QuON: Ein P2P-Protokoll für skalierbare und latenzarme virtuelle Welten

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Dipl.-Inform. Stephan Matthias Krause

aus Bonn

Tag der mündlichen Prüfung: 29.04.2011

Erste Gutachterin:	Prof. Dr. Martina Zitterbart Karlsruher Institut für Technologie (KIT)
Zweiter Gutachter:	Prof. Dr. Wolfgang Effelsberg Universität Mannheim

Danksagung

Ich möchte mich bei den Menschen bedanken, die zur Entstehung dieser Dissertation beigetragen haben.

Zunächst einmal danke ich Frau Prof. Dr. Martina Zitterbart. Sie brachte mir viel Geduld entgegen und sorgte mit wertvollen fachlichen Ratschlägen für das Gelingen der Arbeit. Ich möchte mich auch dafür bedanken, dass sie sich auf das Experiment einliess, mich mit einer halben Stelle am Insititut für Telematik als wissenschaftlicher Mitarbeiter zu beschäftigen.

Bei Herrn Prof. Dr. Wolfgang Effelsberg bedanke ich mich für die bereitwillige Übernahme meines Zweitgutachtens.

Dank gilt auch meinen Kolleginnen und Kollegen am Institut. Die wissenschaftlichen Diskussionen mit ihnen waren mir stets eine große Hilfe, und die fünf Jahre am Institut für Telematik eine sehr bereichernde und angenehme Zeit, nicht zuletzt wegen der guten und kollegialen Atmosphäre. Insbesondere danke ich Dr. Ingmar Baumgart und Bernhard Heep für die gute Zusammenarbeit im Projekt ScaleNet und bei der Entwicklung von OverSim. Ihre fundierten und konstruktiven Anmerkungen sowie der wissenschaftliche Austausch mit ihnen gaben mir viele Denkanstöße für die Entwicklung meiner Arbeit.

Ich bedanke mich auch bei den Studenten, die mit ihrer Studien- oder Diplomarbeit zu dieser Arbeit beigetragen haben. Besonderer Dank gilt Helge Backhaus, dessen hervorragende Studien- und Diplomarbeit einen wesentlichen Beitrag zu dieser Arbeit leisteten, und der mir auch später als Kollege am Institut mit seinen guten Ideen immer als kritischer und inspirierender Ansprechpartner zur Seite stand.

Abschließend möchte ich meinen Freunden und meiner Familie danken, die mich in der Zeit der Dissertation stets auf meinem Weg bestärkt haben und mich tatkräftig unterstützt haben. Meine Eltern haben mit Ihrer Unterstützung den Grundstein für eine erfolgreiche Arbeit gelegt. Ein besonderer Dank geht an meine Frau, die mir in der ganzen Zeit den Rücken freigehalten hat.

Stephan Krause Gaimersheim, im August 2011

Inhaltsverzeichnis

Danksagung **iii**

1 Einleitung **1**
1.1 Problemstellung . . . 2
1.2 Ziele und Beiträge der Arbeit . . . 3
1.3 Gliederung der Arbeit . . . 6

2 Grundlagen **7**
2.1 Virtuelle Welten . . . 7
2.1.1 Arten virtueller Welten . . . 7
2.1.2 Interaktionsmöglichkeiten . . . 9
2.1.3 Anforderungen . . . 10
2.1.4 Einflussgrößen . . . 11
2.2 Peer-to-Peer-Netze . . . 12
2.2.1 Grundprinzipien . . . 12
2.2.2 Unstrukturierte P2P-Netze . . . 13
2.2.3 Strukturierte P2P-Netze . . . 14
2.2.4 Weitere Anwendungen für P2P-Netze . . . 15
2.3 Ausgewählte Aspekte der Kryptographie . . . 16
2.3.1 Hash-Funktionen . . . 16
2.3.2 Public-Key-Kryptographie . . . 16
2.3.3 Signaturen . . . 17

3 Stand der Technik **19**
3.1 Client/Server . . . 20
3.1.1 Technologien zum Verteilen von Teilnehmern . . . 20
3.1.1.1 Sharding . . . 20
3.1.1.2 Instanzen . . . 21
3.1.1.3 Zonen . . . 21
3.1.2 Proxy-Architektur . . . 22
3.1.3 Cloud-Computing . . . 22
3.1.4 Bewertung . . . 23
3.2 P2P . . . 23
3.2.1 Klassifizierung von P2P-Protokollen für virtuelle Welten . . . 23
3.2.2 Vollvermaschte Protokolle . . . 24
3.2.3 Zonenbasierte Protokolle . . . 24
3.2.3.1 ALM-basierte Protokolle . . . 24
3.2.3.2 Supernode-basierte Protokolle . . . 25
3.2.3.3 Ansätze mit dynamischen Zonen . . . 26

3.2.4 Zonenlose Protokolle . . . 26
3.3 Hybride Protokolle . . . 27
3.4 Zusammenfassung . . . 28

4 Das Overlay-Simulationsrahmenwerk OverSim 31
4.1 Anforderungen an Simulationswerkzeuge für Overlay-Netze . . . 31
4.2 Bisherige Simulationswerkzeuge . . . 33
4.3 Basis: OMNeT++ . . . 34
4.4 Architektur . . . 35
4.5 Underlay-Schicht: Netzabstraktion . . . 38
4.5.1 SimpleUnderlay . . . 38
4.5.2 InetUnderlay und ReaseUnderlay . . . 39
4.5.3 Modelle für Echtnetzanbindung . . . 40
4.6 Overlay-Schicht . . . 40
4.7 Anwendungsschicht . . . 41
4.8 Modellierung des Nutzerverhaltens . . . 41
4.8.1 Sitzungsverhalten . . . 41
4.8.2 Applikationsverhalten . . . 45
4.9 Echtnetzanbindung . . . 46
4.10 Interaktive Benutzeroberfläche . . . 47
4.11 Evaluierung und Validierung . . . 48
4.12 Zusammenfassung . . . 50

5 Analyse selektierter P2P-Protokolle für verteilte virtuelle Welten 53
5.1 Protokollauswahl . . . 53
5.1.1 SimMUD . . . 54
5.1.1.1 Funktionsweise . . . 54
5.1.1.2 Implementierung . . . 56
5.1.1.3 Aufwandsabschätzungen . . . 56
5.1.2 PubSubMMOG . . . 57
5.1.2.1 Funktionsweise . . . 57
5.1.2.2 Implementierung . . . 58
5.1.2.3 Aufwandsabschätzungen . . . 59
5.1.3 N-Tree . . . 60
5.1.3.1 Funktionsweise . . . 60
5.1.3.2 Implementierung . . . 61
5.1.3.3 Aufwandsabschätzung . . . 62
5.1.4 Vast . . . 63
5.1.4.1 Funktionsweise . . . 63
5.1.4.2 Implementierung . . . 63
5.1.4.3 Aufwandsabschätzung . . . 65
5.1.5 Vergleich der Aufwandsabschätzungen . . . 65
5.2 Evaluierung . . . 68
5.2.1 Metriken . . . 69
5.2.2 Einflussgröße: Teilnehmerdichte . . . 69
5.2.3 Szenarien . . . 70
5.2.4 Ergebnisse . . . 71
5.2.4.1 Nachrichtenverzögerung . . . 71
5.2.4.2 Bandbreitenbedarf . . . 74

5.2.4.3 Nachbarschaftskenntnis . . . 78
5.2.5 Schlussfolgerungen . . . 80
5.3 Zusammenfassung . . . 81

6 Ein Overlay-Protokoll für verteilte virtuelle Welten: QuON 83
6.1 Grundidee . . . 83
6.2 Mögliche Metriken in QuON . . . 85
6.3 Definitionen . . . 86
6.4 Nachbarschaftsbeziehungen . . . 88
6.4.1 Direkte Nachbarn . . . 88
6.4.2 Verbindungsnachbarn . . . 89
6.4.3 Temporäre Nachbarn . . . 90
6.5 Positionsmeldungen . . . 93
6.6 Finden neuer Nachbarn . . . 95
6.6.1 Finden über direkte Nachbarn . . . 95
6.6.2 Finden über Verbindungsnachbarn . . . 97
6.6.3 Alternative Nachbarsfindung mit geringerem Bandbreitenbedarf 99
6.7 Teilnehmerbeitritt und Verlassen des Overlays . . . 101
6.7.1 Knotenbeitritt . . . 101
6.7.2 Geordnetes Verlassen des Overlays . . . 103
6.7.3 Behandlung von Teilnehmerausfällen . . . 103
6.7.4 Backup-Nachbarn . . . 103
6.7.4.1 Evaluierung . . . 106
6.8 Verbesserung der Nachbarschaftskenntnis bei Nachrichtenverzögerung 109
6.8.1 Evaluierung . . . 110
6.9 Dynamische Interessengebietsgröße . . . 112
6.9.1 Erforderliche Anpassungen des Grundprotokolls . . . 114
6.9.2 Adaptionsstrategien . . . 115
6.9.2.1 Lokale Strategien . . . 115
6.9.2.2 Verteilte Strategien . . . 116
6.9.3 Evaluierung . . . 117
6.10 Aufwandsabschätzung . . . 123
6.11 Zusammenfassung des Protokollablaufs . . . 124

7 Cheating 129
7.1 Cheating in virtuellen Welten . . . 129
7.1.1 Problembeschreibung . . . 129
7.1.2 Klassifizierung von Cheating . . . 130
7.1.3 Cheating in P2P-Systemen . . . 132
7.2 Related Work . . . 133
7.3 Cheatermodell . . . 137
7.4 Authentifizierung und Autorisierung . . . 138
7.5 Vertrauen . . . 140
7.6 Gegenseitige Überwachung . . . 141
7.7 Bestrafung von Cheatern . . . 143
7.8 Kontrolle der Spielfigur . . . 144
7.9 Kryptographische Absicherung . . . 145
7.10 Integration in QuON . . . 146
7.10.1 Überprüfung der Verbindungsnachbarn . . . 146

