

**Studiengang:** Softwaretechnik  
**Betreuer:** Dipl.-Inform. Jörg Hähner  
**Prüfer:** Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel

**Beginn am:** 01.06.2003  
**Beendet am:** 02.01.2004

**CR-Klassifikation:** C.2.2., C.4., E.1., H.2.4.

Diplomarbeit Nr. 2115

## **Räumlich begrenzte Datenreplikation in mobilen ad hoc Netzwerken**

Mario Neynens

Fakultät Elektrotechnik, Informatik,  
Informationstechnik  
Universität Stuttgart  
**Institut für Parallele und Verteilte Systeme**  
**Abteilung Verteilte Systeme**  
Universitätsstr. 38  
70569 Stuttgart

# Inhaltsverzeichnis

1 Einleitung.....	6
2 Umgebungsmodell.....	8
2.1 Informelle Definition eines räumlichen Scopes.....	8
2.2 Modellierungsansätze.....	9
2.3 Bewertung der Modellierungsansätze.....	10
2.3.1 Geometrisches Modell.....	10
2.3.2 Mengenbasiertes Modell.....	12
2.3.3 Hybrides Modell.....	14
2.3.4 Bewertung.....	15
2.4 Hierarchische mengenbasierte Umgebungsmodelle.....	17
2.4.1 Space-Tree.....	17
2.4.2 Lattice.....	18
2.4.3 Bewertung.....	20
3 Scopes.....	21
3.1 Formelle Definition.....	21
3.2 Anforderungen an das Umgebungsmodell.....	22
3.3 Ausdehnungsänderung eines Scopes.....	23
3.4 Positionsänderung.....	25
3.5 Darstellung.....	26
3.5.1 Scopelevel.....	27
4 Consistent Update Diffusion-Verfahren (CUD).....	29
4.1 Systemmodell des CUD.....	29
4.2 Anforderungen für eine Scopeverwaltung im CUD.....	30
5 Scopeverwaltung im CUD.....	32
5.1 Problemstellung.....	32
5.2 Idee.....	32

5.3 erweiterter Objektzustand.....	33
5.4 Verfahren.....	34
5.4.1 Scopebehandlung beim Observer:.....	34
5.4.2 Scopebehandlung beim Datenbankknoten:.....	34
5.5 Schichtenmodell.....	37
5.6 Dynamische Scopes.....	38
5.7 Pufferstrategien.....	41
5.7.1 Ringpuffer.....	41
5.7.2 Timeoutpuffer.....	42
5.7.3 TimeoutGrowingpuffer .....	44
5.7.4 Zusammenfassung.....	45
5.8 Scopeeinfluss auf die Datenhaltung.....	46
6 Simulation.....	48
6.1 Verfahren.....	48
6.2 Umgebungsmodell.....	49
6.3 Bewegung beobachtbarer Objekte.....	51
6.4 Hintergrundlast.....	53
6.5 Bewegungsmodell der Datenbankknoten.....	53
6.6 Anfragemodell.....	54
6.7 Einfluss des Scopepuffers.....	55
6.8 Auswertung.....	55
6.8.1 Bewertung des Scopewachstums.....	55
6.8.2 Szenarien.....	60
6.8.3 Vergleich optimaler Datenbankknoten.....	62
6.8.4 Vergleich optimaler mit simulierten Datenbankknoten.....	65
6.8.5 Betrachtung versendeter Nachrichten .....	67
6.8.6 Betrachtung verpasster Observationen.....	70
7 Verwandte Arbeiten.....	73
7.1 DREAM.....	73

7.2 GeoCast.....	74
8 Weiterführende Arbeiten.....	75
8.1 Lattice als Umgebungsmodell.....	75
8.2 Scopeanpassung aufgrund von Netzwerklast.....	75
8.3 Analyse weiterer Pufferstrategien.....	75
8.4 Weitere Simulationskonfigurationen.....	76
8.5 Künstliche Scopeausdehnung durch Datenbankknoten.....	76
8.6 Problemfall logische Partitionen.....	77
9 Zusammenfassung.....	78
10 Referenzen.....	83

## Abbildungsverzeichnis

Abbildung 1: Space-Tree Beispiel.....	17
Abbildung 2: Lattice-of-Locations Beispiel.....	18
Abbildung 3: Scope als Attribut.....	22
Abbildung 4: Ebenen im Space-Tree.....	24
Abbildung 5: Systemmodell CUD.....	30
Abbildung 6: Schichtenmodell.....	37
Abbildung 7: Simulation und Auswertung.....	49
Abbildung 8: Hierarchie der simulierten Umgebung.....	50
Abbildung 9: Space-Tree der simulierten Umgebung.....	50
Abbildung 10: Beobachtungsmuster der Observer.....	52
Abbildung 11: Profil der Scopegrößenentwicklung für einen DB-Knoten und ein Objekt.....	56
Abbildung 12: Anteile der Scopegrößen.....	58
Abbildung 13: Verlauf der Scopegrößenentwicklung bei optimalen Datenbankknoten.....	64
Abbildung 14: Abweichung optimale / simulierte DB-Knoten.....	66
Abbildung 15: Auswertung versendeter Nachrichten pro Knoten.....	69
Abbildung 16: Anteil verpasster Observationen.....	71

## Tabellenverzeichnis

Tabelle 1: Vergleich der Umgebungsmodelle.....	16
Tabelle 2: Beispiele für Scopelevel.....	27
Tabelle 3: 1. Quadrant des Umgebungsmodells.....	57
Tabelle 4: Positionen der Objektbeobachtungen.....	57
Tabelle 5: Mögliche Scopes.....	58
Tabelle 6: Simulationsparameter.....	61
Tabelle 7: Simulationskonfigurationen.....	62
Tabelle 8: Berechnungsparameter für optimale Datenbankknoten.....	63

# 1 Einleitung

In dieser Arbeit wird das „Consistent-Update-Diffusion“ (CUD)-Verfahren erweitert, das einen schwachen Konsistenzbegriff bei vollständiger Datenreplikation in mobilen adhoc Netzwerken (MANET) etabliert. Detaillierte Informationen, des in der Abteilung Verteilte Systeme der Universität Stuttgart entwickelten Verfahrens, finden sich in [RHB03].

Ein Schwachpunkt dieses Verfahrens ist, dass die Informationsverteilung ohne Berücksichtigung von äußeren Umständen, wie zum Beispiel der Belastungssituation der Netzwerkinfrastruktur, stattfindet. Das heißt, falls ein Knoten Daten empfängt, welche zur Weiterleitung an andere Knoten vorgesehen sind, wird diese Weiterleitung sofort durchgeführt. Diese Vorgehensweise wird „plain flooding“ genannt. Ein Nachteil hierbei ist, dass bei einer großen Anzahl von Knoten, welche Daten weiterleiten, eine deutliche Steigerung des Netzwerkverkehrs auftritt. Die Belastung des Netzwerks wirkt sich negativ auf die Weiterleitung von Nachrichten aus, so dass diese verzögert oder sogar vollkommen verhindert wird. Diese Situation kann durch andere Flooding-Verfahren, wie zum Beispiel dem counter-basierten Ansatz [NTC+99] etwas verbessert werden, bewirkt allerdings keine Änderung in dem Maße, wie es wünschenswert wäre.

Die vorliegende Ausarbeitung stellt eine Erweiterung des „Consistent Update Diffusion“-Verfahrens vor, die diesem Sachverhalt entgegenwirken soll. Hierbei wird die Lokalität von Informationen mit berücksichtigt. Die grundlegende Annahme, hierfür ist, dass Informationen im Normalfall in der räumlichen Nähe des Ortes verwendet werden an dem sie generiert werden.

Der Erfassungsort dient als Grundlage zur Festlegung eines Bereiches in dem Informationen Gültigkeit besitzen. Aufbauend auf dieser Annahme soll sich die Information nur in diesem räumlichen Bereich ausbreiten. Ein solcher Bereich wird als Scope bezeichnet.

Als mögliches Anwendungsszenario dient eine Großbaustelle, auf der mehrere Gebäude errichtet werden. Während der Bauphase steht keine feste Netzwerkinfrastruktur zur Verfügung, weshalb die drahtlose Kommunikation eingesetzt wird. Weiterhin wird mit Hilfe von verschiedensten Sensoren erreicht, dass sich die Zustände und Positionen von Bauarbeitern, Materialien und Werkzeugen erfassen lassen. Unter Verwendung dieser Daten wird ein Informationssystem aufgebaut, welches es einem Bauleiter ermöglicht, sich einen Überblick über die aktuelle Situation und den Fortschritt der Arbeiten zu verschaffen. So könnte er zum Beispiel mit Hilfe dieses Systems Fachkräfte ausfindig machen und deren Arbeiten gezielt koordinieren oder die Position von Spezialwerkzeugen erfassen, welche an verschiedenen Stellen benötigt werden, aber nur in begrenzter Anzahl vorhanden sind. Neben den Personen und Werkzeugen lassen sich auch die Zustände der verbauten oder noch zu

verbauenden Baumaterialien überwachen. Dies ließe zum Beispiel folgende Anfragen zu: „Ist der Beton in einem bestimmten Bereich schon genügend getrocknet?“ oder: „Sind die Fenster innerhalb eines Stockwerkes bereits alle montiert worden?“.

Diese Informationen werden baustellenweit gesammelt und mit Hilfe drahtloser Übertragungstechniken verbreitet. Die Informationen, welche in Bezug zu einem bestimmten Gebäude stehen, breiten sich nur innerhalb dieses Bereiches aus. Auf diese Weise würde der Bauleiter ein angepasstes Bild der Bausituation erhalten, je nachdem bei oder in welchem Gebäude er sich gerade befindet. Eine weitere Unterteilung ließe sich dann beispielsweise zwischen einzelnen Stockwerken errichten, so dass der Ausbreitungsbereich für Informationen nicht mehr das gesamte Gebäude umfasst.

Folgende Fragestellungen werden in dieser Arbeit konkret behandelt:

- Welche Umgebungsmodelle eignen sich um einen Scope in dem gegebenen Umfeld zu definieren?
- Wie und von wem kann ein Scope definiert werden?
- Ändert der Scope sich über die Zeit?
- Was passiert wenn ein Knoten einen Scope betritt oder verlässt?

Diese Arbeit ist wie folgt aufgebaut: In Kapitel 2 unterschiedliche Ansätze für den Aufbau von Umgebungsmodellen analysiert. Hierbei wird besonders hervorgehoben, inwieweit sich bestimmte Ansätze für den Einsatz der räumlich begrenzten Datenreplikation einsetzen lassen. In Kapitel 3 wird näher auf die besonderen Eigenschaften eingegangen, welche ein räumlich begrenzter Bereich aufweisen muss, um sinnvoll für die Einschränkung der Datenausbreitung eingesetzt werden zu können. Kapitel 4 stellt das Consistent-Update-Diffusion (CUD)-Verfahren vor, welches um die räumlich begrenzte Datenreplikation erweitert werden soll. Wie diese Erweiterung durchgeführt werden kann und welche Änderungen sowie zusätzliche Anforderungen hierdurch entstehen, wird in Kapitel 5 näher betrachtet. Weiterhin werden in diesem Kapitel verschiedene Ansätze zur Verwaltung von Scopeinformationen vorgestellt und deren unterschiedliche Auswirkungen auf die Datenausbreitung näher betrachtet. Kapitel 6 beschreibt die zur Evaluation durchgeführten Simulationen und erläutert Ergebnisse und die von daraus gewonnenen Erkenntnisse. In Kapitel 7 werden einige Arbeiten vorgestellt, welche in thematischer Verwandtschaft zu der in dieser Arbeit vorgestellten Vorgehensweise stehen. Welche dieser Arbeiten, aufbauend auf den geschaffenen Grundlagen und Erkenntnissen, als interessante weiterführende Alternativen in Frage kommen, werden in Kapitel 8 näher erläutert. In Kapitel 9 werden abschließend die in dieser Arbeit gewonnenen Erkenntnisse zusammengefasst und ein Fazit gezogen.

## **2 Umgebungsmodell**

Ein Umgebungsmodell stellt die Verbindung zwischen der realen Welt und einer Anwendung dar. Hierfür wird innerhalb eines Umgebungsmodells die reale Welt in einem bestimmten Detaillierungsgrad nachgebildet. Wie detailliert ein solches Modell sein muss, hängt von der Anwendung ab, in der es zum Einsatz kommt. Dieses Kapitel beschäftigt sich mit der Fragestellung, inwiefern bestimmte Umgebungsmodelltypen für die auf räumliche Bereiche beschränkte Informationsausbreitung in mobilen ad hoc Netzwerken geeignet sind. Um einen Bezug zwischen einem eingeschränkten räumlichen Bereich und der modellierten Umgebung herstellen zu können, liefert der folgende Abschnitt eine erste informelle Definition.

### **2.1 Informelle Definition eines räumlichen Scopes**

Ein räumlicher Scope oder kurz Scope definiert einen räumlich eingeschränkten Gültigkeitsbereich. Diese Eigenschaft wird für Informationen über Objekte verwendet, um zu beschreiben, wo diese gültig beziehungsweise ungültig sind. Hierbei legt die Gültigkeit der Information den Bereich fest, in dem diese verbreitet werden soll. Auf die reale Welt bezogen ist ein Scope ein abstrakter Raum. Dieser Raum muss keinen Bezug zu real existierenden Räumen haben und ist auch nicht beschränkt in seiner Ausdehnung. Insbesondere bedeutet dies, dass sich die Ausdehnung und Position eines Scopes im Verlauf einer Anwendung verändern kann.

Jeder Information kann ein beliebiger Scope zugeordnet werden. Bezogen auf das in Kapitel 1 vorgestellte Szenario würde könnte zum Beispiel die Position von Facharbeitern verfolgt werden soll. Je nachdem, ob dies auf Gebäude- oder Etageebene geschieht, wird den jeweiligen Personen entweder die Etage oder das Gebäude als Scope zugeordnet, in denen diese sich aufhalten, um so beispielsweise eine Koordination der Personen zu ermöglichen. Durch die Bewegung der Personen zwischen Etagen und Gebäuden würden die entsprechenden Scopes dann jeweils angepasst werden.

Eine andere Möglichkeit wäre, die Eigenschaften von Baumaterialien innerhalb einzelner Räume zu betrachten. So könnten entsprechende Sensoren erfassen, ob in einem Raum die Wandfarbe schon getrocknet ist. Diese erfasste Information bekäme als Scope den Raum zugeordnet, in dem diese erfasst wurde. Zur Kontrolle dieses Wertes könnte die zuständige Person von Raum zu Raum gehen und diesen jeweils vor Ort überprüfen.

## 2.2 Modellierungsansätze

Für die Darstellung von Umgebungsmodellen gibt es verschiedene, für den gegebenen Anwendungsfall verwendbare Ansätze, die im folgenden kurz vorgestellt werden. Sie unterscheiden sich in der Art und Weise, wie Positionen, Flächen und Räume modelliert werden.

### 1. Geometrisches Modell

Dieses Modell basiert auf geometrischen Figuren, deren Grenzen durch Punkte innerhalb eines standardisierten Koordinatensystems, dargestellt werden. Das am weitesten verbreitete ist das „World Geodetic System 1984“ (WGS84) [DOD97], welches Längen- und Breitengrade verwendet und zusätzlich die Definition einer Höheninformation ermöglicht. Auf Basis dieses Systems lässt sich eine beliebige Position durch drei Koordinaten *Länge*, *Breite* und *Höhe* exakt festlegen. Flächen und Räume werden durch die Positionen ihrer Eckpunkte und deren Verbindungen untereinander zu beliebigen geometrischen Figuren modelliert. Flächen ließen sich derart zum Beispiel durch Polygone modellieren, während für einen Raum zum Würfel verwendet werden könnte.

### 2. Mengenbasiertes Modell

Das mengenbasierte Modell hat als Grundlage Mengen von Positionen. Diese Positionen können innerhalb eines beliebigen Koordinatensystems definiert sein. Die Positionsmengen müssen keinen Bezug zur realen Welt haben. Sie definieren sich über Funktionen der booleschen Algebra, die entscheiden, ob eine Position zu einer Menge gehört oder nicht. Nähere Informationen zu mengenbasierten Umgebungsmodellen finden sich in [KEG93].

### 3. Hybrides Modell

Hybride Modelle kombinieren die Ansätze des mengenbasierten und des geometrischen Modells. Realisiert wird dies, indem Positionsmengen als Attribut ein geometrisches Modell zugeordnet werden kann. Ein geometrisches Modell bezieht sich in diesem Fall ausschließlich auf die Positionsmenge der es zugeordnet wurde und nicht auf die gesamte modellierte Umgebung. In [JIST02] wird die Verwendung und der Einsatz eines hybriden Modells näher beschrieben.

## **2.3 Bewertung der Modellierungsansätze**

Alle drei Ansätze haben ihre individuellen Vor- und Nachteile. In diesem Abschnitt werden die unterschiedlichen Modellierungsansätze miteinander verglichen und ihre Anwendbarkeit für den Einsatz von Scopes auf mobilen Geräten analysiert. Mobile Geräte verfügen über beschränkte Ressourcen an Rechenleistung, Speicherplatz und Energie. Auch der Aufwand der geleistet werden muss um ein Umgebungsmodell nach einem bestimmten Modellierungsansatz zu erstellen, kann ein wichtiger Einflußfaktor für eine Anwendung darstellen.

Unter Berücksichtigung dieser Umstände wurden folgende Punkte näher betrachtet:

- Modellierungsaufwand
- Speicherbedarf
- Aufwand zur Verwaltung von Scopes

Jeder der drei vorgestellten Modellierungsansätze wird auf die oben genannten Aspekte hin analysiert. Der Modellierungsaufwand für ein Umgebungsmodell legt fest, wieviel Aufwand geleistet werden muss, um ein bestimmtes Umgebungsmodell zur Verfügung stellen zu können.

Die verschiedenen Modellierungsansätze verfügen über unterschiedliche Arten und Mengen von Informationen, die gespeichert werden müssen. Eine genauere Betrachtung dieses Aspekts wird unter dem Punkt Speicheraufwand vorgenommen. Der Aspekt Verwaltungsaufwand beinhaltet den Aufwand zur Berechnung der Entscheidung, ob sich eine bestimmte Position innerhalb oder außerhalb eines Scopes befindet und den Aufwand für die unter Umständen notwendigen Berechnungen zur Handhabung der dynamischen räumlichen Eigenschaften von Scopes (siehe Kapitel 2.1).

### **2.3.1 Geometrisches Modell**

Die folgenden Unterabschnitte beurteilen die vorgestellten Aspekte im Hinblick auf den Einsatz eines geometrischen Modells. Neben dem Modellierungsaufwand zur Erstellung eines solchen Modells werden auch der Speicherbedarf für die Ablage der notwendigen modellspezifischen Informationen, sowie der Aufwand typischer, für den Einsatz von Scopes notwendiger Berechnungen näher betrachtet.

### **2.3.1.1 Modellierungsaufwand**

Geometrische Umgebungsmodelle zeichnen sich durch ihre hohe Genauigkeit aus. Allerdings wird dieser Vorteil dadurch erreicht, dass die Umgebung mit sehr hohem Aufwand vermessen werden muss. Für sehr komplexe Umgebungen, wie zum Beispiel große Gebäude oder ganze Straßenzüge, ist dies ein Aufwand, der sich nur lohnt, falls die Anwendung dies zwingend erfordert. Ein Beispiel für eine solche Anwendung wäre die Unterstützung eines Hausmeisters die anzeigt, wo bestimmte Leitungen verlaufen oder Anschlüsse gelegt wurden. Falls die Genauigkeit nicht so hoch sein muss, können im geometrischen Modell beliebige Räume näherungsweise definiert werden, indem ihre Grundfläche modelliert und diese zusätzlich mit einer Höheninformation versehen wird. Da ein Scope in 2.1 als abstrakter dynamischer Raum definiert wurde, kann der Fall eintreten, dass die Grenzen von real existierenden Räumen nicht ausreichen um einen Scope darzustellen. In diesem Fall müssen zusätzlich zu den real existierenden modellierten Räumen die theoretisch möglichen Scopes modelliert werden. Dieser Fall tritt beispielsweise ein, wenn ein Scope einen realen Raum nur teilweise abdecken soll.

### **2.3.1.2 Speicherbedarf**

Ein hoher Detaillierungsgrad des geometrischen Modells beeinflusst den Speicherbedarf für die Modellinformationen. Jedes Polygon, das einen Teil der Umgebung modelliert, muss mit seinen Eckpunkten und deren Verbindungen abgespeichert werden, wobei für jeden Eckpunkt eines Polygons die jeweiligen Koordinaten abgelegt werden müssen. Die Anzahl dieser Koordinaten hängt von der Dimensionierung des zugrundeliegenden Koordinatensystems ab, entsprechend viele Werte pro Punkt müssen gespeichert werden. Die Anzahl der Verbindungen zwischen den Eckpunkten hängt von der Dimensionierung der zu modellierenden geometrischen Figur ab. Theoretisch muss es möglich sein, dass alle Punkte untereinander verbunden werden können. Dies resultiert in einem quadratischen Speicherplatzbedarf. Ein solcher Bedarf ist für den Einsatz auf speicherplatzbeschränkten mobilen Geräten lediglich für eine kleine Anzahl von Punkten vertretbar.

### **2.3.1.3 Berechnungsaufwand**

Für die Entscheidung, ob ein Punkt innerhalb oder außerhalb eines Scopes liegt, muss dieser Punkt in Bezug zu dem Polygon gesetzt werden, das den Scope modelliert. Dies kann mit Hilfe geometrischer Rechenoperationen erreicht werden. Der hierfür notwendige Aufwand bewegt sich je nach Polygon im Bereich von  $O(n)$  und  $O(\log n)$  [KAR00], wobei  $n$  die Anzahl der Kanten des Polygons darstellt. Für geometrische Figuren, die mehr Dimensionen als einfache Polygone umfassen, bewegt sich der Aufwand in ähnlichen Grenzen oder liegt sogar höher.

Ein zusätzlicher Einflussfaktor für den Rechenaufwand wird durch die dynamischen Scopeeigenschaften gegeben. Da ein Scope in seiner Ausdehnung wachsen oder sich verkleinern kann, müssen die entsprechenden Skalierungsoperationen auf die jeweilige geometrische Figur angewendet werden. Falls die dynamische Scopeausdehnung nur auf Grundlage schon im Umgebungsmodell vorhandener Figuren durchgeführt wird, muss für die jeweils zusammengefassten Figuren deren äußere Grenze berechnet werden. Für konvexe Polygone kann dies mit einem Aufwand von  $O(n)$  erreicht werden [KAR00]. Nicht-konvexe Polygone können in eine Menge von konvexen Polygonen zerlegt werden. Dieser Aufwand wäre dann zusätzlich zur Berechnung des Schnittpolygons zu erbringen. Für geometrische Figuren mit größerer Dimensionierung bewegt sich der Aufwand in ähnlichen oder höheren Bereichen. Aufgrund der möglichen Bewegung von Scopes sind diese nicht an feste Koordinaten gebunden, somit muss auch ein Aufwand für die Berechnung neuer Koordinaten für jeden Punkt einer geometrischen Figur in Betracht gezogen werden. Zur Abbildung einfacher Bewegungen wird beispielsweise ein Vektor zu jedem Punkt einer Figur addiert.

### **2.3.2 Mengenbasiertes Modell**

Dieses Kapitel beschreibt in den folgenden Unterabschnitten die für den Einsatz von Scopes relevanten Aspekte im Hinblick auf mengenbasierte Modelle sind dies:

- der notwendige Modellierungsaufwand zur Erstellung eines solchen Modells,
- der Speicherbedarf für die modellspezifischen Informationen und
- der Berechnungsaufwand für typische Berechnungen, welche beim Einsatz von Scopes durchgeführt werden müssen.

#### **2.3.2.1 Modellierungsaufwand**

Die grundlegenden Einheiten des mengenbasierten Modells sind Positionsmengen. Definiert werden diese Mengen über Funktionen der booleschen Algebra. Der Detaillierungsgrad des Modells beeinflusst die Anzahl der Mengen und damit auch die Anzahl der notwendigen Funktionen. Weiterhin wird durch den Detaillierungsgrad auf die Anzahl der Parameter Einfluss genommen, die in einer solchen Funktion berücksichtigt werden müssen. Je genauer ein Raum abgebildet werden soll, umso genauer müssen dessen Grenzen durch die Funktion erfasst werden. Das bedeutet, es müssen mehr Parameter, welche die Raumgrenzen bestimmen, in die Funktion einfließen. Falls die vorhandenen Positionsmengen nicht ausreichen um alle in einer Anwendung möglichen Scopes zu modellieren, muss zusätzlich für jeden nicht darstellbaren Scope eine weitere Funktion definiert werden.

Ein Beispiel für eine solche boolesche Funktion, die anhand der Ausdehnung in die jeweiligen Richtungen eine Fläche modelliert und dann entscheidet ob sich eine beliebige Position,

spezifiziert durch deren Koordinaten, innerhalb dieser Fläche befindet, sähe zum Beispiel folgendermaßen aus:

#### **Boolesche Funktion zu Definition einer Fläche**

- x,y seien beliebige Koordinaten
  - die modellierte Fläche habe eine Ausdehnung von 3x4 Metern.
- Resultierend aus diesen Angaben ergäbe sich folgende Formel:

$$F(x,y) = ((x>0) \text{ UND } (x \leq 3) ) \text{ UND } ((y>0) \text{ UND } (y \leq 4))$$

Alle in einem Umgebungsmodell vorkommenden Räume oder Flächen müssten auf dieser Art und Weise definiert werden. Sollen nun mögliche Scopes nicht nur ganze Räume, sondern zum Beispiel auch einzelne Teilräume abdecken können, so müssten für diese zusätzlichen Möglichkeiten weitere Funktionen nach obigem Schema erstellt werden. Soll zum Beispiel entschieden werden, ob sich eine Position in einer Hälfte der Fläche aus dem obigen Beispiel befindet, so müssten entsprechend zusätzliche Überprüfungen der Parameter mit den Werten  $x=1,5$  und  $y=2$  eingefügt werden. An diesem Beispiel zeigt sich sowohl die Notwendigkeit zusätzlich zu definierender Funktionen, als auch die Erhöhung der Anzahl der zu überprüfenden Parameter. Tiefergehende Information zum Bereich mengenbasierte Umgebungsmodelle und deren Modellierung finden sich in [KEG93].

#### **2.3.2.2 Speicherbedarf**

Der Speicherplatz durch ein mengenbasiertes Umgebungsmodell in Anspruch genommene Speicherplatz ist im wesentlichen für die Funktionen zur Definition der Positionsmengen belegt. Hier entscheidet der Detaillierungsgrad über Anzahl und Umfang der jeweiligen Formeln. Es ist nicht zwingend erforderlich für jede Positionsmenge eine eigene Formel abzulegen. Je nach Struktur der Mengen kann es möglich sein, eine Formel mit verschiedenen Parametern wiederverwenden zu können. In diesem Fall würde die Formel einmal und zusätzlich die jeweiligen Parametersätze gespeichert. Vorstellbar wäre an dieser Stelle eine Funktion, die eine rechteckige Positionsmenge definiert, die dann mit der Position einer Menge in der modellierten Umgebung, sowie einer Länge und einer Breite parameterisiert wird und so für beliebige rechteckige Positionsmengen verwendet werden kann. Der konkrete Speicherbedarf hängt vom Ablageformat der booleschen Funktionen, sowie möglicher Parametersätze ab.

