei Erreichbarkeit durch eine beliebige transparente

Netzschicht definiert ist, die eine asynchrone nachrichtenorientierte Unicast- und Multicastkommunikation zur Verfügung stellt. Proem definiert vier Objekttypen: die Geräte (peers), Personen, die Geräte besitzen (individuals), Sammlungen von Datenelementen (data Space) und Gruppen von Objekten (groups). Gruppen werden dezentral über geheime Gruppenschlüssel verwaltet, die Aufnahme in eine Gruppe erfolgt, wenn eine vorgegebene Zahl Gruppenmitglieder dem zustimmt. Es kann mit der gesamten erreichbaren Nachbarschaft interagiert werden, also je nach Netzwerkrealisierung auch multihop. Die Nachbarschaft wird von einer Nachbarschaftsverwaltung gepflegt, wobei Geräte ihre Existenz im gesamten erreichbaren Netz per Multicast propagieren. In einer Datenbank wird über sämtliche, in der Vergangenheit benachbarten Geräte, Buch geführt. Datenobjekte werden persistent in einem Pool gespeichert, wobei Applikationen mit Hilfe einer Publisher/Subscriber Kommunikation Datenverfügbarkeit propagieren und Interesse an Daten bekunden können. Lokale Interaktion erfolgt ereignisbasiert über eine zentrale Ereignisschlange.

RCSM: Reconfigurable Context-Sensitive Middleware

RCSM [112, 113] realisiert autonome Interaktionen zwischen benachbarten Objekten lokal oder auf Geräten in Kommunikationsreichweite. Geräte sind hierbei mobile Endbenutzergeräte [112], sowie auch Sensoren [113]. Autonome Interaktion wird durch Kontextinformationen ausgelöst, die aus Sensor-, Umgebungs-, und Benutzerinformationen bestehen. In der Interfacebeschreibung, eine Erweiterung der CORBA-IDL, werden für jede Objektmethode Kontextbedingungen angegeben, wann diese Methode aktiviert wird und ob sie Daten erhalten (IN) oder liefern soll (OUT). Aktivierung bedeutet, dass die Methodensignatur propagiert wird, wenn der gefragte Kontext eintritt. Passt die Methodensignatur zu einem von einem benachbarten Objekt vorgegebenen In/Out Mapping, werden die Methoden automatisch verknüpft, d.h. das Ergebnis der Out-Methode wird als Parameter für die passende IN-Methode verwendet. Realisiert ist RCSM als Erweiterung von CORBA. Hierbei wird ausschließlich mit direkt benachbarten Geräten kommuniziert und die ORBs verbunden, falls ein Kontextmatch entstanden ist.

STEAM: Scalable Timed Events And Mobility

STEAM [79] basiert auf einer Ereignisverbreitung ausgehend von fest installierten Sendern zu passierenden, in der Regel hochmobilen Geräten. Die Verbreitung wird auf eine vorgegebene geographische Region um den Sender beschränkt. Die Ereignisfilterung nach Inhalt, Typ und Region erfolgt beim Empfänger. Kommuniziert wird über so genannte *proximity groups*, die über die geographische Region und den Ereignistyp eindeutig bestimmt werden. Die Existenz einer Gruppe wird vom Ereignissender periodisch propagiert.

Die multihop Verbreitung der Ereignisse soll Zuverlässigkeit und die Einhaltung von Auslieferungszeiten ermöglichen.

SOM: Sentient Object Middleware

Diese Middleware ist auf die Erfassung, Aufbereitung und multihop Verbreitung von Kontextinformation ausgelegt [101]. *Sentient Objects* (SO) empfangen Ereignisse von einer Menge an Sensoren. Diese Daten werden nach vorgegebenen applikationsspezifischen Regeln durch eine Inference Engine vorverarbeitet und können so Ereignisse für ein SO erzeugen. Hierbei steht insbesondere die Dienstgüte im Vordergrund. Auch die multihop Kommunikation wird auf Einhaltung der Dienstgüte kontrolliert. Hierbei wird ein Publisher/Subscriber Modell eingesetzt, das dem von STEAM ähnlich ist. Die Ereignisse und auch die Filterung werden mittels XML Beschreibungen repräsentiert. Zur Ereignispropagierung können verschiedene multihop Multicastprotokolle, unter anderem auch probabilistische, verwendet werden.

iFlow

iFlow [70] ist eine auf kontrolliertem Fluten basierende Datasharing Middleware, die für den Einsatz in Sensornetzen und hybriden Netzen gedacht ist. iFlow unterscheidet aus Sicht eines Informationspaketes drei Gerätetypen: *Supplier*, *Relays* und *Consumer*. Der Supplier teilt die Information in mehrere Pakete und streckt sie redundant mittels einer *Tornadocodierung* um einen festen Faktor k. Die Pakete werden per Broadcast verteilt. Die Nachbarknoten entscheiden aufgrund von Wahrscheinlichkeiten, die sich unter anderem aus der Popularität und der Distanz zu anderen Knoten ergeben, über die Speicherung des Paketes (Relay) und über die Weiterleitung. Consumer können Informationen erhalten, indem sie die einzelnen Pakete von verschiedenen Knoten einsammeln, denen sie aufgrund ihrer Bewegung begegnen. Kann nach einer bestimmten Zeit eine Anfrage nicht erfüllt werden, wird der Rest der Pakete direkt beim Supplier angefragt.

LIME: Linda in a mobile Environment

LIME [85] realisiert einen lokal eindeutigen Tupelspace [1][1] zwischen benachbarten Geräten, über den mobile Agenten interagieren. LIME unterscheidet

[1] Ein Tupelspace ist ein Shared-Memory Modell, basierend auf Tupeln, die einer Sequenz getypter Felder entsprechen. Prozesse interagieren nur über diesen Informationsraum und greifen darauf mit den elementaren Operationen *out* zum Einstellen, *in* zum Entfernen und *rd* zum Lesen von Tupeln zu. *in* und *rd* suchen Tupel auf Basis eines angegebenen Musters und existieren in blockierenden und nicht blockierenden Varianten.

drei Teilraumtypen (Scopes): agentenlokal, gerätelokal und kommunikationslokal. Der kommunikationslokale Tupelspace beinhaltet den eigenen gerätelokalen Raum und zusätzlich werden die gerätelokalen Räume der Nachbargeräte eingeblendet. Bei Operationen auf dem gesamten lokal sichtbaren Raum kann angegeben werden, ob nur innerhalb eines bestimmten Scopes operiert werden soll. Somit lassen sich Leseoperationen auf ein bestimmtes Gerät einschränken. Zusätzlich können Ereignisroutinen registriert werden, um über das Auftauchen von Tupeln informiert zu werden. Auch hier ist eine Scopeangabe möglich. Das Zusammenführen von Tupelspaces von Nachbargeräten muss in einer einzigen atomaren Operation erfolgen. Der Kopplungsgrad zwischen den Geräten ist somit sehr hoch.

TinyLIME [17] ist eine Variante, die für ein spezielles Netzwerkszenario entworfen wurde. Das Netz besteht aus einem dünnen, unter Umständen unzusammenhängenden Sensornetz, in dessen Region sich leistungsfähigere mobile Geräte bewegen. Der Kopplungsgrad und der Kopplungsaufwand werden für die Sensorknoten verringert, indem diese nicht automatisch, sondern nur bei expliziter Anforderung in andere Tupelspaces eingebunden werden.

Limone

Limone [26] basiert auf mobilen Agenten, die über Tupelspaces lokal auf einem Gerät oder singlehop mit anderen Agenten interagieren. Jeder Agent hat hier seinen eigenen Tupelspace. Agenten können inklusive ihres gesamten Tupelspaces auf Nachbargeräte migrieren. Agenten geben Filter an, mit denen die Middleware agentenspezifische Nachbarschaftslisten pflegt. An Agenten aus der Nachbarschaftsliste können Operationsaufträge für deren Tupelspaces abgesetzt werden, die synchron erfolgen. Aufträge an Agenten auf Nachbargeräten werden zusätzlich mit Timeouts versehen. Eine Unterscheidung zwischen Timeout und fehlgeschlagener Operation ist nicht möglich. Nachbarschaftslisten können mit Ereignisbehandlungsroutinen versehen werden, die unter anderem das Matchen von Tupels in benachbarten Tupelspaces ermöglichen.

Während LIME einen erheblich höheren Kopplungsgrad unter benachbarten Geräten benötigt, entspricht Limone nicht mehr der eigentlichen Tupelspaceidee, da selbst lokale Agenten nicht über einen gemeinsamen Tupelspace interagieren.