7.10.2 Verweigerung der Nachbarschaftsfindung . . . 147
7.10.3 Sicheres Login . . . 148
7.10.4 Ausschluss von Cheatern mit Ticketrückruf . . . 150
7.11 Evaluierung . . . 150
7.11.1 Implementierung des Prototypes . . . 150
7.11.2 Metriken . . . 151
7.11.3 Untersuchtes Cheat-Verhalten . . . 151
7.11.4 Einflussgrößen . . . 152
7.11.5 Szenarien . . . 152
7.11.6 Ergebnisse . . . 153
7.11.6.1 Cheat-Erkennungsrate . . . 153
7.11.6.2 Cheat-Erkennungszeit . . . 154
7.11.6.3 Fehlerrate . . . 157
7.11.6.4 Bandbreitenbedarf . . . 157
7.11.7 Bewertung . . . 159
7.12 Zusammenfassung . . . 160

8 Evaluierung 161
8.1 Evaluierung im Simulator . . . 161
8.1.1 Metriken . . . 161
8.1.2 Einflussgrößen . . . 162
8.1.3 Szenarien . . . 163
8.1.4 Evaluierung der Vektornormen . . . 164
8.1.5 Möglichkeiten der Bandbreiteneinsparung . . . 168
8.1.6 Teilnehmerbeitritt . . . 171
8.1.7 Einfluss des Sitzungsverhaltens der Teilnehmer . . . 173
8.1.8 Vergleich mit bisherigen Protokollen . . . 175
8.1.9 Schlussfolgerungen . . . 178
8.2 Evaluierung im Forschungsnetz . . . 179
8.2.1 Auswahl des Forschungsnetzes . . . 179
8.2.2 Szenario . . . 179
8.2.3 Ergebnisse . . . 180
8.2.4 Bewertung der Ergebnisse . . . 180
8.3 Zusammenfassung . . . 181

9 Zusammenfassung und Ausblick 183
9.1 Ergebnisse der Arbeit . . . 184
9.2 Zukünftige Arbeiten . . . 186

A Zusätzliche Messergebnisse 189
A.1 Evaluierung bisheriger Protokolle . . . 189
A.1.1 N-Tree . . . 189
A.1.2 SimMUD . . . 190
A.1.3 Vast . . . 191
A.2 Evaluierung Cheat-Erkennung . . . 192
A.3 Evaluierung QuON . . . 193

B Beweise 199
B.1 Korrektheit der Verbindungsnachbarbestimmung . . . 199

B.2 Korrektheit der Nachbarsfindung 208

Literaturverzeichnis **209**

1. Einleitung

Virtuelle Welten sind computerbasierte Anwendungen, in denen eine Vielzahl von Teilnehmern in einer gemeinsamen Umgebung miteinander interagieren können. In der Literatur wurden solche Welten schon vor geraumer Zeit popularisiert[1]. Die Verbreitung breitbandiger Internetanschlüsse und die gestiegenen grafischen Möglichkeiten heutiger Hardware haben die Grundlagen dafür geschaffen, dass virtuelle Welten heute im Internet eine wichtige Rolle spielen.

Die technischen Vorgänger der heutigen virtuellen Welten waren die 1978 entwickelten Multi User Dungeons (MUD) [11]. In diesen textbasierten Systemen konnten mehrere Spieler gemeinsam Rätsel lösen, sich unterhalten oder gegeneinander kämpfen. Zusätzlich bestanden vielfältige weitere Interaktionsmöglichkeiten wie beispielsweise der Handel mit virtuellen Gegenständen.

Aus diesen textbasierten Systemen entwickelten sich im Laufe der 1980er Jahre grafische virtuelle Welten, beispielsweise das von Lucasfilm entwickelte „Habitat“, das 1986 in eine öffentliche Testphase ging. Zu dieser Zeit war die mögliche Anzahl gleichzeitiger Spieler noch relativ gering. So erlaubte das ab 1991 von dem amerikanischen Online-Dienst AOL angebotene „Neverwinter Nights“ zunächst nur 50 gleichzeitig aktive Spieler. Bis zur Abschaltung von „Neverwinter Nights“ 1997 wurde die Kapazität ausgebaut: Von den 115.000 angemeldeten Spielern konnten ungefähr 2.000 gleichzeitig in die Spielwelt eingeloggt sein.

Heutige virtuelle Welten zeichnen sich dadurch aus, dass sie in der Regel eine weit größere Anzahl an gleichzeitigen Teilnehmern zulassen. In einer gemeinsamen Welt können oftmals mehrere tausend bis einige zehntausend Spieler gleichzeitig aktiv sein. Üblicherweise bleiben diese Welten, mit Ausnahme kurzer Wartungsperioden, persistent online. Eine aufgrund ihrer Teilnehmerzahl besonders hervorzuhebende virtuelle Welt ist das *Massively Multiplayer Online Game* (MMOG) „World of War-

[1] Hervorzuheben ist hier besonders der Roman „Neuromancer“ von William Gibson, der 1984 den Begriff des „Cyberspace“ popularisierte.

craft“ des Anbieters Blizzard Entertainment, welches mehr als 12 Millionen zahlende Teilnehmer zählt [22].

Gemeinsam haben alle heutigen kommerziellen virtuellen Welten, dass sie auf eine Client/Server-Architektur aufsetzen. Dies ermöglicht dem Betreiber eine maximale Kontrolle über die Aktionen der Teilnehmer. Allerdings werden dadurch den Welten auch Grenzen gesetzt: Alle Ereignisse müssen von den Teilnehmern an die Server gemeldet werden. Diese Server müssen prüfen, welche anderen Spieler von den Ereignissen betroffen sind, und die Ereignisnachrichten an die betroffenen Spieler weiterleiten. Aufgrund der dadurch erzeugten Netzwerk- und Rechenlast können die verwendeten Servercluster nur eine begrenzte Anzahl an gleichzeitigen Teilnehmern unterstützen. Wollen mehr Spieler an einer Welt teilnehmen, leidet entweder das Spielerlebnis aller aufgrund überlasteter Server, oder Teilnehmer müssen in einer Warteschlange ausharren, bis andere Teilnehmer die Welt verlassen haben. Weiterhin verursachen die Installation und der Betrieb der Servercluster hohe Kosten. Mit Techniken wie Cloud-Computing können zwar Spitzenlasten abgemildert werden, die Gesamtzahl an Servern und und die damit verbundenen Kosten können so jedoch nicht verringert werden.

1.1 Problemstellung

Eine Möglichkeit, eine größere Anzahl an gleichzeitigen Teilnehmern bei geringeren Kosten für den Betreiber im Vergleich zu heutigen virtuellen Welten zu unterstützen, ist das Verwenden von *Peer-to-Peer*-(P2P-)Systemen. Im Gegensatz zur Client/Server-Architektur, in welcher der Server Dienste anbietet und diese Dienste von den Clients genutzt werden, können in einem P2P-System alle Teilnehmer, *Peers* genannt, Dienste sowohl anbieten als auch nutzen. Die Menge der Ressourcen wächst somit linear mit der Anzahl der Nutzer. Aufgrund dieser Eigenschaft können mit P2P-Protokollen skalierbar Dienste bereitgestellt werden. Ein weiterer möglicher Vorteil für P2P-Systeme ergibt sich aus ihrer dezentralen Struktur: Durch den Verzicht auf zentrale Komponenten gibt es, im Gegensatz zu Client/Server-Architekturen, keinen sogenannten *Single Point of Failure*. Der Ausfall einzelner Komponenten kann also ohne Beeinträchtigungen für andere Nutzer aufgefangen werden.

Im heutigen Internet werden P2P-Protokolle größtenteils für Dateitauschsysteme eingesetzt. Wurden zunächst, wie beispielsweise bei der Dateitauschbörse „Napster“, unstrukturierte P2P-Protokolle verwendet, verlagert sich der Fokus heutzutage zunehmend auf strukturierte Protokolle.

Diese auf das Finden und Anbieten von Dateien optimierten Protokolle lassen sich jedoch nicht ohne weiteres für virtuelle Welten adaptieren, da hier deutlich andere Anforderungen an das P2P-Protokoll gestellt werden. Erste Überlegungen, P2P-Protokolle für virtuelle Welten zu nutzen, wurden jedoch schon 1990 von den Architekten des oben erwähnten „Lukasfilm Habitat“ getätigt [99]. In den letzten Jahren stieg das Interesse der Forschung an diesem Thema deutlich, und es entstand eine Vielzahl von möglichen P2P-Protokollen für virtuelle Welten. Es hat sich jedoch in der Forschungsgemeinschaft noch kein Konsens gefunden, welche dieser Protokolle für virtuelle Welten besonders geeignet sind. Im Rahmen dieser Arbeit werden diese Protokolle evaluiert und ein neues Protokoll, *QuON* (kurz für: Quadrantenbasiertes

Overlay-Netzwerk), vorgestellt, das besser als bisherige Protokolle wesentliche Anforderungen virtueller Welten unterstützt. Im Einzelnen sind die im Rahmen dieser Arbeit betrachteten Anforderungen:

- **Skalierbarkeit:** In den hier betrachteten virtuellen Welten sollen tausende von Teilnehmern gleichzeitig in einer gemeinsamen Welt aktiv sein können. Es ist also äußerst wichtig, dass die eingesetzten Protokolle skalierbar sind.
- **Geringe Latenz:** In virtuellen Welten können Teilnehmer auf vielfältige Weise interagieren. Bei vielen dieser Interaktionen ist es wichtig, dass die Kommunikation zwischen den Teilnehmern so wenig wie möglich verzögert wird.
- **Stabilität und Nachbarschaftskenntnis:** In einer virtuellen Welt befinden sich die Teilnehmer in einer gemeinsamen Welt. Um sinnvolle Interaktion zwischen den Teilnehmern zu ermöglichen, ist es wichtig, dass alle Teilnehmer einer virtuellen Welt alle Teilnehmer in ihrer direkten Nachbarschaft und deren Positionen kennen.
- **Cheatingerkennung:** In virtuellen Welten versuchen einige Teilnehmer, sich durch Regelbruch Vorteile gegenüber anderen Teilnehmern zu verschaffen. Diese müssen erkannt und aus der Welt entfernt werden können.
- **Geringer Bandbreitenbedarf:** Da in P2P-Protokollen die Teilnehmer Aufgaben übernehmen, die vormals von einem Server erfüllt worden sind, ist der Bandbreitenbedarf eines P2P-Protokolls für den Nutzer im Regelfall höher als bei einer Client/Server-Architektur. Der maximale Bandbreitenbedarf eines P2P-Protokolls darf jedoch die begrenzten Ressourcen seiner Nutzer nicht überschreiten.