### **2.3.2.3 Berechnungsaufwand**

Die Entscheidung, ob sich ein Punkt innerhalb oder außerhalb eines Scopes befindet, wird durch die boolesche Funktion entschieden, welche die entsprechende Positionsmenge definiert, durch die der Scope dargestellt ist. Die dynamische Scopeausdehnung und -positionierung wird durch Anpassung der Funktionsparameter erreicht, die die Struktur einer konkreten Positionsmenge definieren. Falls eine Scopeausdehnung ausschließlich basierend auf vorhandenen Positionsmengen durchgeführt wird, müssen die Funktionen, welche die einbezogenen Positionsmengen festlegen, zu einer einzigen booleschen Formel zusammengefasst werden. Dies geschieht durch Verknüpfung der einzelnen Funktionen mit der logischen Oder-Funktion. Durch geschickten Aufbau und Anordnung der Funktionen kann so der Berechnungsaufwand unter Umständen noch reduziert werden. Unter der Annahme, dass die einzelnen Positionsmengen in der Hierarchie eines AVL-Baumes angeordnet sind, kann die Inklusion mit einem Aufwand von  $O(\log n)$  bestimmt werden, wobei  $n$  die Anzahl der Knoten im Baum darstellt. Es wird, von der Wurzel ausgehend, der Knoten im Baum gesucht, der die kleinste Menge repräsentiert, in der sich eine gegebene Position befindet. Wird diese Annahme ebenfalls für den Fall der Zusammenfassung zweier Scopes berücksichtigt, so kann dies ebenfalls mit einem Aufwand von  $O(\log n)$  erreicht werden, da in diesem Fall die kleinste Menge gesucht wird, welche die beiden zu vereinigenden Scopes enthält.

### **2.3.3 Hybrides Modell**

Dieses Kapitel beschreibt in seinen Unterabschnitten die wichtigen Aspekte, die für hybride Umgebungsmodelle genauer betrachtet werden müssen, um entscheiden zu können, inwieweit sich diese für den Einsatz von Scopes eignen. Genauer eingegangen wird auf die Punkte Modellierungsaufwand und Speicherbedarf für ein solches Modell, als auch auf den Aufwand der betrieben werden muss, um scopetypische Berechnungen in einem hybriden Umgebungsmodell durchführen zu können.

#### **2.3.3.1 Modellierungsaufwand**

Die Grundlage des hybriden Modells stellt ein mengenbasiertes Umgebungsmodell dar. Aus diesem Grund ist der Modellierungsaufwand an dieser Stelle mit dem eines mengenbasierten Umgebungsmodells vergleichbar. Zusätzlich kommt bei den Positionsmengen, die ein eigenes Koordinatensystem als Attribut zugeordnet haben, der Modellierungsaufwand eines geometrischen Umgebungsmodells hinzu. Der Gesamtaufwand zur Modellierung eines hybriden Umgebungsmodells liegt somit deutlich über dem der anderen Arten.

### **2.3.3.2 Speicherbedarf**

Analog zum Modellierungsaufwand verhält sich der Speicherbedarf eines hybriden Umgebungsmodells. Zusätzlich zu den Funktionen, die die Positionsmengen bestimmen, müssen die innerhalb der Koordinatensysteme definierten geometrischen Figuren abgespeichert werden. Wenn verschiedene Koordinatensysteme zum Einsatz kommen, wird weiterer Speicherplatz zur Ablage der Informationen zu den verschiedenen Koordinatensystemen benötigt, zum Beispiel Basisvektoren oder Transformationsmatrizen für die Umrechnung der Koordinatensysteme untereinander.

### **2.3.3.3 Berechnungsaufwand**

Für die Inklusionsentscheidung in Bezug auf eine Positionsmenge muss, wie beim mengenbasierten Modell, eine boolesche Funktion berechnet werden. Falls ein Koordinatensystem mit in die Berechnung einbezogen werden muss, kommt der entsprechende Berechnungsaufwand für den Fall eines geometrischen Umgebungsmodells hinzu. Zusätzlicher Berechnungsaufwand ist zu berücksichtigen, falls zwischen verschiedenen Koordinatensystemen umgerechnet und transformiert werden muss.

### **2.3.4 Bewertung**

Die vorgestellten Umgebungsmodelle wurden unter Gesichtspunkten einer Anwendung betrachtet, die in einem mobilen ad hoc Netzwerks zum Einsatz kommen soll. Der Modellierungsaufwand spielt in diesem Zusammenhang eine Rolle, da es vorkommen kann, dass ein Umgebungsmodell ebenfalls ad hoc erstellt werden soll. Hier ist das mengenbasierte Umgebungsmodell eine gute Wahl, weil sich ein einfaches Modell mit geringem Aufwand erstellen lässt. Denkbar wäre an dieser Stelle eine Anwendung, die außerhalb von Gebäuden auf freiem Gelände zum Einsatz kommt. Hier ist die Wahrscheinlichkeit gering, dass ein detailliertes Umgebungsmodell, wie zum Beispiel ein Bau- oder Geländeplan, nicht zur Verfügung steht und so bei Bedarf schnell erstellt werden muss. In diesem Fall wäre es denkbar, dass ein Gitternetz über das zu modellierende Gelände gelegt wird.

Mobile Geräte sind unter verschiedenen Gesichtspunkten in ihren Ressourcen beschränkt. So gibt es zum Beispiel einen beschränkten Energievorrat, der schneller verbraucht werden würde, falls viele aufwendige Berechnungen durchzuführen sind. Die begrenzte Rechenleistung eines mobilen Geräts würde diesen Einflußfaktor sogar noch weiter verstärken. Diese Punkte sprechen eindeutig gegen ein geometrisches oder hybrides Umgebungsmodell. Eine weitere knappe Ressource auf mobilen Geräten stellt der Speicherplatz dar. Der hohe Speicherplatzbedarf eines geometrischen Umgebungsmodells stellt in diesem Falle einen klaren Nachteil dar. Ähnliches gilt für ein hybrides Modell, das auch geometrische Komponenten enthalten kann. Tabelle 1 stellt zusammenfassend die

Eigenschaften der Umgebungsmodelle in Bezug auf mobile Geräte dar, welche in MANETs eingesetzt werden sollen.

	<i>Geometrisch</i>	<i>Mengenbasiert</i>	<i>Hybrid</i>
<i>Modellierungsaufwand</i>	<i>Hoch</i>	<i>Gering</i>	<i>Hoch</i>
<i>Speicherbedarf</i>	<i>Hoch</i>	<i>Gering</i>	<i>Moderat</i>
<i>Berechnungsaufwand</i>	<i>Hoch</i>	<i>Gering</i>	<i>Moderat/Hoch</i>

**Tabelle 1:**  
**Vergleich der Umgebungsmodelle**

Der geringe Modellierungsaufwand und Speicherbedarf sind klare Gründe für den Einsatz eines mengenorientierten Umgebungsmodells zur Verwaltung von Scopes. Im Gegensatz zum geometrischen und hybriden Umgebungsmodell hält sich auch der Berechnungsaufwand in überschaubaren Grenzen. Bei hybriden Modellen ist der Berechnungsaufwand davon abhängig, wie oft Berechnungen in den geometrischen Komponenten durchgeführt werden müssen. Falls für eine Anwendung ein höherer Modellierungsaufwand und unter Umständen ein höherer Berechnungsaufwand vertretbar ist, kann ein hybrides Umgebungsmodell verwendet werden. Im Vergleich zu mengenbasierten Umgebungsmodellen wird damit, eine genauere Darstellung der realen Umgebung erreicht. Die höhere Genauigkeit muss dann allerdings nur an den Stellen berücksichtigt werden, wo diese auch wirklich benötigt wird.

[BEDU03] beschäftigt sich näher mit dem Einsatz von Umgebungsmodellen in Anwendungen im Bereich des Ubiquitous Computing. Hier werden verschiedene Aspekte betrachtet, die von Anwendungen in Bezug auf Umgebungsmodelle berücksichtigt werden müssen.

## **2.4 Hierarchische mengenbasierte Umgebungsmodelle**

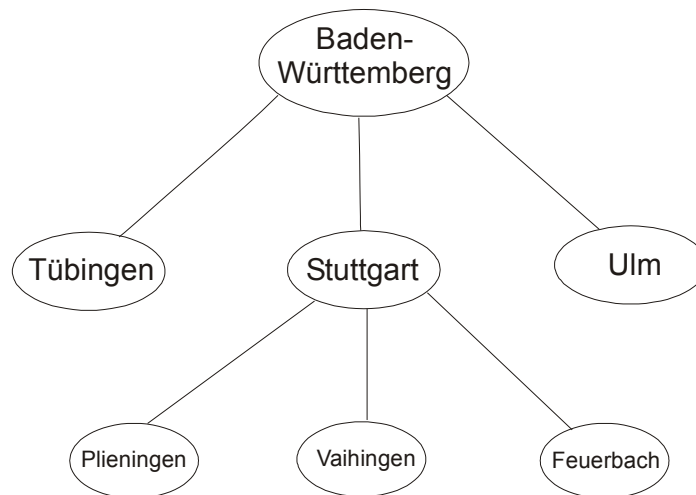
Ein Nachteil des rein mengenbasierten Umgebungsmodell ist es, dass sich Strukturen aus der realen Welt schlecht nachbilden lassen. So ist anhand einer Positionsmenge nicht erkennbar, dass sie die Positionen eines Raumes innerhalb eines Gebäudes enthält. Abhilfe schafft hier der Ansatz, dass eine Hierarchie auf den Positionsmengen definiert wird. Mit Hilfe der Hierarchie kann dargestellt werden, dass sich eine Positionsmenge aus verschiedenen Untermengen zusammensetzt. Auf diese Weise kann zum Beispiel für jeden Raum innerhalb eines Gebäudes eine Positionsmenge definiert werden. Diese Raum-Positionsmengen können zu Etagen-Positionsmengen und die Etagen-Positionsmengen schließlich zu einer Positionsmenge für das gesamte Gebäude zusammengefasst werden.

Die Zusammenfassung von Mengen ist gleichzusetzen mit deren Vereinigung. Auf die booleschen Funktionen, die eine Positionsmenge definieren, bezogen heißt dies, dass diese mit dem booleschen Oder-Operator verknüpft werden. Diese Vorgehensweise beschreibt die logische Konstruktion des hierarchischen mengenbasierten Umgebungsmodells oder kurz hierarchischen Umgebungsmodells.

Im Folgenden werden zwei Vorgehensweisen zum Aufbau einer Mengenhierarchie vorgestellt. In Bezug auf Anwendungen im Bereich mobiler ad hoc Netzwerke wird deren Eignung für die Handhabung von Scopes näher betrachtet.

### **2.4.1 Space-Tree**

In [JIST02] wird ein hybrides Umgebungsmodell vorgestellt, dessen mengenbasierte Komponente eine baumartige Struktur aufweist. Die Autoren nennen dieses Konzept Space-Tree. Zur Konstruktion eines solchen Space-Tree wird die reale Umgebung in abstrakte Räume unterteilt, die durch verschieden große Positionsmengen repräsentiert werden. Das Konzept des Space-Tree sieht vor, dass die Wurzel des Baumes aus der Positionsmenge besteht, welche die gesamte reale Umgebung abdeckt, in der eine Anwendung ausgeführt wird. Diese Positionsmenge wird in disjunkte Untermengen aufgeteilt. Die Mengen stehen somit in einer Ober- / Untermengenbeziehung zueinander, die durch die Kanten des Space-Tree dargestellt wird. Die Aufteilung der Positionsmengen in weitere Untermengen wird soweit fortgesetzt, bis ein für die jeweilige Anwendung ausreichender Detaillierungsgrad erreicht ist. Die Blätter des Baumes bilden, nach erfolgreicher Konstruktion, alle in der Umgebung einzeln adressierbaren Positionen bzw. Positionsmengen.

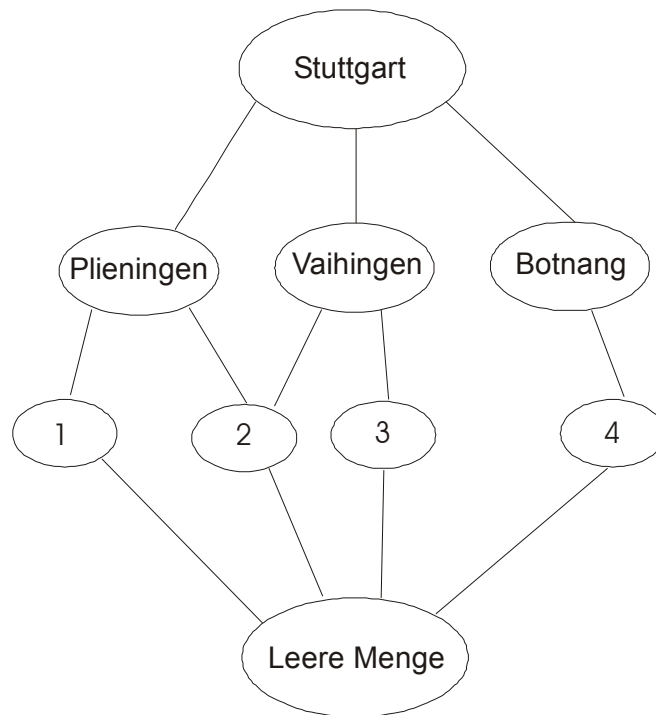


**Abbildung 1: Space-Tree Beispiel**

Abbildung 1 zeigt einen Space-Tree, der ausschnittsweise das Land Baden-Württemberg modelliert. Es werden einige Städte und etwas genauer die Stadt Stuttgart, mit einigen Stadtteilen dargestellt.

### **2.4.2 Lattice**

Die in [KEG93] beschriebene Grundlage des Lattice-Ansatzes sind Positionsmengen, bei denen eine Halbordnung auf ihren Elementen definiert ist (partially ordered sets, posets). Zusätzlich wird die Anforderung gestellt, dass es eine gemeinsame Über- und eine gemeinsame Untermenge gibt. Das bedeutet, es gibt eine Positionsmenge in der alle Untermengen enthalten sind und eine Positionsmenge, die selbst in allen Positionsmengen enthalten ist. Die Ordnungsrelation auf den Positionsmengen wird durch die räumliche Inklusion definiert. Die dadurch entstehende Struktur die dadurch entsteht nennt sich *Lattice* (zu deutsch: Verband), welche durch ein sogenanntes Hasse-Diagramm dargestellt werden kann. Hierbei werden Obermengen oberhalb der in ihnen enthaltenen Untermengen abgebildet und durch Kanten miteinander verbunden. Die Knoten des Hasse-Diagramms bestehen aus allen möglichen Teil- und Schnittmengen. Hierbei wird eine Schnittmenge im Hasse-Diagramm unterhalb derjenigen Positionsmengen abgebildet und mit ihnen verbunden, deren gemeinsame Positionen sie enthält. Diese Aufteilung in Unter- bzw. Schnittmengen wird solange durchgeführt, bis der gewünschte Detaillierungsgrad erreicht ist.



**Abbildung 2: Lattice-of-Locations Beispiel**

Abbildung 2 verdeutlicht den Aufbau eines Lattice anhand eines konkreten Beispiels, das die Stadt Stuttgart auszugsweise darstellt. Zu beachten ist hier die Unterteilung der Stadtteile in Straßen. Anstatt konkreter Straßennamen wurden die symbolischen Bezeichner 1,2,3 und 4 verwendet, um die Übersichtlichkeit in der Abbildung zu wahren. Die Straße 2 verbindet die Stadtteile Plieningen und Vaihingen und ist somit in beiden Stadtteilen vorhanden. Aus diesem Grund ist sie als Teilmenge modelliert und verfügt über Verbindungen zu beiden Stadtteilen. Alle anderen modellierten Straßen verlaufen ausschließlich innerhalb der Stadtteile. An dieser Stelle wird die höhere Mächtigkeit im Vergleich zum Space-Tree deutlich. Die Straße 2 hätte in einem Space-Tree in zwei Blattknoten auftauchen müssen oder könnte somit nicht zugeordnet werden, da eine eindeutige Zuordnung zu einem Stadtteil auf der Detaillierungsebene einzelner Straßen nicht möglich ist.

### 2.4.3 Bewertung

Für beide Modelle gibt es keine Vorgaben wie die reale Welt auf diese übertragen werden soll. Worin sie sich unterscheiden ist der Speicherplatz, der benötigt wird, um das jeweilige Modell abzuspeichern. Weiterhin ist der für einen Lattice höhere Konstruktionsaufwand zu beachten, der sich allerdings teilweise automatisieren lässt. So wird in [KEG93] zum Beispiel beschrieben, wie sich aus einem Space-Tree automatisch ein Lattice generieren lässt. Ein Vorteil des Lattice ist der höhere Freiheitsgrad bei der Abbildung der realen Umgebung, der erreicht wird, indem Überlappungen realer Räume, im Gegensatz zum Space-Tree, mit berücksichtigt werden können. Dieser erhöhte Freiheitsgrad wirkt sich allerdings auch auf die Komplexität der einzusetzenden Algorithmen aus, die auf einem Lattice ausgeführt werden. Falls zum Beispiel in einem Space-Tree überprüft werden soll, ob eine Positionsmenge in einer anderen Menge enthalten ist, so muss der Pfad zur Wurzel hinaufgewandert werden. Bei einem Lattice ist unter Umständen nicht bekannt, in welche Richtung die Suche durchgeführt werden soll, da eine Inklusion einmal in einer Obemenge oder eine Inklusion in einer Teilmenge vorkommen kann. Bezogen auf das Hasse Diagramm würde im ersten Fall nach oben und im zweiten Fall nach unten gewandert werden. In Abbildung 2 müsste zum Beispiel sowohl die Leere Menge als auch die Positionsmenge Plieningen überprüft werden, falls eine solche Entscheidung für den Knoten 2 zu treffen wäre.

Unter dem Aspekt, dass mobile Geräte über beschränkte Ressourcen, wie Speicherplatz und Energievorrat verfügen, bietet sich die Verwendung der Space-Tree-Hierarchie für ein Umgebungsmodell an. Ein weiteres Kriterium für diese Struktur stellt die Tatsache dar, dass sich der Konstruktionsaufwand in überschaubaren Grenzen hält. Diese Tatsache gestattet es unter Umständen, abhängig von der konkreten Anwendung, ad hoc eine mengenbasierte Hierarchie der realen Umgebung erstellen zu können.

Die endgültige Entscheidung für die eine oder andere Hierarchiestruktur kann neben der Situation beschränkter Ressourcen auch von der konkreten Anwendung und von der zu modellierenden realen Umgebung abhängen. Falls es zum Beispiel darauf ankommt, ein kleines fein strukturiertes Gebiet zu modellieren, könnte sich die Erstellung eines Lattice als vorteilhaft erweisen.

Die in diesem Kapitel durchgeführten Betrachtungen haben ergeben, dass das Space-Tree-Modell eine gute Wahl für den Einsatz und die Verwaltung von Scopes ist. Aus diesem Grund wird für den weiteren Verlauf dieser Arbeit das Space-Tree-Modell zur Modellierung einer Umgebung verwendet. Neben den besonderen Anforderungen, die für die Handhabung und Verwaltung von Scopes entstehen, werden bei diesem Modell auch die Einschränkungen berücksichtigt, die durch den Einsatz mobiler Geräte zum Tragen kommen.

## 3 Scopes

In diesem Kapitel wird detailliert auf Scopes und deren Handhabung eingegangen. Es werden weiterhin Anforderungen erstellt, die von einem Umgebungsmodell hierfür erfüllt werden müssen. Weiterhin werden besondere Eigenschaften von Scopes erläutert und wie sich diese in einem Umgebungsmodell abbilden lassen.

### 3.1 Formelle Definition

In Kapitel 2.1 wurde eine informelle Definition für einen Scope beschrieben. An dieser Stelle soll eine formale Definition festgelegt werden, damit eine genauere Beschreibung der Handhabung von Scopes ermöglicht werden kann. Ein Scope ist, analog zu Umgebungsdefinitionen in mengenbasierten Umgebungsmodellen, eine Menge von Positionen. Eine solche Menge wird durch eine boolesche Funktion definiert. Im Gegensatz zu den Positionsmengen, welche die Umgebung in einem mengenbasierten Umgebungsmodell festlegen, verfügt ein Scope über zusätzliche Eigenschaften. Diese werden im Folgenden dargestellt:

- Ein Scope muss nicht an eine feste Position in der realen Umgebung gebunden sein. Daraus folgt, dass sich ein Scope, bezogen auf die reale Umgebung, bewegen kann. So ist es zum Beispiel denkbar, dass sich ein Scope zwischen unterschiedlichen Räumen eines Gebäudes bewegt.
- Ein Scope kann sich in seiner Ausdehnung verändern. Das bedeutet, die Anzahl der Positionen innerhalb eines Scopes ist nicht notwendigerweise fest definiert. Beispielsweise kann ein Scope zu einem Zeitpunkt einen Raum umfassen und etwas später eine ganze Etage abdecken.

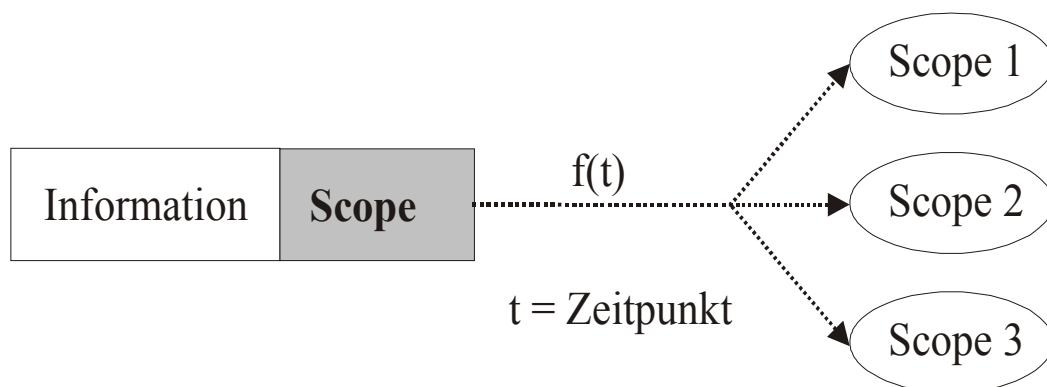
#### **Definition:**

**Ein Scope ist eine dynamische Positionsmenge, die sich im Verlauf der Zeit ändern kann.**

Informationen, welche mit Hilfe von Scopes verbreitet werden sollen, erhalten ein zusätzliches Attribut. Der Wertebereich dieses Attributs umfasst die, jeweils in einer Anwendung einzusetzenden Scopes.

Die Sichtweise, die in diesem Fall für Scopes verwendet wird, entspricht einer Funktion, welche in Abhängigkeit von unterschiedlichen Faktoren, zum Beispiel der Zeit, den jeweils zu verwendenden Scope festlegt. Die für diese Funktion zu verwendenden Parameter sind von der Strategie abhängig, die für die Bestimmung von Scopes in einer Anwendung zum Einsatz kommen soll. Eine andere mögliche Alternative sind zum Beispiel konkrete Positionsangaben.

Abbildung 3 stellt diesen Sachverhalt plastisch dar. Man sieht die Zuordnung eines Scopes zu einer Information, dessen konkreter Wert von einer Funktion in Abhängigkeit von der Zeit festgelegt wird.



**Abbildung 3:**  
Scope als Attribut

### **3.2 Anforderungen an das Umgebungsmodell**

Für die Verwaltung von Scopes wird ein Umgebungsmodell benötigt. Da sich hierfür, wie in Kapitel 2 analysiert, ein hierarchisches Umgebungsmodell anbietet, ist die Definition eines Scopes durch eine Positionsmenge von Vorteil. Dieser Vorteil spiegelt sich darin wieder, dass die grundlegenden Elemente eines hierarchischen Umgebungsmodells ebenfalls Positionsmengen sind. An dieser Stelle wird die Annahme getroffen, dass das hierarchische Umgebungsmodell in einem Space-Tree organisiert ist und dass die Mengen auf einer Ebene ähnlich große Bereiche der realen Welt modellieren.

Durch die Variabilität von Scopes werden zusätzliche Anforderungen an das Umgebungsmodell gestellt. Diese spiegeln sich im konkreten dadurch wieder, dass die einen Scope beschreibenden booleschen Funktionen variabel gehalten werden müssen. Die Variabilität kann durch die bedarfsweise Anpassung dieser Funktionen dargestellt werden. Weiterhin wird im folgenden angenommen, dass ein Scope nur durch im Umgebungsmodell vorhandene Positionsmengen definiert werden kann. Der Vorteil dieser Einschränkung ist, dass keine Änderungen oder Erweiterungen am Umgebungsmodell durchgeführt werden müssen. Für einen Scope muss lediglich der Knoten bekannt sein, welcher, bezogen auf den

Space-Tree, die Wurzel des Teilbaums darstellt, der die entsprechende Positionsmenge definiert. Würde diese Einschränkung nicht berücksichtigt werden, müssten zum Beispiel zusätzliche Positionsmengen für Scopes festgelegt werden, die Bereiche der realen Umgebung abdecken, die nicht von bestehenden Positionsmengen modelliert werden.

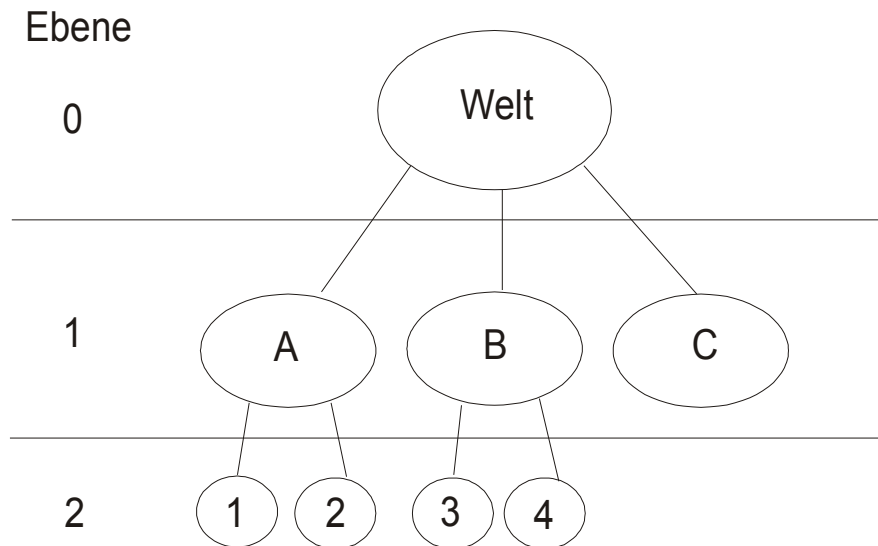
Diese Einschränkung ist hilfreich für Anwendungsfälle, bei denen ein Umgebungsmodell automatisiert zu erstellen ist, zum Beispiel bei der Einteilung einer Fläche in Quadranten, auf denen eine Hierarchie definiert werden soll. In diesem Fall wird eine Definition der Eckdaten für die Erstellung des Umgebungsmodells benötigt, wie zum Beispiel die Größe der Gesamtfläche, Anzahl der Quadranten und eine Vorgehensweise für die Festlegung von Scopes. Mobile Klienten wären mit Hilfe dieser Informationen in der Lage, Umgebungs- und Scopeinformationen lokal zu verwalten. Die Art und der Umfang der notwendigen Informationen ist anwendungsabhängig. Wie die Operationen auf Scopes in einem hierarchischen Umgebungsmodell dargestellt werden können, verdeutlichen die folgenden Abschnitte.

### **3.3 Ausdehnungsänderung eines Scopes**

Eine Veränderung der Scopeausdehnung entspricht einer Erhöhung beziehungsweise Reduzierung der Anzahl der in der Positionsmenge enthaltenen Elemente, die einen Scope definiert. Bezogen auf einen Space-Tree bedeutet dies, dass zur Definition des veränderten Scopes ein Vorgänger- oder Nachfolgeknoten des Knotens dient, der den ursprünglichen Scope festlegt. Eine Vergrößerung wird im Space-Tree dadurch modelliert, dass ein Vorgängerknoten ausgewählt wird. Die Anzahl der Ebenen, die dabei überwunden wird, ist abhängig von der neu gewählten Ausdehnung. Als maximale Ausdehnung wird der Wurzelknoten des Space-Tree genutzt. Eine Vergrößerung hat zur Folge, dass neue Untermengen zur ursprünglichen Positionsmenge hinzugefügt werden. Hierbei werden die booleschen Funktionen, welche sämtliche in der neuen Positionsmenge enthaltenen Untermengen definieren, mit dem booleschen Oder-Operator verknüpft. Dies entspricht der Vereinigung dieser Mengen.