MESHMdl

MESHMdl [51] ist eine tuplespacebasierte Middleware, die primär auf singlehop Interaktion basiert. Applikationskomponenten interagieren mit anderen Komponenten und der Middleware ausschließlich über den gerätelokalen Tupelspace. Unterschieden wird zwischen normalen, virtuellen und persistenten

Tupels. Persistente Tupels werden in einem nicht flüchtigen Speicher abgelegt und werden zum Lesen daraus entfernt. Virtuelle Tupel werden von der Middleware erzeugt und repräsentieren andere Applikationen sowie direkt benachbarte Geräte. Die virtuellen Tupel der Nachbargeräte beinhalten auch eine Repräsentierung des Tupelspace des Nachbargerätes, über den Operationen auf dem eigentlichen Tupelspace durchgeführt werden. Nachbartupel werden wieder gelöscht, wenn die Nachbarschaft beendet ist. Um Tupels automatisiert auszutauschen, werden *Xectors* eingesetzt. Diese beinhalten ein Tupeltemplate und haben die Aufgabe, passende Tupels zu sammeln, zu verbreiten oder abzulehnen. Darüber hinaus kann auch eine Funktion zur Manipulation des Tupels und die Geräteadresse des Zieltuplespaces angegeben werden. Applikationen werden auch als Agenten bezeichnet, da konzeptionell vorgesehen ist, dass sie auch auf benachbarte Geräte migrieren können, um unterbrechungsfrei interagieren zu können.

TOTA: Tuples on the air

Die *TOTA* [74] Middleware basiert auf mobilen Agenten die auf gerätelokalen Tupelspaces basieren. Im Tupelspace erzeugte Tupel werden nach einer tupelspezifischen Verbreitungsregel im Netzwerk propagiert. Die Verbreitungsregel wird im Tupel realisiert und entscheidet, ob ein Tupel lokal verbleibt und ob es, unter Umständen dupliziert, weiterverbreitet werden soll. Diese Entscheidung kann von der Nachbarschaft und den lokal vorhandenen anderen Tupeln abhängig gemacht werden. Die Verbreitung kann auch später und wiederholt erfolgen. Hierbei kann die Verbreitungsregel auch Änderungen am Tupelinhalt vornehmen. Beispiele für Verbreitungsregeln sind gerichtetes Fluten und Geocast. Die mobilen Agenten können auf Nachbargeräte migrieren. Die Migration wird von den Agenten selbst entscheiden.

In [73] wird TOTA um eine Maintainanceregel in den Tupeln erweitert. Diese kann Änderungen am Tupelinhalt an alle Kopien des Tupels in der Verbreitungsregion propagieren. Darüber hinaus kann hier zusätzlich auf Tupelspaces direkt benachbarter Geräte zugegriffen werden.

XMIDDLE

Applikationen auf Basis von *XMIDDLE* [76] interagieren über XML Bäume, die mit direkten Nachbarn ausgetauscht und teilautomatisiert abgeglichen werden. Ein Gerät kann sich an einen Teilbaum des XML-Baums eines Nachbargerätes binden. Dadurch wird der Teilbaum in seinen lokalen Baum kopiert. Operationen auf dem Teilbaum werden aber immer von dem Ursprungsgerät durchgeführt, solange es sich noch in direkter Nachbarschaft befindet. Änderungen werden an alle Geräte in der Nachbarschaft propagiert, die den Teilbaum eingebunden haben. Ist das Ursprungsgerät nicht erreichbar, kann auf

der lokalen Kopie weitergearbeitet werden. Somit ist ein Abgleich der XML Bäume notwendig, wenn zwei Geräte sich treffen, die denselben Teilbaum besitzen. Entstehen beim automatischen Abgleich Konflikte, werden applikationsspezifische Abgleichfunktionen aufgerufen. Ein erfolgreicher Abgleich erzeugt eine *Edition*, die anhand einer laufenden Nummer und der Kennungen der zwei beteiligten Geräte eindeutig ist. Lokale Änderungen an Editionen werden in der Editionskennung vermerkt und sind *Versionen*. Änderungen in Teilbäumen ändern auch die Editionskennungen des Elternknoten.

EMMA: Epidemic Messaging Middleware for Ad hoc networks

EMMA [88] ist eine nachrichtenorientierte Middleware. Geräte besitzen Messagequeues, in die sich Gerätenachbarn mit verlängerbaren Zeitdeltas eintragen können. Gerätenachbarn sind alle, sich in Sendereichweite befindlichen Geräte, es können sich aber auch Geräte aufgrund von multihop Kommunikation eintragen. Ist einer der Empfänger nicht erreichbar, wird ein epidemisches Verbreitungsverfahren verwendet. Nachrichten haben hierbei eine bestimmte Lebensdauer und werden solange weiterpropagiert, bis diese abgelaufen ist. Dazu werden mit neuen Nachbarn Nachrichtenlisten ausgetauscht und abgeglichen. Nachrichten können je nach Typ über denselben Mechanismus, also Unicastkommunikation und im Fehlerfall Fluten der Nachricht, bestätigt werden. Wurde die Nachricht epidemisch verbreitet, wird auch das Acknowledgement geflutet. Somit wird die Nachrichten noch vor Ablauf ihrer Lebenszeit von den meisten Geräten entfernt.

GAS: Gadgetware Architectural Style

GAS [57, 58] ist eine Middleware für Sensor-/Aktornetzwerkcluster. Die Knoten des Netzwerks sind Alltagsobjekte, so genannte *eGadgets*, die Sensor-, Aktor-, Kommunikations-, und Rechenfähigkeiten besitzen und gemeinschaftlich eine Ambient Intelligence Umgebung bilden. eGadgets werden als Objekte modelliert. Jedes reale Gerät besitzt einen virtuellen Repräsentanten, der die Fähigkeiten des Objektes darstellt und diese anderen eGadgets zur Verfügung stellt. Dies geschieht mittels *Plugs*, die, falls sie kompatibel sind, zwischen zwei eGadgets eine Beziehung herstellen können. Assoziierte eGadgets können zu einer funktionalen Konfiguration zusammengefasst werden, die einem gemeinsamen Zweck dienen.

eGadgets propagieren aktiv ihre Fähigkeiten (Plugs), die von anderen zwischengespeichert werden. Kommunikation zwischen eGadgets erfolgt über eine DSR-Variante, wobei eine eigene Objektadressierung verwendet wird. Die Kommunikation ist asynchron und nachrichtenorientiert. Die Nachrichten liegen immer in einem XML Format vor. Verbindungen zwischen eGadgets sind daher eher lose, d.h. es besteht keine permanente Verbindung da nur Assoziationen gespeichert werden. Kommunikation ist immer lokal oder findet

über wenige Hops statt, da sich die Knoten in dem vorgesehene Szenario in einer näheren, räumlich beschränkten Umgebung befinden. Dienstsuche und -matching wird von jedem Gerät selbst realisiert, wobei auf eine global eindeutige Ontologie zurückgegriffen wird, die auf jedem Gerät vorhanden ist.

10.1 Diskussion

Die vorgestellten Middlewaresysteme lassen sich in Bezug auf die verwendeten Kommunikationsmechanismen in drei Klassen unterteilen:

Rein auf singlehop Interaktion basieren RCSM, Lime, Limone, MeshMDL und XMIDDLE. Diese Systeme sind deshalb nur in Netzen einsetzbar, in denen eine direkte Erreichbarkeit der Kommunikationspartner gegeben ist. Auf Basis der tupelspacebasierten Ansätze, Lime, Limone und MeshMDL, kann von Applikationen Store-and-Forward Kommunikation realisiert werden, da die Middlewaresysteme ein einfaches ereignisbasiertes Triggern von Methoden aufgrund erscheinender Tupel und somit auch auf Geräten und deren Eigenschaften ermöglichen. Das vorgestellte Middlewaresystem ermöglicht neben der direkten, ereignisbasierten singlehop Interaktion mit benachbarten Geräten auch eine direkte Unterstützung für Store-and-Forward Techniken. Hier existieren aber keine fertigen Lösungen, vielmehr muss auch die Anwendung Entscheidungsroutinen zur Verfügung stellen, um Store-and-Forward Kommunikation zu ermöglichen. XMIDDLE berücksichtigt explizit die Trennung und spätere Wiederherstellung einer Verbindung benachbarter Geräte. Allerdings ist eine Weiterverbreitung von XML-Teilbaumkopien nicht vorgesehen, um somit eine Unterstützung für multihop Propagierung von Informationen zu erhalten. Dafür muss ein neuer Teilbaum erzeugt und die Informationen kopiert werden.

Auf regionaler Interaktion mit Hilfe einer transparenten Kommunikationsschicht basieren Proem, EMMA und GAS. Diese Systeme sind speziell für in der Ausdehnung stark beschränkte ad-hoc Netze konzipiert. Dieses Wissen wird aber nicht für eine Adaption der unteren Schichten verwendet – die Systeme basieren transparent auf einer Transportschicht. Eine anwendungsspezifische Einschränkung der Reichweite von Kommunikationsverfahren, wie es bei dem in dieser Arbeit vorgestellten System möglich ist, bieten diese Systeme nicht. Eine applikationsübergreifende Einschränkung auf die angenommene Netzgröße ließe sich aber bei allen Systeme durchführen, damit die in der Praxis durch Überschneidungen meist größeren Netze dennoch beschränkt bleiben. Nur EMMA berücksichtigt einen möglichen Fehlerfall der Kommunikation. Das System setzt dann aber auf Fluten, was in größeren Netzen nicht tragfähig ist.