Als Rahmenbedingung wird davon ausgegangen, dass die Teilnehmer an einer virtuellen Welt mit einem breitbandingen Anschluss an eine IP-Infrastruktur ausgestattet sind. Dabei wird vorausgesetzt, dass zwischen den Teilnehmern Ende-zu-Ende-Verbindungen aufgebaut werden können. Wie im Kontext von P2P-Protokollen üblich, ist mit ständiger Knotenfluktuation zu rechnen. Diese ergibt sich sowohl durch Fehlerzustände wie Verbindungsabbrüche als auch durch das geordnete Beenden einer Spiel-Sitzung durch den Teilnehmer.

1.2 Ziele und Beiträge der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung eines skalierbaren und latenzarmen P2P-Protokolls für virtuelle Welten. Als wichtige Anforderung soll mit diesem Protokoll die Stabilität und der Zusammenhang des Netzes gewährleistet bleiben und eine gute Nachbarschaftskenntnis erreicht werden können. Weiterhin sollen Cheater, die sich regelwidrige Vorteile vor anderen Teilnehmern verschaffen wollen, erkannt und aus dem Netz entfernt werden können. Dabei darf die maximal benötigte Bandbreite die Ressourcen eines durchschnittlichen Internetnutzers nicht überschreiten.

Die Beiträge der Dissertation liegen somit in folgenden Bereichen:

- Entwicklung eines P2P-Protokolls für virtuelle Welten mit geringer Latenz bei geringem Bandbreitenbedarf.

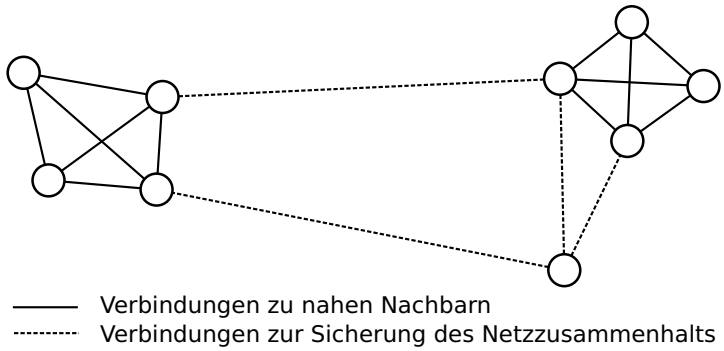


Abbildung 1.1 Die Grundidee bei Mutual-Notification-Protokollen. Gezeigt sind die Positionen der Teilnehmer in der virtuellen Welt sowie die Verbindungen zwischen den Teilnehmern.

  - Entwicklung und Evaluierung des quadrantenbasierten P2P-Protokolls QuON.
  - Entwicklung und Evaluierung von Mechanismen zur Wahrung der Stabilität und Nachbarschaftskenntnis.
  - Entwicklung und Evaluierung von Mechanismen zur Vermeidung von durch hohe Spielerdichten verursachte Überlastsituationen.
  - Entwicklung und Evaluierung von Cheatingerkennungsmaßnahmen.

- Entwicklung des Overlay-Frameworks OverSim.

Das im Rahmen dieser Arbeit entwickelte *P2P-Protokoll für virtuelle Welten, QuON,* soll die oben erläuterten Anforderungen virtueller Welten besser erfüllen als bisherige Protokolle. Viele der bisherigen Protokolle basieren auf einer Unterteilung der Spielwelt in Zonen [58, 79, 144, 149]. Diese Aufteilung führt, wie in Kapitel 5 gezeigt wird, in der Regel zu hohen Latenzen. Aus diesem Grund basiert QuON auf dem zonenlosen Mutual Notification-Ansatz. In Mutual-Notification-Protokollen werden alle Ereignisse direkt vom Verursacher an alle interessierten Teilnehmer kommuniziert. Dies sind in der Regel alle anderen Teilnehmer, die sich in der virtuellen Welt in einem gewissem Umkreis um die Position des Verursachers befinden.

Dazu bauen Teilnehmer, die in der Spielwelt nahe beieinander stehen, wie in Abbildung 1.1 dargestellt, direkte Verbindungen miteinander auf. Dadurch kann zwischen unmittelbar interagierenden Teilnehmern in der Regel eine geringe Latenz gewährleistet werden. Um eine Partitionierung des Netzes zu verhindern, müssen Verbindungen zwischen weiter entfernten Teilnehmern gebildet werden. Um den Bandbreitenbedarf zu beschränken, sollte die Anzahl dieser Verbindungen nicht zu hoch gewählt werden.

Die bislang vorgeschlagenen Mutual-Notification-Protokolle wie VAST [67] zeigen zwar das Potential dieses Ansatzes, können jedoch die oben beschriebenen Anforderungen nicht voll erfüllen. Insbesondere ist die Nachbarschaftskenntnis bisheriger Mutual Notification-Protokolle in der Regel unzureichend. Zusätzlich liegt die benötigte Bandbreite bisheriger Protokolle deutlich über den Ressourcen durchschnittlicher Internetnutzer.

Um diese Probleme zu vermeiden, teilt jeder Teilnehmer in QuON die Spielwelt in vier Quadranten auf. Statt der in bisherigen Protokollen verwendeten impliziten Nachbarschaftsfindung aus Voronoi- oder Delaunaygraphen wählt ein Teilnehmer in QuON nun aus jedem Quadranten explizit einen Teilnehmer als Verbindungsnachbarn. Dabei werden immer möglichst nahe Teilnehmer als Verbindungsnachbarn gewählt. Somit werden unnötige weit reichende Verbindungen vermieden und der Protokolloverhead reduziert. Nur falls keine nahen Teilnehmer vorhanden sind, werden weiter entfernte Nachbarn als Verbindungsnachbarn gewählt. Mit diesen Nachbarschaftsbeziehungen wird der Netzzusammenhalt gewährleistet und eine gute Nachbarschaftskenntnis erreicht: Die Evaluation in Kapitel 8 zeigt, dass QuON so eine Nachbarschaftskenntnis von über 99,9% erreicht. Dies bedeutet, dass 99,9% der Teilnehmer korrekt über die Positionen aller Nachbarn informiert sind. Dabei beträgt die Latenz bei der Zustellung von Ereignisnachrichten im Durchschnitt 90ms und entspricht damit den guten Werten bisheriger Mutual Notification Protokolle. Im Vergleich zu bisherigen Mutual Notification Protokollen ist die benötigte Bandbreite um bis zu 80% geringer.

Ein mögliches Problem bei P2P-Protokollen für virtuelle Welten sind durch sogenannte *Hotspots* verursachte *Überlastsituationen*. Hotspots sind Bereiche, in denen eine Vielzahl von Teilnehmern auf sehr engem Raum steht. Da normalerweise Spieler Ereignisnachrichten an alle anderen Spieler in ihrem Interessengebiet senden, führen Hotspots zu einer großen Menge an Nachrichten und somit zu einer Überlastung der Teilnehmer. Eine mögliche Lösung dieses Problems ist das dynamische Anpassen der Größe des Interessengebiets an die Lastsituation. Droht eine Überlast, können die Teilnehmer ihr Interessengebiet dynamisch verkleinern. Eine solche dynamische Anpassung der Interessengebiete kann jedoch die Nachbarschaftskenntnis negativ beeinflussen. In dieser Arbeit wird ein Adaptionsalgorithmus vorgestellt, der in Überlastsituationen den Bandbreitenbedarf deutlich senkt und dabei die Nachbarschaftskenntnis nur geringfügig verringert. Dies wird erreicht, indem Teilnehmer nicht nur auf lokales Wissen zurückgreifen, sondern die Interessengebietsgröße anderer Teilnehmer mit berücksichtigen. Mit dieser Information können Hotspots vorab erkannt werden; somit kann ein Teilnehmer sein Interessengebiet verkleinern, bevor die Überlastsituation eintritt. In Evaluationen wurde gezeigt, dass die benötigte Spitzenbandbreite in Hotspot-Szenarien mit diesen Techniken um 75% gesenkt werden kann.

Das Problem der *Cheatingerkennung* wurde in den meisten Vorschlägen für P2P-Protokolle für virtuelle Welten bislang ignoriert. Dediziert vorgeschlagene Anti-Cheating-Protokolle lassen sich in der Regel nicht direkt in die vorgeschlagenen Protokolle integrieren, ohne die Erfüllung der oben beschriebenen Anforderungen deutlich zu verschlechtern. So steigern Anti-Cheating-Protokolle, die eine Synchronisation der Nachrichten der Teilnehmer erfordern [14, 39], oft die Nachrichtenverzögerung deutlich. Andere Anti-Cheating-Protokolle fordern, dass zentrale Komponenten einen großen Anteil der versendeten Nachrichten begutachten oder weiterverteilen [147] müssen. Dadurch wird die Skalierbarkeit der Protokolle stark eingeschränkt. In dieser Arbeit wird eine Cheatingerkennung vorgestellt, die sich direkt in QuON integrieren lässt. Dabei müssen keine vergrößerten Latenzen in Kauf genommen werden und die Skalierbarkeit wird nicht beeinträchtigt. Hierzu kontrollieren sich die Teilnehmer gegenseitig. In konkreten Verdachtsfällen werden die des Cheatings verdächtigen Spieler von einem vertrauenswürdigen Beobachter überprüft, bevor sie gege-

benenfalls für ihr Fehlverhalten bestraft werden. Dadurch wird der Einfluss falscher Anschuldigungen gering gehalten.