Die Verkleinerung wird im Space-Tree durch Auswahl eines Nachfolgeknotens nachgebildet. Die Anzahl der Ebenen, die bei diesem Vorgang in Richtung der Blattknoten hin durchwandert werden, ist wie im Fall der Vergrößerung von der neuen Ausdehnung abhängig. Die kleinste Ausdehnung, die erreicht werden kann, ist die eines Blattknotens. Die baumartige Struktur des Umgebungsmodells bewirkt, dass ein Knoten (ausgenommen die Blattknoten) mehrere Nachfolger haben kann. Für den Fall der Verkleinerung müssen Entscheidungsmechanismen bereitgestellt werden, die eine Auswahl des entsprechenden Nachfolgeknotens ermöglichen. In Abbildung 4 wird dies am Beispiel des Knotens A verdeutlicht. Falls ein Scope durch diesen Knoten definiert ist und verkleinert werden soll,

muss entschieden werden, ob dieser Schritt auf den Knoten 1 oder den Knoten 2 durchgeführt werden soll. Die konkrete Festlegung solcher Mechanismen ist von der jeweiligen Anwendung abhängig. Dabei sind verschiedene Ansätze denkbar, zum Beispiel soll immer die größere Untermenge als neuer Scope verwendet wird.



**Abbildung 4: Ebenen im Space-Tree**

Ein anderer Ansatz ist es, sich an gegebenen Positionsinformationen zu orientieren und die Positionsmenge auszuwählen, welche die definierte Position enthält. Ein Scope würde bei dieser Vorgehensweise gezielt positioniert werden.

Bezogen auf die booleschen Beschreibungsfunktionen für die Positionsmenge des Scopes bedeutet dies, dass diejenigen Teilformeln, die Positionsmengen beschreiben, die nicht mehr zu dem neuen Scope gehören, aus der ursprünglichen Beschreibungsformel entfernt werden müssen. Dies kommt einer Abspaltung ausgewählter Teilmengen gleich.

Hiermit wird deutlich, dass der Vorgang der Scopeverkleinerung für eine Anwendung unter Umständen sehr sorgfältig geplant werden muss, da entsprechende Kriterien und Mechanismen zur Auswahl eines neuen Knotens für die Scopefestlegung, zur Verfügung gestellt werden müssen.

### **3.4 Positionsänderung**

Ausgehend von den in Kapitel 3.2 getroffenen Annahmen ist ein Scope durch einen beliebigen Knoten im Space-Tree definiert. Eine Eigenschaft von Scopes ist es, dass sie ihre Position verändern können. Diese Änderung muss sich entsprechend im Umgebungsmodell nachvollziehen lassen. Bewegt sich ein Scope in der realen Umgebung in ein Gebiet, welches durch einen anderen Teil des Space-Tree modelliert wird, so muss diese Änderung entsprechend abgebildet werden. Das bedeutet, dass der Scope durch einen anderen Knoten im Space-Tree repräsentiert wird.

Weiterhin ist es möglich, dass der neu auszuwählende Knoten auf einer anderen Ebene des Space-Tree liegt als der Ausgangsknoten. Hierbei sind zwei Fälle zu berücksichtigen: Der neue Knoten kann auf einer höheren oder auf einer tieferen Ebene angesiedelt sein. Ein Wechsel in eine höhere Ebene ist unproblematisch, allerdings ist zu beachten, dass bei einem solchen Schritt auch die Scopeausdehnung verändert wird.

Die Berücksichtigung der Ausdehnung spielt eine Rolle, falls der Zielknoten in einer tieferen Ebene des Space-Tree liegt. Dann muss unter Umständen entschieden werden, ob der Scope verkleinert werden soll oder nicht. Die zu wählende Strategie ist von der Anwendung abhängig.

Generell gilt: Soll keine Ausdehnungsänderung im Zuge einer Scopebewegung durchgeführt werden, so darf der Zielknoten nicht auf einer anderen Ebene liegen als der Ausgangsknoten. Eine solche Strategie ist nur in einem Umgebungsmodell realisierbar, in dem alle Blattknoten auf einer Ebene liegen. Alternativ wird die minimale Scopegröße derart definiert, dass diese auf einer Ebene des Space-Tree liegt, in welcher eine Bewegung durch die gesamte modellierte Umgebung möglich ist, ohne dass diese Ebene gewechselt werden muss.

In Abbildung 4 wäre eine Positionsänderung von Knoten 4 in einen Bereich, welcher durch den Knoten C modelliert wird, nur zusammen mit einer Ausdehnungsvergrößerung möglich. Beim umgekehrten Weg von Knoten C in einen Bereich, der durch den Knoten B modelliert wird, muss unter Umständen entschieden werden, ob der Knoten B als neuer Scope dient oder einer der Knoten des Teilbaums, von dem Knoten B die Wurzel darstellt. Die kleinste Ebene, in der eine Bewegung durch die gesamte modellierte Umgebung ohne Ausdehnungsänderung möglich ist, wäre die Ebene 1.

Sind im Rahmen einer Bewegung mögliche Ausdehnungsänderungen eines Scopes zu berücksichtigen, so muss eine Anwendung Kriterien und Mechanismen zur Festlegung eines entsprechenden Nachfolgeknotens bereitstellen. Im wesentlichen kommen an dieser Stelle die Aspekte der Ausdehnungsänderung zum Tragen, wie sie im Kapitel 3.3 beschrieben wurden.

### 3.5 Darstellung

In den vorangehenden Abschnitte wurden Scopes definiert und ihre Handhabung erläutert. Dieser Abschnitt stellt einen Lösungsansatz vor, wie ein Scope dargestellt werden kann. Für die Arbeit mit Scopes ist es notwendig, eine Positionsmenge innerhalb eines Umgebungsmodells als Scope festlegen zu können. Hierfür wird ein geeignetes Mittel benötigt, um eine bestimmte Positionsmenge innerhalb eines Umgebungsmodells zu adressieren. Durch die Verwendung eines hierarchischen Modells ist die Verwendung des URI-Konzeptes (Universe Resource Identifier) [BFI+98] zur Adressierung einer Positionsmenge gut geeignet, weil sich hiermit beliebige Elemente in hierarchischen Strukturen in geeigneter Weise adressieren lassen. Der Einsatz von URIs in Bezug zu Umgebungsmodellen wird in [JIST02] und [DURO03b] näher vorgestellt. Unter Annahme der Verwendung von Space-Trees ist der wesentliche Aspekt beim Einsatz von URIs der, dass der Pfad von der Wurzel zum jeweils zu adressierenden Knoten in Stringform repräsentiert wird. Hierbei werden die Namen aller Knoten auf dem Pfad, getrennt durch ,/'-Zeichen, aneinandergesetzt. Falls nun eine konkrete Position, repräsentiert durch einen Blattknoten im Space-Tree, adressiert werden soll, könnte dies zum Beispiel folgendermaßen aussehen:

***URI-Form eines Scopes***

*scope://Stuttgart/Vaihingen/Universitätsstraße/38*

Das zugrundeliegende Umgebungsmodell repräsentiert die Stadt Stuttgart, aufgeteilt in Stadtteile, welche wiederum in Straßen und Häuser mit Hausnummern unterteilt sind. Die Blattknoten repräsentieren die einzelnen Gebäude einer Straße, die durch Hausnummern voneinander unterschieden werden. Ein Vorteil beim Einsatz von URIs ist, dass bei geeigneter Benennung der Positionsmengen eine für den Anwender gut lesbare Darstellung ermöglicht wird. Das obige Beispiel stellt eine andere Form der allgemein gebräuchlichen Darstellung von Adressangaben für Gebäude dar.

Ein Scope muss nicht notwendigerweise ausschließlich durch Blattknoten eines Space-Tree festgelegt sein. Falls ein Scope einen größeren Bereich abdecken soll als dies durch einzelne Blattknoten möglich wäre, wird ein entsprechend kürzerer URI gebildet. Zur Abgrenzung von Positionsangaben, wird zusätzlich ein ,\*' -Zeichen am Ende angefügt. Dies bedeutet, dass anstelle des ,\*' ein beliebiger Inhalt in der URI auftreten darf. Falls nun ein Scope zum Beispiel einen ganzen Stadtteil abdecken soll, würde folgende Darstellung gewählt werden:

**Stadtteil modelliert durch einen Scope**  
*scope://Stuttgart/Vaihingen/\**

Mit diesen Mitteln lassen sich die dynamischen Eigenschaften eines Scopes in entsprechender Weise darstellen. Eine Positionsänderung resultiert in einer Änderung des letzten Elements des URI, ausgenommen das ,\*' -Zeichen. Falls sich ein Scope zwischen Gebäuden oder Stadtteilen bewegt, würde dies folgendermaßen dargestellt werden:

**Bewegung zwischen Gebäuden:**

*scope://Stuttgart/Vaihingen/Universitätsstraße/38*

*scope://Stuttgart/Vaihingen/Universitätsstraße/36*

**Bewegung zwischen Stadtteilen:**

*scope://Stuttgart/Vaihingen/\**

*scope://Stuttgart/Botnang/\**.

Eine Vergrößerung erfolgt durch Ersetzen der Bezeichnungen der zusammenfassenden Positionsmengen durch das ,\*' -Zeichen. Bei einer Verkleinerung wird der URI entsprechend wieder um ausgewählte Bezeichnungen von Positionsmengen erweitert.

### 3.5.1 Scopelevel

Die Anzahl der Bezeichner von Positionsmengen innerhalb der Scopedarstellung, legt den sogenannten Level des Scopes fest. Dieser bestimmt, abhängig vom Umgebungsmodell, die ungefähre Größe des Bereichs, der durch einen Scope abgedeckt wird. Je kleiner der Level ist, umso größer ist der abgedeckte Bereich.

Tabelle 2 verdeutlicht diesen Sachverhalt.

<i>Scope</i>	<i>Level</i>
<i>scope://Stuttgart/Vaihingen/Universitätsstraße/38</i>	4
<i>scope://Stuttgart/Botnang/*</i>	2

**Tabelle 2:**  
**Beispiele für Scopelevel**

Ein Scope mit Level 1 würde die gesamte modellierte Welt abdecken. Die maximale Größe des Levels ist abhängig vom jeweiligen Umgebungsmodell. Über diesen Wert ist es möglich, eine ungefähre Größe von Scopes zu definieren. Wie präzise dieser Wert ist, hängt von den Definitionen der Positionsmengen des Umgebungsmodells ab. In 3.2 wurde angenommen, dass die Knoten einer Ebene des Space-Tree ähnlich große Bereiche der realen Welt modellieren. Unter dieser Annahme kann anhand des Scopelevels die Größe eines Scopes angegeben werden. Im weiteren Verlauf dieser Arbeit werden aus diesem Grund die Begriffe Scopelevel und Scopegröße des öfteren als Synonym füreinander verwendet.

## 4 Consistent Update Diffusion-Verfahren (CUD)

CUD ist ein Verfahren, welches einen schwachen Konsistenzbegriff etabliert, der es gestattet eine konsistente Datenreplikation auf einer verteilten Datenbank zu realisieren. Das Einsatzgebiet dieses Verfahrens sind Mobile Ad Hoc Netzwerke (MANETs). Das Systemmodell, welches diesem zugrunde liegt wird im folgenden Abschnitt vorgestellt.

### 4.1 Systemmodell des CUD

Die Grundbausteine des Systemmodells sind beobachtbare Objekte, Beobachter (Observer) Datenbankknoten und Klienten.

*Beobachtbare Objekte* weisen Eigenschaften auf, die von Observern erfasst werden können. Hierbei gibt es keine Einschränkung, was ein beobachtbares Objekt sein kann. Beispiele für erfassbare Eigenschaften sind Temperatur, Geschwindigkeit oder Ähnliches. Der Zustand eines beobachtbaren Objektes wird mit einem Zustandsdatensatz beschrieben, in dem neben den für das CUD relevanten Werten die beobachteten Objekteigenschaften enthalten sind.

*Observer* verfügen über Sensoren, um bei den Eigenschaften unterschiedlicher beobachtbarer Objekte bestimmte Zustandsänderungen erfassen zu können. Sie agieren unabhängig voneinander und verfügen nicht über synchronisierte Uhren. Die erfassten Zustandsänderungen werden an Datenbankknoten übermittelt.

*Datenbankknoten* sammeln die von Observern erfassten Zustandsänderungen und bilden auf diese Weise eine verteilte Datenbank, welche die jeweils aktuellsten Zustände (in Bezug auf die Realzeit) der beobachtbaren Objekte zur Verfügung stellt. Damit die verteilte Datenbank in einem konsistenten Zustand gehalten wird, tauschen die Datenbankknoten ihr Wissen um Objektzustandsänderungen untereinander aus.

Die Anfragen an Datenbankknoten gehen von *Klienten* aus, die Informationen über die Objektzustände abfragen können. Es wird angenommen, dass der Programmcode des Datenbankknoten und des Klienten auf dem selben mobilen Gerät ausgeführt werden, damit der Aufwand einer zuverlässigen Kommunikation zwischen diesen vernachlässigt werden kann.

Zusätzlich wird angenommen, dass ein Klient ausschließlich auf den lokalen Datenbestand zugreift und somit nicht mit entfernten Datenbankknoten kommuniziert.

Observer, Datenbankknoten und beobachtbare Objekte können sich beliebig und unabhängig voneinander in der realen Umgebung bewegen. Die Kommunikation von Klient und Datenbankknoten beruht auf lokalen Kommunikationsmechanismen, während für alle anderen Kommunikationen drahtlose Medien, zum Beispiel Bluetooth [BLCO99] oder WLAN

[WLST99], genutzt werden können.

Die Informationsausbreitung wird mit Hilfe von Broadcasts erreicht, welche unter Einsatz eines Flooding-Verfahrens realisiert sind. Um von unterschiedlichen Flooding-Verfahren abstrahieren zu können, wird eine allgemeine Funktion  $f\_forward$  angenommen, welche die Nachrichtenausbreitung unabhängig vom tatsächlich eingesetzten Flooding-Verfahren durchführt. Weiterhin wird angenommen, dass  $f\_forward$  mit best-effort-Semantik arbeitet und zufallsbasierte Algorithmen einsetzt, um Kollisionen im Netzwerk zu vermeiden.

Abbildung 5 verdeutlicht die in diesem Abschnitt erläuterten Zusammenhänge.

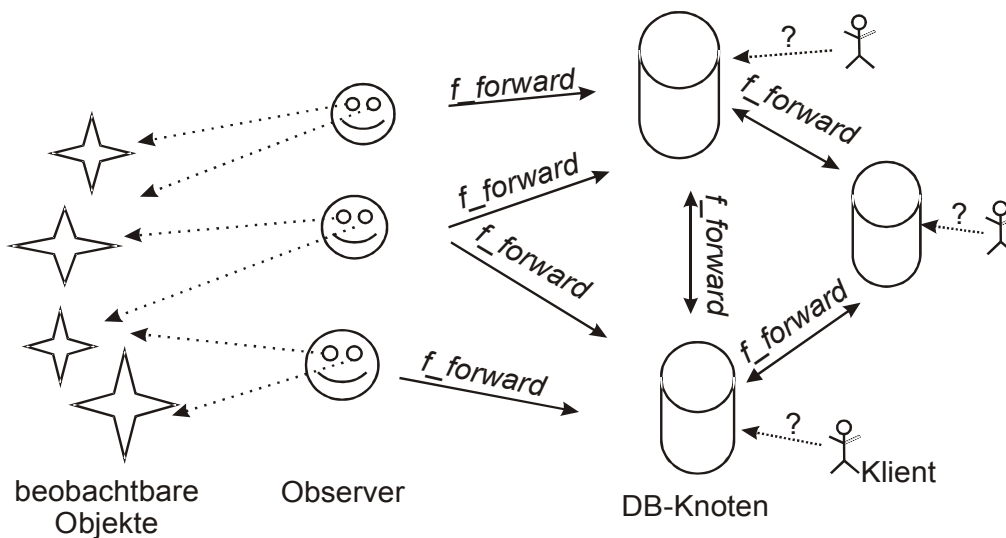


Abbildung 5: Systemmodell CUD

## 4.2 Anforderungen für eine Scopeverwaltung im CUD

Ergänzend zu dem im Kapitel 4.1 vorgestellten Systemmodell, müssen zusätzliche Annahmen formuliert werden, damit der Einsatz von Scopes ermöglicht werden kann.

Scopes stellen einen räumlichen Bezug zu Objektinformationen her. Damit dieser räumliche Bezug berücksichtigt werden kann, muss jeder Knoten, also sowohl Datenbankknoten als auch Observer, seine Position kennen. Für die Ermittlung der eigenen Position kann in freiem Gelände das satellitengestützte „Global Positioning System“ (GPS) [KAP96] verwendet werden, während für Anwendungen in geschlossenen Räumen sich Verfahren wie das ultraschallgestützte ActiveBat [WJH97] oder das mit Infrarot arbeitende ActiveBadge-Verfahren [WHF+92] eignen.

Die festgestellte Position muss in Bezug zu der realen Umgebung gesetzt werden können, in der eine Anwendung zur Ausführung kommt. Erreicht wird dies mit Hilfe eines Umgebungsmodells, welches jedem Knoten lokal zur Verfügung steht. Diese Annahme wird

getroffen um die Kommunikation mit einem Location-Service vernachlässigen zu können. Hierbei handelt es sich um ein hierarchisches mengebasiertes Modell, damit Scopes entsprechend gehandhabt werden können.

Für die Scopeverwaltung müssen ausschließlich die relativen Bewegungen von Datenbankknoten und Objekten zueinander berücksichtigt werden, was o.B.d.A. keine nennenswerten Einschränkungen für die Gesamtergebnisse zur Folge hat, da eine dynamische Scopeentwicklung nur von dieser relativen Bewegung abhängig ist. Aus diesem Grund wird in Bezug auf die Observer, eine weitere Einschränkung zur Vereinfachung gemacht. Um zusätzliche Dynamik in den Objektbeobachtungen zu vermeiden, wird im folgenden angenommen, dass sich Observer nicht bewegen, sondern einmalig an festen Orten in der realen Umgebung positioniert werden.

Im Gegensatz zum ursprünglichen CUD werden nicht alle Datenbankeinträge auf allen Datenbankknoten repliziert, sondern nur auf Datenbankknoten, welche sich innerhalb eines Scopes für ein bestimmtes Objekt befinden. Aus diesem Grund ist die Verfügbarkeit der Objektzustandsinformationen immer auf den zugehörigen Scope beschränkt.

## **5 Scopeverwaltung im CUD**

Dieses Kapitel beschreibt, wie die Scopeverwaltung in das CUD-Verfahren integriert wurde. Nach der Skizzierung der Problemstellung und wird der gewählte konkrete Ansatz vorgestellt. Die darauf folgenden Abschnitte stellen Lösungsmöglichkeiten für die Handhabung der dynamischen Scopeeigenschaften vor und beschreiben den Einfluss, den die Scopeverwaltung auf die verteilte Datenbank nimmt.

### **5.1 Problemstellung**

Das Systemmodell des CUD sieht Objekte vor, deren Zustandsänderungen von Observern erfasst werden. Im ursprünglichen Verfahren wurden diese Änderungen ohne Berücksichtigung weiterer Umstände mit Hilfe eines Flooding-Mechanismus unter den Knoten des MANETs ausgetauscht.

Dies führte zu Situationen, in denen eine starke Belastung des Netzwerkes auftrat insbesondere, wenn viele Knoten Broadcasts mit Zustandsinformationen übertrugen. Diese Belastungssituation führte dazu, dass sich Zustandsänderungen immer schlechter verbreiteten. Insbesondere resultierte es darin, dass Nachrichten mit Zustandsänderungen immer mehr Zeit benötigten, um sich über die Datenbankknoten hinweg zu verbreiten oder auch komplett verloren gingen, da die Empfangspuffer der Geräte aufgrund des übermäßigen Datenverkehrs überlastet waren.

Eine solche Situation ist nicht wünschenswert, weshalb das im Folgenden beschriebene Konzept entwickelt wurde.

### **5.2 Idee**

Um die Netzwerklast zu beschränken wird angenommen, dass ein beobachteter Objektzustand nur in einem bestimmten räumlichen Bereich von Interesse ist. Das bedeutet, die Zustandsänderung soll sich nur in einem eingeschränkten räumlichen Bereich ausbreiten. Über diesen Bereich hinaus findet keine Weiterleitung der Information statt. Die Lokalität von Informationen wird auf diese Weise berücksichtigt.

Als Beispiel dient eine Großbaustelle. Es werden auf einem Gelände verschiedene Gebäude errichtet, wobei für jedes Gebäude ein Vorarbeiter vorgesehen ist. Die Vorarbeiter seien für die Bauarbeiter in einem Gebäude verantwortlich. Ein System, das eine Koordination der Arbeiter ermöglicht, sollte zum Beispiel die Informationen, welche für einen Vorarbeiter interessant sind, lediglich innerhalb des Gebäudes verbreiten, für das er zuständig ist. Ein

solches System könnte beispielsweise die Positionen verschiedener Handwerker oder Spezialwerkzeuge verwalten oder auch Informationen sammeln, die den Baufortschritt dokumentieren, wie zum Beispiel den Trocknungsgrad von Beton in einem bestimmten Bereich der Baustelle. In diesem Beispiel umfasst der eingeschränkte räumliche Bereich somit ein Gebäude. Denkbar wäre auch ein Bauleiter, der sich von Gebäude zu Gebäude bewegt und eine jeweils angepasste Situation präsentiert bekommt und auswerten kann.

Für Szenarien dieser Art bietet sich der Einsatz von Scopes an. Der Bereich, in dem sich eine Information ausbreiten kann, wird durch einen Scope begrenzt. Innerhalb eines Scopes werden die Informationen in einem MANET mit Hilfe von Broadcasts verbreitet, da auf einer Baustelle noch keine Netzwerkinfrastruktur zur Verfügung steht.

Das Umgebungsmodell, welches diesem Szenario zugrunde liegen würde, könnte zum Beispiel mit Hilfe eines Bauplans für die zu errichtenden Gebäude erstellt werden.

### **5.3 erweiterter Objektzustand**

Bei Beobachtung einer Zustandsänderung wird der Ort der Beobachtung berücksichtigt. Hier wird der Zustandsdatensatz, welcher eine beobachtete Zustandsänderung beschreibt, um ein Attribut ergänzt, in dem der Scope für eine Informationsausbreitung eingetragen wird. Das Format dieses Eintrags orientiert sich an der in Kapitel 3.2 vorgestellten Darstellung für Scopes. Da dieser Zustandsdatensatz mit jeder Nachricht übertragen wird, führt diese Vorgehensweise, im Vergleich zum ursprünglichen Verfahren, zu einer Steigerung des übertragenen Datenvolumens und damit auch zu einer Erhöhung der Netzwerklast.

In einer realen Anwendung würde sich zur Vermeidung einer überhöhten Netzwerkbelastung eine kompaktere Darstellung der Scopeinformationen anbieten.

Eine Möglichkeit wäre, die Knoten des Space-Tree, der das Umgebungsmodell darstellt, durchnummerieren. In diesem Fall muss lediglich die Nummer des Knotens übertragen werden, welcher den zu verwendenden Scope repräsentiert. Da angenommen wird, dass ein identisches Umgebungsmodell lokal auf jedem Knoten zur Verfügung steht, kann diese Knotenbezeichnung korrekt interpretiert und in das Umgebungsmodell eingeordnet werden.

Die URI-Form wird in dieser Ausarbeitung eingesetzt, um eine möglichst plastische Darstellung von Scopes zu erhalten.

## **5.4 Verfahren**

Dieser Abschnitt beschreibt wie Scopeinformationen bei Observern und Datenbankknoten gehandhabt und berücksichtigt werden, um die Ausbreitung von Zustandsänderungen, gemäß den Vorgaben der Scopeverwaltung, einzugrenzen.

### **5.4.1 Scopebehandlung beim Observer:**

Ein Observer verfügt über Sensoren, die einen bestimmten Objektzustand erfassen können, zum Beispiel eine Temperatur oder eine Schadstoffkonzentration. Über diese Sensoren ist ein Observer in der Lage, Veränderungen im beobachteten Zustand zu erkennen. Gemäß dem CUD würde er in diesem Falle eine O\_UPDATE-Nachricht erzeugen und senden.

Mit einer O\_UPDATE-Nachricht signalisiert ein Observer also, dass er an einem bestimmten Objekt eine Zustandsänderung erfasst hat. Damit nun die Ausbreitung dieser Information eingeschränkt werden kann, muss zusätzlich die Position der Veränderung erfasst werden.

Es wird angenommen, dass jedem Observer seine eigene Position bekannt ist. Die Observerposition stellt somit eine Annäherung an den Ort der Beobachtung dar. Mit entsprechender Sensorik ließe sich der Grad der Annäherung verbessern, falls der Observer in der Lage ist die genaue Objektposition zu erfassen.

Diese Positionsinformation wird im Zustandsdatensatz, welcher einen Objektzustand beschreibt, abgelegt und mit der O\_UPDATE-Nachricht per Broadcast gesendet.

### **5.4.2 Scopebehandlung beim Datenbankknoten:**

Ein Datenbankknoten kann Positionsinformationen von Observern und Scopeinformationen von anderen Datenbankknoten empfangen. Diese werden unterschiedlich behandelt. Die verschiedenen Vorgehensweisen werden im folgenden detailliert vorgestellt.

#### **5.4.2.1 Scopeempfang vom Observer**

Die O\_UPDATE-Nachrichten der Observer werden von Datenbankknoten empfangen. Beim Empfang dieser Nachrichten wird nicht überprüft, ob sich die eigene Position innerhalb des empfangenen Scopes befindet. Das bedeutet, dass Observer-Informationen grundsätzlich immer angenommen werden. Käme es zu einer Überprüfung an dieser Stelle, so würde eine Zustandsänderung nur von den Datenbankknoten akzeptiert werden, welche sich in der selben Positionsmenge befinden, wie der sendende Observer. Daraus würde resultieren, dass sich ein Scope niemals über die Positionsmenge hinweg ausdehnen könnte, in der sich der jeweilige Observer befindet.

Die Ausdehnung von Scopes wird dadurch initiiert, dass ein Datenbankknoten für das gleiche Objekt von verschiedenen positionierten Observern Zustandsinformationen empfängt, von denen

dann die Scopes zu einem neuen, größeren Scope zusammengefasst werden können. Die minimal erreichbare Ausbreitung ist somit von der Sendeleistung eines Observers abhängig. Je weiter dieser Senden kann, desto mehr Datenbankknoten sind in der Lage, eine Observerinformation zu empfangen, auch über die Positionsmenge hinaus, in der sich ein Observer befindet.

Diese Vorgehensweise spricht nicht gegen das ursprüngliche Verfahren. Das Ziel ist, einen möglichst konsistenten und vollständigen Datenbestand von Zustandsinformationen zu erhalten. Die Annahme der Observernachrichten ohne Berücksichtigung des Scopes trägt dazu bei, dass Datenbankknoten möglichst viele Zustände verschiedener Objekte empfangen und speichern können.

Wird aufgrund einer Observernachricht der Objektzustand innerhalb der lokalen Datenbank verändert, so würde der Datenbankknoten eine N\_UPDATE Nachricht senden. Bei diesem Vorgang wird dann überprüft, ob die eigene Position innerhalb des unter Umständen neu berechneten Scopes liegt, welcher in der zu sendenden Nachricht enthalten ist.

Ist dies nicht der Fall, würde die Nachricht nicht gesendet werden. Damit wird verhindert, dass sich Objektzustandsinformationen über deren Scope hinaus ausdehnen, da neue Zustände von Observern zwar angenommen, aber nicht ohne Scopeüberprüfung weiterpropagiert werden würden.

Falls eine bessere Replikation erreicht werden soll, kann auf die Überprüfung des Scopes beim Senden der N\_UPDATE-Nachricht verzichtet werden, weil auch beim Empfang dieses Nachrichtentyps eine Prüfung der eigenen Position in Bezug auf den empfangenen Scope stattfindet. Diese Vorgehensweise würde allerdings die Netzwerkklast steigern. Ein Vorteil dieser Modifikation wäre, dass Datenbankknoten, die sich außerhalb der Sendereichweite eines Observers, aber innerhalb eines gültigen Scopes befinden, Zustandsinformationen von Datenbankknoten, die sich außerhalb des Scopes befinden annehmen können. Die Datenbankknoten außerhalb des Scopes würden eine Art Brücke für Zustandsinformationen darstellen.