STEAM, SOM und iFlow basieren auf speziellen Kommunikationsverfahren. STEAM und SOM setzen Publisher/Subscriber auf Basis positionsbasiertem Flutens ein, da in einem hochmobilen Umfeld andere Kommunikationsstrategien meist fehlschlagen. iFlow nutzt kontrolliertes Fluten, um Daten an

viele Geräte zu verteilen. Diese speziellen Kommunikationsverfahren werden von der vorgestellten Middleware nicht unterstützt. Insbesondere das hochmobile Szenario von STEAM ist nur bedingt realisierbar, da das Erzeugen von Gerätерepräsentanten für sehr kurzfristige Verbindungen nicht geeignet ist. Die Verbreitung von Tupeln bei TOTA lässt sich mit der autonomen Bewegung von MobileStates auf Basis von Store-and-Forward Kommunikation vergleichen, wobei in beiden Fällen Migration und Duplikation verwendbar ist. TOTA Tupel sind aber nur für den Verbreitungsvorgang aktiv, und die Kommunikationsverfahren sind vollständig im propagierten Tupel enthalten. Bei TOTA werden keine schon auf den Geräten vorhandenen Standardverfahren verwendet.

Unterscheiden lassen sich die Systeme auch aufgrund des notwendigen Kopplungsgrades der Geräte untereinander. Der Kopplungsgrad zwischen benachbarten Geräten ist bei den tupelspacebasierten singelhop Systemen sehr hoch, da hier eine permanente Verbindung mit Nachbargeräten eingegangen wird. Bei XMIDDLE ist der Kopplungsgrad geringer, weil durch lokale Kopien und erlaubte unverbundene Operationen auf den Kopien die Nachbargeräte spontan wegfallen können. Proem und GAS setzen einen höheren Kopplungsgrad auch multihop voraus, da die Kommunikation transparent erfolgt und als relativ zuverlässig vorausgesetzt wird. EMMA reduziert den Kopplungsgrad, da ein möglicher Fehlerfall der Kommunikation explizit berücksichtigt und dann auf Fluten zurückgegriffen wird. Bei STEAM und iFlow ist der Kopplungsgrad aufgrund des Flutens sehr gering, iFlow setzt aber ein zusammenhängendes Netz voraus um Informationsquellen auch direkt ansprechen zu können. Auch bei TOTA ist der Kopplungsgrad gering, da sich die Tupel autonom, auch über Netzpartitionen hinaus, verbreiten. Werden aber Maintananceregeln verwendet, ist der Kopplungsgrad höher, da die Quelle die verbreiteten Tupel wieder verändern kann. Die vorgestellte Middlewareplattform besitzt im Vergleich zu den singlehop Systemen aufgrund der Erzeugung von Gerätерepräsentanten nur einen leichten Kopplungsgrad mit direkten Nachbarn. Der Kopplungsgrad der Geräte hängt jedoch bei dieser Middlewareplattform von den darauf realisierten Anwendungen ab. So ist beispielsweise der Kopplungsgrad bei UbiQuiz sehr gering, da der En-Passant Abgleich von Objekten jederzeit abbrechen darf. Bei DistScript hingegen ist er erheblich höher, da im Veranstaltungskontext ein zusammenhängendes Netz und die Erreichbarkeit eines bestimmten Gerätes vorausgesetzt wird.

Insgesamt ist die vorgestellte Middleware flexibler auf gegebene Netzsituationen einrichtbar – und dies auch zur Laufzeit. Somit lässt sie sich auch in spezielleren Netzwerkszenarien der anderen Systeme einsetzen. Ein großer Nachteil ist hier aber der enorme Transparenzverlust. Anwendungen müssen entweder diese Adaptionsaufgabe selbst durchführen, oder es müssen Repräsentanten für diese speziellen Netzwerkszenarien realisiert werden, die dann von einer Anwendung in ähnlich transparenter Weise genutzt werden können, wie in den anderen Systemen.

11

Schlusswort

Die vorliegende Arbeit stellt ein Middlewaremodell für mobile ad-hoc Netzwerke vor. Es wurde ein Ansatz gewählt, der die Dynamik eines mobilen ad-hoc Netzes in sich berücksichtigt. Das ad-hoc Netzwerk, Middleware- und Applikationskomponenten werden als mobile Objekte modelliert, die untereinander ereignisbasiert interagieren. Die Kommunikationsmechanismen, auf deren Basis eine entfernte Interaktion erfolgt, sind nicht transparent in einer Middlewareschicht gekapselt, sondern auf sie kann direkt zugegriffen werden, sie können adaptiert und kombiniert werden. Somit ist auch zur Laufzeit eine Anpassung an eine geänderte Netzumgebung möglich. Durch Informationen über Objektnachbarschaften und Umgebungsinformationen, so genannten Spuren, wird sowohl diese Adaption, als auch die Entscheidung zur Interaktion mit anderen unterstützt. Durch Verbreiten und Sammeln von Spurinformationen entstehen über die Zeit Erfahrungen über das Netz, die in Zukunft eine Entscheidungshilfe liefern können. Spuren beinhalten auch Applikationswissen über eine mögliche Struktur und Eigenschaften des Netzes. Ist eine Anwendung in der Lage, Annahmen über das Netz zu treffen, in dem sie etabliert wurde, kann auch dieses Wissen helfen, besser mit der Dynamik mobiler ad-hoc Netze umzugehen. Mobile Objekte stellen aber nicht nur einzelne Objektinstanzen wie z.B. Dienste oder Anwendungskomponenten dar. Hierbei kann es sich auch um Organisationskonzepte handeln, in denen beispielsweise Geräte mit bestimmten Eigenschaften zusammengefasst werden. Als wichtige Eigenschaft gilt hier der räumliche Zusammenhang über die Zeit, so dass in einer solchen Gerätegruppe Objekte in direkte Interaktionsreichweite mit anderen Objekten der Gruppe gelangen können. In der Arbeit wurden zwei dieser Gruppenorganisationskonzepte definiert. Die lokale Gruppe mit zeitlichem Zusammenhang und die ortsbasierte Gruppe. In beiden Fällen ist eine Kommunikation mit den Gruppenmitgliedern auch auf Basis von Store-and-Forward Kommunikation möglich, mit der sich die Nachricht über die Zeit in der Gruppe verbreiten kann. Als Beispiel für die ad-hoc Kommunikation in

einer ortsbasierten Gruppe wurde der Veranstaltungskontext der Anwendung Distributed Script dargestellt.

Neben den vorgestellten Gruppenkonzepten lassen sich noch weitere Gruppen identifizieren, denen die Geräte des ad-hoc Netzwerkes zugeordnet werden können. Neben den durch Anwendungen definierten Gruppen, die beispielsweise thematische Interessen ausdrücken, werden Geräte auch durch die Installation von Anwendungen Gruppen zugeordnet. Auf Anwendungsebene findet Interaktion primär zwischen Geräten statt, die dieselben Anwendungen installiert haben. Die Applikation UbiBay nutzt zwar eine Vielzahl weiterer Geräte als transparente Router und die Interaktion der Auktionsagenten findet bei UbiBay auf beliebigen Geräten statt – auf Anwendungsebene interagieren aber die Geräte, die die Agenten erzeugt haben. UbiBay definiert also eine Gruppe von Geräten, die aufgrund einer geographischen Nähe zu einem Ort, dem Marktplatz, durch die Nutzung der Anwendung miteinander interagieren. Auch die Verbreitung von Nachrichten über das Informationsradio definiert eine Gerätegruppe. Sie beinhaltet alle Geräte, die sich in der geographischen Region der Propagierung aufhalten. Eine weitere, auf Kommunikation basierende Gruppe ist die der direkten Nachbargeräte. Die Gruppe ist aus der lokalen Sicht der einzelnen Geräte bestimmt, besitzt aber in ad-hoc Netzwerken eine große Bedeutung. Kommunikation mit anderen erfolgt immer über diese Gruppe, und häufig ist die Interaktion völlig auf diese Gruppe beschränkt. Es lassen sich also eine Vielzahl Gruppen identifizieren, in denen sich ein Gerät befinden kann und die über installierte Anwendungen aber auch über Kommunikation definiert sind. Darüber hinaus kann sich ein mobiles Gerät in einer Vielzahl physischer Netze befinden. Schon heute können "Smartphones" und PDAs gleichzeitig über Bluetooth, WLAN, GPRS und UMTS kommunizieren. Auch diese Eigenschaft lässt sich als Gruppenzugehörigkeit modellieren – so kann ein Gerät gleichzeitig Teil eines sehr leistungsschwachen Sensornetzes, mehrerer leistungsstärkerer aber unzuverlässiger physischer ad-hoc Netze, sowie Teil einer zuverlässigen Infrastruktur sein. Diese Gruppen können untereinander interagieren und Synergieeffekte nutzen. So können Gruppen Kommunikationsdienstleistungen für andere erbringen, beispielsweise um Nachrichten über ein Infrastrukturnetz zu kommunizieren oder um von einem Sensornetz Daten an ein Gerät zu liefern, welches über ein ad-hoc Netzwerk angebunden ist. Darüber hinaus können die Gruppen die bisher gruppenintern verwendeten Spurinformationen anderen Gruppen zur Verfügung stellen. Somit kann vorhandenes Wissen über das Netzwerk und seine Eigenschaften auch von anderen genutzt werden. Ein Beispiel ist das Wissen um einen Veranstaltungskontext. Hier kann das Wissen über eine für bestimmte Zeiten dichte und stabile Netzregion zur Etablierung eines Anwendungsmarktplatzes verwendet werden. Um diese Synergieeffekte ausnutzen zu können, muss die Middleware um Mechanismen erweitert werden, die die Synergieverhandlungen der Gruppen ermöglichen und diese anbahnen.