Um P2P-Protokolle für virtuelle Welten evaluieren zu können, ist der Einsatz von *Simulationswerkzeugen* hilfreich. Bisherige Simulatoren [57, 70, 90, 125] können jedoch wichtige Anforderungen an eine flexible, skalierbare und realistische Simulation von P2P-Protokollen nicht erfüllen [100]. Aus diesem Grunde wurde im Rahmen dieser Arbeit in Zusammenarbeit mit Kollegen das Simulations-Rahmenwerk *OverSim* [15] entwickelt. OverSim ermöglicht die skalierbare Simulation von P2P-Protokollen in verschiedenen Szenarien. Durch seine modulare Architektur lassen sich alle Komponenten transparent austauschen. So können beispielsweise verschiedene Underlay-Modelle eingesetzt werden, die sich in der Abstraktion der unterliegenden Netzwerkschichten unterscheiden. Durch spezielle Underlay-Modelle eignet sich OverSim auch für die Untersuchung von P2P-Protokollen in Testnetzen wie PlanetLab oder G-Lab. Dabei bleibt OverSim skalierbar, einfache Simulationen sind auch mit über 100.000 Knoten in realistischer Zeit durchführbar [15].

1.3 Gliederung der Arbeit

Die weitere Arbeit ist wie folgt gegliedert: In Kapitel 2 werden zunächst die Grundlagen dargelegt, die zum Verständnis der weiteren Arbeit notwendig sind. Hier werden auch die Anforderungen, die ein P2P-Protokoll für virtuelle Welten erfüllen muss, näher erläutert. In dem darauf folgenden Kapitel 3 wird der Stand der Technik heutiger virtueller Welten diskutiert.

Im Rahmen der Arbeit wurde gemeinsam mit Kollegen das Simulations-Rahmenwerk OverSim entwickelt. Dieses wird in Kapitel 4 vorgestellt. Dabei wird auf die einzelnen Komponenten des Rahmenwerks sowie auf die Besonderheiten bei der Simulation von virtuellen Welten eingegangen. OverSim wird später verwendet, um die Leistungsfähigkeit der in dieser Arbeit betrachteten Protokolle zu evaluieren.

Um die Entwicklung eines neuen P2P-Protokolls für virtuelle Welten vorzubereiten, werden in Kapitel 5 bisherige Protokollvorschläge mit OverSim evaluiert. Insbesondere werden die Stärken und Schwächen der Protokolle bezüglich der im Grundlagenkapitel vorgestellten Anforderungen erarbeitet.

Anhand der bei der Evaluierung gewonnenen Erkenntnisse wird in Kapitel 6 das neue quadrantenbasierte Protokoll *QuON* entworfen, welches die Schwächen bestehender Protokolle vermeiden soll. Es werden die Basisarchitektur und grundlegende Protokollmechanismen aufgezeigt. Zusätzlich werden Mechanismen vorgestellt, mit denen die Leistung des Protokolls in Situationen mit hoher Knotenfluktuation oder hoher Teilnehmerdichte verbessert werden kann. In Kapitel 7 wird besprochen, wie QuON um eine effiziente Methode zur Erkennung von Cheating erweitert werden kann.

In Kapitel 8 wird das in den vorangehenden Kapiteln entwickelte Protokoll evaluiert. Dabei werden umfassende Simulationen des Protokolls mit OverSim vorgestellt und die Ergebnisse mit Evaluierungsergebnissen bestehender Protokollvorschläge verglichen. Weiterhin wird eine Installation von QuON auf dem deutschlandweiten Testbett G-Lab vorgestellt.

Abschließend wird die Arbeit im Kapitel 9 zusammengefasst und ein Ausblick auf weitergehende Forschungsmöglichkeiten gegeben.

2. Grundlagen

In diesem Kapitel werden für das Verständnis der folgenden Kapitel benötigte Grundlagen vorgestellt. Im Einzelnen werden Grundprinzipien und wichtige Eigenschaften virtueller Welten und von Peer-to-Peer-Netzen sowie einige Aspekte der Kryptographie erläutert.

2.1 Virtuelle Welten

Virtuelle Welten sind vernetzte Systeme, in denen zahlreiche Nutzer miteinander kommunizieren oder in sonstiger Weise miteinander interagieren können. Die Teilnehmer einer solchen Welt werden durch sogenannte *Avatare*[1] repräsentiert. Mit diesen Avataren können sie sich in der virtuellen Welt bewegen. Die Interaktionsmöglichkeiten zwischen den Teilnehmern werden auch durch die räumliche Position der Avatare in der virtuellen Welt bestimmt. So ist beispielsweise ein Unterhaltung nur bei räumlicher Nähe der Avatare zueinander möglich.

2.1.1 Arten virtueller Welten

Virtuelle Welten lassen sich in zwei Klassen einteilen: *Soziale virtuelle Welten* und *Massively Multiplayer Online Games* (kurz: MMOGs). Soziale virtuelle Welten sind auf die soziale Interaktion zwischen den Teilnehmern fokussiert. Der Schwerpunkt der Interaktionsmöglichkeiten liegt hier in der Regel auf den Kommunikationsmöglichkeiten, also text- oder sprachbasierter Unterhaltung, dem Versenden von Nachrichten oder der öffentlichen Diskussion. Ein weiterer wichtiger Aspekt von sozialen virtuellen Welten ist das Erstellen von und der Handel mit virtuellen Gegenständen.

Die momentan größte soziale virtuelle Welt ist *„Second Life“* [92] des Anbieters Linden Labs mit mehr als 21 Millionen angemeldeten Nutzern [93]. Da die Nutzung von

[1] Der Begriff Avatar wurde 1992 durch den Roman „Snow Crash“ von Neal Stevenson popularisiert. Er bezeichnet dort eine grafische Repräsentation von Personen der physischen Welt in einer virtuellen Welt.

Second Life kostenlos ist und einmal erstellte Nutzerkonten nicht auslaufen, umfasst diese Zahl nahezu alle Teilnehmer, die je in Second Life aktiv waren. Im November und Dezember 2010 waren etwas mehr als 1,3 Millionen Nutzer in dieser virtuellen Welt aktiv [93]. Ein Hauptmerkmal von Second Life ist die Möglichkeit für Nutzer, 3D-Objekte in der Welt zu erstellen und mit einer Programmiersprache, der „Linden Scripting Language“, mit Animationen oder zusätzlichen Funktionen zu versehen. Diese Objekte können in der virtuellen Welt gehandelt werden. Die in der virtuellen Welt verwendete Währung lässt sich in US$ umtauschen. Das Handelsvolumen in der virtuellen Welt und auf den angeschlossenen E-Commerce-Seiten lässt sich so in US$ umrechnen und betrug im 1. Quartal des Jahres 2010 über 2,4 Millionen US$. Einen ähnlichen Ansatz wie Second Life verfolgt das bereits länger am Markt befindliche, jedoch kommerziell weniger erfolgreiche *„ActiveWorlds“* [2]. *„Entropia Universe“* [46] hält den aktuellen Weltrekort für den Verkauf des teuersten virtuellen Grundstücks mit einem Verkaufspreis von 335.000 US$ [62]. Einige soziale virtuelle Welten wie *„Smeet“* [127] integrieren sich in soziale Netzwerke wie Facebook.

Neben kommerziellen Anbietern virtueller Welten gibt es auch Open Source-Initiativen wie *„Open Wonderland“* [105] oder das zu Second Life kompatible *„OpenSimulator“* [106], die Rahmenwerke zur Erstellung virtueller Welten entwickeln. Diese ermöglichen es Unternehmen und Privatanwendern, virtuelle Welten für kleinere Nutzergruppen anzubieten. Diese Welten können teilweise verbunden werden, sodass eine Web-ähnliche Struktur aus verbundenen Welten entstehen kann.

Ein Spezialfall sozialer virtueller Welten sind virtuelle Welten für Ausbildungs- und Lernzwecke. Hier ist ein Handel mit virtuellen Gegenständen in der Regel nicht vorgesehen. Stattdessen werden von der virtuellen Welt Hilfsmittel zur Wissensvermittlung angeboten. Beispiele für eine solche virtuelle Welt sind das an der Duke University, NC, USA entwickelte *„Open Cobalt“* [94] und das an der Indiana University, IN, USA entwickelte *„Quest Atlantis“* [10].

Der größte Teil der virtuellen Welten sind jedoch keine sozialen virtuellen Welten, sondern MMOGs. Bei MMOGs steht im Gegensatz zu sozialen virtuellen Welten das Online-Spiel im Vordergrund. Dabei können Spieler in der virtuellen Welt gegeneinander oder gemeinsam gegen computergesteuerte Gegner kämpfen. Abgrenzungsmerkmale zu anderen Online-Spielen sind die Größe sowie die Persistenz der virtuellen Welt. Während sich in normalen Online-Spielen meist weniger als 100 Spieler in einer gemeinsamen Spielumgebung befinden und diese Spielumgebung vor jeder Spielrunde in einen festgelegten initialen Zustand gebracht wird, ist in MMOGs die Zahl der gleichzeitig in einer Welt aktiven Spieler deutlich größer. Zudem läuft die Spielwelt in MMOGs kontinuierlich weiter. Ein Zurücksetzen der Welt findet im Gegensatz zu normalen Online-Spielen nicht statt. Die meisten MMOGs stammen aus dem Spielgenre der Rollenspiele.

Das MMOG *„World of Warcraft“* [23] ist die virtuelle Welt mit der größten Nutzerzahl. Im Oktober 2010 waren bei dem Spiel mehr als 12 Millionen zahlende Nutzer angemeldet [22]. Durch die große Zahl zahlender Nutzer ist World of Warcraft auch die kommerziell erfolgreichste virtuelle Welt.

Das MMOG mit den meisten in einer gemeinsamen Welt gleichzeitig aktiven Spielern ist *„EVE Online“* [28]. Hier wird auf die bei MMOGs oft verwendete Unterteilung der Spieler auf getrennte Realms oder Shards (siehe hierzu Abschnitt 3.1.1.1) verzichtet. Zu Spitzenzeiten sind in EVE Online über 60.000 Spieler gleichzeitig in einer

gemeinsamen Welt online [27, 104]. Das größte nach dem *„Free to Play“*-Modell, bei dem der Anbieter keine Gebühren für das Spielen verlangt sonders sich über den Verkauf von Spielgegenständen finanziert, angebotene MMOG ist *„RuneScape“* [69] mit mehr als 10 Millionen Teilnehmern.