Eine bessere Steuerung der Annahme der Zustandsinformationen von Observern kann über die Verwendung sogenannter Empfangssscopes erreicht werden. Bei dieser Modifikation wird jedem Datenbankknoten ein Scope zugeteilt in dem er sich befinden muss, damit Observerinformation im Sinne der Scopeverwaltung als gültig angesehen und akzeptiert werden. Bewegt sich ein Datenbankknoten aus einem Empfangssscope hinaus, wird er keine O\_UPDATE Nachrichten von Observern mehr akzeptieren. Sind für alle Datenbankknoten eines MANETs entsprechende Empfangssscopes festgelegt, kann die Replikation von Zustandsinformationen entsprechend gesteuert werden. Da Observernachrichten nur in eingeschränkten Bereichen akzeptiert werden, ist sichergestellt, dass sich ein Scope nicht über

diese Bereiche hinweg ausdehnen kann. Hierbei ist allerdings zu beachten, dass die Empfangssscopes entsprechend auf jedem Datenbankknoten im MANET definiert sein müssen. Haben Datenbankknoten unterschiedlich ausgedehnte Empfangssscopes so kann sich der Bereich der Replikation bis auf die Größe des größten definierten Scopes ausdehnen, falls sich diese Datenbankknoten innerhalb ihrer Empfangssscopes aufhalten.

#### **5.4.2.2 Scopeempfang von Datenbankknoten**

Ein Datenbankknoten reagiert bei Änderung einer Objektzustandsinformation in seiner lokalen Datenbank mit dem Senden einer N\_UPDATE-Nachricht, die diese Zustandsänderung beim sendenden Datenbankknoten beschreibt.

Eine N\_UPDATE-Nachricht enthält den neuen Objektzustand und eine Scopeinformation für das zugehörige Objekt. Die Scopeinformation kann von einem Datenbankknoten modifiziert werden um die dynamischen Scopeeigenschaften zu realisieren.

Datenbankknoten, welche diese Nachrichten empfangen nutzen diese, um ihren eigenen Datenbestand in einem möglichst, in Bezug auf die Realzeit, aktuellen Zustand zu versetzen. Der Grad der Aktualität ist von den, durch das CUD festgelegten, Konsistenzbedingungen abhängig.

Sowohl beim Senden als auch beim Empfangen von N\_UPDATE-Nachrichten, wird eine Überprüfung durchgeführt, ob der empfangene beziehungsweise zu sendende Scope die eigene Position enthält. Ist dies nicht der Fall, wird der Sende- beziehungsweise Empfangsvorgang nicht durchgeführt.

Mit dieser Vorgehensweise wird erreicht, dass keine Objektinformation über deren Scopegrenzen hinaus verbreitet wird. Befindet sich ein Datenbankknoten außerhalb eines Scopes, wird er die daraus empfangenen Informationen nicht berücksichtigen.

## 5.5 Schichtenmodell

Als Struktur des Gesamtsystems der Scopeverwaltung beim CUD wurde ein Schichtenmodell gewählt, das in Abbildung 6 dargestellt wird.

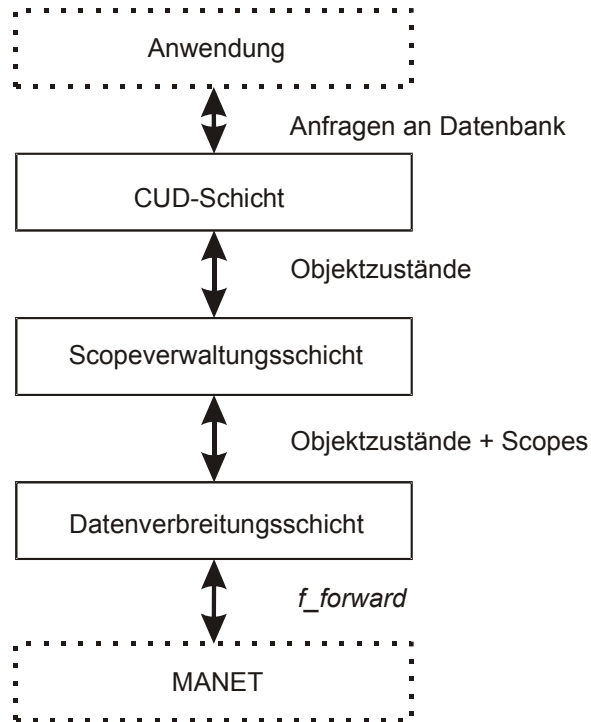


Abbildung 6: Schichtenmodell

Auf unterster Ebene ist die Schicht, die für die Datenausbreitung zuständig ist. Diese kann beispielsweise ein beliebiges Flooding-Verfahren zum Lösen dieser Aufgabe einsetzen.

Oberhalb dieser Schicht befindet sich die Scopeverwaltung, die anhand der empfangenen Scopeinformation entscheidet, ob ein empfangener Objektzustand angenommen oder abgelehnt wird.

Über der Scopeverwaltung befindet sich die CUD-Schicht, welche die Konsistenz der zu replizierenden Daten gewährleistet.

Wird ein Objektzustand von der CUD-Schicht akzeptiert und soll weiterverbreitet werden, so wird dieser, in Form des Zustandsdatensatzes, inklusive der zugehörigen Scopeinformation an die Scopeverwaltungsschicht übermittelt, so dass hier der Scope für die nächste Sendeoperation ermittelt werden kann. Ist die aktuelle Position des Datenbankknotens in dem zu sendenden Scope enthalten, so werden die Informationen an die Verbreitungsschicht übergeben und von dieser übertragen. Beim Senden einer Zustandsinformation werden die

Scopes für jedes Objekt zwischengespeichert um so eine Zusammenfassung zu größeren Scopes zu ermöglichen. Die hierbei zum Einsatz kommenden Vorgehensweisen werden im folgenden Kapitel 5.6 detailliert beschrieben.

Innerhalb der CUD-Schicht findet keine Verarbeitung der Scopeinformation aus dem Zustandsdatensatz statt, sie wird nur durchgereicht. Alternativ zum Durchschleifen der Scopeinformation wird diese bereits beim Empfang in der Scopeverwaltungsschicht für jedes Objekt gespeichert und kann dann für eine mögliche Neuberechnung des Scopes verwendet werden, falls für ein Objekt eine entsprechende N\_UPDATE-Nachricht zu senden ist. Diese Vorgehensweise stellt eine saubere Trennung der Schichten sicher, weil jede Schicht nur die Informationen präsentiert bekommt und handhabt, die auch wirklich für sie von Bedeutung sind. Die Scopeverwaltungsschicht verfügt über eine Schnittstelle, um die jeweilige Position eines Datenbankknotens ermitteln zu können. Weiterhin erfolgt in ihr der Zugriff auf das Umgebungsmodell, das benötigt wird, um Scopes verwalten zu können.

## **5.6 Dynamische Scopes**

Im Systemmodell wird davon ausgegangen, dass sich sowohl die Knoten des MANETs, als auch die zu beobachtenden Objekte innerhalb der realen Welt bewegen können. Zustandsänderungen eines Objekts sollen sich in einem begrenzten Bereich um den Beobachtungsort herum ausbreiten. Sie sind also von der Position des beobachteten Objekts abhängig.

Dabei ist es sinnvoll, den Bereich der Ausbreitung an den Bewegungsbereich des Objektes anzupassen. Diese Anpassung wird von den Datenbankknoten vorgenommen werden, da diese Zustandsänderungen eines Objekts von verschiedenen Observern empfangen können. Würde ein Observer das selbe Objekt im Verlauf der Anwendung ausschließlich an der gleichen Position beobachten, so würde sich der Ausbreitungsbereich auf diese Position beschränken.

Wird ein Objekt an verschiedenen Positionen beobachtet und registriert der Datenbankknoten dies aufgrund der von den Observern empfangenen O\_UPDATE-Nachrichten, so ist dieser in der Lage, den Scope entsprechend der Beobachtungspositionen anzupassen.

Zu diesem Zweck verfügt jeder Datenbankknoten über einen Scopepuffer, in dem die unterschiedlichen für ein Objekt empfangenen Scopes abgelegt werden. Die gepufferten Scopeinformationen werden vor Sendung einer N\_UPDATE-Nachricht ausgewertet. Bei diesem Vorgang wird der Scope ermittelt, der alle im Scopepuffer abgelegten Scopes enthält. Es wird die kleinste Obermenge berechnet, die den Bereich abdeckt, in dem ein Objekt beobachtet wurde.

Das folgende Beispiel verdeutlicht den Vorgang der Scopezusammenfassung.

<p><b>Inhalt des Scopepuffers:</b></p> <p>a) <i>scope://Stuttgart/Vaihingen/Allamndring/8</i></p> <p>b) <i>scope://Stuttgart/Vaihingen/Allmandring/10</i></p> <p>c) <i>scope://Stuttgart/Vaihingen/Universitätsstraße/*</i></p> <p><b>Zusammenfassung von a und b:</b></p> <p><i>scope://Stuttgart/Vaihingen/Allmandring/*</i></p> <p><b>Zusammenfassung von a,b und c:</b></p> <p><i>scope://Stuttgart/Vaihingen/*</i></p>
---

Der neu zu berechnende Scope wird also anhand der Darstellung der gepufferten Scopes ermittelt werden kann: Gesucht wird das längste Prefix, welches in allen zu berücksichtigenden Scopedarstellungen vorhanden ist.

Im Verlauf einer Anwendung kann es vorkommen, dass immer mehr unterschiedliche Scope- und Positionsinformationen für ein Objekt gepuffert werden, so dass der Bereich der Informationsausbreitung immer weiter wächst. Je weiter diese Entwicklung für ein Objekt fortschreitet, desto mehr verhält sich die Informationsausbreitung für das jeweilige Objekt wie im ursprünglichen CUD, da irgendwann die größte Positionsmenge, welche die gesamte modellierte Welt umfasst, als Scope-Obermenge für ein Objekt berechnet werden würde. Solange diese Scopeausdehnung nur für eine begrenzte Anzahl von Objekten stattfindet, ist dies nicht problematisch.

In Anwendungen, in denen sich viele Objekte innerhalb großer Bereiche der realen Umgebung bewegen, kann es zu den Performanceproblemen führen, die auch das CUD aufweist (siehe Kapitel 5.1). Es müssen Maßnahmen getroffen werden, die ein übermäßiges Scopewachstum verhindern.

Eine Möglichkeit ist, eine maximale Scopegröße für jedes Objekt festzulegen. Diese Einschränkung kann die Informationsausbreitung für ein Objekt allerdings stark beschränken, falls dessen Bewegungsbereich größer ist als die maximal mögliche Scopegröße. Zusätzlich ist der Verwaltungsaufwand für die möglicherweise unterschiedlichen Scopegrößen der

beobachtbaren Objekte in Betracht zu ziehen. Es müssten zusätzliche Informationen übertragen werden, da zu jedem Objekt prinzipiell eine andere maximale Größe definiert werden kann. Das bedeutet, dass neben dem Scope einer Objektinformation auch die maximale Größe mit übertragen werden muss, um zu verhindern, dass ein Scope als „zu groß“ berechnet wird. Dies kann umgangen werden, indem Scopes unmittelbar bis auf ihre maximal mögliche Größe ausdehnt werden, was allerdings in einer Informationsausbreitung in einem relativ großen Bereich resultieren kann und in dem Fall zu einer höheren Netzwerklast führt.

Besser ist es, wenn sich die Scopegröße an das Bewegungsmuster eines Objekts anpasst. Erreicht wird dies, indem die aktuellsten Scopeinformationen zur Bestimmung eines neuen Scopes berücksichtigt werden. Durch die Begrenzung der Anzahl der zu berücksichtigten Scopeinformationen wird verhindert, dass ein Scope übermäßig wächst.

Generell gilt die Regel:

**Mit der Anzahl unterschiedlicher Scopeinformationen, die berücksichtigt werden, steigt die Wahrscheinlichkeit, dass ein Scope sich ausdehnt.**

Um diese Anzahl zu begrenzen, gibt es verschiedene Ansätze. Eine Strategie wäre, Scopeinformationen altern zu lassen. Veraltete Scopeinformationen werden aus dem Puffer entfernt und für die zukünftige Bestimmung neuer Scopes nicht mehr berücksichtigt. Das heißt, falls ein Objekt innerhalb eines Bereiches der realen Umgebung über einen Zeitraum hinweg nicht mehr beobachtet wird, so wird dieser Bereich nicht mehr berücksichtigt.

Eine andere Strategie wäre, einen Puffer zu verwenden, der nur eine bestimmte Anzahl von Scopes speichern kann. Hier werden lediglich die zuletzt empfangenen Scopeinformationen zur Bestimmung eines neuen Scopes verwendet.

Das folgende Kapitel stellt einige mögliche Strategien zur Verwaltung von Scopes vor, die auf verschiedenste Arten das Wachstum eines Scopes begrenzen.

## **5.7 Pufferstrategien**

Die folgenden Abschnitte betrachten verschiedene Pufferstrategien detaillierter. Im Wesentlichen gibt es zwei Charakteristika an denen sich Pufferstrategien unterscheiden lassen: Zum einen gibt es Strategien, welche die Zeit als eine Obergrenze verwenden und zum anderen wird der zur Verfügung gestellte Speicherplatz als eine Obergrenze für die Ablage von Scopeinformationen verwendet.

### **5.7.1 Ringpuffer**

Ein Ringpuffer verhindert übermäßiges Scopewachstum dadurch, dass er nur eine begrenzte Anzahl von Scopes speichert. Die maximale Größe eines Scopes wird nicht beschränkt.

Diese wird durch die Kombination der Einträge im Puffer beeinflusst und kann prinzipiell bis auf die maximal mögliche Größe eines Scopes anwachsen. Dieser Fall tritt ein, wenn eine Kombination von Scopes gepuffert wird, deren Obermenge der Positionsmenge entspricht, welche die gesamte reale Umgebung abdeckt.

Wird davon ausgegangen, dass regelmäßig neue Scopeinformationen empfangen werden, so werden die im Puffer vorhandenen Einträge mit neueren Informationen überschrieben. Dies kann in einer Ausdehnungsänderung eines Scopes resultieren. Das bedeutet, dass ein Scope durchaus für einen bestimmten Zeitraum bis auf die gesamte Umgebung ausgedehnt wird, aber aufgrund aktuellerer Informationen auch wieder auf eine kleinere Ausdehnung zusammenfallen kann.

Ein Schwachpunkt des Ringpuffers ist seine Abhängigkeit vom Sendeverhalten anderer Knoten. Dies umfasst sowohl Observer als auch Datenbankknoten. Falls der selbe Knoten in schneller Folge neue Zustandsinformationen sendet, kann es passieren, dass die selbe Scopeinformation mehrmals im Puffer abgelegt wird und im Extremfall den gesamten Puffer füllt. Die kleinste gemeinsame Obermenge entspräche dann lediglich dieser einen Positionsmenge.

Eine größere Positionsmenge kann nur von Datenbankknoten empfangen werden, da nur diese Einfluss auf die Ausdehnung eines Scopes nehmen können. Datenbankknoten senden nur dann neue Scopeinformationen, wenn sich der in ihnen gespeicherte Objektzustand verändert hat. Es kann somit angenommen werden, dass der empfangene Scope immer an aktuelle Zustandsinformationen gekoppelt ist und seine Ausdehnung auf entsprechend aktuellen Scopeinformationen beruht.

Werden von einem Observer in kurzer Zeit viele Scopeinformationen empfangen und führt dies zu einer kompletten Füllung des Puffers mit der selben Scopeinformation, so resultiert das im Kollabieren des Scopes auf eine kleine Positionsmenge.

Dieses Verhalten ist im Sinne der Anwendung von Scopes nicht falsch, da sich die aktuellste Information immer in dem Bereich ausbreitet, wo sie beobachtet wurde. Ist es beabsichtigt diesen Bereich so klein wie möglich zu halten, so wäre diese Vorgehensweise durchaus angebracht. Allerdings schränkt dies eine Scopeanpassung an das Bewegungsmuster eines Objekts dahingehend ein, dass lediglich die Positionsmenge berücksichtigt wird, in der das Objekt zuletzt beobachtet wurde. Dies gilt nur unter der Annahme, dass Observer so schnell Informationen senden, dass der Puffer mit diesen gefüllt wird.

Diesen Einflüssen kann mit verschiedenen Einfügestrategien entgegengewirkt werden. Es wäre beispielsweise möglich zu prüfen, ob sich ein Scope schon im Puffer befindet um so zu verhindern, dass Einträge doppelt abgelegt werden. Eine andere Strategie ist, nur Scopes anzunehmen, welche die resultierende Obermenge erweitern würden.

Bei diesen Einfügestrategien werden weniger Scopeinformationen in den Puffer geschrieben als beim ursprünglichen Ansatz. Im Vergleich zu einem Ringpuffer, der keine spezielle Einfügestrategie verwendet, können die Scopeinformationen effektiver genutzt werden. Dieser Nutzen ist allerdings vom Sendeverhalten der Observer, den Bewegungsmustern der beobachtbaren Objekte und der Datenbankknoten abhängig, da diese Informationen die Art und auch die Berücksichtigung der Scopeinformationen beeinflussen können. Der Nachteil hierbei ist, dass sich aufgrund der empfangenen Scopeinformation ein Scope unerwünscht weit ausbreiten kann, weil Scopeinformation seltener überschrieben werden. Es gilt also, eine angepasste Größe für den Ringpuffer festzulegen, die von der Anzahl der zu empfangenden Zustandsänderungen und der jeweiligen Einfügestrategie abhängt.

Eine weitere zu berücksichtigende Abhängigkeit für die Einfügestrategie können die Art der Scopeinformationen darstellen, die eng mit den Bewegungsmustern der beobachtbaren Objekte zusammenhängen. Dies umfasst sowohl Positionsinformationen von Observern als auch Scopeinformationen von Datenbankknoten.

Der Vorteil des Ringpuffers ist, dass die Anforderungen für Speicherplatz und die Komplexität der Obermengenberechnung gut abgeschätzt werden können, weil diese Faktoren ausschließlich von der Puffergröße abhängen, von der angenommen wird, dass sie sich im Verlauf der Anwendung nicht mehr ändert.

### **5.7.2 Timeoutpuffer**

Die Idee des Timeoutpuffers ist, dass Scopeinformationen nur für eine gewisse Zeitspanne gespeichert werden. Für die Berechnung neuer Scopes werden nur Scopeinformationen berücksichtigt, die innerhalb eines vorgegebenen Zeitfensters empfangen wurden. Zu jeder Scopeinformation, die in den Puffer eingetragen wird, wird der Zeitpunkt der Eintragung mitgespeichert. Vor jeder Scopeberechnung werden Einträge, die einen vorgegebenen

Timeout-Wert überschritten haben, aus dem Puffer entfernt.

Ein Scope kann sich beliebig weit ausdehnen, würde aber nach Ablauf der durch den Timeout definierten Zeitspanne wieder zusammenfallen, falls die neu empfangenen Scopeinformationen dies gestatten. Decken die frischen Scopeinformationen ebenfalls den gesamten Bereich ab, bleibt der Scope in seiner Ausdehnung bestehen.

Sind alle Scopeinformationen für ein Objekt veraltet und werden keine neuen Zustandsinformationen für ein Objekt mehr empfangen, so kann kein neuer Scope für dieses Objekt berechnet werden. Das bedeutet, dass die Zustandsinformation nicht mehr weiterpropagiert werden kann.

Um ein versehentliches Leeren des Puffers aufgrund des Timeouts zu verhindern, sollte sichergestellt sein, dass der Timeout so gewählt ist, dass immer mindestens eine Scopeinformation im Puffer vorhanden ist. Bei dieser Vorgehensweise kann es sinnvoll sein, die Annahme zu treffen, dass Observer Objektzustände in regelmässigen Zeitintervallen senden, was eine entsprechende Aktualisierung der Zustandsinformation in den Datenbankknoten zur Folge hat. Werden unter diesen Bedingungen keine neuen Scopeinformationen empfangen, kann angenommen werden, dass sich ein Datenbankknoten nicht mehr im Gültigkeitsbereich für Zustandsinformationen eines Objektes befindet.

Die Schwierigkeit ist, einen passenden Timeoutwert zu bestimmen. Wird dieser zu klein gewählt, kann es sein, dass zu früh keine Scopeinformationen mehr zur Verfügung stehen und dadurch die Ausbreitung von Zustandsinformationen abbricht. Der Scope würde in diesem Fall zu schnell kollabieren. Eine optimale Anpassung an das Bewegungsmuster eines Objektes ist unter diesen Bedingungen nicht gewährleistet. Bei zu großer Auslegung des Timeouts kann sich ein Scope unter Umständen zu weit ausdehnen. Ein Seiteneffekt von großen Timeoutwerten ist es, dass theoretisch auch mehr Speicherplatz für Scopeinformationen in Anspruch genommen werden kann, da in einem größeren Zeitraum auch mehr Informationen empfangen werden können.

Diesem Effekt kann mit Einfügestrategien entgegengewirkt werden, indem verhindert wird, dass zum Beispiel Scopeinformationen doppelt gespeichert werden oder nur Informationen abgelegt werden, die in einer größeren Obermenge resultieren würden. Bei der Vermeidung von Duplikaten sollte allerdings der Zeitpunkt des Einfügens in den Puffer, auf den Wert angepasst werden, welcher der aktuellsten Eintragung entspricht, um so zu verhindern, dass eine Positionsinformation zu früh nicht mehr für die Obermengenberechnung berücksichtigt werden würde. Der jeweilige Eintrag wird also, anstatt ihn mehrfach abzulegen, lediglich wieder aufgefrischt. Diese Vorgehensweise hat somit keinen Einfluss auf das eigentliche Pufferverhalten, da ein Eintrag erst verschwindet, wenn sein Eintragungszeitpunkt den Timeout überschreitet. Haben zwei identische Einträge unterschiedliche

Eintragungszeitpunkte, werden sie bis zum Veralten eines Eintrags beide für die Obermengenermittlung berücksichtigt. Da die Scopeinformation identisch ist, hat dies auf das Resultat keinen Einfluss. Es ist daher ohne Einschränkung möglich, ältere Einträge durch jüngere identische Einträge zu ersetzen, weil ein Einfluss auf die Obermenge erst eintritt, wenn der zuletzt abgelegte Eintrag veraltet und aus dem Puffer entfernt wird.

Bei der ausschließlichen Annahme von Scopeinformationen, die in einer Vergrößerung der Obermenge resultieren, kann es eine nachteilige Entwicklung des Scopes geben. Scopeinformationen, die schon durch im Puffer vorhandene Einträge abgedeckt sind, werden aufgrund der Einfügestrategie nicht angenommen. Das bedeutet, dass diese Informationen für eine weitere Scopeentwicklung auch nicht zur Verfügung stehen. Dies kann dazu führen, dass sich ein Scope, wenn seine ursprünglichen Scopeinformationen veralten, seine Position und Ausdehnung sprunghaft ändert, da mögliche Zwischenschritte aufgrund der Annahmestrategie nicht akzeptiert wurden.

Es wird deutlich, dass der Verwaltungsaufwand für einen Timeoutpuffer eine nicht zu vernachlässigende Größe darstellt und die Aufwände für Speicherplatz und Obermengenermittlung je nach Füllstand des Puffers variieren. Die Puffergröße, gemessen an der Anzahl der Einträge, ist nach oben hin nicht beschränkt. Je mehr Scopeinformationen gepuffert werden, umso mehr Informationen müssen gespeichert und für eine Obermengenermittlung in Betracht gezogen werden. Der Vorteil dieses Puffers ist, dass er sich an dynamisches Senderverhalten von Datenbankknoten und Observern gut anpassen kann, solange genügend Ressourcen zur Verfügung stehen.

### **5.7.3 TimeoutGrowingpuffer**

Das Prinzip des TimeoutGrowingpuffer ist, ein Scopewachstum zeitlich zu begrenzen. Das bedeutet, dass ein Scope nur über einen bestimmten Zeitraum hinweg beliebig wachsen kann, wobei ein Timeout für die Dauer des Scopewachstums definiert wird.

Die verwendete Annahmestrategie ist: Es werden nur Scopeinformationen angenommen, die ein weiteres Wachstum ermöglichen.

Ist ein Wachstumstimeout abgelaufen, gibt es verschiedene Möglichkeiten, fortzufahren. Nach Ablauf des Timeouts wird die nächste eintreffende Scopeinformation als neue Grundlage für ein Scopewachstum angenommen. Hierbei macht es einen Unterschied, ob es sich um eine Positionsinformation eines Observers oder um eine Scopeinformation von einem Datenbankknoten handelt. Letztere kann eine größere Positionsmenge abdecken, als dies mit der Positionsinformation eines Observers möglich wäre. Somit ergibt sich eine größere Ausgangsausdehnung als Grundlage für die Berechnung neuer Scopes. Das hat den Vorteil, dass der neue Scope eine gewisse Ausgangsausdehnung hat. Bei Annahme einer

Positionsinformation ist dies nicht gegeben, allerdings ist der Scope so optimal positioniert und ausgedehnt, da er ausschließlich den Bereich abdeckt, aus dem der aktuellste Objektzustand stammt. Der Vorteil einer gewissen Ausgangsausdehnung ist, dass sich die Objektinformation schneller ausbreitet, da eine Wachstumsphase bis auf diese vorgegebene Ausdehnung entfällt. Der Nachteil dieser Vorgehensweise ist, dass ein Scope sehr sprunghaft positioniert werden und wachsen oder schrumpfen kann. Die Sprünge sind durch den Timeout vorhersagbar und dadurch kontrollierbar, indem zum Beispiel als neue Grundlage nur Positionsinformationen von Observern oder von Datenbankknoten berechnete Scopeinformationen zugelassen werden.

Soll eine weite Informationsausbreitung sichergestellt werden, dient ein Scope als Grundlage. Geht es darum, möglichst aktuell und ortsnahe zu replizieren, wird eine Positionsinformation als neue Grundlage gewählt. Es ist zu beachten, dass, falls ein Scope bis auf maximale Größe angewachsen ist, eine Ausdehnungsänderung erst nach Ablauf des Timeouts wieder möglich ist. Der Timeoutwert sollte deshalb so gewählt werden, dass eine zu große Ausdehnung verhindert wird, beziehungsweise sichergestellt ist, dass die maximale Ausdehnung nicht zu lange aufrechterhalten wird.

Der Vorteil dieser Pufferstrategie ist die minimale Anforderung an den Speicherplatz, da nur der aktuelle Scope für ein Objekt gespeichert werden muss. Eine Neuberechnung der Obermenge findet nur statt, wenn eine Scopeinformation empfangen wird, die noch nicht im bisher gespeicherten Scope enthalten ist. Dies reduziert den nötigen Aufwand für die Neuberechnung von Scopes. Allerdings muss für jede angenommene Scopeinformation geprüft werden, ob sie schon abgedeckt wird oder nicht. Eine Beeinflussung durch das Sendeverhalten anderer Knoten des MANETs ist ebenfalls nicht gegeben, da bedingt durch die Einfügestrategie eine Filterung der Scopeinformationen stattfindet. Die Anwendung dieser Pufferstrategie ermöglicht gute Vorhersagen in Bezug auf Speicher- und Berechnungsaufwand. Sind also zum Beispiel überwiegend Geräte mit geringem Speicherplatz im Einsatz, kann der TimeoutGrowingpuffer eine gute Alternative zu anderen Pufferstrategien darstellen.

#### **5.7.4 Zusammenfassung**

Bei der Auswahl einer geeigneten Pufferstrategie sind unterschiedliche Faktoren zu berücksichtigen. Generell ergibt sich das Problem, dass ein Scope möglichst gut an das Bewegungsmuster eines Objekts angepasst sein sollte. Eine gute Anpassung wird erreicht, wenn möglichst viele Scopeinformationen für ein Objekt berücksichtigt werden. Das Problem der Balance zwischen guter Anpassung und übermäßigem Wachstum, aufgrund zu groß dimensionierter Puffer, besteht für jede verwendete Strategie.