In ad-hoc Netzwerken ist zur multihop Kommunikation nicht ein bestimmtes Routingverfahren das Verfahren der Wahl. Es können stattdessen, je nach Netzwerksituation, unterschiedliche Verfahren zum Einsatz kommen. Dies liegt in der sich ständig ändernden Netzstruktur und den Netzbedingungen begründet. Daher ist es notwendig, mehrere Verfahren zur Auswahl zu haben, die zur Laufzeit gewechselt oder auch parallel betrieben werden können. Die Auswahl kann transparent von einer Middlewareschicht oder durch eine Anwendung erfolgen. Die Anwendung muss aber das notwendige Wissen über die Middlewarekomponenten besitzen oder der Middleware ihre Anforderungen mitteilen können. Aufgrund der freien Wahl der Kommunikationsmechanismen können diese auch zu neuen Kommunikationsmustern zusammengefügt werden. Um eine beliebige Rekombination und Adaption zur Laufzeit zu ermöglichen und auch neue Kommunikationsverfahren einbringen zu können, muss der dafür notwendige Kode mitverbreitet werden. Neben den damit einhergehenden Sicherheitsaspekten steht in ad-hoc Netzwerken auch die Frage nach der Tragfähigkeit des damit verbundenen Kommunikationsaufwandes im Vordergrund. Zumindest die Verbreitung des Kodes von Middlewarebasisdiensten, wie beispielsweise Kommunikationsverfahren, ist aber zwingend erforderlich, wenn neue Mechanismen in einer etablierten Umgebung eingesetzt werden sollen. Ein starres System, wie es in drahtgebundenen Infrastrukturnetzen der Fall ist, ist selbst dort nur bedingt tauglich. Ein Beispiel ist die Einführung von IPv6 [20], die bis heute nicht vollzogen ist. Dies liegt unter anderem an der Notwendigkeit der administrierten Einführung. Auch für Infrastrukturnetze wird über Lösungsmöglichkeiten nachgedacht, eine höhere Flexibilität zu schaffen, beispielsweise auf Basis der Konzepte des Active Routings [77].

Multihop Routing rein auf Basis von drahtloser Kommunikation ist nur bedingt einsetzbar. Aufgrund des geteilten Mediums kann lediglich über sehr wenige Hops direkt kommuniziert werden. Eine einfache Überlegung verdeutlicht dies: kommunizieren alle Knoten im Schnitt über n Hops, steht für jeden Knoten $1/n$ der in drahtlosen Medien sehr geringen Bandbreite zur Verfügung. In der Praxis ist dies noch erheblich weniger, da eine Nachrichtenübertragung auch auf die Knoten in der Nachbarschaft Einfluss hat. Somit ist der Einsatz von multihop Routing meist nur sehr lokal im direkten geographischen Umfeld sinnvoll, oder um hierdurch über wenige Hops Zugang zu einem Infrastrukturnetz zu erhalten. Nachrichten über größere Entfernungen zu transportieren, ist dennoch sinnvoll, wenn es sich nicht um Individualkommunikation handelt, sondern die Kommunikation für die meisten der involvierten Geräte von Nutzen ist. Dürfen bei der Individualkommunikation n unbeteiligte Geräte genutzt werden, können bei m Empfängern $m*n$ Geräte involviert werden. Allerdings wird dies von Multicastverfahren nicht berücksichtigt. Hopbeschränkungen wie im Falle des Unicast sind entweder zu restriktiv oder zu großzügig, da dem Verfahren das Wissen über die Tiefe eines Multicastbaums und über die Verteilung der Gruppenmitglieder fehlt. Multicastkommunikationsverfahren, die mit einer anonymen Gruppenmenge arbeiten, welche beliebig im Netz

verstreut ist, können also diese Kosten/Nutzen-Rechnung für das Gesamtnetz nicht erfüllen. Um dies dennoch zu erreichen, besteht die Möglichkeit, das Wissen über den Aufenthaltsort eines Großteils der Gruppenmitglieder, eine ungefähre Schätzung der Gruppengröße und die zu erwartende Netzdistanz zum Erreichen dieses Ortes zu nutzen. Mit diesem Wissen kann der Locationcast zur Ortskontextkommunikation eingesetzt werden, und der dabei zurückgelegte Unicastweg kann entsprechend länger sein als für eine Individualkommunikation. Ebenfalls ein attraktiver Lösungsansatz ist die Verwendung der En-Passant Kommunikation zur Gruppenkommunikation. Hier kann die benötigte Netzlast drastisch reduziert werden, indem nur für interessierte Geräte Nachrichten dupliziert und kommuniziert werden – meist aber gepaart mit einer drastischen Erhöhung der Auslieferungslatenz. Besitzen die Empfänger aufgrund ihrer Gruppenzugehörigkeit nicht nur eine thematische, sondern auch eine geographische Nähe, kann aber dennoch in kurzer Zeit eine hohe Verbreitung erreicht werden. Da der Nachrichtentransport auch aufgrund der Mobilität der Geräte durchgeführt wird, ist die En-Passant Kommunikation eine Möglichkeit, permanente Netzpartitionen allein auf Basis des ad-hoc Netzwerks zu überwinden. Dies gilt auch für die Verwendung des En-Passant Verfahrens zur Unicastkommunikation. Eine Kombination mit direkten Kommunikationsverfahren kann in zusammenhängenden Netzen die Latenz reduzieren und die Zuverlässigkeit erhöhen. Da Store-and-Forward Mechanismen rein auf lokalem Wissen über das Netz und "gedächtnislos", d.h. unabhängig von der Netzstruktur beim Empfang der Nachricht, operieren sollten, sind hier nur Greedyverfahren sinnvoll einsetzbar. Weil sich die umgebende Netzstruktur eines Gerätes bis zur nächsten Weiterleitung völlig ändern kann, sind die existierenden Planargraphenroutingverfahren hierfür ungeeignet. Daher ist auch der Einsatz der direkten multihop Kommunikation notwendig. Unabhängig von der Store-and-Forward Kommunikation wird sie verwendet, wenn Objekte, die sich nicht in direkter Kommunikationsreichweite befinden, interagieren wollen oder nachrichteneffizient eine Information zu einer Zielregion kommuniziert werden soll. Daraus folgt die Notwendigkeit, beide Kommunikationsvarianten parallel zu betreiben. In der Arbeit wird dieser Notwendigkeit entsprochen und somit können die Vorteile beider Ansätze miteinander kombiniert werden.

Direkte Kommunikation über viele Hops ist auch aufgrund der Netzdynamik nur bedingt einsetzbar, da zusätzlich zur eigentlichen Nachrichtenkommunikation ein in Abhängigkeit der Netzdynamik steigender Protokollaufwand notwendig ist, um die Auslieferung einer Nachricht zu ermöglichen. Selbst bei positionsbasierten Greedyverfahren steigt der Aufwand, da eine höhere Dynamik eine höhere Beaconingfrequenz erfordert, um potenzielle Weiterleitungsziele erkennen zu können. Der Einsatz von reaktivem Beaconing ist aber nur bei geringem Nachrichtenaufkommen sinnvoll, da hier in jedem Routingschritt die singlehop Nachbarschaft angefragt wird. Planargraphroutingverfahren sind bisher aufgrund der starken Annahmen an das Netz nicht praxistauglich. Darüber hinaus gilt für alle zustandsfreien Verfahren mit garantier-

ter Auslieferung, dass sie im schlechtesten Fall mindestens n^2 Routingschritte benötigten, um eine Nachricht auszuliefern (n Länge des kürzesten Pfades) [64]. Aufgrund der dadurch notwendigen Beschränkung der Hopzahl ist in der Praxis selbst bei nahen Zielen unter Umständen keine Erreichbarkeit gegeben. So ist bisher nur eine Kombination von zustandslosen Greedyverfahren mit linkbasierten Verfahren als Rückfall- oder Recoverystrategie in der Praxis zur direkten Kommunikation einsetzbar.