Neben den Rollenspielen werden auch Spiele anderer Genres als MMOG angeboten. Bei dem Spiel *„PlanetSide“* [128] handelt es sich um einen sogenannten „First Person Shooter“ (kurz: FPS), bei dem der Spieler seinen Avatar nicht von außen sieht, sondern aus der Ego-Perspektive spielt. Spiele dieses Genres sind für ihre besonders hohen Anforderungen an die Netzverzögerung bekannt.

Die Grenzen zwischen sozialen virtuellen Welten und MMOGs sind teilweise fließend. Bereits die 1986 veröffentlichte soziale virtuelle Welt *„Habitat“*, eine der ersten virtuellen Welten [99], ermöglichte neben der sozialen Interaktion auch Kämpfe zwischen den Spielern. Auch Second Life ermöglicht den Teilnehmern in einigen Bereichen der Spielwelt das Tragen und Benützen von Waffen. Andererseits lassen viele MMOGs auch den Handel oder andere soziale Interaktionen zu.

2.1.2 Interaktionsmöglichkeiten

Die einfachste Interaktionsform in virtuellen Welten ist die *Unterhaltung* zwischen Teilnehmern. Diese erfolgt oft auf Basis von Textnachrichten. Einige virtuelle Welten ermöglichen auch eine Sprachkommunikation zwischen Teilnehmern.

Eine besonders in sozialen virtuellen Welten häufig genutzte Interaktion ist der *Handel* virtueller Gegenstände. Hierbei kann ein Teilnehmer einen solchen Gegenstand gegen eine Spielwährung oder im Tausch gegen einen anderen Gegenstand einem anderen Teilnehmer weitergeben.

Eine in MMOGs wichtige Interaktionsform ist der *Kampf* zwischen Teilnehmern oder zwischen einem Teilnehmer und einem computergesteuerten Gegner. Der Ablauf dieser Interaktion ist stark von der konkreten virtuellen Welt abhängig. In der Regel können Teilnehmer verschiedene Nah- und Fernkampfwaffen sowie Zaubersprüche oder andere fiktive Kampftechniken nutzen. Dabei verliert ein Spieler durch erlittene Treffer Lebenspunkte. Diese können teilweise durch freundlich gesinnte Teilnehmer „geheilt“ werden. Erreicht die Zahl der Lebenspunkte eines Teilnehmers null, „stirbt“ der Teilnehmer und kann bis zu einer späteren Wiederbelebung nicht mehr mit anderen Teilnehmern interagieren. Da der Kampf in MMOGs üblicherweise in Echtzeit abläuft, stellt diese Interaktionsform hohe Anforderungen an die Nachrichtenverzögerung. Die von einem Teilnehmer ausgehenden Interaktionen wie Unterhaltung, Handel und Kampf werden mit *Ereignisnachrichten* an andere Teilnehmer kommuniziert.

Da alle Teilnehmer die *Bewegung* der anderen Teilnehmer in der virtuellen Welt sehen können, kann auch die Bewegung als Interaktion gewertet werden. Um die Bewegungen anderer Teilnehmer zeitnah visualisieren zu können, müssen sowohl regelmäßig von allen Teilnehmern die aktuellen Positionen mit einer *Positionsmeldung* gemeldet, als auch die Positionsmeldungen der anderen Teilnehmer ausgewertet werden. In der Praxis werden fehlende Positionsdaten oft durch Techniken wie *Koppeln* (englisch: dead reckoning) ergänzt. Dies kann jedoch fehlende Positionsdaten nur kurzfristig ersetzen. Bei längeren Ausfällen divergieren geschätzte und tatsächliche Position zu stark.

Die Interaktionsmöglichkeiten in virtuellen Welten basieren in der Regel auf der Position der kommunizierenden Teilnehmer in der virtuellen Welt. Dabei wird gefordert, dass die Positionen der Teilnehmer nicht zu weit voneinander entfernt sind. Die maximale Entfernung ist dabei von der konkreten Interaktion abhängig. Für das Handeln von Gegenständen kann beispielsweise gefordert werden, dass die Teilnehmer in der virtuellen Welt unmittelbar nebeneinander stehen. Eine Unterhaltung zwischen Teilnehmern kann erfordern, dass sich diese in Hörweite befinden. Die maximale Entfernung, über die Teilnehmer interagieren können, lässt sich zur Definition des *Interessengebiets* eines Teilnehmers nutzen. Das Interessengebiet ist der Bereich der virtuellen Welt, der alle Teilnehmer umfasst, mit denen ein gegebener Teilnehmer zu einem Zeitpunkt interagieren kann.

Virtuelle Welten fördern oft das Zusammenspiel mehrerer Teilnehmer. Deshalb schließen sich Teilnehmer oft zu Gruppen zusammen, die sich gemeinsam in der Welt bewegen und ein gemeinsames Ziel verfolgen. In einem MMOG könnte solch ein Zusammenschluss beispielsweise zum Besiegen eines besonders starken Gegners oder zum Kampf gegen eine andere Teilnehmergruppe erfolgen. Der Zusammenschluss der Teilnehmer kann kurzfristig erfolgen und nur für eine begrenzte Zeit bestehen.

2.1.3 Anforderungen

Das Ermöglichen der Interaktion von Nutzern ist das wichtigste Ziel virtueller Welten. Damit dieses Ziel bestmöglich erreicht werden kann, müssen einige Anforderungen erfüllt werden. In dieser Arbeit werden insbesondere die folgenden Anforderungen virtueller Welten betrachtet:

- **Skalierbarkeit:** Das Ziel virtueller Welten ist es, so vielen Teilnehmern wie möglich in einer gemeinsamen Welt Interaktionsmöglichkeiten zu bieten. Deswegen ist Skalierbarkeit bezüglich der Teilnehmerzahl eine wesentliche Anforderung virtueller Welten. Bei bisherigen Client/Server-basierten virtuellen Welten ist, wie in Abschnitt 3.1.4 ausführlicher aufgeführt, die Skalierbarkeit eingeschränkt. Bei der Verwendung von P2P-Protokollen können auch die Ressourcen der Teilnehmer genutzt werden. In diesen Protokollen steigen die Gesamtressourcen mit der Zahl der Teilnehmer. Sofern der Ressourcenbedarf der virtuellen Welt nicht stärker mit der Teilnehmerzahl steigt als das Ressourcenangebot der Peers können P2P-Protokolle skalierbare virtuelle Welten ermöglichen.

- **Geringe Latenz:** Viele Interaktionen in virtuellen Welten erfordern eine verzögerungsarme Kommunikation zwischen den Teilnehmern. Die maximale akzeptable Verzögerung ist dabei von der Art der Interaktion abhängig. Die höchsten Anforderungen stellt hier die Kampf-Interaktion in MMOGs. Eine Verzögerung von mehr als 250 ms wird von Spielern von MMOGs als Einschränkung empfunden [31, 126], Verzögerungen von mehr als 500 ms nicht toleriert [35]. Grundsätzlich gilt bei Echtzeit-Interaktionen, dass die Nachrichtenverzögerungen so gering wie möglich gehalten werden sollen.

- **Stabilität und Nachbarschaftskenntnis:** Teilnehmer in virtuellen Welten können mit allen Teilnehmern interagieren, die sich in ihrem Interessengebiet befinden. Die *Nachbarschaftskenntnis* wird hier definiert als der Anteil der

Teilnehmer, der alle anderen Teilnehmer kennt, die sich in seinem Interessengebiet befinden. Eine hohe Nachbarschaftskenntnis ist Voraussetzung dafür, dass Teilnehmer nicht in ihrer Interaktionsmöglichkeit eingeschränkt sind. Eine hohe Nachbarschaftskenntnis lässt sich nur erreichen, wenn das Netz als solches stabil bleibt. Treten beispielsweise Netzpartitionen auf, wie dies unter anderem bei dem P2P-Protokoll Vast geschehen kann [7], können sich Teilnehmer aus unterschiedlichen Partitionen in der virtuellen Welt einander nicht mehr sehen und nicht mehr miteinander interagieren.

- **Cheat-Erkennung:** *Cheating*, also der Versuch böswilliger Teilnehmer, sich durch Regelbruch Vorteile gegenüber ehrlichen Spielern zu verschaffen, ist ein gängiges Problem in virtuellen Welten. Hierdurch wird das Spiel-Erlebnis der ehrlichen Spieler verschlechtert. Aus diesem Grund müssen Cheater aus der virtuellen Welt entfernt werden. Hierzu ist es notwendig, das Cheating erkannt werden kann und Cheater aus dem Spiel ausgeschlossen werden können. Bei klassischen Client/Server-basierten virtuellen Welten kann dies von dem zentralen Server durchgeführt werden. Bei P2P-Protokollen muss eine verteilte Cheat-Erkennung eingeführt werden.

- **Geringer Bandbreitenbedarf:** Nutzer virtueller Welten verfügen in der Regel über Breitbandanbindungen ans Internet. Diese sind oft asymmetrisch. In Empfangsrichtung sind 4 Mbit/s bis zu 50 Mbit/s üblich. In Senderichtung ist die verfügbare Bandbreite üblicherweise deutlich geringer, eine Bandbreitenbeschränkung bei 1 Mbit/s ist üblich [50]. Der Bandbreitenbedarf einer virtuellen Welt darf die begrenzten Ressourcen des Nutzers nicht überschreiben.