Deutliche Unterschiede zeigen sich bei Anforderung an Speicherplatz und Aufwand zur

Berechnung der Obermenge. Die flexibelste Lösung stellt ein Timeoutpuffer dar, der doppelte Einträge eliminiert und lediglich die Einfügezeitpunkte anpasst. Diese Pufferstrategie hat auch die potenziell höchsten Anforderungen an Speicherplatz, da die Menge der zu puffern den Scopeinformationen im Rahmen des Timeouts, nicht beschränkt ist. Dieser Punkt wirkt sich ebenfalls auf die Komplexität der Obermengenberechnung aus, da unterschiedlich viele Scopeinformationen berücksichtigt werden müssen.

Einen guten Kompromiss in Bezug auf Speicherplatzbedarf und Berechnungskomplexität stellt der Ringpuffer dar. Durch die feste Vorgabe der Anzahl der Speicherplätze lassen sich über diese beiden Faktoren eindeutige Aussagen treffen.

Die geringsten Anforderungen an den Speicherplatz stellt der TimeoutGrowingPuffer, bei dem nur ein Scope pro Objekt gespeichert werden muss. Erkauft wird dieser Vorteil mit einem erhöhten Aufwand bei der Annahme von Scopeinformationen, da hier sofort geprüft wird, ob die neu empfangene Scopeinformation schon durch den Pufferinhalt abgedeckt ist oder nicht. Nicht außer Acht gelassen werden sollte die sprunghafte Scopeentwicklung, die sich allerdings durch den Timeout des Wachstums und die Auswahl der neuen Berechnungsgrundlage für einen Scope kontrollieren lässt.

Die Ansiedlung des Scopepuffers liegt, bezogen auf das in Kapitel 5.5 beschriebene Schichtenmodell, in der Scopeverwaltungsschicht. Eine technische Realisierungsmöglichkeit wäre, dass diese Schicht über eine einheitliche Schnittstelle zum Scopepuffer verfügt, so dass eine entsprechend leichte Konfiguration des Pufferverhaltens möglich ist, da nur die Klasse des zu instantiiierenden Puffers übergeben werden muss. Alle anderen pufferspezifischen Parameter, wie zum Beispiel die Dimensionierung, können dann in der jeweiligen Pufferklasse individuell gehandhabt werden.

## **5.8 Scopeinfluss auf die Datenhaltung**

Mit Hilfe der Scopes wird Objektzustandsinformationen ein räumlicher Gültigkeitsbereich zugeordnet. Dieser Abschnitt betrachtet die Verhaltensweise von Datenbankknoten, wenn sie sich in einen neuen Gültigkeitsbereich hineinbewegen oder einen solchen verlassen.

Die eigene Position eines Datenbankknotens wird bei jedem Empfang von Scopeinformationen anderer Datenbankknoten in Bezug dazu gesetzt. Befindet sich ein Datenbankknoten innerhalb eines empfangenen Scopes, werden die empfangenen Zustandsinformationen gemäß CUD ausgewertet. Aufgrund dieser Vorgehensweise werden automatisch die Daten verarbeitet, in dessen Gültigkeitsbereich ein Datenbankknoten sich befindet. Die Bewegung des Datenbankknotens wird dabei dahingehend berücksichtigt, dass dieser automatisch seine eigene Position permanent aktualisiert. Verlässt ein

Datenbankknoten den Gültigkeitsbereich einer Zustandsinformation, so sollte er diese Situation entsprechend berücksichtigen. Nach Verlassen eines Scopes können sich Informationen in der lokalen Datenbank befinden, welche aufgrund ihrer Scopeinformationen ungültig sind und deshalb entfernt werden können. Das Bewegungsmusters eines Datenbankknotens könnte prinzipiell so verlaufen, dass dieser sich immer wieder in einen Scope hinein und hinausbewegt. Bei jedem Verlassen würden die ungültigen Zustandsinformationen entfernt werden. Es kann hierbei vorkommen, dass sich ein Knoten innerhalb des Scopes eines Objekts befindet, aber aufgrund seines Bewegungsmusters keine aktuellen Informationen zu diesem hat, obwohl dies theoretisch möglich wäre. Aus diesem Grund bietet es sich an, einen Bereich rund um einen Datenbankknoten herum zu definieren, der es ermöglicht, dass ein Knoten in kurzer Zeit einen Scope verlassen und wieder betreten kann, ohne dass Zustandsinformationen gelöscht werden. Erreicht wird dies durch abschneiden einer vorgegebenen Anzahl von Ebenen der URI-Darstellung der eigenen Position. Diese Vorgehensweise ähnelt der Definition des Empfangssscopes wie sie in 5.4.2.1 vorgestellt wurde. Der Unterschied zum Empfangsscope ist, dass sich die Position dieses Datenhaltungssscopes immer an der eigenen Position orientieren muß. Würde die Position des Datenhaltungssscopes beliebig gewählt werden, bewertet der Datenbankknoten Scopeinformationen nur dann als gültig, wenn er sich in dem angegebenen Bereich aufhalten würde. Dies widerspricht der Idee, des Konzepts des Datenhaltungssscopes, die Bewegung über Scopegrenzen hinaus in Bezug auf die Datenhaltung zu kompensieren. Es würde diese Situation nur noch verschlimmern, da mehr Informationen als ungültig bewertet werden würden, als es das Scope-Konzept für das CUD vorsieht. Die Ausdehnung des Datenhaltungssscopes kann variabel gehalten werden. Hierbei ist zu beachten, dass die Größe immer mindestens der des Empfangssscopes entspricht. Sonst würden Zustandsinformationen in bestimmten Situationen im Sinne des Empfangssscopes als gültig und im Sinne des Datenhaltungssscopes als ungültig behandelt werden. Dieser Fall tritt ein, wenn Scopeinformationen angenommen werden, die innerhalb des Empfangssscopes, aber außerhalb des kleineren Datenhaltungssscopes liegen. Eine größere Ausdehnung des Empfangssscopes ist ohne Probleme möglich, da so die empfangenen Informationen im Sinne der Datenhaltung immer als gültig angesehen werden.

## 6 Simulation

Um die Leistungsfähigkeit von Scopes im CUD zu ermitteln, wurden Simulationen mit dem Netzwerksimulator ns2 (Version 2.26) [NS2] durchgeführt. Die hierfür geschaffenen Voraussetzungen, die Zusammensetzung der einzelnen Szenarien und die Durchführung der Simulationsdurchgänge, werden im folgenden Abschnitt vorgestellt.

### 6.1 Verfahren

Das MANET, in dem das modifizierte CUD-Verfahren zum Einsatz kommt, wurde mit Hilfe des Netzwerksimulators ns2 [NS2] unter Verwendung des MAC-Protokolls 802.11 simuliert. Die Implementierung basiert auf der ursprünglichen CUD-Implementierung, die auch in [RHB03] zum Einsatz kam. Die ursprüngliche Implementierung wurde um die Scopeverwaltung erweitert. Weiterhin wurde ein einfaches Umgebungsmodell implementiert, wie es im Kapitel 6.2 genauer beschrieben wird. Die Datenbankknoten verfügen über einen Scopepuffer, der die Scopeanpassung nach einer der in Kapitel 5.6 beschriebenen Strategien durchführt. Der mit den Nachrichten übertragene Zustandsdatensatz wurde um ein Attribut für den Scope ergänzt. Die Steuerung der Simulation wird mit einem TCL-Skript, in dem die Simulationsparameter, wie zum Beispiel die Laufzeit oder die Anzahl der Datenbankknoten spezifiziert werden, durchgeführt. Dieses wurde ebenfalls an die neuen Gegebenheiten angepasst. Es umfasst zusätzlich die Implementierung eines neuen Beobachtungsschemas für die beobachtbaren Objekte, auf welches detailliert in Kapitel 6.3 eingegangen wird.

Im Verlauf einer Simulation werden Meldungen generiert, die den genauen Ablauf dokumentieren:

- beim Empfang und Senden einer Nachricht beim Datenbankknoten
- beim Senden einer Nachricht beim Observer
- bei einem Positionswechsel eines Datenbankknotens, in Bezug auf das mengenbasierte Umgebungsmodell
- beim Annehmen und Ablehnen einer Zustandsinformation.

Der Inhalt der generierten Meldungen erlaubt eine genaue Interpretation des jeweiligen Vorgangs. Zur besseren Auswertung werden die Informationen aus den generierten Meldungen in einer Datenbank gespeichert. Die Daten werden mit Hilfe einer Tabellenkalkulation und einer Statistiksoftware ausgewertet. Abbildung 7 verdeutlicht die beschriebene Vorgehensweise.

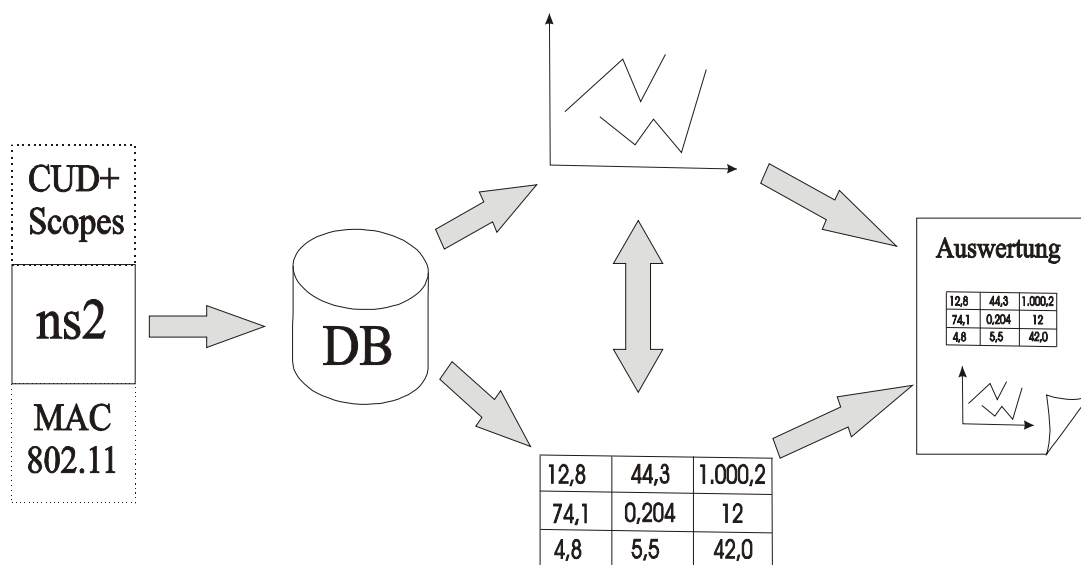


Abbildung 7: Simulation und Auswertung

## 6.2 Umgebungsmodell

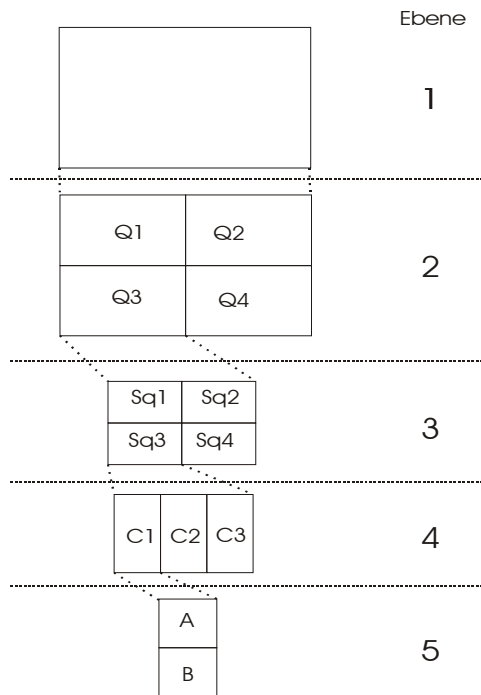
Die Grundlage für die Handhabung von Scopes ist das Umgebungsmodell. Für alle Szenarien wird ein einheitliches Modell verwendet, welches im Folgenden beschrieben wird.

Die simulierte Umgebung ist eine ebene Fläche mit einer Ausdehnung von 3000m \* 2000m (=6 km<sup>2</sup>). Auf dieser Fläche wird eine Unterteilung in Rechtecke mit einer Kantenlänge von je 250 m vorgenommen. Auf diese Weise erhält man ein Raster bestehend aus 12x8 Rechtecken. Die kleinste Ausdehnung, die ein Scope annehmen kann, besteht aus einem solchen Rechteck. Um eine Skalierung der Scopegröße zu ermöglichen wurde folgende Hierarchie auf dem Raster aus Rechtecken definiert:

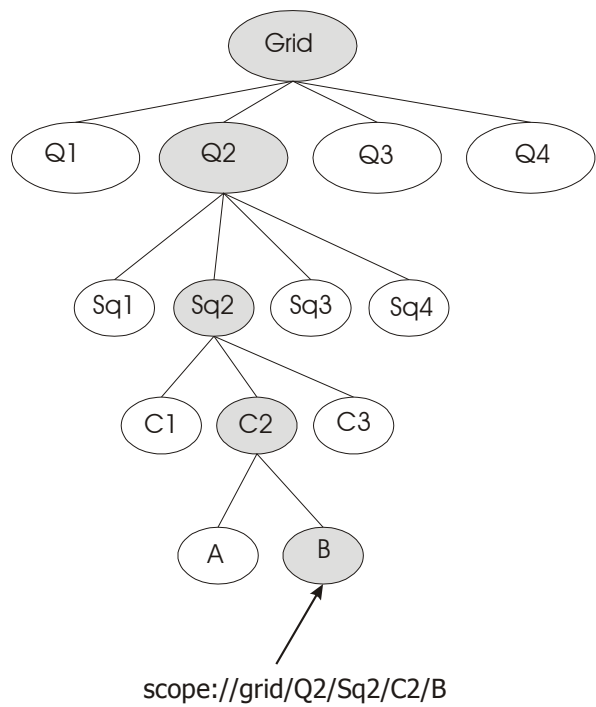
- die erste Ebene stellt die gesamte Fläche dar bestehend aus den 12x8 Rechtecken
- die zweite Ebene wird durch eine Aufteilung in vier gleichmäßige Quadranten geschaffen, welche jeweils 6x4 Rechtecke enthalten
- die dritte Ebene unterteilt jeden Quadranten in vier gleichmäßige Unterquadranten mit je 2x3 Rechtecken ein
- die vierte Ebene stellt eine Unterteilung der Unterquadranten in je zwei übereinanderliegende Rechtecke dar; es wird also eine Unterteilung in 1x2 Rechtecken vorgenommen
- in der fünften Ebene wird jedes einzelne Rechteck als Unterteilung betrachtet

Analog zu den hier aufgezählten Ebenen gestaltet sich der Scopelevel in der Darstellung für Scopes (siehe Kapitel 3.5.1).

Die vorgestellte Hierarchie wird mit Hilfe eines Space-Tree modelliert. Die Abbildungen 8 und 9 stellen die simulierte Umgebung als Fläche, sowie ausschnittsweise den daraus resultierenden Space-Tree dar:



**Abbildung 8: Hierarchie der simulierten Umgebung**



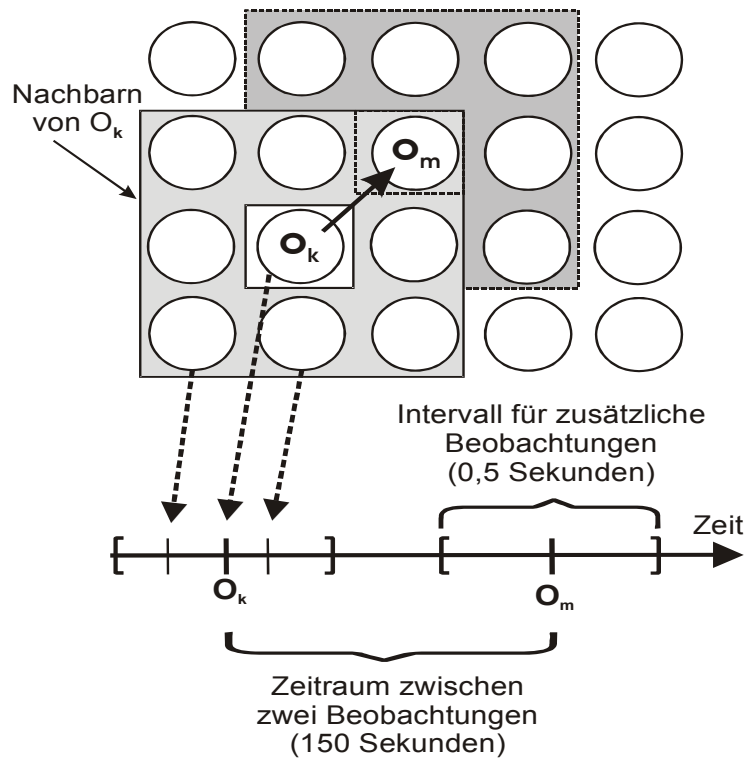
**Abbildung 9: Space-Tree der simulierten Umgebung**

In Abbildung 9 wird ein Beispiel für die Scopedarstellung anhand des Space-Tree gezeigt. Die markierten Knoten bilden den Pfad zu der Position, die durch die URI in der Abbildung wiedergegeben ist. Der Scopelevel (Kapitel 3.5.1) kann bei diesem Umgebungsmodell zwischen den Werten 1 und 5 festgelegt werden.

### 6.3 Bewegung beobachtbarer Objekte

Um die Dynamik der Scopeentwicklung einzugrenzen, werden für die Observer feste Positionen innerhalb der simulierten Umgebung festgelegt. Diese Anordnung orientiert sich am Umgebungsmodell. Es wird in jedem der 96 Rechtecke aus der untersten Hierarchieebene ein Observer mittig platziert. Im Verlauf jeder Simulation werden 10 beobachtbare Objekte von den Observern erfasst. Zu Beginn jeder Simulation wird für jedes Objekt zufällig ein Observer ausgewählt, der die erste Zustandsänderung wahrnimmt und den Datenbankknoten in seiner Sendereichweite mitteilt. Der Radius der Sendereichweite beträgt für jede Simulation 250 m. Damit ist sichergestellt, dass jeder Datenbankknoten innerhalb eines Rechtecks in dem sich ein Observer befindet, dessen Nachrichten auch empfangen kann. Ein Empfang der Observerinformation über ein Rechteck hinaus ist unter diesen Gegebenheiten ebenfalls gewährleistet. Der nächste Observer, welcher ein Objekt beobachtet, wird alle 150 Sekunden zufällig aus den direkten Nachbarn des zuletzt aktiven Observers ausgewählt. Ein beobachtbares Objekt kann somit alle 150 Sekunden seine Position wechseln, was einer Geschwindigkeit von ca. 6 km/h entspricht. Zusätzlich wird mit einer Wahrscheinlichkeit von 50 % von bis zu drei Nachbarn des zuletzt aktiven Observers die selbe Zustandsänderung erfasst und propagiert. Bei Auswahl von mehr als einem Nachbarn wird zusätzlich sichergestellt, dass die ausgewählten Observer ebenfalls benachbart sind, um zu verhindern, dass Beobachtungen von Observern generiert werden, die bezogen auf den ursprünglichen Observer entgegengesetzt positioniert sind. Diese Maßnahme wird getroffen, um eine Bewegung eines beobachtbaren Objekts eindeutig festlegen zu können, was die Anpassung von Scopes, in Bezug auf die Objektbewegung, begünstigt.

Die zusätzlichen Beobachtungen werden nicht zeitgleich mit dem aktiven Observer gesendet, sondern versetzt innerhalb eines Zeitintervalls  $[t-t_{obs}; t+t_{obs}]$  übertragen, wobei  $t_{obs}$  dem Zeitpunkt der Beobachtung des aktiven Observers entspricht. Der Parameter  $t$  wird zufällig aus einem Intervall  $[0..0,25]$  Sekunden gewählt. Die Entscheidung ob ein Observer vor oder nach der ursprünglichen Beobachtung reagiert, wird mit einer Wahrscheinlichkeit von 50% getroffen. Dieser Mechanismus verhindert, dass ausschließlich gleichzeitige Beobachtungen benachbarter Observer generiert werden, schließt diesen Fall jedoch nicht vollständig aus. Damit eine repräsentative Menge an Beobachtungen durchgeführt und unter den Datenbankknoten verbreitet werden kann, wird die simulierte Zeit bei allen Simulationen mit 7200 Sekunden (2 Stunden) festgelegt.



**Abbildung 10: Beobachtungsmuster der Observer**

Abbildung 10 zeigt einen Ausschnitt aus der simulierten Umgebung. Zu einem bestimmten Zeitpunkt beobachtet der Observer  $O_k$  ein Objekt. Zusätzlich beobachten zwei seiner Nachbarn das gleiche Objekt, wobei deren Beobachtungen zeitversetzt stattfinden. Nach 150 Sekunden würde der Observer  $O_m$  das Objekt als nächstes beobachten.

Durch die gleichverteilte Auswahl des nächsten Observers, der ein Objekt beobachten soll, wird ein zufälliges Bewegungsmuster für die beobachtbaren Objekte generiert. Im Verlauf einer Simulation können sich alle beobachtbaren Objekte über die gesamte modellierte Umgebung hinweg bewegen.

Zur besseren Analyse des Einflusses der Beobachtungsrate wurden zusätzlich Simulationen mit einem Zeitraum von 75 Sekunden zwischen zwei Beobachtungen durchgeführt. Hierbei ist zu beachten, dass sich parallel mit der Verkleinerung des Zeitraums die Geschwindigkeit der beobachtbaren Objekte erhöht.

## **6.4 Hintergrundlast**

Innerhalb der Simulationen werden 10 beobachtbare Objekte explizit berücksichtigt. Um das Verhalten des Verfahrens unter Belastung besser beurteilen zu können, werden Beobachtungen weiterer Objekte generiert, welche als Scope die komplette Umgebung zugeordnet bekommen, so dass sich deren Zustandsinformationen über alle Datenbankknoten hinweg ausbreiten. Diese Vorgehensweise wird gewählt, damit es zu einer gleichmäßigen Auslastung des gesamten MANETs kommt. Würden für diese Objekte die Scopes wie für die relevanten Objekte verwaltet werden, so würde sich die zusätzliche Belastung nur in einzelnen Scopes auswirken. Dieser Zustand ist nicht erwünscht, da die Hintergrundlast für alle beteiligten Datenbankknoten gleichmäßig sein soll. Das für die Generierung der Hintergrundlast verwendete Verfahren wird im folgenden vorgestellt.

Jeder Observer generiert innerhalb eines Zeitintervalls [0..150] Sekunden mit einer vorgegebenen Wahrscheinlichkeit zusätzliche Beobachtungen. Die Wahrscheinlichkeit für diese Generierung wird auf 7,8 % festgelegt, was bei 96 Observern im Durchschnitt 7,5 zusätzlicher Beobachtungen pro Beobachtung eines relevanten Objekts entspricht. Um das Verfahren ohne zusätzliche Netzwerkbelastung zu testen, muss die Wahrscheinlichkeit der Generierung zusätzlicher Beobachtungen auf 0 % gesenkt werden. Dann würden ausschließlich Beobachtungen der 10 relevanten beobachtbaren Objekte generiert werden. Mit dieser Vorgehensweise ist es möglich, den Einfluss des zusätzlichen Netzwerkverkehrs auf das Verfahren zu bewerten.

## **6.5 Bewegungsmodell der Datenbankknoten**

Die Bewegungsmuster der Datenbankknoten wurde mit Hilfe des CanuMobiSim-Werkzeugs [SHB+03] generiert. Hierbei werden die Knoten zu Beginn der Simulation zufällig in der simulierten Umgebung platziert. Anschließend wird unter Anwendung des Random-Waypoint-Schemas für jeden Knoten eine neue Zielpositionen festgelegt, zu welchen diese sich dann hinbewegen. Die Geschwindigkeit der Bewegung wird in einem Bereich zwischen 5 km/h und 7 km/h gewählt. Hat ein Knoten eine Zielposition erreicht, wählt er unmittelbar danach eine zufällige neue Position aus und setzt seine Bewegung fort. Somit befindet sich jeder Datenbankknoten über den simulierten Zeitraum hinweg in permanenter Bewegung. Um den Einfluss eines konkreten generierten Bewegungsmusters vernachlässigen zu können, werden Simulationen mit 5 verschiedenen Bewegungsmustern durchgeführt und untereinander verglichen.

Die hierbei beobachteten Verhaltensweisen in Bezug auf die Scopeentwicklung waren nahezu identisch. In [BRS03] wird gesagt, dass sich beim Einsatz eines Random-Waypoint-Schemas

für die Knotenbewegung eine Gleichverteilung der Konten über die gesamte modellierte Umgebung hinweg erreicht wird. Es kann also davon ausgegangen werden, dass sich alle Datenbankknoten in der gesamten modellierten Umgebung bewegen werden. Weiterhin besteht keine Abhängigkeit zwischen einer konkreten Scopeentwicklung und einem bestimmten Bewegungsmuster, solange sichergestellt ist, dass dieses nach dem Random-Waypoint-Schema strukturiert ist und die betrachtete Zeitspanne genügend groß gewählt wird. Unter Berücksichtigung dieser Randbedingungen wurden die detaillierter betrachteten Simulationen mit einem Rand-Waypoint-Bewegungsmuster durchgeführt. Für den Fall, dass eine Scopeentwicklung ungewöhnliche Tendenzen aufweisen sollte, würden weitere Simulationen unter identischen Bedingungen, jedoch mit einem anderen Random-Waypoint-Muster wiederholt werden. Diese Vorgehensweise sichert ab, dass durch Zufall doch ein ungewöhnlicher Einfluss des Bewegungsmusters auf die Scopeentwicklung entsteht.

## **6.6 Anfragemodell**

Das Anfragemodell simuliert einen Anwender, der zu einem bestimmten Objekt Informationen abfragt. Zu diesem Zweck werden von den Datenbankknoten zu bestimmten Ereignissen während des Simulationsdurchlaufs alle im Speicher des DB-Knotens befindlichen Objekte, sowie der für diese Objekte berechnete Scope ausgegeben. Eine Ausgabe dieser Art geschieht, falls ein Datenbankknoten eine neue Objektbeobachtung akzeptiert oder wenn ein Positionswechsel stattfindet. Im ersten Fall kann es sein, dass sich der berechnete Scope für ein Objekt verändert, da die neue Beobachtung eine Positionsinformation beinhalten kann, die in dem vorherigen Scope noch nicht enthalten war. Nach einem Positionswechsel des Datenbankknotens kann es vorkommen, dass eine Objektinformation nicht mehr gültig ist, da der Datenbankknoten den zu diesem Objekt zugehörigen Scope verlassen hat und deshalb dieses Objekt aus seiner lokalen Datenbank entfernt hat. Diese Vorgehensweise beschreibt den Extremfall eines möglichen Anfrageverhaltens, weil zu jedem Zeitpunkt, an dem möglicherweise eine Änderung stattfinden kann, eine Abfrage durchgeführt wird. Ein abgeschwächter Ansatz wäre zum Beispiel ein zeitbasiertes Anfragemodell, das nur nach Ablauf einer gewissen Zeitspanne Objektinformationen abfragt.

Durch diese Ausgaben ist es möglich, eine Bewertung vornehmen zu können, welche Anfragen in Bezug auf ein vorgegebenes Objekt ein Datenbankknoten zu einem bestimmten Zeitpunkt beantworten kann. Wird unter Berücksichtigung dieser Informationen eine Menge von Datenbankknoten betrachtet, kann beurteilt werden, wie gut sich eine Objektbeobachtung innerhalb eines Scopes verbreitet hat. Zusätzlich ist bei dieser Perspektive eine Einschätzung möglich, wie gut sich ein Scope an Objektbeobachtungen angepasst hat.