Die vorgestellte Middleware ermöglicht eine flexible Verwendung unterschiedlicher Kommunikationsverfahren. Diese Flexibilität verursacht aber im Gegenzug einen starken Transparenzverlust, der die Anwendungsentwicklung erheblich aufwendiger macht. Die Realisierung der Kommunikation für spezielle Netz- und Anwendungsszenarien auf Basis von Repräsentanten kann einen Teil dieses Aufwandes verbergen. Hier muss eine möglichst allgemeine Realisierung erfolgen, um eine breite Klasse von Anwendungen abdecken zu können und somit ein hohe Wiederverwendbarkeit zu erreichen. Dennoch ist vom Anwendungsentwickler immer noch ein umfangreicheres Wissen gefragt – zum einen, um die passende Kommunikationskomponente zu finden und zum anderen muss er die Funktionalität, die Eigenschaften und insbesondere die Auswirkungen der Kommunikationskomponente auf das Netz kennen.

In Netzen auf Basis personengebundener Geräte kann die En-Passant Kommunikation durch Altruismus und Synergie erheblich verbessert werden. Werden Nachrichten auch von unbeteiligten Geräten weitertransportiert, kann die Auslieferungslatenz reduziert und die Auslieferungswahrscheinlichkeit erhöht werden. Beispielsweise kann die Smallworldeigenschaft sozialer Netze ausgenutzt werden, um den Nachrichtentransport aufgrund von Nutzermobilität zu verbessern. Das Injizieren einer Nachricht in einen oder mehrere Cluster, in denen sich das Ziel potenziell aufhält, kann eine aufwendige Zielsuche vermeiden. Während die Multicastkommunikation innerhalb eines Clusters (entspricht der En-Passant Gruppenkommunikation wie in [44] beschrieben) einfach realisierbar ist, bleiben für weitere Kommunikationsparadigmen noch viele Fragen offen. So können Außenstehende einen Cluster nur erreichen, wenn sie ein Clustermitglied direkt erreichen können oder kennen und wieder treffen. Bei der Multicast- und im Falle der Unicastkommunikation fehlt eine konkretere Zielfindung in diesem zu Routingzwecken eigentlich idealen Smallworldgraphen. Hierfür kann aber, ähnlich dem Milgram-Experiment, in gewissem Maße geographische Information über das Ziel helfen. Eine Weiterleitung erfolgt also auch auf Basis möglicher zukünftiger Aufenthaltsregionen des Ziels. Zu klären ist, inwiefern Hubs in Smallworldnetzen erkannt und nutzbar gemacht werden können. Ein weiteres Problem ist die experimentelle Validierung von Verfahren in solchen Netzen. Gerade für die Modellierung von mobilen ad-hoc Netzen auf Basis sozialer Netze sind Bewegungsmodelle notwendig, welche die Eigenschaften der Netze simulativ abbilden können. Komplexere Mobilitätsmodelle, wie das in der Arbeit vorgestellte Pathnet mit Benutzerstundenplänen, können soziale Netze erzeugen. Sie besitzen aber

den Nachteil, dass das entstandene soziale Netz konstruiert und nicht randomisiert erzeugt wird. Es gibt einige Ansätze (z.B. [86]), die Eigenschaften sozialer Netze in mobilen ad-hoc Netzen nachbilden. In diesem Beispiel wurden Netzcluster auf Basis anonymer Knoten erzeugt, so dass das entstandene ad-hoc Netz permanente Häufungscluster besitzt, die Knoten über die Zeit aber dennoch rein zufällige Nachbarschaften besitzen. Erst vor kurzem wurde dies verbessert [87]. Mit dem dort vorgestellten Community Modell entsteht ein Clustering, welches auch über die Zeit eine bzw. mehrere Gruppenzugehörigkeiten der Geräte ausbildet und darüber hinaus ein Netz erzeugt, das Smallworldeigenschaften besitzt. Mit EpiSims existiert ein Simulator, der die Kleinstadt Portland (USA) und das Verhalten der Bewohner nachbildet. Dieser wurde zur Pockensimulation [4] entwickelt und eingesetzt. Der Simulator wird zwar anderen zur Verfügung gestellt, für seinen Einsatz sind aber umfangreiche Sozial- und Meldedaten einer zu simulierenden Kleinstadt notwendig, da die Portlanddaten aus Datenschutzgründen nicht herausgegeben werden dürfen.

Die lokale Verbreitung und das Sammeln von Spuren lässt sich mit der in selbstorganisierenden Systemen häufig vorkommenden Kommunikationsform *Stigmergy* vergleichen. Hierbei wird über leichte Veränderungen in der direkten Umgebung kommuniziert, um globale Veränderungen im System zu erreichen. Fischschwärme beispielsweise bewegen sich gemeinschaftlich, indem Richtungsänderungen im Schwarm aufgrund von Wellenimpulsen an direkte Nachbarn weitergegeben werden. Ein Schwarm ändert die Richtung, wenn genügend Fische solche Impulse auslösen – beispielsweise bei Angriff eines Feindes (aus [13]). Eine derartige Interaktionsform lässt sich auch auf Basis von Spurpropagierung erreichen. Hierbei werden kurze Informationen über das Gerätebeaconing verbreitet und über eine feste Hopzahl weitergereicht bzw. in aggregierter Form im Hintergrund geflutet. In [73] wurde diese Koordinationsform in ähnlicher Weise untersucht. Hier wird die Information durch direktes und periodisches Fluten propagiert, was zu einer geringen Verbreitungslatenz aber einer hohen Netzlast führt. Diese Organisationsform ist insbesondere in Sensornetzen attraktiv, da hier eine große Gerätemenge aufgrund geringer Informationsmengen koordiniert werden müssen. In gewissem Maße ist dies mit dem Einsatz von Spurinformationen als Hinweise im Beaconing vergleichbar, wie es die Applikation DistScript einsetzt, um eine für die Gerätegruppe globale Verbreitung von Information zu ermöglichen.

Applikationen in ad-hoc Netzwerken können häufig keine hohen Anforderungen an das Netzwerk stellen, da die Dynamik des Netzes diese nicht gewährleisten kann. Daher sind meist nur Applikationen mit hoher Toleranz bezüglich Zuverlässigkeit, Datenkonsistenz und Informationsverlust realisierbar. Eine Applikationsklasse, die diese Toleranz besitzt, basiert primär auf der Verbreitung von Informationen aus einem Informationsraum von großem Interesse, dessen Einzelinformationen aber geringe Wichtigkeit haben. Diese Informationen können daher häufig dupliziert werden, und der Verlust ein-

zelner Informationen ist tolerierbar. Ein Beispiel aus dieser Klasse ist die Applikation UbiQuiz. Hier haben alle Teilnehmer Interesse an großen Teilen des Applikationsinformationsraums, es ist aber nicht notwendig, dass alle Geräte alle für sie interessanten Informationen erhalten, da die Applikation selbst mit einer geringen Teilmenge des Informationsraums arbeiten kann. Tolerant bezüglich Informations- bzw. Nachrichtenverlust ist im gewissen Maße auch die Applikation UbiBay. Da ihr vorgesehenes Einsatzgebiet lokale Auktionen mit geringem Wert sind, kann ein Totalverlust einer Auktion toleriert werden – das Wiedereinstellen der Auktion ist meist unproblematisch. Auch ein Teilverlust von Informationen, beispielsweise der einseitige Verlust einer getroffenen Auktionsvereinbarung, ist unkritisch. Da im nachfolgenden Güteraustausch auf zuverlässigere Kommunikationswege (z.B. eine telefonische Vereinbarung der Übergabe) zurückgegriffen werden sollte, kann es dennoch zu einer Einigung der Teilnehmer kommen. Weitere Applikationsbeispiele dieser Klasse sind öffentliche schwarze Bretter und Mitfahrzentralen, die auch in der herkömmlichen Aushangvariante keine Zuverlässigkeitsgarantie geben können. Interaktive Anwendungen sind in diesen Netzen nur realisierbar, wenn die Nutzerinteraktion mit lokaler Kommunikation auskommt. Über größere Entfernungen ist nach obiger Diskussion eine Individualkommunikation nicht tragfähig. UbiBay erlaubt zwar durch eine Store-and-Forward Unicastkommunikation auch über größere Entfernungen den Transport von Individualnachrichten. Eine direkte Interaktion mit einer gestarteten Auktion ist aber aufgrund von Netzpartition nicht immer möglich oder ist meist von einer großen Latenz geprägt. In der Praxis sollte die Interaktion mit einem Auktionsmarktplatz auch nur über wenige Weiterleitungsschritte erfolgen. Für die Überwindung von größeren geographischen Distanzen ist also die Verwendung der Gerätemobilität notwendig, um Kommunikationsschritte einsparen zu können. Somit ist eine direkte Nutzerinteraktion mit einer Auktion über größere Entfernungen rein auf Basis des ad-hoc Netzes ausgeschlossen. Daher sind auf dieser Basis nur Applikationen realisierbar, für die diese Latenz vernachlässigbar ist und die Nachrichtenverbreitung über Stunden statt wenigen Sekunden erfolgen darf. Ist die Anwendungskommunikation allerdings lokal beschränkt, kann sogar eine höhere Zuverlässigkeit ermöglicht werden. Anwendungen, die ihre primäre Interaktion auf die Orte beschränken, an denen genügend weitere Anwendungsteilnehmer zu finden sind, wie in der Beispielapplikation Distributed Script, können erheblich höhere Ansprüche an das Netz stellen. Garantien können aufgrund der Dynamik und der Mobilität der Teilnehmer aber nie gegeben werden.