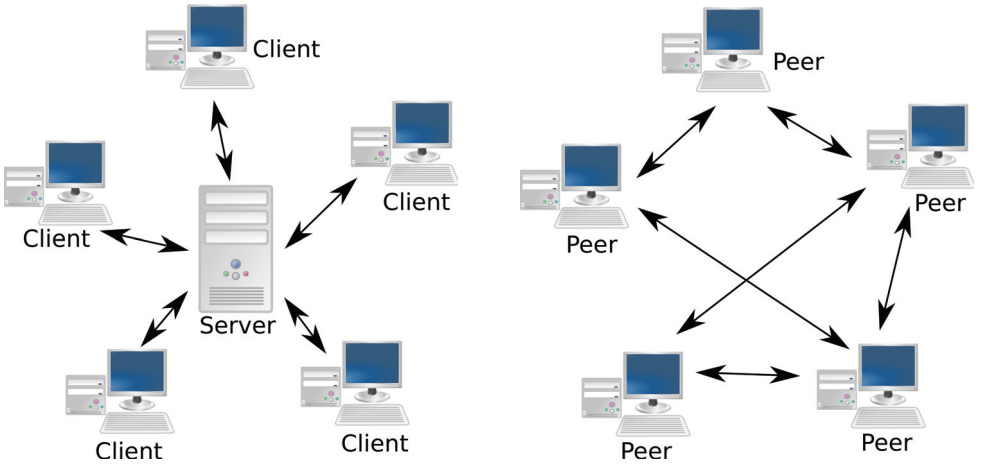
2.1.4 Einflussgrößen

Verschiedene Größen können Protokolle für virtuelle Welten beeinflussen. Die wichtigsten sind:

- **Teilnehmerzahl:** Die Teilnehmerzahl der virtuellen Welt ist bei Client/Server-basierten virtuellen Welten die wichtigste Einflussgröße. Die Zahl der vom Server zu empfangenden und zu sendenden Nachrichten und die Menge der vom Server zu bearbeitenden und zu speichernden Daten sind proportional zur Zahl der Teilnehmer. Die Leistungsfähigkeit des Servers setzt so eine Obergrenze für die Zahl der möglichen Teilnehmer der virtuellen Welt.

- **Teilnehmerdichte:** Die Teilnehmerdichte gibt an, wie viele Teilnehmer sich in einer bestimmten Fläche der virtuellen Welt aufhalten. Da Teilnehmer in virtuellen Welten mit räumlich nahen Teilnehmern interagieren können, steigt die Zahl der Interaktionen mit der Teilnehmerdichte. Dies kann sowohl bei Client/Server-basierten als auch bei P2P-basierten virtuellen Welten die Last der Teilnehmer und der an der Kommunikation der Teilnehmer beteiligten Systeme steigern.

- **Sitzungsverhalten der Teilnehmer:** Das Sitzungsverhalten der Teilnehmer beinhaltet Eigenschaften wie die durchschnittliche Sitzungszeit, die Verteilung der Sitzungszeiten und die Ausfallwahrscheinlichkeit der Teilnehmer. Besonders in P2P-basierten virtuellen Welten können kurze Sitzungszeiten und hohe



(a) Client/Server **(b)** Peer-to-Peer

Abbildung 2.1 Struktur eines Client/Server-Systems und eines P2P-Netzes.

Ausfallwahrscheinlichkeiten die Stabilität des Protokolls und somit die Nachbarschaftskenntnis gefährden.

2.2 Peer-to-Peer-Netze

In dieser Arbeit werden virtuelle Welten auf Basis von Peer-to-Peer-Protokollen betrachtet. Die Grundkonzepte und wichtige Einsatzmöglichkeiten von Peer-to-Peer-Protokollen werden in den folgenden Abschnitten vorgestellt.

2.2.1 Grundprinzipien

In klassischen Client/Server-Systemen stellt der Server Ressourcen bereit. Die Clients nutzen diese Ressourcen. Ein solches System ist in Abbildung 2.1a dargestellt. Da nur der Server Ressourcen bereitstellt, gibt es in diesen Systemen ein Skalierungsproblem: Nutzen mehr Clients das System, werden mehr Ressourcen benötigt. Der Ressourcenbedarf steigt dabei mindestens linear mit der Zahl der Nutzer. Der Server kann jedoch nur konstante Ressourcen bereitstellen. Dies begrenzt die Zahl der von einem Server unterstützten Clients. In einem Peer-to-Peer-System (kurz: P2P) sind alle Peers gleichberechtigt. Alle Peers können die Ressourcen anderer Peers nutzen, stellen aber auch eigene Ressourcen bereit. Dies ist in Abbildung 2.1b gezeigt. In diesen Systemen steigen die Ressourcen linear mit der Zahl der Nutzer. Dies ist in Abbildung 2.2 dargestellt. P2P-Systeme können so besser skalieren als Client/Server-Systeme. Eine Voraussetzung hierfür ist allerdings, dass der Ressourcenbedarf nicht deutlich mehr als linear mit der Teilnehmerzahl steigt.

Die Kommunikation in P2P-Netzen erfolgt oft in der Form eines *Overlays*. Dies bezeichnet eine virtuelle Netztopologie auf Basis einer bestehenden physischen Topologie. Dabei werden in der Overlay-Schicht Verbindungen zwischen Teilnehmern aufgebaut, welche für die Kommunikation und die Weiterleitung von Nachrichten genutzt werden. Zwischen in der Overlay-Schicht direkt verbundenen Teilnehmern können in der unterliegenden Topologie mehrere Zwischenknoten liegen. Weiterhin kann es vorkommen, dass in der unterliegenden Topologie Teilnehmer direkt verbunden sind, in der Overlay-Topologie jedoch keine direkte Verbindung besteht. Eine solche Overlay-Topologie ist in Abbildung 2.3 gezeigt.

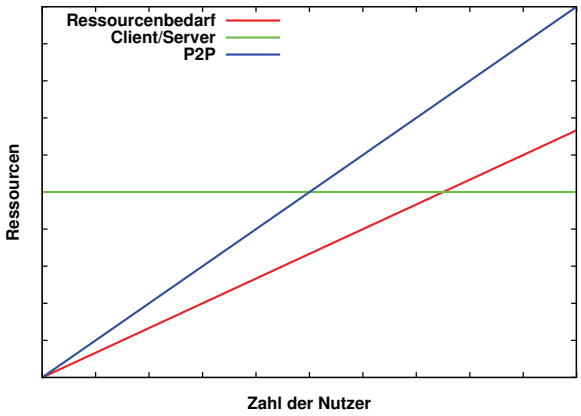


Abbildung 2.2 Minimaler Ressourcenbedarf und verfügbare Ressourcen in Client/Server- und P2P-Systemen.

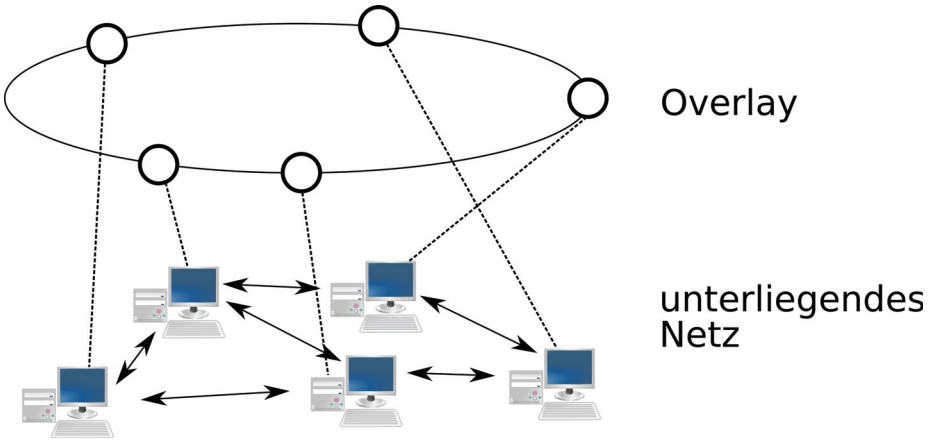


Abbildung 2.3 Schematische Overlay- und Underlay-Topologie.

2.2.2 Unstrukturierte P2P-Netze

P2P-Netze wurden Ende der 90er Jahre durch Dateitauschbörsen populär. In diesen Dateitauschbörsen können Nutzer Dateien anderer Nutzer suchen und herunterladen. Als Vorreiter dieser Dateitauschbörsen gilt die Software *Napster*, die 1998 entwickelt wurde. Benötigte Napster noch einen zentralen Index für die verfügbaren Dateien, konnten spätere Systeme wie *Gnutella* ganz auf zentrale Komponenten verzichten.

Bei den eingesetzten P2P-Netzen handelte es sich oft um *unstrukturierte* P2P-Netze. In diesen Netzen sind Verbindungen zwischen den Teilnehmern zufällig. Es wird keine definierte Netztopologie angestrebt. Die Suche nach Ressourcen erfolgt in der Regel durch *begrenztes Fluten* oder *Random Walk*-Ansätze. Unstrukturierte P2P-Netze eignen sich gut zur Nutzung häufiger Ressourcen. Die Suche nach selten im Netz verfügbaren Ressourcen oder die Suche nach spezifischen Teilnehmern des Netzes scheitert in der Regel an der für ein vollständiges Durchsuchen des Netzes benötigten Zeit und dem dadurch verursachten Kommunikationsoverhead.

Wurden in frühen unstrukturierten P2P-Protokollen wie Gnutella alle Knoten als gleich betrachtet, differenzierten spätere Protokolle wie KaZaA zwischen normalen und besonders leistungsfähigen Knoten [29]. Diese leistungsfähigen Knoten wurden

mit zusätzlichen Aufgaben belegt. Diese Art von Knoten, die spezielle zusätzliche Aufgaben in P2P-Netzen übernehmen, nennt man *Supernode*.

Aufbauend auf dem Konzept der Supernodes wurden auch Protokolle wie Gia [29] entwickelt, die eine graduelle Aufteilung der Last entsprechend der Ressourcen eines Knotens erlauben.

2.2.3 Strukturierte P2P-Netze

In strukturierten P2P-Netzen werden die Verbindungen zwischen den Netzknoten so erstellt, dass eine spezifische Topologie entsteht. Diese Topologien ermöglichen das effiziente Lokalisieren von Teilnehmern und Ressourcen im P2P-Netz.

Hierzu wählen Teilnehmer eines strukturierten P2P-Netzes einen eindeutigen Schlüssel als Identifikator. Als Schlüsselraum dient die zyklische Gruppe $(\mathbb{Z}/n\mathbb{Z}, +)$ mit $n := 2^l$ für eine gegebene Schlüssellänge l. Die Netzknoten lassen sich so anhand ihrer Schlüssel in einem Ring anordnen.

Auf diesem Ring lässt sich nun ein *schlüsselbasiertes Routing* (englisch: Key Based Routing, kurz: KBR) definieren. Jedem Knoten wird dazu, abhängig von seinem Schlüssel, ein Teil des Schlüsselraumes zugeordnet. So kann beispielsweise jeder Knoten für die Menge an Schlüsseln verantwortlich sein, die seinem Schlüssel am nächsten sind. Alternativ kann auch jedem Knoten der Bereich des Schlüsselraumes zwischen seinem Schlüssel und dem Schlüssel des nachfolgenden Knoten zugeordnet werden. Beim schlüsselbasierten Routing kann nun eine Nachricht an einen beliebigen Schlüssel versendet werden. Sie wird dann dem Knoten zugestellt, der für den entsprechenden Teil des Schlüsselraumes zuständig ist.