## **6.7 Einfluss des Scopepuffers**

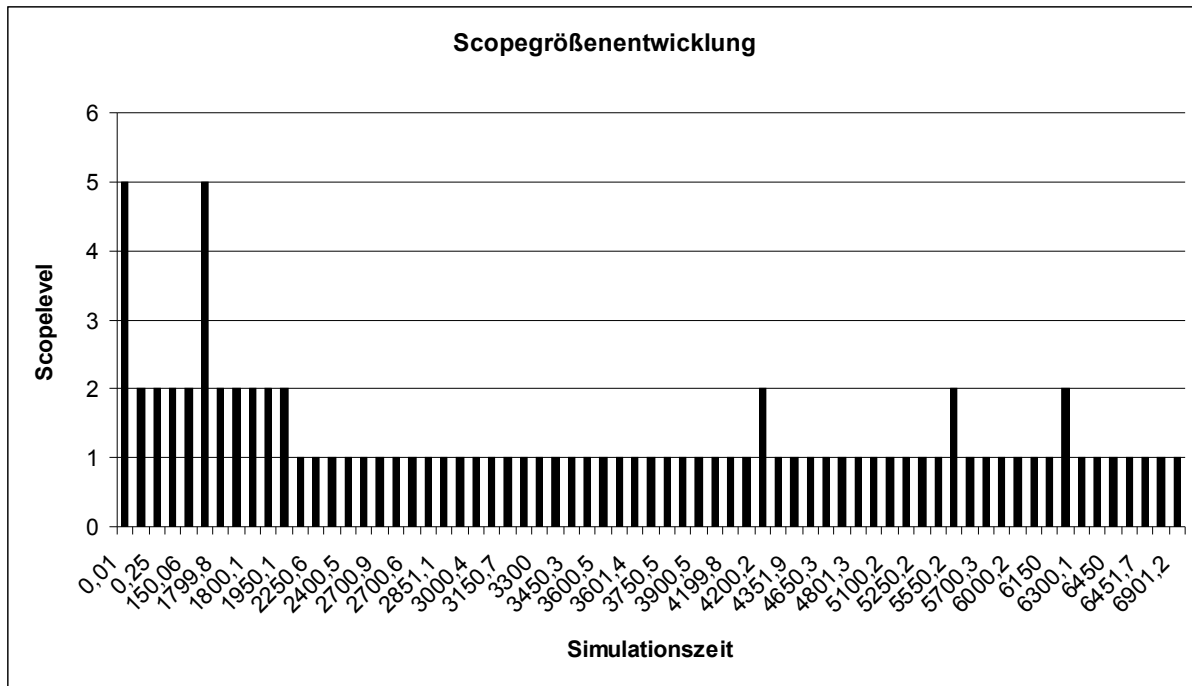
Zur Eingrenzung des Aufwandes wurde für die Durchführung der Simulationen ausschließlich der Ringpuffer in unterschiedlichen Größen berücksichtigt. Die Entscheidung zugunsten des Ringpuffers wurde getroffen, da das Verhalten des Ringpuffers sehr gut abschätzbar ist. Insbesondere sind hier die Unabhängigkeit von der Zeit und Art der empfangenen Scopeinformation zu nennen, welche beim Timeout- und TimeoutGrowingpuffer entscheidende Einflußfaktoren darstellen. Die genannten Einflußfaktoren stellen Variablen dar, welche im Verlauf einer Simulation nur mit einem erhöhten Aufwand zu kontrollieren und nachzuvollziehen sind. Eine Aufgabe für zukünftige Arbeiten könnte sein, entsprechende Auswertungen für die verbleibenden Strategien durchzuführen.

## **6.8 Auswertung**

Das Ziel der Simulationen ist es, eine Einschätzung der einzelnen Puffergrößen in Bezug auf das Verhalten dynamischer Scopes vorzunehmen. Die folgenden Unterkapitel beschreiben die Bedingungen der durchgeführten Auswertungen detaillierter und erläutert die hieraus gewonnenen Erkenntnisse.

### **6.8.1 Bewertung des Scopewachstums**

Zur Bewertung des Scopewachstums wurden die unterschiedlichen Scopegrößen, die bei den Datenbankknoten innerhalb einer Simulation angenommen wurden, genauer betrachtet. Durch die unterschiedlichen Scopeinformationen, welche ein Datenbankknoten im Verlauf einer Simulation berücksichtigt, nimmt die Scopegröße, die dieser für ein Objekt berechnet, jeweils ab oder zu. Auf diese Weise ergibt sich für jeden Datenbankknoten und jedes Objekt, das dieser beobachtet, ein Profil der Scopegrößenentwicklung. Abbildung 11 zeigt ein Beispiel zu einem solchen Profil für einen Datenbankknoten und ein Objekt.



**Abbildung 11: Profil der Scopegrößenentwicklung für einen DB-Knoten und ein Objekt**

Sie stellt die Größen der von einem Datenbankknoten im Verlauf einer Simulation berechneten Scopes dar. Es wird deutlich, dass der Datenbankknoten etwa alle 150 Sekunden einen neuen Wert berechnet, was mit der Observationsrate der Simulation übereinstimmt, aus der die dargestellten Werte entnommen wurden.

Anfangs wird ein Scope der Größe 5 berechnet, was darauf hindeutet, dass zu diesem Zeitpunkt eine Zustandsinformation von einem Observer empfangen wurde. Danach fällt der Scope auf eine Scopegröße mit einem Level von 2 ab. Dieses Verhalten beschreibt den für Scopes typischen Einschwingvorgang. An dieser Stelle wurden vom Datenbankknoten weitere Scopeinformationen empfangen, welche den Scope auf Unterquadranten-Ebene erweitert haben. Es wurde somit Zustandsinformationen von benachbarten Observern berücksichtigt, welche den Scope auf das Gebiet, in dem sich diese befanden, haben anwachsen lassen.

Etwas später bei der Simulationszeit von 1799,8 Sekunden, wird erneut ein Scope mit Level 5 berechnet. Zu diesem Zeitpunkt hatte der Datenbankknoten den Scope für das dargestellte Objekt verlassen und sich anschließend später wieder in diesen hineinbewegt. Dadurch hat er erneut eine Observation zum dargestellten Objekt erhalten und den Scope anfangs auf den Level 5 berechnet. Anschließend wächst der Scope bis zum Level 1. Der Einschwingvorgang für die Scopegröße findet somit erneut statt.

Es ist auffällig, dass relativ oft ein Scopelevel der Größe 1 berechnet wird. Aus diesem Grund wurde das Verhalten der Scopegrößenentwicklung für alle Datenbankknoten einer Simulation genauer untersucht. Hierbei stellte sich heraus, dass ca. 90% der berechneten Scopegrößen innerhalb einer Simulation in einem Scopelevel von 1 resultierten. Die restlichen 10%

verteilen sich auf die verbleibenden Scopegrößen. Die Ursache hierfür ist die hohe Objektivität. Da sich die beobachtbaren Objekte in der gesamten modellierten Umgebung bewegen können, dehnt sich der Scope bis auf diese Fläche aus. Die Struktur des Umgebungsmodells fördert ein schnelles Wachstum von Scopes noch zusätzlich. Empfängt ein Datenbankknoten zwei Observationen aus benachbarten Quadranten der simulierten Umgebung, so berechnet er schon mit zwei gepufferten Scopes einen Level von 1.

Zur Belegung dieser These wurden Simulationen mit eingeschränkter Objektivität durchgeführt. In diesen Simulationen wurden die Beobachtungen und Bewegung eines einzelnen Objekts in einem Quadranten der modellierten Umgebung berücksichtigt. Um vergleichbare Werte mit den anderen durchgeführten Simulationen zu erhalten, wurde die Mobilität der Datenbankknoten ebenfalls auf diesen Bereich beschränkt. Die Anzahl der Datenbankknoten und die die simulierte Zeit blieben identisch.

0	8	16	24	32	40
1	9	17	25	33	41
2	10	18	26	34	42
3	11	19	27	35	43

**Tabelle 3:**  
**1. Quadrant des Umgebungsmodells**

Tabelle 3 stellt den ersten Quadranten der modellierten Umgebung und dessen Einteilung in Unterquadranten dar (siehe Kapitel 6.2). Weiterhin sind die Nummern der in den jeweiligen Zellen platzierten Observer aufgeführt. Die Objektbeobachtung wurde von den grau hinterlegten Observern in der sich über die Simulationszeit wiederholenden Reihenfolge 8,17,26,35,26,17 durchgeführt. Die aus diesen Beobachtungen resultierenden Positionen und die von Datenbankknoten berechenbaren Scopes sind in den Tabellen 4 und 5 aufgeführt. Gleichzeitige Beobachtungen benachbarter Observer wurden nicht berücksichtigt.

<i>Observernummer</i>	<i>Scope</i>
8	scope://grid/Q1/SQ1/S2/a
17	scope://grid/Q1/SQ1/S3/b
26	scope://grid/Q1/SQ3/S1/a
35	scope://grid/Q1/SQ3/S2/b

**Tabelle 4: Positionen der Objektbeobachtungen**

<i>Scope</i>	<i>Level</i>
scope://grid/Q1/SQ1/*	3
scope://grid/Q1/SQ3/*	3
scope://grid/Q1/*	2

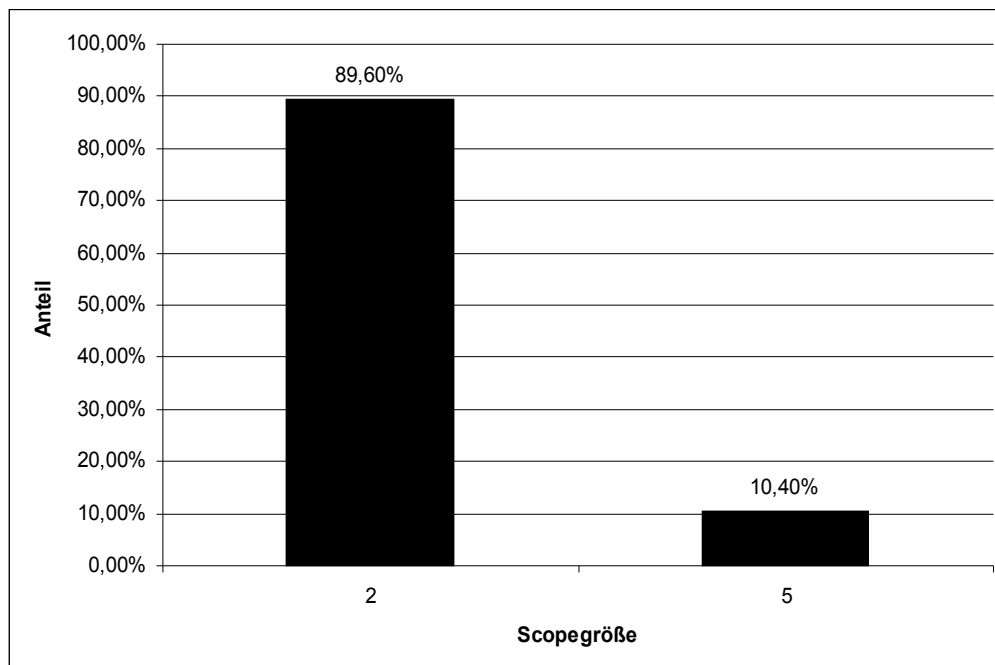
**Tabelle 5**  
**Mögliche Scopes**

Die berechenbaren Scopelevel schwanken somit zwischen 5 und 2. Aufgrund der Tatsache, dass die Objektbewegungen nur in einem Quadranten simuliert werden, darf ein Scope sich niemals über diesen hinweg ausdehnen. Es ist zu beachten, dass der Level 4 niemals berechnet werden kann, da es keine zwei Beobachtungen innerhalb der gleichen Spalte gibt. Das heisst, die einzigen möglichen berechenbaren Scopegrößen sind die Level 2,3 und 5.

Die innerhalb dieser Bedingungen berücksichtigten Ringpuffergrößen sind 3 Scopes und 5 Scopes.

### 6.8.1.1 Auswertung

Die unter den beschriebenen Bedingungen durchgeführten Simulationen resultierten in einer Verteilung der Scopegrößen, wie sie in Abbildung 12 dargestellt ist.



**Abbildung 12: Anteile der Scopegrößen**

Es wird deutlich, dass sich, analog zu den anderen Simulationen, ebenfalls eine Verteilung von 90% auf die maximal mögliche Scopegröße und 10% auf die verbleibenden Größen ergibt. Weiterhin ist erkennbar, dass sich ein Scope niemals in einen größeren Bereich ausdehnt als dies vorgesehen ist.

Es fällt ebenfalls auf, dass nicht alle möglichen Scopegrößen berechnet werden. Ein Scopelevel von 3 wird nie erreicht. Dies liegt daran, dass sich ein Datenbankknoten über einen relativ langen Zeitraum hinweg innerhalb eines Unterquadranten aufhalten müsste, um zwei Observationen aus diesem Bereich derart berücksichtigen zu können, die in einem Scope resultieren, der diesen Bereich abdecken würde. Aufgrund der hohen Datenbankknotenmobilität ist dieser Fall sehr unwahrscheinlich. Weiterhin werden Observationen an den Observern 17 und 26 doppelt so oft durchgeführt wie bei den anderen beiden. Die Kombination dieser beiden Positionsmengen resultiert in einem Scope, der die maximal erreichbare Größe von 2 hat. Dieser Sachverhalt erhöht die Wahrscheinlichkeit, dass dieser Scope öfter berechnet wird, als die anderen möglichen Alternativen.

Die erzielten Werte sind für die Puffergröße 3 und 5 identisch, was an der Anzahl und der Position der gemachten Beobachtungen liegt. Aufgrund der Einschränkung der möglichen Scopes würde jede Puffergröße ein ähnliches Verhalten zeigen.

#### **6.8.1.2 Problemfall: Scope auf Weltebene**

Der Fall, dass ziemlich viele oder im Extremfall sogar alle Datenbankknoten einen Scope auf Weltebene ausdehnen, stellt einen besonderen Fall dar. Bei einer angenommenen Puffergröße von 2 Einträgen, würde eine Reduzierung der Scopeausdehnung erst eintreten, falls ein Datenbankknoten 2 Observationen empfängt, die an Positionen gemacht wurden, deren Zusammenfassung in einer kleineren Scopeausdehnung resultieren würde. Erst dann würden sich aktuellere Objektzustände in diesem kleineren Scope verbreiten. Die verbleibenden Datenbankknoten, welche sich außerhalb des neuen Scopes befinden, behielten die Scopeausdehnung auf Weltebene bei. Dies ist kein wesentlicher Nachteil, da diese Information erst erneut propagiert werden würde, falls die Datenbankknoten neue Objektzustandsinformationen empfangen würden. Hierbei ist zu beachten, dass sich bei Empfang einer Nachricht der von den Datenbankknoten berechnete Scope sofort wieder auf Weltebene ausdehnen würde, da die Zusammenfassung eines beliebigen Scopes mit einem Scope auf Weltebene wieder in einem Scope auf Weltebene resultiert.

Um dies zu verhindern, ist es sinnvoll, einen generellen Timeout zu definieren, der festlegt, wie lange Zustandsinformationen mit zugehörigem Scope von einem Datenbankknoten gehalten werden sollen. Dieser Ansatz wird ebenfalls in [RHB03] vorgestellt. Es wird festgelegt, dass, falls über einen bestimmten Zeitraum keine neue Zustandsinformationen für ein Objekt empfangen wurden, die zugehörigen Daten aus der lokalen Datenbank des

jeweiligen Knotens gelöscht werden. Dann dienen Zustandsinformationen, die dann von einem Datenbankknoten empfangen werden, als neue Grundlage für eine Scopeberechnung.

Die andauernde Ausbreitung auf Weltebene, so wie sie in den vorhergehenden Simulationen erreicht wurde, könnte mit der Wahl einer entsprechenden Einfügestrategie eingeschränkt werden. Würde ein Datenbankknoten Scopes nur dann in den Puffer eintragen, falls diese noch nicht gepuffert sind, so würde eine Verbesserung erreicht. Angenommen ein Datenbankknoten hält in seinem Puffer einen Scope für die Weltebene und würde als nächstes den gleichen Scope wieder empfangen. Dadurch, dass dieser Scope sich schon im Puffer befindet, wird er nicht erneut abgelegt. Bei einer Puffergröße von 2 Einträgen würde dann nach Erhalt von zwei unterschiedlichen Scopes, welche implizit einen kleineren Level als die Weltebene aufweisen, zu einer Ausdehnungsverkleinerung des Scopes führen.

Käme in diesem Fall die Einfügestrategie zum Einsatz, dass nur noch Scopes gepuffert werden, welche noch nicht durch die Kombination der bisher im Puffer enthaltenen Scopes abgedeckt sind, so würde sich ein nachteiliger Effekt einstellen. Es würde darin resultieren, dass der Scope auf Weltebene ausgedehnt bleiben würde, da die Kombination der Puffereinträge den gesamten möglichen Bereich abdeckt. Das heißt, jeder kleinere empfangene Scope würde nicht gepuffert werden, da dieser Bereich schon abgedeckt ist.

### **6.8.2 Szenarien**

Um das Verhalten der Scopeentwicklung unter kontrollierten Bedingungen analysieren zu können, wurden verschiedene Simulationsszenarien erstellt, deren Parameter im Folgenden detailliert beschrieben werden. Für zusätzliche Analysen, deren Bedingungen von den hier genannten abweichen, werden die entsprechenden Angaben in den jeweiligen Abschnitten gesondert aufgeführt. Die Verfahren zur Bewegung von beobachtbaren Objekten und Datenbankknoten, sowie für die Generierung der Hintergrundlast und das Anfragemodell, wurden in den vorangegangenen Kapitel detailliert vorgestellt. Gleiches gilt für das zum Einsatz kommende Umgebungsmodell.

Die Anzahl der Datenbankknoten, welche sich innerhalb der modellierten Umgebung bewegen, wurde auf 375 Knoten festgelegt. Jeder dieser Knoten kann sich uneingeschränkt über die gesamte Umgebung nach dem Random-Waypoint-Muster bewegen.

Die Größe des Ringpuffers wurde zwischen den Werten 3,5,7 und 10 Einträgen variiert. Die Größe des Ringpuffers ändert sich über den Simulationsverlauf hinweg nicht und ist bei jedem Datenbankknoten identisch.

Um den Einfluss der Hintergrundlast auf die Scopeentwicklung bei einer gegebenen Puffergröße feststellen zu können, wurde für jede der genannten Scopegrößen eine Simulation

mit und ohne Hintergrundlast durchgeführt. Das Niveau der Hintergrundlast lag bei 7,8%, was bei 96 Obergrenzen 7,5 zusätzlich erfasste Beobachtungen ausmacht.

Die Bewegungsmuster der beobachteten Objekte basieren auf dem in Kapitel 6.4 vorgestellten Verfahren. Der Simulationszeitraum, in dem Beobachtungen durchgeführt und von Datenbankknoten verarbeitet wurden, betrug für alle Simulationen 7200 Sekunden beziehungsweise 2 Stunden. Gemäß den Angaben in Bezug auf die Bewegungsgeschwindigkeit der Datenbankknoten und der beobachteten Objekte, haben diese in der berücksichtigten Zeitspanne genügend Zeit, sich über einen weiten Bereich der simulierten Umgebung zu bewegen.

Ein weiterer Punkt welcher Berücksichtigung in der Auslegung des beobachteten Zeitraums fand, ist die Generierung einer repräsentativen Anzahl von Beobachtungen. Die Gesamtanzahl der erfassten Beobachtungen ist neben dem Simulationszeitraum ebenfalls von der Observationsrate abhängig. Dieser Wert beschreibt den Zeitraum zwischen zwei unabhängigen Observationsen. Damit gemeint ist die Zeit, die vergeht, wenn ein Observer einen Objektzustand erfasst und dann ein weiterer Observer einen neuen Zustand des gleichen Objekts erfasst. Dieser Wert unterscheidet sich von den gleichzeitigen Beobachtungen benachbarter Observer, die einen identischen Objektzustand wahrnehmen (siehe Kapitel 6.3). Im Verlauf der Simulationen wurden jeweils 10 beobachtbare Objekte berücksichtigt.

Tabelle 6 listet alle Parameter der erstellten Simulationsszenarien zusammenfassend auf.

<i>Parameter</i>	<i>Wertebereich</i>
Anzahl der Datenbankknoten	375
Anzahl beobachtbarer Objekte	10
Hintergrundlast	0% und 7,8%
Observationsrate	1/150, 1/300 Sekunden
Laufzeit	7200 Sekunden
Puffergrößen	3,5,7,10 Einträge

**Tabelle 6:**  
**Simulationsparameter**

Basierend auf diesen Simulationen wurden je 4 Simulationen (für jede Puffergröße eine) sowohl mit als auch ohne Hintergrundlast mit einer Observationsrate von 1/300 Sekunden durchgeführt. Weiterhin wurden wieder mit variierender Hintergrundlast je 4 Simulationen mit einer Observationsrate von 1/150 Sekunden durchgeführt.

Tabelle 7 zeigt die konkreten Simulationskonfigurationen detaillierter:

<i>Puffergröße</i>	<i>Hintergrundlast</i>	<i>Observationsrate</i>
3,5,7,10 Scopes	0,00%	1/300 Sekunden
3,5,7,10 Scopes	7,80%	1/300 Sekunden
3,5,7,10 Scopes	0,00%	1/150 Sekunden
3,7,7,10 Scopes	7,80%	1/150 Sekunden

**Tabelle 7:**  
**Simulationskonfigurationen**

Es wurden somit insgesamt  $4 \cdot 4 = 16$  Simulationen durchgeführt.

### 6.8.3 Vergleich optimaler Datenbankknoten

Für die Bewertung unterschiedlicher Pufferdimensionierungen ist es wichtig herauszufinden, wie sich diese in Bezug auf variierende Randbedingungen, wie zum Beispiel unterschiedliches Beobachtungsverhalten von Observern, voneinander abgrenzen. Für einen Anwender des Verfahren ist es interessant zu wissen, bei welchen Gegebenheiten eine bestimmte Puffergröße optimal eingesetzt werden kann oder wie das Verhalten einer gegebenen Puffergröße von äußeren Bedingungen beeinflusst wird.

Um für diese Anforderungen Anhaltspunkte zu finden, wurde auf der Basis simulierter Observationen das Verhalten sogenannter optimaler Datenbankknoten berechnet. Ein optimaler Datenbankknoten unterscheidet sich von einem simulierten Datenbankknoten dadurch, dass er alle Observationen eines Objekts empfängt und gemäß dem in dieser Arbeit vorgestellten Verfahren verarbeitet. Er befindet sich niemals außerhalb eines zu einem Objekt gehörenden Scopes.

Das optimale Datenbankknoten-Verhalten wurde mit unterschiedlichen Puffergrößen berechnet. Als variierende Bedingung wurden unterschiedliche Beobachtungsmuster betrachtet, bei denen das Zeitintervall zwischen zwei Beobachtungen im Bereich von 75 bis 300 Sekunden variiert wurde. Dieser Wert wird als Beobachtungsrate bezeichnet. Der Zeitraum, in dem Beobachtungen berücksichtigt wurden lag für alle Berechnungen bei 7200 Sekunden.

Die Größe des Ringpuffers wurde zwischen 2 und 10 Einträgen variiert. Eine Größe von einem Scope würde keinen Sinn ergeben, da ein Objektscope sich nicht ausdehnen könnte. Es würde lediglich die Positionsmenge der zuletzt berücksichtigten Beobachtung als Scope verwendet werden. Die Obergrenze von 10 Scopes wurde anfangs willkürlich gewählt und im

Verlauf der Simulationen als genügend groß für eine Obergrenze verifiziert. Tabelle 8 stellt alle variierten Parameter der Berechnungen dar:

<i>Parameter</i>	<i>Verwendete Werte</i>
Puffergröße	2,3,5,7,10 Einträge
Zeit zwischen zwei Beobachtungen	75, 150, 300 Sekunden
Beobachtungszeitraum	7200 Sekunden (2 Stunden)

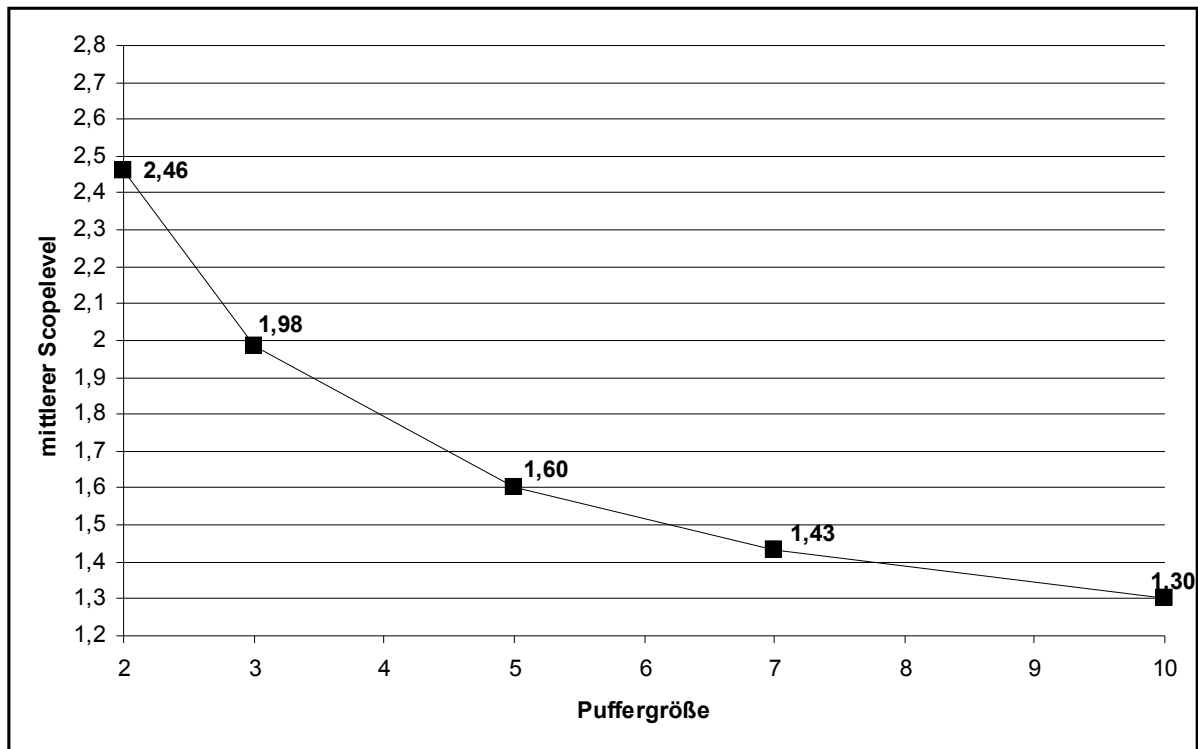
**Tabelle 8:**  
**Berechnungsparameter für optimale Datenbankknoten**

Die Hintergrundlast wurde in den durchgeführten Berechnungen nicht betrachtet, da sie keinen Einfluss auf das Verhalten eines optimalen Datenbankknotens hat. Ein höhere Hintergrundlast würde bei einem simulierten Datenbankknoten dazu führen, dass dieser unter Umständen einzelne Beobachtungen verpasst. Für einen optimalen Datenbankknoten wurde im Gegensatz dazu angenommen, dass dieser alle Observationen berücksichtigt.

### **6.8.3.1 Auswertung**

Das Resultat der durchgeführten Berechnungen ist der Mittelwert des für eine Puffergröße und eine Beobachtungsrate erreichten Scopelevels. Die Auswertung dieser Werte führte zu dem Resultat, dass sich unabhängig von der Beobachtungsrate ein ähnlicher Wert für den mittleren Scopelevel in Bezug auf eine gegebene Puffergröße ergab. Daraus lässt sich folgern, dass die Beobachtungsrate unter den gegebenen Bedingungen keinen nennenswerten Einfluss auf die Scopeentwicklung nimmt.

In allen Szenarien verteilte sich die mittlere Scopegröße derart auf die unterschiedlichen Ringpuffergrößen, wie in Abbildung 13 dargestellt. Der dargestellte Kurvenverlauf zeigt den Durchschnitt der mittleren Scopegröße, der für die oben beschriebenen Parameter berechnet wurde.