Literaturverzeichnis

1. Ahuja, S. ; Carriero, N. ; Gelernter, D.: Linda and Friends. In: *IEEE Computer* 19 (1986), August, S. 26–34
2. Ashbrook, Daniel ; Starner, Thad: Using GPS to learn significant locations and predict movement across multiple users. In: *Personal and Ubiquitous Computing* 7 (2003), Oktober, Nr. 5, S. 275 – 286
3. Barabasi, Albert-Laszlo ; Albert, Reka ; Jeong, Hawoong: Scale-free characteristics of random networks: the topology of the world-wide web. In: *Physica A: Statistical Mechanics and its Applications* 281 (2000), Juni, Nr. 1-4, S. 69–77
4. Barrett, Chris L. ; Eubank, Stephen G. ; Smith, James P.: If smallpox strikes Portland... In: *Scientific American* 292 (2005), März, Nr. 3, S. 42–9
5. Barriere, Lali ; Fraigniaud, Pierre ; Narajanan, Lata ; Opatrny, Jaroslav: Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges. In: *Proceedings of the 5th ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M 01)*, 2001, S. 19–27
6. Benicke, Oliver: *Entwurf und Implementierung eines Managers zum Erfassen und Unterstützen von Feldversuchen in mobilen Ad-Hoc Netzwerken*, Fachhochschule Trier, Diplomarbeit, 2005
7. Blair, Gordon S. ; Coulson, G. ; Robin, P. ; Papathomas, M.: An architecture for next generation middleware. In: *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer-Verlag, 1998
8. Blazevic, Ljubica ; Giordano, Silvia ; Boudec, Jean-Yves L.: Self organized routing in wide area mobile ad-hoc networks. In: *Global Telecommunications Conference (GLOBECOM '01)*, 2001
9. Bose, Prosenjit ; Morin, Pat ; Stojmenovic, Ivan ; Urrutia, Jorge: Routing with Guaranteed Delivery in ad hoc Wireless Networks. In: *Proceedings of the 3rd ACM International Workshop on discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M 99)*. Seattle, WA, August20 1999, S. 48–55
10. Broch, Josh ; Maltz, David A. ; Johnson, David B. ; Hu, Yih-Chun ; Jetcheva, Jorjeta: A Performance Comparison of Multi-Hop Wireless Ad Hoc

Network Routing Protocols. In: *Mobile Computing and Networking*, 1998, S. 85–97

11. Brown, Nat ; Kindel, Charlie: *Distributed Component Object Model Protocol – DCOM/1.0.* Internet draft, Januar 1998
12. Buchegger, Sonja ; Boudec, Jean-Yves L.: Performance Analysis of the CONFIDANT Protocol Cooperation Of Nodes – Fairness In Dynamic Ad-hoc NeTworks. In: *Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC02).* Lausanne, Switzerland, Januar 2002, S. 226–236
13. Camazine, Scott ; Deneubourg, Jean-Louis ; Franks, Nigel R. ; Sneyd, James ; Theraula, Guy ; Bonabeau, Eric: *Self-Organization in Biological Systems.* Princeton University Press, 2003. – ISBN: 0-691-11624-5
14. Capkun, Srdjan ; Hamdi, Maher ; Hubaux, Jean-Pierre: GPS-free Positioning in Mobile Ad Hoc Networks. In: *Cluster Computing* 5 (2002), April, Nr. 2, S. 157 – 167
15. Chao, William ; Macker, Joseph P. ; Weston, Jeffery W.: NRL – Mobile Network Emulator / Naval Research Laboratory. Washington, DC, USA, 2003. – Forschungsbericht
16. Cugola, Gianpaolo ; Jacobsen, H.-Arno: Using publish/subscribe middleware for mobile systems. In: *Mobile Computing and Communications Review* 6 (2002), Oktober, Nr. 4, S. 25 – 33
17. Curino, Carlo ; Giani, Matteo ; Giorgetta, Marco ; Giusti, Alessandro ; Murphy, Amy L. ; Picco, Gian P.: TinyLIME: Bridging Mobile and Sensor Networks through Middleware. In: *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM05)*, 2005, S. 61–72
18. Datta, Anwitaman ; Quarteroni, Silvia ; Aberer, Karl: Autonomous Gossiping: A Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Wireless Mobile Ad-Hoc Networks. In: *Semantics of a Networked World: Semantics for Grid Databases, First International IFIP Conference, ICSNW 2004.* Paris, France, Juni 2004, S. 126
19. db4objects Inc. (Hrsg.): *DB4O – Database for Objects.* San Mateo, California, USA: db4objects Inc., März 2006. – http://www.db4o.com/
20. Deering, S. ; Hinden, R.: *Internet Protocol, Version 6 (IPv6) Specification.* RFC 2460, Dezember 1998
21. Engel, Michael ; Smith, Matthew ; Hanemann, Sven ; Freisleben, Bernd: Wireless Ad-Hoc Network Emulation Using Microkernel-Based Virtual Linux Systems. In: *5th EUROSIM Congress on Modeling and Simulation.* Marne la Vallee, France, 2004, S. 198–203
22. Eppstein, David: Finding the k Shortest Paths. In: *SIAM J. Computing* 28 (1998), Nr. 2, S. 652–673
23. Fall, Kevin ; Varadhan, Kannan: *The ns Manual. The VINT Project – A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC.* http://www.isi.edu/nsnam, 1989-2006
24. Finn, Gregory G.: Routing and Addressing Problems in Large Metropolitan-scale Internetworks / Information Sciences Institute (ISI). 1987 (ISI/RR-87-180). – Forschungsbericht
25. Flynn, Juan ; Tewari, Hitesh ; Mahony, Donal O.: JEmu: A Real-Time Emulation System for Mobile Ad-Hoc Networks. In: *First Joint IEI/IEE Symposium on Telecommunications Systems Research.* Dublin, Ireland, November 2001

26. Fok, Chien-Liang ; Roma, Gruia-Catalin ; Hackmann, Gregory: A Lightweight Coordination Middleware for Mobile Computing. In: *Lecture Notes in Computer Science* 2949 (2004), Januar, S. 135 – 151
27. Franz, Walter: *FleetNet – internet on the road. Ad hoc radio network for inter-vehicle communications.* http://www.et2.tu-harburg.de/fleetnet, Januar 2002
28. Frey, Hannes: *Geographisches Routing – Grundlagen und Basisalgorithmen*, Universität Trier, Diss., November 2006
29. Frey, Hannes ; Görgen, Daniel: Geographical Cluster Based Routing in Sensing-Covered Networks. In: *Proceedings of the 2nd International Workshop on Wireless Ad Hoc Networking (WWAN 2005).* Columbus, Ohio, USA : IEEE Computer Society, Juni 6–9 2005, S. 885–891
30. Frey, Hannes ; Görgen, Daniel: Planar graph routing on geographical clusters. In: *Ad Hoc Networks, Special issue on Data Communication and Topology Control in Ad Hoc Networks* 3 (2005), September, Nr. 5, S. 560–574
31. Frey, Hannes ; Görgen, Daniel: Geographical Cluster based Routing in Sensing-Covered Networks. In: *IEEE Transactions on Parallel and Distributed Systems: Special issue on Localized Communication and Topology Protocols for ad hoc Networks* 17 (2006), April, Nr. 4
32. Frey, Hannes ; Görgen, Daniel ; Lehnert, Johannes K. ; Sturm, Peter: Erfahrungsbericht zur praktischen Umsetzung eines Auktionssystems für großflächige mobile multihop Ad-hoc-Netzwerke. In: *GI Betriebssysteme, Frühjahrstreffen 2003.* Erlangen, Germany, April 3–4 2003
33. Frey, Hannes ; Görgen, Daniel ; Lehnert, Johannes K. ; Sturm, Peter: A Java-based uniform workbench for simulating and executing distributed mobile applications. In: *FIDJI 2003 International Workshop on Scientific Engineering of Distributed Java Applications.* Luxembourg, 2003, S. 116 – 127
34. Frey, Hannes ; Görgen, Daniel ; Lehnert, Johannes K. ; Sturm, Peter: Auctions in mobile multihop ad-hoc networks following the marketplace communication pattern. In: *Wireless Information Systems 2004.* Porto, Portugal, 2004, S. 161–169
35. Frey, Hannes ; Lehnert, Johannes K. ; Görgen, Daniel ; Sturm, Peter: A generic background dissemination service for mobile ad-hoc networks / University of Trier, Germany. 2004 (TR-04-01). – Forschungsbericht
36. Frey, Hannes ; Stojmenovic, Ivan: Geographic and Energy aware Routing in Sensor Networks. In: Stojmenovic, Ivan (Hrsg.): *Handbook on Sensor Networks (ISBN 0-471-68472-4).* Wiley, Oktober 2005
37. Garbarenko, Yevgen: *Informationsverteilung in mobilen Ad-hoc-Netzwerken*, Universität Trier, Diplomarbeit, November 2004
38. Gast, Matthew S.: *802.11 Wireless Networks.* 2nd. O'Reilly, 2005. – ISBN 0-596-10052-3
39. Gerla, Sung-Ju L. ; Chiang, M. Ching-Chuan: On-demand multicast routing protocol. In: *Wireless Communications and Networking Conference* Bd. 3. New Orleans, LA, USA, September 1999, S. 1298–1302
40. Giordano, S. ; Hamdi, M.: Mobility management: The virtual home region / EPFL, Lausanne, Switzerland. 1999 (SSC/1999/037). – Forschungsbericht
41. Görgen, Daniel: *Marktplatz-basierte Agreementprotokolle in Ad-hoc-Netzen*, Universität Trier, Diplomarbeit, 2002
42. Görgen, Daniel ; Frey, Hannes ; Hiedels, Christian: JANE – A Simulation Platform for Ad Hoc Network Applications. In: *to appear in Demos of the 9th*