Durch geschickte Auswahl der Verbindungen zwischen Knoten kann gewährleistet werden, dass eine Nachricht bei n Teilnehmern des P2P-Netzes in $O(\log n)$ Routingschritten zugestellt werden kann. Im KBR-Protokoll *Chord* [130] hält beispielsweise jeder Knoten, neben einer Verbindung zu Vor- und Nachfolgeknoten im Schlüsselraum, bei einer Schlüssellänge von l eine Verbindung zu maximal l weiteren Knoten. Diese wählt er aus, indem er für alle Zahlen i zwischen 1 und l zu seinem eigenen Schlüssel 2^i hinzuaddiert und eine Verbindung zu dem Knoten aufbaut, dem der resultierende Schlüssel zugeordnet ist. In Abbildung 2.4 sind die aus dieser Verbindungswahl resultierenden Verbindungen eines Teilnehmers mit dem Schlüssel 1 in einem Chord-Netz der Schlüssellänge 4 bit, bestehend aus den Teilnehmern mit den Schlüsseln 0, 1, 2, 3, 5, 6, 8, 9, 11, 12, 13 und 14, dargestellt.

In einem solchen Netz werden Nachrichten weitergeleitet, indem jeder Teilnehmer die Nachricht an denjenigen Teilnehmer weiterleitet, dessen Schlüssel dem Zielschlüssel der Nachricht nach der von dem jeweiligen Protokoll verwendeten Metrik am nächsten ist. Dies ist in Abbildung 2.5 im Beispiel dargestellt. Hier soll eine Nachricht von Teilnehmer 1 des oben gezeigten Chord-Netzes der Schlüssellänge 4 bit an den Schlüssel 12 versendet werden. Teilnehmer 1 hat aufgrund der oben beschriebenen Auswahl der Verbindungen eine Verbindung zu Teilnehmer 9. Dieser wiederum leitet die Nachricht an Teilnehmer 11 weiter, der sie an das Ziel, Teilnehmer 12, weiterleitet.

Neben dem hier beschriebenen Chord gibt es viele weitere KBR-Protokolle, beispielsweise Pastry [121], Koorde [73], Kademlia [96], Bamboo [115] und Broose [55], die sich in der Overlay-Topologie und den Stabilisierungsverfahren unterscheiden.

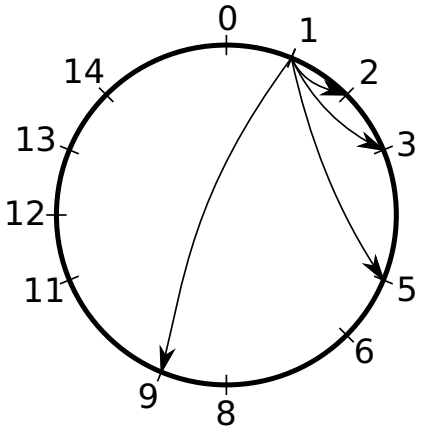


Abbildung 2.4 Schematische Darstellung der Verbindungen eines Mitglieds eines Chord-Netzes.

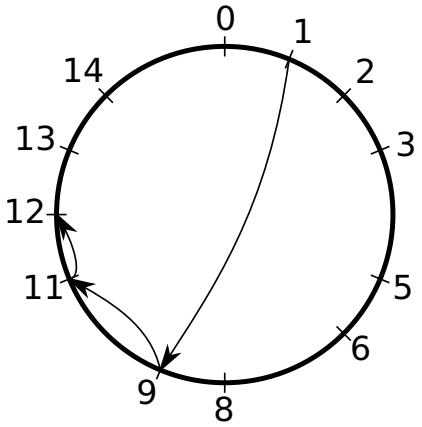


Abbildung 2.5 Weiterleitung einer Nachricht in Chord.

Ein wichtiger Dienst, der mit schlüsselbasiertem Routing realisiert werden kann, ist die *verteilte Hashtabelle* (englisch: Distributed Hash Table, kurz: DHT). Mit diesem Dienst können Daten in einem P2P-Netz gespeichert werden. Dazu werden die Daten mit einem Schlüssel versehen. Die Daten werden dann als Schlüssel-Wert-Paar von dem Knoten gespeichert, dem der Schlüssel zugeordnet ist. Durch Replikation der Daten kann die Zuverlässigkeit der Speicherung bei Knotenausfällen erhöht werden.

2.2.4 Weitere Anwendungen für P2P-Netze

Eine weitere wichtige Anwendung für P2P-Netze ist *Application Layer Multicast* (kurz: ALM). ALM ermöglicht das Versenden von Nachrichten an Gruppen von Teilnehmern. Der Dienst ähnelt dem Multicast auf IP-Schicht. IP-Multicast ist jedoch für die meisten Nutzer nicht zugänglich und in der Regel nicht netzübergreifend möglich. Durch Anbieten eines Multicast-Dienst über ein P2P-Protokoll kann der Dienst allen Netznutzern ermöglicht werden.

Das Kernproblem eines ALM-Protokolls ist es, die Teilnehmer so zu verbinden, dass eine effiziente Weiterleitung der Nachrichten an alle Mitglieder einer Multicast-Gruppe gewährleistet ist. Hierbei gibt es verschiedene Lösungsansätze. Eine gängige Methode ist das Verbinden der Teilnehmer in einer Baumstruktur. Um das Aufbauen

einer Baumstruktur zu erleichtern, kann als Basis ein KBR-Protokoll verwendet werden. Diesen Ansatz wählt beispielsweise das Protokoll Scribe [26], welches auf dem KBR-Protokoll Pastry [121] aufsetzt.

Andere ALM-Protokolle wie Nice [9] bilden eine Baumstruktur, ohne auf ein anderes Protokoll aufzusetzen. Anstatt einer Baumstruktur können die Teilnehmer eines ALM-Protokolls auch in einer anderen Struktur, wie beispielsweise einem Mesh, verbunden werden. Das Protokoll Narada [34] setzt beispielsweise ein Mesh ein, um einen *central point of failure* zu vermeiden.

Eine weitere Anwendungsmöglichkeit für P2P-Protokolle sind virtuelle Welten. In diesen kann durch Verwendung des P2P-Protokolls auf einen zentralen Server verzichtet werden. In den folgenden Kapiteln wird genauer auf P2P-Protokolle für virtuelle Welten eingegangen.

2.3 Ausgewählte Aspekte der Kryptographie

Für einige Elemente der in Kapitel 7 vorgestellten Cheating-Erkennung werden kryptographische Funktionen verwendet. Diese werden im Folgenden kurz vorgestellt.

2.3.1 Hash-Funktionen

Kryptographische Hash-Funktionen bilden beliebige Zeichenfolgen auf Zeichenfolgen einer festen Länge ab. Dabei handelt es sich um *Einwegfunktionen*. Das bedeutet, dass es mit realistischem Aufwand für eine Hashfunktion H unmöglich ist, zu einem gegebenen Ausgabewert y einen Eingabewert x zu finden, so dass $H(x) = y$ gilt. Es ist also nicht möglich, aus dem Ausgabewert mit realistischem Aufwand Information über den Eingabewert zu gewinnen. Weiterhin wird für Hashfunktionen *schwache Kollisionsresistenz* gefordert. Dies bedeutet, dass es für eine gegebene Eingabe x mit realistischem Aufwand nicht möglich ist eine Eingabe $x^{'}$ zu finden, für die $H(x) = H(x^{'})$ gilt.

Zusätzlich lässt sich eine *starke Kollisionsresistenz* fordern. Bei starker Kollisionsresistenz lassen sich mit realistischem Aufwand keine zwei verschiedenen Eingabewerte x und $x^{'}$ konstruieren, die denselben Ausgabewert liefern. Bei der starken Kollisionsresistenz können also beide Eingabewerte frei gewählt werden, während bei der schwachen Kollisionsresistenz ein Eingabewert fest vorgegeben ist. Ein Beispiel für eine solche Hashfunktion ist das von dem amerikanischen „National Institute of Standards and Technology“ standardisierte *SHA-1* [45].

Mit kryptographischen Hash-Funktionen lässt sich die Integrität von Daten belegen. Eine weitere Anwendung ist, wie in Abschnitt 2.3.3 beschrieben, die digitale Signatur.

2.3.2 Public-Key-Kryptographie

In klassischen Kryptosystemen wird zur Ver- und Entschlüsselung einer Nachricht derselbe Schlüssel verwendet. Dies ermöglicht zwar die sichere Kommunikation zwischen zwei Kommunikationspartnern, ist allerdings in der Kommunikation von Gruppen ineffizient. Entweder müssen zwischen allen Teilnehmern der Gruppe paarweise Schlüssel ausgetauscht werden, was in einem quadratischen Aufwand für die Schlüsselerzeugung und Speicherung resultiert, oder alle Teilnehmer der Gruppe verwenden

einen gemeinsamen Schlüssel, was das Entfernen von Teilnehmern aus der Gruppe ohne Schlüsselwechsel unmöglich macht.

In Public-Key-Kryptosystemen existieren hingegen für jeden Teilnehmer zwei Schlüssel, ein öffentlicher und ein privater Schlüssel. Dabei lässt sich aus dem öffentlichen Schlüssel der private Schlüssel nicht mit realistischem Aufwand errechnen. Wird eine Nachricht mit dem öffentlichen Schlüssel verschlüsselt, kann sie nur mit dem dazugehörigen privaten Schlüssel wieder entschlüsselt werden. Umgekehrt lassen sich Nachrichten, die mit dem privaten Schlüssel verschlüsselt sind, mit dem öffentlichen Schlüssel entschlüsseln. Dies wird für die in Abschnitt 2.3.3 vorgestellten Signaturen verwendet.