**Abbildung 13: Verlauf der Scopegrößenentwicklung bei optimalen Datenbankknoten**

Es wird deutlich, dass je größer ein Puffer dimensioniert wird, der mittlere Scopelevel umso kleiner ist und entsprechend größer die Fläche, welche abgedeckt werden würde. Dieses Verhalten resultiert daher, dass jeweils mehr Scopeinformationen zur Bildung eines neuen Scopes, der alle gepufferten Scopes beinhaltet, berücksichtigt werden.

Ein Scopelevel von 1 würde die gesamte modellierte Umgebung abdecken. Das in Kapitel 6.2 vorgestellte Umgebungsmodell weist einen maximalen Scopelevel von 5 auf. Der optimale Wert der mittleren Scopegröße läge hier zwischen 2 und 3, welche von einem Ringpuffer mit Größe 2 bis 4 Einträgen erreicht wird. Die abgedeckte Fläche würde dann einen Quadranten (Scopelevel 2) beziehungsweise einen Unterquadranten (Scopelevel 3) in der modellierten Umgebung abdecken.

Ein Ringpuffer der Größe 10 erreicht ungefähr einen mittleren Scopelevel der Größe 1, was den schlechtesten Fall darstellt, da hier ein Scope bis auf die größtmögliche Ausdehnung anwachsen würde.

Es wird deutlich, dass nur wenige Scopeinformationen berücksichtigt werden müssen, um eine gute Scopeentwicklung zu erreichen. Übersteigt die Puffergröße den Wert 4, so steigt die mittlere Scopegröße in Wertebereiche die als ungünstig angesehen werden, da der berechnete Scope im Mittel einen Großteil der modellierten Umgebung abdeckt.

#### **6.8.4 Vergleich optimaler mit simulierten Datenbankknoten**

Geprüft wird für jeden Datenbankknoten, inwieweit sich der berechnete Scope für ein Objekt von einem idealen Scope unterscheidet, welcher bei Berücksichtigung aller der für ein Objekt generierten Beobachtungen berechnet werden müsste. Zu diesem Zweck wurde ein idealer Datenbankknoten simuliert, der alle Observationen eines Objekts erfasst und auf Basis dieser Informationen einen Scope berechnet (siehe Kapitel 6.8.3).

Unter der Annahme, dass ein simulierter Datenbankknoten nur eine Teilmenge der gesamten Beobachtungen empfängt, kann es sein, dass dieser einen vom Optimum abweichenden Scope berechnet. Aus diesem Grund wurde die Abweichung im Scopelevel der simulierten Datenbankknoten zu den optimalen Datenbankknoten genauer betrachtet.

In Kapitel 6.3 und 6.5 wurde gezeigt, dass die Auslegung der Beobachtungs- und Bewegungsmuster sicherstellen, dass sich die Objektbeobachtungen und die Bewegung der Datenbankknoten im Verlauf der Zeit gleichmäßig auf die simulierte Umgebung verteilen. Aus diesem Grund kann die mittlere Scopegröße für ein Objekt über alle Datenbankknoten hinweg berechnet werden, da sich alle Datenbankknoten für ein beliebiges beobachtetes Objekt ähnlich in Bezug auf dessen berücksichtigte Beobachtungen und Bewegungen verhalten. Die Konzentration auf das Verhalten einzelner Knoten ist an dieser Stelle wenig sinnvoll, weil jeder Datenbankknoten pro Objekt eine unterschiedliche Anzahl von Beobachtungen berücksichtigt. Es kann weiterhin durchaus vorkommen, dass einzelne Datenbankknoten für ein Objekt keine Beobachtungen empfangen, weil sie sich permanent außerhalb des zu diesem Objekt gehörigen Scopes bewegt haben. Wird stattdessen über alle Datenbankknoten hinweg gemittelt, sind die Ergebnisse repräsentativer Natur.

### 6.8.4.1 Auswertung

Zur Auswertung wurde der mittlere Scopelevel für eine gegebene Puffergröße ermittelt. Dieser Wert setzt sich aus den einzelnen gemittelten Scopeleveln für jede der für eine Puffergröße durchgeführten Simulationen zusammen. Abbildung 14 veranschaulicht die ermittelten Resultate.

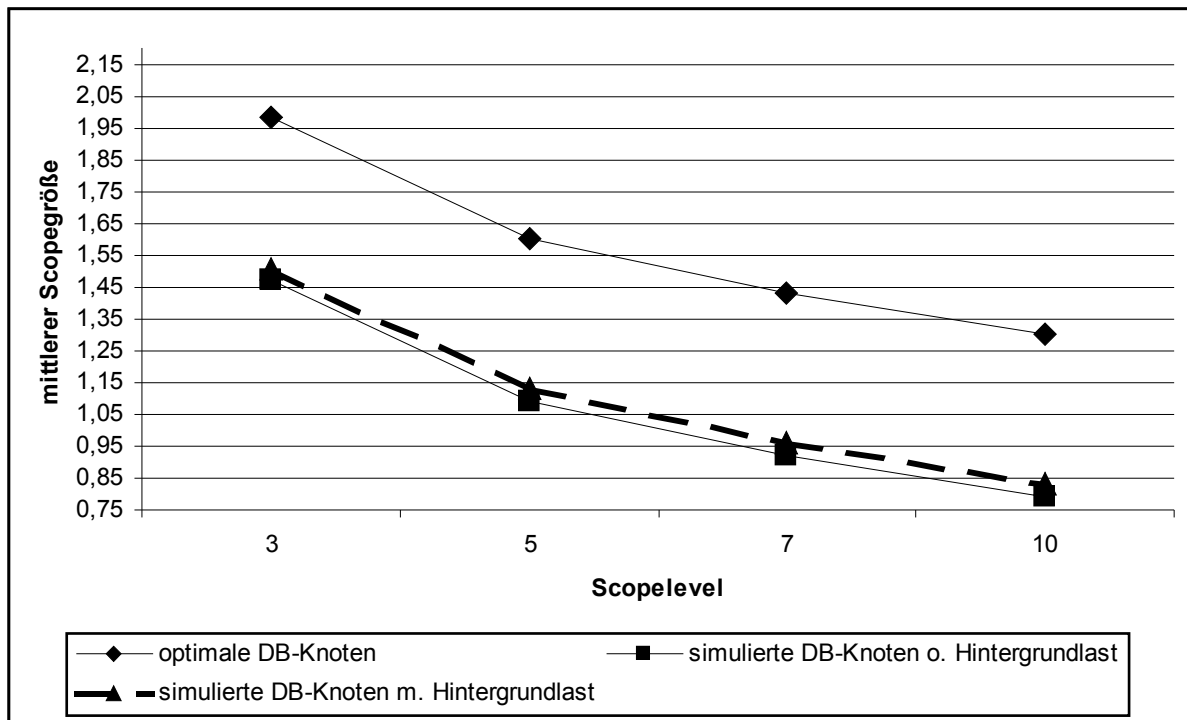


Abbildung 14: Abweichung optimale / simulierte DB-Knoten

Es ist sehr gut erkennbar, dass sich die simulierten Datenbankknoten in Bezug auf die mittlere Scopegröße wie die optimalen Datenbankknoten verhalten. Interessant an dieser Feststellung ist, dass die simulierten Datenbankknoten nur einen Teil der gemachten Beobachtungen berücksichtigen und trotzdem ein dem Idealfall ähnliches Verhalten aufweisen. Hieraus lässt sich schlussfolgern, dass nur ein geringer Anteil an Observationen berücksichtigt werden muss, um ein optimales Verhalten zu erreichen.

Die Abweichung im Scopelevel ohne Hintergrundlast beträgt -0,507 Einheiten in der mittleren Scopegröße, bezogen auf den Level. Wird die Hintergrundlast mitberücksichtigt, so beträgt die Abweichung vom optimalen Scope -0,470 Einheiten im Scopelevel. Für die Berechnung der Abweichung vom Optimum wurde der Mittelwert der Einzelabweichung der jeweiligen Simulationen ermittelt. Aus diesem Grund sind im Diagramm Werte kleiner als 1 aufgeführt. Ein Scope, mit einem Level kleiner als 1, ist in dem in dieser Arbeit verwendeten Umgebungsmodell als real auftretender Wert nicht möglich.

Die Abweichung im Scopelevel nach unten hin bedeutet, dass die simulierten Datenbankknoten einen größeren Scope ermitteln, als dies die optimalen Datenbankknoten tun. Diese Abweichung beruht darauf, dass die simulierten Datenbankknoten nur eine Teilmenge der gesamten gemachten Observations berücksichtigen. Dadurch ist es ihnen nicht möglich, eine mit dem Optimum absolut identische Scopeentwicklung zu erreichen. Hierbei kommt besonders der Fall zum tragen, dass Datenbankknoten Beobachtungen berücksichtigen, welche zeitlich gesehen weit auseinanderliegen. Innerhalb dieses nicht berücksichtigten großen Zeitraums kann sich ein beobachtbares Objekt über eine relativ weite Distanz hinweg bewegt haben. Daraus resultiert, dass ein Datenbankknoten zwei Beobachtungen zusammenfasst, die räumlich betrachtet weit auseinanderliegen, was in einem großen Scope resultiert. Ein weiterer Einflußfaktor ist, dass simulierte Datenbankknoten neben den Scopeinformationen von Observern auch von anderen Datenbankknoten zusammengefasste Scopes berücksichtigen. Dieses Verhalten zeigen optimale Datenbankknoten nicht.

Eine Möglichkeit, ein solches Verhalten zu kompensieren, wäre, eine Richtlinie für die Berechnung neuer Scopes festzulegen, bei der die Anzahl der im Schnitt abweichenden Scopelevel-Einheiten zum Scopelevel für den neuen Scope hinzuaddiert werden würde. Dann wäre die mittlere Entwicklung der Scopegröße nahezu identisch mit den Werten der optimalen Datenbankknoten.

Die Differenz der Abweichung für die Werte mit und ohne Hintergrundlast ist sehr gering, was die Vermutung nahelegt, dass diese unter Berücksichtigung der gegebenen Simulationsparameter keinen nennenswerten Einfluss auf die Scopentwicklung nimmt.

### **6.8.5 Betrachtung versendeter Nachrichten**

Im ursprünglichen CUD-Verfahren wurde plain-flooding eingesetzt, um die Nachrichten mit Objektzustandsinformationen im Netzwerk zu verbreiten. Diese Vorgehensweise resultierte darin, dass jeder Datenbankknoten pro berücksichtigter Beobachtung eine Nachricht verschickt hat. Das Resultat hieraus war, dass unter bestimmten Bedingungen eine hohe Netzwerklast auftrat, die eine zuverlässige Ausbreitung der Zustandsinformationen im MANET negativ beeinflusste. Die Scopeverwaltung wurde entwickelt, um diesem Sachverhalt entgegenzuwirken.

Die Nachrichtenausbreitung ist durch die Einführung von Scopes nur auf eingegrenzte räumliche Bereiche beschränkt. Dies hat zur Folge, dass nicht mehr alle Datenbankknoten alle im Verlauf einer Simulation gemachten Beobachtungen berücksichtigen. Da eine Nachricht von einem Datenbankknoten nur gesendet wird, falls dieser eine Zustandsinformation im Sinne der Scopeverwaltung und den Kriterien des CUD akzeptiert, sollte dieser Wert beim Einsatz von Scopes geringer ausfallen als im ursprünglichen Verfahren.

Durch die so erreichte Reduktion der Anzahl der gesendeten Nachrichten, wird die Netzwerklast in geringerem Maße erhöht, als dies ohne Scopeverwaltung der Fall gewesen ist. Eine leichte Erhöhung findet im Vergleich zum ursprünglichen Verfahren durch die Vergrößerung der zu sendenden Nachrichten statt. Durch das Anfügen der Scopeinformation an den Zustandsdatensatz, welcher eine Zustandsänderung eines Objektes beschreibt, werden pro gesendeter Nachricht mehr Daten übertragen. Dieser zusätzliche Faktor ist bei geeigneter Scopekodierung allerdings konstant und kann aus diesem Grund an dieser Stelle vernachlässigt werden.

Um bewerten zu können, inwieweit der Einsatz von Scopes auf diese Größe Einfluss nimmt wurde die Anzahl der von jedem Datenbankknoten im Schnitt versendeten Nachrichten pro Beobachtung analysiert.

#### **6.8.5.1 Auswertung**

Die Grundlage der Auswertung stellt die im Durchschnitt gesendete Anzahl von Nachrichten dar, welche pro Beobachtung von jedem Datenbankknoten gesendet wird. Es wurde jeweils der Mittelwert aus allen Simulationen mit und ohne Hintergrundlast, sowie mit variierender Observationsrate ermittelt.

Ein Wert von 0,3 bedeutet, dass ein Knoten nur ungefähr ein Drittel der Menge an Nachrichten verschickt, wie er es tun würde, falls er für jede gemachte Beobachtung eine Nachricht verschickte. Je geringer dieser Wert ausfällt umso mehr wird die Belastung des MANETs in positiv beeinflusst, da pro Beobachtung im Schnitt weniger Informationen verteilt werden müssen. Zugleich lässt sich aus den ermittelten Werten die Verbesserung der Situation in Bezug auf das ursprüngliche Verfahren herleiten, da hier als Basis eine Nachricht pro Beobachtung vorgegeben ist. Je kleiner die ermittelten Werte sind umso besser Verhält sich das Verfahren unter Einsatz der Scopeverwaltung gegenüber der Ausgangsbasis.

Ein Faktor, der die berücksichtigten Nachrichten beeinflussen kann, ist die Hintergrundlast. Wird diese mit in Betracht gezogen, so kann es sein, dass einzelne Beobachtungen verloren gehen, da die Netzwerklast auf einem höheren Niveau liegt. Der Grund hierfür ist, dass der Empfangspuffer eines Datenbankknotens voll ist und deshalb nicht mehr alle Nachrichten verarbeiten kann, die er theoretisch empfangen müsste.

Zur Beurteilung dieser Einflussgröße wurden Simulationen mit und ohne Hintergrundlast für jede Puffergröße durchgeführt und miteinander verglichen. Außerdem wurde die Auswertung sowohl für die einfache Observationsrate mit 1/150 Sekunden zwischen zwei Beobachtungen, als auch die doppelte Observationsrate mit dem halbierten Zeitintervall zwischen zwei Beobachtungen durchgeführt.

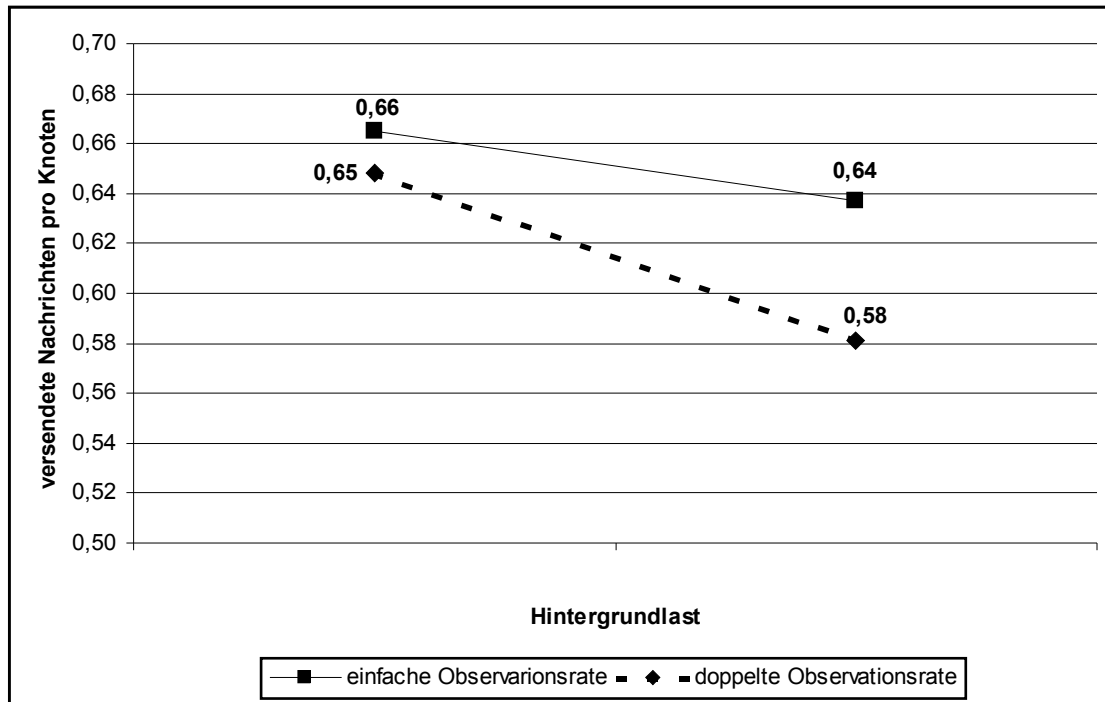


Abbildung 15: Auswertung versendeter Nachrichten pro Knoten

Abbildung 15 zeigt die unter Einsatz der Scopeverwaltung erreichten Werte. Es wird deutlich, dass gegenüber dem ursprünglichen Verfahren eine eindeutige Reduzierung der Anzahl der gesendeten Nachrichten erreicht wird. Im ursprünglichen Verfahren wird eine Nachricht pro Beobachtung gesendet. Im Durchschnitt werden etwa 0,4 Nachrichten pro Knoten weniger übertragen, was einer Verbesserung von 40 % entspricht. Das bedeutet, dass im Schnitt 40% mehr Beobachtungen berücksichtigt werden können, bevor ähnliche Belastungsprobleme im MANET auftreten wie es im ursprünglichen Verfahren der Fall war.

Die erreichten Werte sind in Bezug auf die Observationsrate nahezu identisch. Das bestärkt die schon getroffene Annahme, dass die Observationsrate keinen nennenswerten Einfluss auf die Scopentwicklung nimmt. Diese Annahme gilt an dieser Stelle nur für die innerhalb der jeweiligen Simulationen berücksichtigten Parameter.

Mit zunehmender Hintergrundlast werden jeweils weniger Nachrichten pro Knoten gesendet. Dies beruht auf der oben beschriebenen Tatsache, dass Nachrichten aufgrund der Netzwerklast verloren gehen und so die Verbreitung von Zustandsinformation beschränken. Der Einfluss der Hintergrundlast ist sowohl für die einfache, als auch die doppelte Observationsrate identisch, wobei die Kurve, welche sich auf die doppelte Observationsrate bezieht, etwas stärker abweicht. Dies lässt sich damit erklären, dass der Gesamtanteil der durch die Hintergrundlast verloren gegangenen Nachrichten, bedingt durch die höhere Observationsrate, ebenfalls einen höheren Wert annimmt. An dieser Stelle kommt der

Zusammenhang zum tragen, dass mit steigender Observationsrate mehr Nachrichten verbreitet werden müssen. Je mehr Nachrichten verbreitet werden sollen, umso mehr gehen bei steigender Hintergrundlast verloren. Hieraus folgt, dass der Anteil der verlorenen Nachrichten mit zunehmender Hintergrundlast ansteigt.

### **6.8.6 Betrachtung verpasster Observationen**

Eine weitere Größe, mit der der Einsatz einer Scopeverwaltung mit dem ursprünglichen Verfahren verglichen werden kann, ist der Anteil der verpassten Observationen. Der Optimalfall im CUD ist es, dass jeder Datenbankknoten alle Observationen berücksichtigen kann und somit keine verpasst. Unter Verwendung der Scopeverwaltung ist diese Aussage nur noch teilweise zu erreichen. Befindet sich ein Datenbankknoten außerhalb eines Scopes, so verpasst er automatisch alle Observationen, welche innerhalb dieses gemacht werden. Auf diese Weise verpasst ein Datenbankknoten anteilig mehr Observationen, was aber nicht als negativ bewertet werden kann.

Durch die Erhöhung der Hintergrundlast sollte es zu einem höheren Anteil an verpassten Observationen kommen, da die Nachrichten, welche zur Propagation von Observationen versendet werden, verloren gehen können. Inwieweit sich die Hintergrundlast auf den Anteil der verpassten Observationen unter Einsatz der Scopeverwaltung auswirkt, wurde aus diesem Grunde genauer betrachtet.

Ein weiterer variabler Faktor, der in dieser Betrachtung berücksichtigt wurde, ist die Observationsrate. Es wurde das ursprüngliche Verfahren mit einfacher Observationsrate mit dem Einsatz der Scopeverwaltung verglichen. Weiterhin wurde eine Simulationsreihe mit doppelter Observationsrate (75 Sekunden Abstand zwischen zwei Observationen) durchgeführt.

Die genaue Analyse erfolgte derart, dass der mittlere Anteil aller verpassten Observationen für alle Datenbankknoten über den gesamten Simulationsverlauf hinweg ermittelt wurde. Hierbei wurden die Werte für alle Puffergrößen zusammengefasst. Auf diese Weise wurden die Werte mit und ohne Hintergrundlast ermittelt.

### 6.8.6.1 Auswertung

Abbildung 16 veranschaulicht die erzielten Ergebnisse.

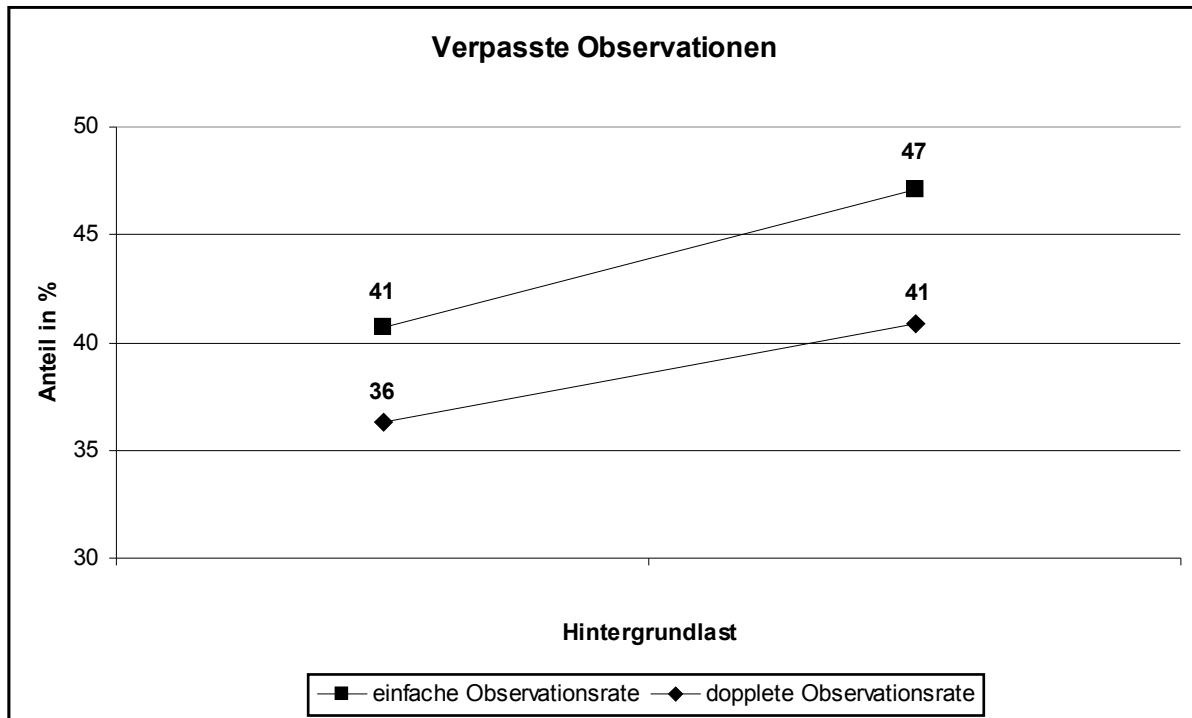


Abbildung 16: Anteil verpasster Observationen

Werden die Werte des ursprünglichen CUD-Verfahrens, unter Berücksichtigung vergleichbarer Szenarien, mit denen des Verfahrens mit Scopeverwaltung, jeweils ohne Hintergrundlast verglichen, so wird eine Verschiebung um 10 % nach oben hin deutlich. Der Anteil verpasster Nachrichten lag beim ursprünglichen Verfahren bei etwa 30 %. Das bedeutet, dass die Scopeverwaltung bewirkt, dass 10 % mehr Nachrichten verpasst werden. Dieser Faktor wird durch die Knoten erreicht, die sich außerhalb eines Scopes für ein Objekt befinden und aus diesem Grund dessen Observations verpassen. Diese Aussage kann nur für die Simulationen mit einfacher Simulationsrate gemacht werden, da für die doppelte Observationsrate keine vergleichbaren Werte zum ursprünglichen Verfahren zur Verfügung stehen.

Der Wert der anteilig verpassten Observations liegt bei doppelter Observationsrate etwas niedriger, was daran liegt, dass Datenbankknoten mehr unterschiedliche Scopeinformationen empfangen und so einen Scope schneller wachsen lassen. In einem größeren Scope befinden sich anteilig mehr Datenbankknoten, welche die Zustandsinformationen empfangen. Dadurch ist der Gesamtanteil verpasster Observations etwas niedriger.

Die Gesamtauswirkung bei steigender Hintergrundlast ist bei einfacher und doppelter Observationsrate identisch, was in Abbildung 16 daran erkennbar ist, dass die Kurven nahezu parallel verlaufen. Es werden mit steigender Hintergrundlast anteilig mehr Observationen verpasst. Dies resultiert, wie schon im vorherigen Abschnitt angedeutet daraus, dass Nachrichten aufgrund der höheren Netzwerkbelastung verloren gehen.

## 7 Verwandte Arbeiten

In diesem Kapitel werden Arbeiten behandelt, die thematisch mit dieser Arbeit in Verbindung stehen. Es werden Ähnlichkeiten verglichen und Unterschiede dargestellt.

### 7.1 DREAM

DREAM [BCS+98] steht als Abkürzung für Distance Routing Effect Algorithm for Mobility und beschreibt ein neuartiges lokationsbasiertes Routing-Protokoll für MANETs. Beim Routing von Daten zwischen Knoten werden die relativen Positionen von Knoten zueinander berücksichtigt. Damit dieses Vorgehen möglich wird, tauschen die Knoten eines MANETs permanent ihre eigenen Positionen untereinander aus, welche in einer Lokationstabelle auf jedem Knoten gespeichert werden. Aus diesen Positionsinformationen kann ein Knoten die Entfernung anderer Knoten in Bezug auf seine eigene Position bestimmen. Beim Aussenden der eigenen Positionierungsinformation wird der sogenannte Distanzeffekt berücksichtigt. Dieser beschreibt das Phänomen, dass falls zwei Knoten sich mit der gleichen Geschwindigkeit bewegen, ein näherer Knoten sich scheinbar schneller an einem vorbei bewegt, als ein entfernterer. Aus diesem Grund müssen entferntere Knoten weniger oft über eigene Positionsänderungen benachrichtigt werden, als diejenigen, die sich in der näheren Umgebung befinden. Erreicht wird dies mit Zeitangaben, welche die Lebensdauer einer Positionsinformation beschreiben, in der diese weiterpropagiert werden. Soll nun ein Datenpaket an einen bestimmten Knoten gesendet werden, so wird dieses in die Richtung geleitet, in der sich der Zielknoten befindet. Hierzu wird eine Auswahl der nächstliegenden Knoten gesendet, welche sich in dieser Richtung befinden.

DREAM unterscheidet sich von dem in dieser Arbeit beschriebenen Verfahren dadurch, dass es auf einem geographischen Umgebungsmodell basiert. Weiterhin wird nicht mit Broadcasts für die eigentliche Informationsausbreitung gearbeitet, sondern mit einer direkten Auswahl von Knoten, die eine Information empfangen, beziehungsweise weiterleiten sollen. Außerdem müssen Zustandsinformationen über die anderen Knoten im MANET gehalten werden, damit dieses Verfahren eingesetzt werden kann. Diese Anforderung ist für eine Scopeveraltung im CUD nicht notwendig.

Um die Information in eine bestimmte Richtung zu leiten, wird erreicht, dass sich die Information, durch die gezielte Auswahl von Knoten, in einem definierten Bereich auf ihren Zielknoten hin ausbreitet. Dieses Vorgehen kann als Bilden eines Scopes angesehen werden. Die Informationen sind nur in einem Bereich gültig, der zwischen dem Sender und dem Empfänger eines Datenpakets liegt. Dieser Scope wird durch die gezielte Auswahl von Knoten realisiert, die sich innerhalb eines solchen Bereiches befinden.