International Symposium of Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2006

43. Görgen, Daniel ; Frey, Hannes ; Hiedels, Christian: JANE – The Java Ad hoc Network Environment. In: *in preparation for the 40th Annual Simulation Symposium*, 2007
44. Görgen, Daniel ; Frey, Hannes ; Hutter, Christian: Information dissemination based on the en-passant communication pattern. In: *Fachtagung "Kommunikation in Verteilten Systemen" (KiVS)*. Kaiserslautern, Germany, Apr 2005, S. 129–141
45. Görgen, Daniel ; Frey, Hannes ; Lehnert, Johannes K. ; Sturm, Peter: Marketplaces as Communication Patterns in Mobile Ad-Hoc Networks. In: *Kommunikation in Verteilten Systemen (KiVS 03)*. Leipzig, Germany, Apr 2003, S. 183–194
46. Görgen, Daniel ; Lehnert, Johannes K. ; Frey, Hannes ; Sturm, Peter: SELMA: A Middleware Platform for Self-Organzing Distributed Applications in Mobile Multihop Ad-hoc Networks. In: *Communication Networks and Distributed Systems Modeling and Simulation CNDS'04*. San Diego, California, USA, 2004
47. Görgen, Daniel ; Transier, Matthias ; Sturm, Peter ; Effelsberg., Wolfgang: Distributed Script – Prototyping a Mobile Application for Multi-hop Ad-hoc Networks. / Universität Trier. Trier, Germany, März 2006 (TR-06-02). – Forschungsbericht
48. Granelli, Fabrizio ; Kliazovich, Dzmitry: Cross-layering for performance improvement in multi-hop wireless networks. In: *8th International Symposium on Parallel Architectures,Algorithms and Networks. ISPAN 2005*, 2005, S. 6–
49. Heikki, Helin ; Pauli, Misikangas: Supporting Seamless Roaming in FIPA Nomadic Application Architecture. In: *Workshop on AI in Mobile Systems (AIMS '01)*. Seattle, USA, August 2001
50. Heinzelman, Wendi R. ; Kulik, Joanna ; Balakrishnan, Hari: Adaptive protocols for information dissemination in wireless sensor networks. In: *5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999
51. Herrmann, Klaus: MESHMdl – A Middleware for Self-Organization in Ad Hoc Networks. In: *23rd International Conference on Distributed Computing Systems*. Providence, Rhode Island, USA, Mai 2003
52. Hightower, Jeffrey ; Borriello, Gaetano: Location systems for ubiquitous computing. In: *Computer* 34 (2001), August, Nr. 8, S. 57–66
53. Hirschler, Jan: *Synthese von Szenarien zur Simulation von Ad-hoc-Netzwerken*, Universität Trier, Diplomarbeit, 2003
54. Hutter, Christian: *ADS an Ad-hoc-Directoryservice for JANE*. persönliche Kommunikation, 2006
55. Jain, Rahil ; Puri, Anuj ; Sengupta, Raja: Geographical Routing Using Partial Information for Wireless Ad Hoc Networks. In: *IEEE Personal Communication* (2001), Februar, S. 48–57
56. Johnson, David B. ; Maltz, David A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: *Mobile Computing*. Kluwer Academic Publishers, 1996
57. Kameas, A. ; Mavrommati, I. ; Ringas, D. ; Wason, P.: eComP: an architecture that supports P2P networking among ubiquitous computing devices. In: *Second International Conference on Peer-to-Peer Computing. (P2P 2002)*. Linköping, Schweden, September 2002, S. 57– 64

58. Kameas, Achilles ; Bellis, Stephen ; Mavrommati, Irene ; Delaney, Kieran ; Colley, Martin ; Pounds-Cornish, Anthony: An Architecture that Treats Everyday Objects as Communicating Tangible Components. In: *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*. Dallas-Fort Worth, USA, März 2003, S. 115–
59. Kleinrock, Leonard ; Silvester, John: Optimum transmission radii for packet radio networks or why six is a magic number. In: *National Telecommunications Conference (NTC78)* Bd. 1. Birmingham, Alabama, USA, Dezember 1978, S. 4.3.1–4.3.5.
60. Ko, Young-Bae ; Vaidya, Nitin H.: Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In: *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98)*. Dallas, TX, USA : ACM Press, Oktober 25–30 1998, S. 66–75
61. Ko, Young-Bae ; Vaidya, Nitin H.: Geocasting in mobile ad hoc networks: location-based multicastalgorithms. In: *Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA99)*. New Orleans, LA, USA, Februar 1999
62. Kortuem, Gerd ; Schneider, Jay ; Preuitt, Dustin ; Thompson, Thaddeus G. C. ; Fickas, Stephen ; Segall, Zary: When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad hoc Networks. In: *First International Conference on Peer-to-Peer Computing (P2P'01)*. Linköpings, Schweden, August 2001, S. 75–85
63. Kranakis, Evangelos ; Singh, Harvinder ; Urrutia, Jorge: Compass Routing on Geometric Networks. In: *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*. Vancouver, August 1999, S. 51–54
64. Kuhn, Fabian ; Wattenhofer, Roger ; Zollinger, Aaron: Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. In: *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing & Communications (DIAL M 02)*. New York : ACM Press, September 28 2002, S. 24–33
65. Ledoux, T.: OpenCorba: A reactive open broker. In: *Springer LNCS* 1616 (1999), S. 197ff
66. Lehnert, Johannes K.: *Entwicklungs- und Simulationsunterstützung fur mobile Anwendungen in multihop Ad-Hoc-Netzen*, Universität Trier, Diss., Juli 2004
67. Lehnert, Johannes K. ; Görgen, Daniel ; Frey, Hannes ; Sturm, Peter: A Scalable Workbench for Implementing and Evaluating Distributed Applications in Mobile Ad Hoc Networks. In: *Western Simulation MultiConference WMC'04*. San Diego, California, USA, 2004
68. Lenz, Markus: *Integration einer realitätsnahen MAC-Layer für Wireless LAN in eine Simulationsumgebung für multihop Ad-Hoc-Netze*, Universität Trier, Diplomarbeit, 2006
69. Li, Jinyang ; Blake, Charles ; Couto, Douglas S. J. D. ; Lee, Hu I. ; Morris, Robert: Capacity of Ad Hoc Wireless Networks. In: *7th ACM International Conference on Mobile Computing and Networking (MobiCom01)*. Rome, Italy, Juli 2001, S. 61 – 69
70. Li, Zongpeng ; Li, Baochun ; Xu, Dongyan ; Zhou, Xin: iFlow: Middleware-assisted Rendezvous-based Information Access for Mobile Ad Hoc Applications. In: *International Conference on Mobile Systems, Applications, and Services (MobiSys03)*. San Francisco, CA, USA, Mai 2003

71. Lochert, Chr. ; Hartenstein, H. ; Tian, J. ; Füssler, H. ; Hermann, D. ; Mauve, M.: A routing strategy for vehicular ad hoc networks in city environments. In: *IEEE Intelligent Vehicles Symposium*. Columbus, Ohio, Juni 2003
72. Madueño, Maribel ; Vidal, Josep: Joint physical-MAC Layer design of the broadcast protocol in ad hoc networks. In: *IEEE Journal on Selected Areas in Communications* 23 (2005), Januar, Nr. 1, S. 65– 75
73. Mamei, Marco ; Zambonelli, Franco: Programming stigmergic coordination with the TOTA middleware. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS05)*. Netherlands, 2005, S. 415–422
74. Mamei, Marco ; Zambonelli, Franco ; Leonardi, Letizia: Tuples On The Air: A Middleware for Context-Aware Computing in Dynamic Networks. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCSW 03)*. Washington, DC, USA : IEEE Computer Society, 2003, S. 342
75. Mamoud, Qusay H.: *Getting Started With JavaSpaces Technology: Beyond Conventional Distributed Programming Paradigms*. Santa Clara, California, USA: Sun Microsystems, Inc., Juli 2005. – http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/
76. Mascolo, Cecilia ; Capra, Licia ; Zachariadis, Stefanos ; Emmerich, Wolfgang: XMIDDLE: A Data-Sharing Middleware for Mobile Computing. In: *In Personal and Wireless Communications Journal* 21 (2002), April, Nr. 1, S. 77 – 103
77. Maxemchuk, Nicholas F. ; Low, Steven H.: Active routing. In: *Selected Areas in Communications* 19 (2001), März, Nr. 3, S. 552–565
78. Mayrhofer, Rene: An architecture for context prediction. In: *Advances in Pervasive Computing*, 2004, S. 65–72
79. Meier, Rene ; Cahill, Vinny: STEAM: Event-Based Middleware for Wireless Ad Hoc Networks. In: *in Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*. Vienna, Austria, 2002, S. 639–644
80. Michiardi, Pietro ; Molva, Refik: Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: *Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, 2002, S. 107 – 121
81. Milgram, Stanley: The Small World Problem. In: *Psychology Today* (1967), S. 60–67
82. Miller, Brent A. ; Bisdikian, Chatschick: *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice Hall PTR, 2001. – ISBN 0-13-090294-2
83. Moore, Gordon E.: Cramming more components onto integrated circuits. In: *Electronics* 38 (1965), April 19, Nr. 8
84. Morais, Cordeiro Carlos d. ; Gossain, Hrishikesh ; Agrawal, Dharma P.: Multicast over wireless mobile ad hoc networks: present and future directions. In: *IEEE Network* 17 (2003), Februar, Nr. 1, S. 52– 59
85. Murphy, A.L. ; Picco, G.P. ; Roman, G.-C.: LIME: A middleware for physical and logical mobility. In: *Proc. of the 21st Int. Conf. on Distributed Computing Systems (ICDCS01)*, 2001, S. 524–533

86. Musolesi, Mirco ; Hailes, Stephen ; Mascolo, Cecilia: An Ad Hoc Mobility Model Founded on Social Network Theory. In: *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Venice, Italy : ACM Press, October 2004, 20-24
87. Musolesi, Mirco ; Mascolo, Cecilia: A Community based Mobility Model for Ad Hoc Network Research. In: *Proceedings of the 2nd ACM/SIGMOBILE International Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN'06)*, ACM Press, May 2006
88. Musolesi, Mirco ; Mascolo, Cecilia ; Hailes, Stephen: EMMA: Epidemic Messaging Middleware for Ad hoc networks. In: *Personal Ubiquitous Computing* 10 (2006), Februar, Nr. 1, S. 28–36
89. Ni, Sze-Yao ; Tseng, Yu-Chee ; Chen, Yuh-Shyan ; Sheu, Jang-Ping: The broadcast storm problem in a mobile ad hoc network. In: *In Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM99)*. Seattle, Washington, USA., August 1999, S. 151–162
90. Object Management Group: Common Object Request Broker Architecture: Core Specification. 2004 (3.0.3 Editorial changes). – Forschungsbericht. – http://www.omg.org/cgi-bin/doc?formal/04-03-12
91. Perkins, Charles ; Bhagwat, Pravin: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In: *Proceedings of the ACM International Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*, 1994, S. 234–244
92. Perkins, Charles E. ; Royer, Elizabeth M.: Ad-hoc On-Demand Distance Vector Routing. In: *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*. New Orleans, LA, Februar 1999, S. 90–100
93. Petzold, Jan ; Bagci, Faruk ; Ungerer, Wolfgang T. ; Vintan, Lucian: Global State Context Prediction Techniques Applied to a Smart Office Building. In: *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS04)*. San Diego, CA, USA, Januar 2004
94. Ramanathan, S. ; Steenstrup, Martha: A Survey of Routing Techniques for Mobile Communications Networks. In: *Mobile Networks and Applications* 1 (1996), Nr. 2, S. 89–104
95. Rappaport, Theodore S.: *Wireless Communication: Principles and Practice*. 2nd. Prentince Hall, 2002. – ISBN 0-13-042232-0
96. Royer, Elizabeth M. ; Perkins, Charles E.: Multicast operation of the ad-hoc on-demand distance vector routing protocol. In: *5th annual ACM/IEEE international conference on Mobile computing and networking*. Seattle, Washington, United States, 1999, S. 207 – 218
97. Royer, Elizabeth M. ; Toh, Chai-Koeng: A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. In: *IEEE Personal Communications* 6 (1999), April, Nr. 2, S. 46–55
98. Sagduyu, Yalin Evren ; Ephremides, Anthony: Crosslayer design for distributed MAC and network coding in wireless ad hoc networks. In: *International Symposium on Information Theory, 2005. ISIT 2005. Proceedings.*, 2005, S. 1863– 1867
99. Shakkottai, Sanjay ; Rappaport, Theodore S. ; Karlsson, Peter C.: Cross-layer design for wireless networks. In: *Communications Magazine, IEEE* 41 (2003), Oktober, Nr. 10, S. 74– 80

100. Soni, Samit ; Gupta, Rakesh ; Pirkul, Hasan: Survivable Network Design: The State of the Art. In: *Information Systems Frontiers* 1 (1999), Oktober, Nr. 3, S. 303 – 315
101. Sørensen, Carl-Fredrik ; Wu, Maomao ; Sivaharan, Thirunavukkarasu ; Blair, Gordon ; Okanda, Paul ; Friday, Adrian ; Limon, Hector D.: A Context-Aware Middleware for Applications in Mobile Ad Hoc Environments. In: *2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC04)*, 2004
102. Stojmenovic, Ivan ; Russell, Mark ; Vukojevic, Bosko: Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks. In: *Computers and Informatics* 21 (2002), Nr. 2, S. 149–165
103. Sturm, Peter ; Frey, Hannes ; Görgen, Daniel ; Lehnert, Johannes K.: Supporting Smart Applications in Multihop Ad-Hoc Networks - The GecGo Middleware -. In: *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'04)*. Wellington, New Zealand, 2004, S. 718–726
104. Takagi, Hideaki ; Kleinrock, Leonard: Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. In: *IEEE Transactions on Communications* 32 (1984), März, Nr. 3, S. 246–257
105. Tian, Jing ; Haehner, Joerg ; Becker, Christian ; Stepanov, Illya ; Rothermel, Kurt: Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation. In: *35th Annual Simulation Symposium*, 2002, S. 337–
106. Transier, Matthias ; Füssler, Holger ; Widmer, Jörg ; Mauve, Martin ; Effelsberg, Wolfgang: A Hierarchical Approach to Position-Based Multicast for Mobile Ad-hoc Networks. In: *Wireless Networks (to appear)* (2006)
107. Varga, András: The OMNeT++ Discrete Event Simulation System. In: *European Simulation Multiconference (ESM'2001)*. Prague, Czech Republic., Juni 2001
108. Vukojevic, Bosko ; Stojmenovic, Ivan: A Routing Strategy and Quorum Based Location Update Scheme for Ad Hoc Wireless Networks / Comuter Science, SITE, University of Ottawa. 1999 (TR-99-09). – Forschungsbericht
109. Weiser, Marc: The computer of the 21st century. In: *Scientific American* 265 (1991), September, Nr. 3, S. 94–104
110. Williams, Brad ; Camp, Tracy: Comparison of broadcasting techniques for mobile ad hoc networks. In: *3rd ACM international symposium on Mobile ad hoc networking & computing*. Lausanne, Switzerland, 2002, S. 194 – 205
111. Xie, Jason ; Talpade, Rajesh R. ; McAuley, Anthony ; Liu, Mingyan: AMRoute: Ad Hoc Multicast Routing Protocol. In: *Mobile Networks and Applications* 7 (2002), Dezember, Nr. 6, S. 429 – 439
112. Yau, Stephen S. ; Karim, Fariaz: Reconfigurable Context-Sensitive Middleware for ADS Applications in Mobile Ad-Hoc Network Environments. In: *Proc. 5th IEEE Int'l Symp. on Autonomous Decentralized Systems (ISADS 2001)*. Dallas, USA, März 2001, S. 319–326
113. Yau, Stephen S. ; Karim, Fariaz: An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments. In: *Real-Time Systems* 26 (2004), Nr. 1, S. 29–61
114. Zeng, Xiang ; Bagrodia, Rajive ; Gerla, Mario: GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In: *Workshop on Parallel and Distributed Simulation*. Banff, Alberta, Canada, Mai 1998, S. 154–161