Der Einsatz von Zeitangaben für die Propagierung der Positionsinformationen bewirkt ebenfalls eine Bildung von Scopes. Durch die unterschiedliche Lebensdauer werden unterschiedlich große räumliche Bereiche definiert, in denen die Positionsinformationen einzelner Knoten gültig sein sollen. Der Ausdehnungsbereich eines Scopes wird durch die Lebensdauer einer Positionsinformation definiert. Je länger eine Positionsangabe weitergeleitet werden soll, umso größer ist der hieraus resultierende Scope. In dem in dieser Arbeit vorgestellten Verfahren wird ebenfalls ein dem Distanzeffekt ähnliches Verhalten erreicht. Durch die dynamische Scopeanpassung wird im Fall einer Vergrößerung eine verzögerte Benachrichtigung von Knoten erreicht, die sich in einer weiteren räumlichen Entfernung zu einer Objektbeobachtung befinden.

## **7.2 GeoCast**

In [KOVA02] werden verschiedene floodingbasierte Geocast-Verfahren für den Einsatz in mobilen ad hoc Netzwerken vorgestellt. GeoCast ist ein Verfahren ähnlich dem Multicast, bei dem Nachrichtenpakete an mehrere zu einer Gruppe gehörende Knoten gesendet werden. Speziell für GeoCast-Verfahren gilt, dass sich ein Knoten in einer definierten räumlichen Region befinden muss, damit dieser als Empfänger in Frage kommt. Möchte ein Sender ein Datenpaket an Knoten in einer bestimmten Region schicken, so stellen GeoCast-Verfahren das Routing in diese Region und auch die Verteilung innerhalb einer solchen sicher. Dieser Punkt unterscheidet GeoCast von dem in dieser Arbeit vorgestellten Verfahren, bei dem kein Routing stattfindet. Es wäre mit dem Fall des Geocastings vergleichbar, falls sich sowohl der Sender als auch alle Empfänger in der Zielregion für Datenpakete befinden. Dies entspräche zum Beispiel einem Observer, der den Datenbankknoten in seiner Umgebung eine Objektbeobachtung mitteilt. Da allerdings auch beim GeoCast definierte räumliche Bereiche eine Rolle spielen, in denen Daten verbreitet werden sollen, ist die Verwandtschaft zu einer Scopeverwaltung zu erkennen. So gleichen sich zum Beispiel beide Verfahren darin, dass Knoten Informationen zugestellt bekommen, wenn sie sich innerhalb eines bestimmten räumlichen Bereiches aufhalten und bei dessen Verlassen keine weiteren Nachrichten mehr erhalten, die für die Region bestimmt sind. Der Weg der Nachrichten von einem Sendeknoten zum Zielbereich hin, wird durch eine sogenannte forwarding-zone beschrieben, die den räumlichen Bereich beschreibt, durch den Nachrichten geroutet werden müssen, um die Zielregion zu erreichen. Diese forwarding-zone stellt somit auch eine Art Scope dar. Die Informationsausbreitung innerhalb der forwarding-zone und dem Zielbereich basiert sowohl bei den in [KOVA02] beschriebenen Verfahren, als auch bei dem in dieser Arbeit vorgestellten Verfahren auf einem Flooding-Mechanismus. Abweichend wird bei den vorgestellten GeoCast-Verfahren angenommen, dass mit exakten Positionsinformationen gearbeitet wird, was bedeutet, dass diese auf einem geometrischen Umgebungsmodell aufbauen.

## **8 Weiterführende Arbeiten**

Die folgenden Unterabschnitte stellen mögliche weiterführende Arbeiten auf. Es werden mögliche Ergänzungen zur Scopeverwaltung skizziert, als auch weiterführende Untersuchungen vorgeschlagen.

### ***8.1 Lattice als Umgebungsmodell***

In Kapitel 2.4.2 wurde der Lattice als ein mögliches Umgebungsmodell für den Einsatz von Scopes vorgestellt. Ein mögliche Aufgabe für weiterführende Arbeiten wäre, die Scopeverwaltung auf dieses Umgebungsmodell anzupassen. Dies würde in einer flexibleren Handhabung von Scopes resultieren. Die Anpassungen würden Problemlösungen in den Operationen auf Scopes erfordern, insbesondere sind hier die Ausdehnungsänderung und Positionsänderung von Scopes innerhalb des neuen Umgebungsmodells zu nennen. Besonders hervorzuheben sei hier eine Strategie zur Findung einer geeigneten zusammenfassenden Ober- oder Teilmenge.

### ***8.2 Scopeanpassung aufgrund von Netzwerklast***

Ein weiteres interessantes Gebiet wäre es, Mechanismen zu entwickeln, die eine Scopeverwaltung basierend auf der aktuellen Belastungssituation eines mobilen ad hoc Netzwerkes durchführen. Vorstellbar wäre, dass in einem bestimmten Teil des Netzwerkes eine hohe Belastung erkannt wird und als Reaktion zu Eindämmung dieser Belastung für Objektinformationen Scopes vergeben werden. Hierfür sind Indikatoren notwendig, welche die Belastung eines Netzwerkes derartig wiedergeben, dass aufgrund der gelieferten Werte eine Entscheidung für den Einsatz einer Scopeverwaltung gefunden wird.

### ***8.3 Analyse weiterer Pufferstrategien***

Diese Arbeit konzentrierte sich in der Analyse auf den Ringpuffer. Zukünftige Arbeiten könnten sich mit der Analyse der anderen in dieser Arbeit vorgestellten Pufferstrategien dem Timeout- und TimeoutGrowing-Puffer, befassen. Dabei gilt es, weitere Kriterien für die Begrenzung eines Scopewachstums zu finden.

## **8.4 Weitere Simulationskonfigurationen**

Um das Verhalten der Scopeentwicklung für unterschiedliche Umgebungsbedingungen besser einschätzen zu können, wäre es sinnvoll, Simulationen mit mehr Variationen der Parameter durchzuführen.

So kann zum Beispiel die Hintergrundlast oder die Beobachtungsrate in einem weiteren Spektrum variiert werden.

Interessant ist es, auch das Verhalten innerhalb anderer Umgebungen als einer planen Fläche zu betrachten und koordinierte Bewegungen von Objekten und Datenbankknoten miteinzubeziehen. Hier wären graphbasierte Bewegungsmodelle ein interessanter Ansatzpunkt. Diese Analysen würden Aussagen über die Scopeentwicklung unter Gesichtspunkten ermöglichen, welche sich an realen Szenarien orientieren. Untersuchungen, die alternative Hierarchievarianten auf der in dieser Arbeit vorgestellten Umgebung verwenden, sollten ebenfalls interessante Ergebnisse in Bezug auf die Abhängigkeit der Scopeentwicklung zum Umgebungsmodell liefern. Zusätzlich könnten auch andere Hierarchieformen, wie zum Beispiel der Lattice of Locations [KEG93], genauer betrachtet werden.

Ein weiterer Ansatzpunkt für Variationen wären mehr Simulationskonfigurationen mit unterschiedlichen Objekt- und Knotengeschwindigkeiten.

## **8.5 Künstliche Scopeausdehnung durch Datenbankknoten**

Eine mögliche Erweiterung der Scopeverwaltung könnte es sein, dass Datenbankknoten die Möglichkeit bekommen, einen Scope soweit auszudehnen, dass sie eine gewünschte Information zugestellt bekommen. Ein möglicher Ansatz hierfür wäre, dass ein Datenbankknoten eine Art neutralen Objektzustand verbreitet. Dieser Zustandsinformation könnte eine Objekt-ID beinhalten, die das Objekt beschreibt, von dem Informationen empfangen werden sollen. Weiterhin wäre dieser Zustandsinformation ein Scope zugeordnet, der so groß ist, dass er einen möglichen Beobachtungsort sowie die eigene Position eines Datenbankknotens abdeckt. Diese neutrale Zustandsinformation veranlasst, dass Datenbankknoten, welche diese empfangen, für das vorgegebene Objekt den Scope in den Puffer schreiben, der sowohl die mögliche Beobachungsposition, als auch die Position des interessierten Datenbankknotens berücksichtigt. Für dieses Verfahren sind einige Erweiterungen nötig und sein Einsatz wäre auch mit einigen Seiteneffekten verbunden, die genauer betrachtet werden müssten.

## **8.6 Problemfall logische Partitionen**

Ein weiterer interessanter Punkt, welcher näher in Bezug auf den Einsatz von Scopes betrachtet werden kann, sind sogenannte logische Partitionen. Dieses Phänomen beschreibt die Situation, dass ein Scope eine Umgebung abdeckt, welche einen Bereich enthält in dem, zum Beispiel bedingt durch ein Hindernis, keine Nachrichten übertragen werden können. Eine Lösung für ein solches Problem könnte sein, dass Datenbankknoten, welche außerhalb dieses Scopes liegen, eine Art Brücke zwischen den Knoten innerhalb des Scopes schaffen, der durch das Hindernis logisch partitioniert wurde. Ein erster Ansatz hierfür wurde in Kapitel 5.4.2.1 geliefert. Hier wurde erläutert, dass es mit einer angepassten Annahmestrategie eines Datenbankknoten möglich ist, eine einfache Brücke für einen solchen Fall zu etablieren. Einfach bedeutet in diesem Zusammenhang, dass die Brücke nur über einen Hop hinweg reicht. Zur Überbrückung größerer logischer Partitionen wären unter Umständen mehrere Hops notwendig, um eine logische Partition zu umgehen.

Die Suche nach entsprechenden Lösungsmöglichkeiten für diese Problematik stellt eine interessante Erweiterung für die Scopeverwaltung dar. Hiermit ließen sich Probleme beseitigen, die in realen Anwendungen auftreten können, die mit dem in dieser Arbeit vorgestellten Ansatz noch nicht lösbar sind.

## 9 Zusammenfassung

Dieses Kapitel fasst abschließend alle wichtigen Fragestellungen, sowie die daraus gewonnenen Erkenntnisse zusammen.

Die Ausgangsbasis zur Entwicklung einer Scopeverwaltung wurde vom „Consistent-Update-Diffusion-Verfahren“ (CUD) geliefert. Mit Hilfe dieses Verfahrens ist es möglich, eine verteilte Datenbank innerhalb eines mobilen ad hoc Netzwerks (MANET) zu erstellen und den Datenbestand unter dem Bezugspunkt der Realzeit in einem konsistenten Zustand zu erhalten. Der Inhalt dieser verteilten Datenbank, welche durch die sogenannten Datenbankknoten gebildet wird, stellt die Zustände verschiedener Objekte dar, die innerhalb einer Anwendung von sogenannten Observern überwacht und entsprechend verbreitet werden. Um dies zu erreichen, tauschen die beteiligten Knoten Nachrichten untereinander aus, welche generiert werden, falls neue Objektzustände beobachtet wurden oder sich der Datenbestand eines Knotens verändert hat. Das Problem bei dieser Vorgehensweise ist, dass mit steigender Anzahl an Änderungen, sich analog dazu auch die Menge der zu übertragenden Nachrichten netzwerkweit erhöht. Die Nachrichtenverteilung innerhalb des MANET geschieht mit einem flooding-basierten Ansatz. Bei Verwendung dieser Art der Nachrichtenverteilung besteht eine Abhängigkeit zur Netzwerklast. Wird diese zu sehr erhöht, breiten sich Nachrichten nur noch sehr langsam unter den Knoten aus oder gehen im schlimmsten Fall ganz verloren.

Um dieser Problematik vorzubeugen, soll die Lokalität von Informationen ausgenutzt werden. Lokalität von Informationen bedeutet, dass eine Information in der Regel in der Nähe des Ortes benötigt wird, an dem sie generiert wurde. Unter Ausnutzung dieser Annahme wurde ein Scope definiert, der einen begrenzten räumlichen Bereich beschreibt. Übertragen auf das CUD-Verfahren bedeutet dies, dass Zustandsinformationen über beobachtete Objekte nun nicht mehr im gesamten MANET verbreitet werden, sondern nur noch in einem eingeschränkten räumlichen Bereich um den Ort herum, an dem diese generiert wurden.

Durch die Bewegung der beobachtbaren Objekte kommt es vor, dass diese an unterschiedlichen Positionen im Verlauf einer Anwendung beobachtet werden. Ein Scope sollte sich dieser Bewegung anpassen. Hieraus resultieren zwei für Scopes besondere Eigenschaften: Erstens kann ein Scope seine Position wechseln, und zweitens kann sich die Ausdehnung eines Scopes beliebig verändern. Mit Hilfe dieser Eigenschaften ist es möglich, einen Scope an die Objektbewegung anzupassen. Das bedeutet, dass die Zustandsinformationen zu einem Objekt sich in dem Bereich ausbreiten, in dem sich ein beobachtbares Objekt bewegt. Es wird somit nicht nur die einzelne Position einer Objektbeobachtung berücksichtigt, sondern zusätzlich der Raum, in welchem mehrere Beobachtung im Verlauf einer Zeitspanne gemacht wurden.

Um eine derartige Scopeverwaltung zu ermöglichen, spielen die Positionen der Objektbeobachtungen, welche als Informationsquelle dienen eine Rolle. Weiterhin muss jedem Knoten im MANET seine Position bekannt sein, damit dieser entscheiden kann, ob er sich innerhalb oder außerhalb eines Scopes befindet und dementsprechend Objektzustandsinformationen berücksichtigen und weiterleiten soll oder nicht. Es wird somit angenommen, dass jedem beteiligten Knoten, also sowohl Observern, als auch Datenbankknoten, seine eigene Position bekannt ist. Damit eine Anwendung mit Positionsinformationen arbeiten kann, wird ein Modell der Umgebung benötigt, in der diese eingesetzt werden soll.

Für die Auswahl eines solchen Umgebungsmodells, wurden drei verschiedene Ansätze auf ihre Tauglichkeit für den Einsatz einer Scopeverwaltung auf mobilen, ressourcenbeschränkten Geräten hin überprüft. Der Einsatz eines geometrischen oder eines hybriden Umgebungsmodells ist aufgrund des relativ hohen Modellierungsaufwandes und des notwendigen Speicherbedarfs nicht gut geeignet. Zusätzlich ist in diesen beiden Ansätzen ein großer Rechenaufwand notwendig um typische, für den Einsatz von Scopes notwendige, Berechnungen innerhalb dieser durchzuführen. Dies spricht eindeutig gegen den Einsatz eines dieser Ansätze auf mobilen Geräten, welche oft nur eine relativ geringe Rechenleistung und auch nur einen begrenzten Energievorrat aufweisen. Die Wahl fiel somit auf ein mengenbasiertes Umgebungsmodell, welches die gestellten Anforderungen gut erfüllt. Somit halten sich der Modellierungsaufwand und der Speicherplatz in engen Grenzen. Ebenso sind Berechnungen zur Verwaltung von Scopes im Verhältnis zu den anderen Ansätzen relativ aufwandsarm realisierbar.

Die grundlegenden Bausteine des mengenbasierten Umgebungsmodells sind Positionsmengen. Zur Abbildung der dynamischen Scopeeigenschaften, der Scopebewegung und der Ausdehnungsänderung ist es notwendig, eine Hierarchie auf den Positionsmengen zu definieren. Das bedeutet, dass eine Positionsmenge sich aus beliebigen Teilmengen zusammensetzen kann. Eine Objektbewegung wird nun durch Auswahl entsprechender Positionsmengen abgebildet. Die Ausdehnung eines Scopes in eine beliebige Richtung wird durch Auswahl einer Positionsmenge modelliert, die alle Teilmengen enthält, über die sich ein Scope ausdehnen soll.

Für mengenbasierte Umgebungsmodelle wurden zwei Hierarchieformen auf die Einsatzmöglichkeit einer Scopeverwaltung auf mobilen Geräten näher betrachtet. Der latticebasierte Ansatz bietet sehr flexible Möglichkeiten für die Verwaltung von Scopes, weist allerdings einen relativ hohen Modellierungsaufwand auf und benötigt weiterhin mehr Speicherplatz als die Folgenden vorgestellten Alternative. Aus diesen Gründen fiel die Wahl auf den Space-Tree, welcher die Positionsmengen in einer baumartigen Struktur anordnet. Das

für die Scopeverwaltung zugrunde liegende Umgebungsmodell ist somit ein mengenbasiertes Modell, welches als Hierarchieform den Space-Tree verwendet.

Um die Objektbewegung zur Bildung eines Scopes berücksichtigen zu können, ist es notwendig, eine bestimmte Anzahl an Positionen von Objektbeobachtungen zwischenzuspeichern. Die Zusammenfassung dieser Positionsinformation würde in einem Scope resultieren, der alle Positionen abdeckt in denen ein Objekt beobachtet wurde. Problematisch an dieser Vorgehensweise ist, dass sich im Fall vieler Positionsinformationen ein Scope zu weit ausdehnt. Der Extremfall wäre, dass ein Scope die gesamte modellierte Umgebung abdeckt. Weiterhin ist es sinnvoll, einen Scope nur anhand der Positionen zu bilden, an denen ein Objekt zuletzt beobachtet wurde. Auf diese Weise breitet sich ein Scope lediglich in dem Bereich aus, in dem ein Objekt zuletzt gesehen wurde. Dies gewährleistet eine gute Anpassung an die Objektmobilität.

Zur Beschränkung der Anzahl der zwischenzuspeichernden Scopes wurden drei Ansätze für mögliche Pufferstrategien vorgestellt. Beim Timeout-Puffer werden Scopeinformationen nur über einen bestimmten Zeitraum hinweg abgelegt. Läuft der definierte Timeout ab, so werden die entsprechenden Scopeinformationen aus dem Puffer gelöscht und nicht mehr für weitere Zusammenfassungen von Scopes berücksichtigt. Der TimeoutGrowingpuffer verwendet ebenfalls einen zeitbasierten Ansatz zur Begrenzung der Anzahl der zu berücksichtigenden Scopes. Hierbei wird ein Zeitraum definiert, innerhalb dessen ein Scope wachsen soll. Nach Ablauf dieses Zeitraums wird die nächste eintreffende Scopeinformation als neue Ausgangsbasis für ein Scopewachstum gewählt. Der sogenannte Ringpuffer beschränkt die Anzahl der Scopes, die in ihm abgelegt werden können. Durch das Überschreiben der jeweils am längsten im Puffer enthaltenen Information wird eine Anpassung an die aktuelle Situation der Objektmobilität erreicht.

Zur Integration der Scopeverwaltung in eine Anwendung wurde ein Schichtenmodell vorgestellt. Hierbei wurde darauf geachtet, eine saubere Trennung zwischen den einzelnen beteiligten Komponenten zu erreichen. Auf unterster Ebene befindet sich die für die Datenausbreitung zuständige Schicht. Darauf folgt die Schicht, welche entscheidet, ob Informationen aufgrund ihrer Scopeinformation angenommen oder abgelehnt werden sollen. Auf dieser Scopeverwaltungsschicht sitzt die Schicht, welche die Datenkonsistenz im Sinne des CUD-Verfahrens gewährleistet. Es wurde gezeigt, dass es möglich ist die Scopeverwaltung sauber vom Rest abzutrennen. Dies gestattet es ein, scopeorientiertes Vorgehen in beliebige Anwendungen zu integrieren.

Zur genaueren Auswertung der Auswirkungen einer Scopeverwaltung wurden Simulationen mit Hilfe des Netzwerksimulators ns2 durchgeführt. Hierfür wurde als Umgebung eine 6 km<sup>2</sup> große Fläche angenommen, welche in Quadranten unterteilt wurde, auf denen eine Hierarchie im Sinne des Space-Tree modelliert wurde. Zur detaillierten Untersuchung wurden zwei

Klassen von Simulationen durchgeführt. Zum einen wurde das Verhalten in einem kleinen Gebiet der simulierten Umgebung mit gezielten Objektbeobachtungen simuliert, und zum anderen wurden Simulationen durchgeführt, in denen sich die beobachtbaren Objekte über die gesamte Umgebung hinweg bewegten. Untersucht wurden das Verhalten bei Einsatz eines Ringpuffers mit unterschiedlichen Größen. Als Einflussfaktoren wurden sowohl die Hintergrundlast als auch die Beobachtungsrate variiert.

Es stellte sich heraus, dass die für die Simulationen gewählte Strukturierung der Umgebung sich negativ auf die Scopeentwicklung auswirkte. Es wurde beobachtet, dass 90% der Knoten Scopes auf maximaler Ausdehnung berechneten. Es wurde deutlich, dass die Struktur des Umgebungsmodells in Kombination mit der Bewegung der beobachtbaren Objekte und der Datenbankknoten für dieses Verhalten verantwortlich sind.

Zur Untersuchung inwieweit eine Scopeausdehnung von einer Puffergröße abhängt, wurden sogenannte optimale Datenbankknoten berechnet. Diese Knoten zeichnen sich dadurch aus, dass sie alle Beobachtungen eines Objekts berücksichtigen und sich somit niemals außerhalb des zu einem Objekt gehörenden Scopes befinden. Weiterhin berücksichtigen sie nur Informationen von Observern und nicht von anderen Datenbankknoten. Die Ergebnisse dieser Untersuchung zeigten, dass es einen klaren, nicht linearen Zusammenhang zwischen der Puffergröße und der daraus resultierenden durchschnittlichen Puffergröße gibt. Es zeigte sich, dass bei größer dimensionierten Puffer, auch der von diesem im Verlauf einer Anwendung berechnete Scope wachsen würde. Dies liegt daran, dass ein größerer Puffer mehr Scopeinformationen zur Berechnung eines zusammenfassenden Scopes berücksichtigt.

Ein Vergleich von optimalen Datenbankknoten mit simulierten Datenbankknoten zeigte, dass diese sich in der Scopeentwicklung nur geringfügig unterscheiden. Das heisst, simulierte Datenbankknoten zeigten ein dem Optimalfall ähnliches Verhalten. Die resultierende Abweichung ist auf die Tatsache zurückzuführen, dass simulierte Datenbankknoten nicht wie der optimale Datenbankknoten alle Beobachtungen eines Objekts berücksichtigen sondern nur jeweils eine Teilmenge. Weiterhin berücksichtigen simulierte Datenbankknoten neben Observer-Informationen auch Scopeinformationen von Datenbankknoten, welche unter Umständen schon Scopes mit einer gewissen Ausdehnung berechnet haben.

Ziel des Einsatzes der Scopeverwaltung war es, die Netzwerklast, die beim Einsatz des CUD-Verfahren auftritt, zu begrenzen. Inwieweit dies erreicht wurde, zeigte die Betrachtung der gesendeten Nachrichten. Hierbei zeigte sich eine deutliche Verbesserung gegenüber dem ursprünglichen Verfahren, das ohne Scopeverwaltung arbeitete.

Ein weiterer Wert, welcher mit dem ursprünglichen Verfahren verglichen wurde, war der Anteil der verpassten Beobachtungen. Dieser sollte bei Einsatz der Scopeverwaltung höher liegen als beim ursprünglichen Verfahren, da ein Datenbankknoten nur Beobachtungen aus

dem Scope berücksichtigt, in dem er sich gerade aufhält. Alle anderen Beobachtungen werden von diesem absichtlich verpasst. Diese Vermutung wurde bei Auswertung der Simulationen bestätigt.

Der Einfluss der Hintergrundlast auf das Verfahren wirkte sich bei Einsatz der Scopeverwaltung ähnlich aus wie beim ursprünglichen Verfahren. Der Vorteil, der bei der Analyse der Anzahl der versendeten Nachrichten ermittelt wurde, sollte es dem Verfahren mit Scopeverwaltung gestatten, auch bei höherer Hintergrundlast noch zuverlässig arbeiten zu können.

Abschließend kann gesagt werden, dass der Einsatz einer Scopeverwaltung eine sinnvolle Möglichkeit zur Beschränkung der Netzwerklast darstellt. Der zusätzliche Aufwand, der hierfür betrieben werden muss, lässt sich in klaren Grenzen halten. Entscheidende Faktoren, welche diesen beeinflussen, sind die Wahl und der Aufbau des Umgebungsmodells, sowie eine entsprechende Berücksichtigung der Mobilität von Datenbankknoten und beobachtbaren Objekten. Die große Anzahl an Möglichkeiten der zukünftigen Arbeiten zeigt, dass es noch einige Aspekte gibt, die beim Einsatz von Scopes näher zu betrachten sind. Diese Arbeit sollte erste Grundlagen liefern, auf die im weiteren aufgebaut werden kann.

## 10 Referenzen

- [BCS+98] Basagni, S.; Chlamtac, I.; Syrotiuk, V. R.; Woodward, B. A. „A distance routing effect algorithm for mobility (DREAM)“. In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom'98, pp. 76-84, (1998 )
- [BEDU03] Becker, Christian; Dürr Frank “On Location Models for Ubiquitous Computing”, zur Veröffentlichung eingereicht, (2003)
- [BFI+98] Berners-Lee T., Fielding R., Irvine U., Mansiter L., „Uniform resource identifiers (URI): Generic Syntax“, IETF RFC 2396 (1998)
- [BLCO99] Bluetooth Consortium, „Bluetooth specification volume“ (1999)
- [BRS03] Bettstetter C., Resta G., Santi P., „The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks“, IEEE Transactions on Mobile Computing, vol. 2, no. 3, pp. 257-269, (2003)
- [DOD97] Department of Defense, „World Geodetic System 1984, its definition and relationship with local geodetic systems“, Technical Report 8350.2, Third Edition, National Imagery and Mapping Agency, (1997)
- [DURO03] Dürr F., Rothermel K., “On a Location Model for Fine-Grained Geocast”, Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003), pp. 18-35
- [JIST02] C. Jiang, P. Steenkiste, “A Hybrid Model with a Computable Location Identifier for Ubiquitous Computing”, In Proceedings of the fourth International Conference on Ubiquitous Computing (UbiComp 2002)
- [KAP96] Kaplan, E.D., Ed., „Understanding GPS: principles and applications“, Artech House, Boston, MA, (1996)
- [KAR00] Oliver Karch, „Skript: Algorithmische Geometrie“, verfügbar unter [http://www-info1.informatik.uni-wuerzburg.de/vorlesungen/WS9899/algo\\_geo/](http://www-info1.informatik.uni-wuerzburg.de/vorlesungen/WS9899/algo_geo/), (2000)
- [KEG93] W. Kainz, M. Eggenhofer, I. Greasley, “Modeling Spatial Relations and Operations with Partially Ordered Sets”, In International Journal of Geographic Information Systems (1993)
- [KOVA02] Ko, Young-Bae; Vaidya Nitin H, “Flooding-Bases Geocasting Protocols for Mobile Ad Hoc Networks”, In: Mobile Networks and Applications 7, pp. 471-480, (2002)
- [NS2] Network Simulator ns2. Verfügbar unter <http://www.isi.edu/nsnam/ns/>

- [NTC+99] Ni, Sze-Yao; Tseng, Yu-Chee; Chen, Yuh-Shyan; Sheu, Jang-Ping; „The Broadcast Storm Problem in a Mobile Ad Hoc Network“, Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MOBICOM), pp. 151-162, (1999)
- [RHB03] K. Rothermel, J. Hähner, C. Becker, “Consistent Update Diffusion in Mobile Ad Hoc Networks”, zur Veröffentlichung eingereicht (2003)
- [SHB+03] Stepanov, I.; Hähner, J.; Becker, C.; Tian, J.; Rothermel, K., „A Meta-Model and Framework for User Mobility in Mobile Networks“, In: Proceedings of the 11th International Conference on Networking 2003 (ICON 2003)
- [WHF+92] Roy Want, Andy Hopper, Victoria Falcao, Jonathan Gibbons, „The Active Badge Location System“, ACM Transactions on Information Systems 10, pp 91-102, (1992)
- [WJH97] Andy Ward, Alan Jones, Andy Hopper, „A New Location Technique for the Active Office“, IEEE Personal Communications, Vol. 4, No. 5, pp 42-47, Technical Report 97.10, (1997)
- [WLST99] ANSI/IEEE Standard 802.11, (1999), verfügbar unter <http://standards.ieee.org>