Mittels Public-Key-Kryptographie lässt sich so eine effiziente verschlüsselte Kommunikation zwischen vielen Teilnehmern realisieren. Hierzu verteilt jeder Teilnehmer seinen öffentlichen Schlüssel an die übrigen Teilnehmer. Ein Teilnehmer kann so Nachrichten mit den öffentlichen Schlüsseln der gewünschten Empfänger verschlüsseln. Diese sind mit ihren privaten Schlüsseln in der Lage, die Nachricht zu entschlüsseln. Im Gegensatz zu klassischen symmetrischen Kryptosystemen ist hier für die Speicherung der Schlüssel nur linearer Aufwand nötig.

Die Grundideen einer asymmetrischen Verschlüsselung wurden 1976 von Diffie und Hellman publiziert [41]. 1978 entstand daraus das auch heute noch viel verwendete Krytosystem RSA von Rivest et al. [118].

2.3.3 Signaturen

Mittels Public-Key-Kryptographie und kryptographischen Hash-Funktionen lassen sich sichere digitale Signaturen konstruieren. Diese können die Authentizität und die Integrität einer Nachricht beweisen.

Hierzu wird von einer Nachricht ein Hashwert erzeugt und mit dem privaten Schlüssel des Absenders verschlüsselt. Der verschlüsselte Hashwert wird zusammen mit der Nachricht an den Empfänger versendet. Dieser kann nun den Hashwert mit dem vorab bekannt gemachten öffentlichen Schlüssel des Absenders entschlüsseln. Daraufhin kann er selber einen Hashwert der Nachricht ermitteln und mit dem entschlüsselten Wert vergleichen. Stimmen die Werte überein, kann er sicher sein, dass die Nachricht nach der Signierung nicht geändert worden ist. Weiterhin ist sichergestellt, dass der Absender der Nachricht über den zu dem öffentlichen Schlüssel gehörigen privaten Schlüssel verfügt. Ein solches Signaturverfahren wird beispielsweise von dem Kryptosystem RSA [118] bereitgestellt.

3. Stand der Technik

Virtuelle Welten sind Gegenstand aktiver Forschung. Gleichzeitig werden sie von einigen kommerziellen Anbietern im Internet angeboten. In diesem Kapitel wird der Stand der aktuellen Forschung und der Stand der Technik bei kommerziellen virtuellen Welten dargestellt.

In Abbildung 3.1 sind die in kommerziellen virtuellen Welten verwendeten und in der Forschung betrachteten Technologien für virtuelle Welten dargestellt. Zunächst lassen sich Client/Server-Technologien und P2P-Protokolle unterscheiden. Client/Server-Technologien lassen sich in Technologien zum Verteilen von Teilnehmern auf Server – Sharding, Instanzen und Zonen –, Proxy-Architekturen und Cloud Computing unterteilen. P2P-Protokolle lassen sich in vollvermaschte Protokolle, zonenbasierte Protokolle und zonenlose Protokolle unterteilen. Bei zonenbasierten Protokollen kann zwischen ALM-basierten Protokollen, Supernode-basierten Protokollen und Protokollen mit dynamischen Zonen unterschieden werden. Ein Sonderfall sind hybride Protokolle, die Client/Server-Technologien mit Eigenschaften von P2P-Protokollen verbinden.

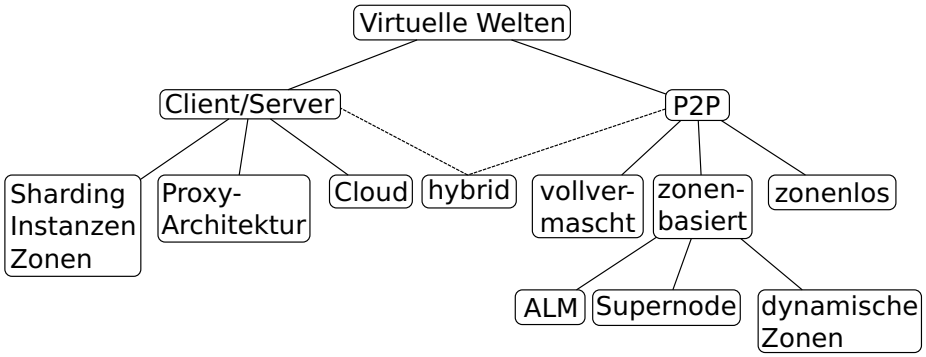


Abbildung 3.1 Technologien für virtuelle Welten.

3.1 Client/Server

Alle momentan auf dem Markt befindlichen kommerziellen virtuellen Welten basieren auf dem Client/Server-Prinzip. Dabei wird die virtuelle Welt von zentralen Servern oder Serverclustern bereitgestellt. Die Teilnehmer fungieren als Clients, welche die Ressourcen der Server nutzen. Das Hauptproblem dieses Ansatzes ist die Skalierbarkeit. Steigt die Teilnehmerzahl der virtuellen Welt über eine gewisse Grenze, sind die Ressourcen der Server ausgelastet. Dies hat zur Folge, dass bei Eintreffen weiterer Teilnehmer die Server überlastet werden und das Spiel-Erlebnis aller Teilnehmer leidet. Alternativ können Warteschlangen eingeführt werden, sodass weitere Teilnehmer die virtuelle Welt nur betreten können, wenn andere Teilnehmer die Welt verlassen. In den folgenden Abschnitten werden einige Ansätze vorgestellt, mit denen Betreiber virtueller Welten versuchen, das Skalierungsproblem zu mindern.

3.1.1 Technologien zum Verteilen von Teilnehmern

Die ersten verfügbaren virtuellen Welten wie „Habitat“ [99] verwendeten einen zentralen Server oder Servercluster, der nur einer festen Teilnehmerzahl das Spielen ermöglichte. Um einer größeren Zahl an Personen die Teilnahme zu ermöglichen, führten die Betreiber späterer virtueller Welten die Aufteilung der Teilnehmerbasis auf verschiedene Server oder Servercluster ein. In den folgenden Abschnitten sollen die drei wesentlichen Möglichkeiten einer solchen Aufteilung – Sharding, Instanzen und Zonen – kurz vorgestellt werden.

3.1.1.1 Sharding

Die einfachste Form der Aufteilung von Teilnehmern auf verschiedene Server stellen die sogenannten *Shards* oder auch *Realms* dar. Beides bezeichnet das Aufteilen der virtuellen Welt in mehrere initial identische Kopien. Jeder Teilnehmer kann zu Beginn wählen, welche Kopie der Welt er nutzen möchte. Steigt die Teilnehmerzahl, wird eine neue Kopie angelegt. Durch das Hinzufügen beliebig vieler Kopien können so prinzipiell beliebig viele Teilnehmer unterstützt werden. Dabei müssen allerdings für jede neue Kopie auch neue Server bereitgestellt werden.

Da die einzelnen Kopien vollständig voneinander unabhängig sind, ergeben sich durch diese Technik aber auch weitere Implikationen für die virtuelle Welt. So können sich verschiedene Kopien im Laufe der Zeit unterschiedlich entwickeln. Die Teilnehmer können jedoch stets nur die Entwicklung ihrer jeweiligen Kopie verfolgen. Außerdem haben die Teilnehmer unterschiedlicher Kopien keine Möglichkeit, miteinander zu interagieren. Von den hier vorgstellten Techniken ist Sharding mit den größten Einschränkungen für die Teilnehmer verbunden.

Sharding wird jedoch aufgrund seiner einfachen Realisierbarkeit von einem Großteil der heute verfügbaren virtuellen Welten, insbesondere bei MMOGs, eingesetzt. Dabei wird teilweise versucht, Teilnehmern den Wechsel zwischen verschiedenen Kopien zu ermöglichen, damit beispielsweise auf eine Kopie gewechselt werden kann, die von befreundeten Teilnehmern genutzt wird. Da dies ein relativ aufwändiger Vorgang ist, ist so ein Wechsel in der Regel nur beschränkt oft möglich und teilweise mit Zusatzkosten für den Teilnehmer verbunden [124].

3.1.1.2 Instanzen

Instanzen sind eine dem Sharding verwandte Technik, um Teilnehmer einer virtuellen Welt auf mehrere Server zu verteilen. Instanziierung arbeitet jedoch nicht mit statischen Kopien wie Sharding. Stattdessen werden die Kopien dynamisch angelegt. Weiterhin wird nicht die gesamte virtuelle Welt, sondern nur einzelne Teile kopiert.

In MMOGs ist es üblich, mit Instanzen kleinere fest definierte Gebiete der virtuellen Welt mit meist besonders starken Gegnern abzuteilen. Für jede Spielergruppe, die ein solches Gebiet betritt, wird eine neue Kopie erstellt. Verlässt der letzte Spieler die instanziierte Kopie, wird sie wieder gelöscht.

Da diese für Instanzen genutzten Gebiete mit starken Gegnern von einem verhältnismäßig großen Teil der Spieler genutzt werden, kann somit ein großer Teil der Last auf Instanz-Server verteilt werden. Diese Art der Instanziierung ist weit weniger invasiv als Sharding, da Teilnehmer nach Verlassen des Gebietes wieder mit allen anderen Teilnehmern interagieren können.

Eine Variante der Instanziierung sind transparente Instanzen. Hier werden bestimmte Gebiete der virtuellen Welt kopiert, wenn die Teilnehmerzahl in dem betreffenden Gebiet einen bestimmten Schwellenwert überschreitet. Dies ermöglicht eine Lastverteilung bei besonders stark frequentierten Bereichen der virtuellen Welt. Dabei können sich für die Teilnehmer jedoch Unannehmlichkeiten ergeben: Wollen sie sich mit anderen Teilnehmern an einem bestimmten Ort der virtuellen Welt treffen, werden vom Server jedoch auf verschiedene Instanzen aufgeteilt, können sie sich nicht sehen, obwohl sie sich eigentlich an derselben Stelle der virtuellen Welt befinden. Einige virtuelle Welten wie „Age of Conan“ [54] oder „Guild Wars“ [101], die diese Art der Instanziierung verwenden, ermöglichen es Teilnehmern deshalb, aktiv in eine andere Instanz zu wechseln.

Grundsätzlich ist die hier beschriebene Technik auf Gebiete beschränkt, die keine grundsätzlichen Veränderungen erfahren. Da die Kopien nur temporär angelegt werden und einzelne Teilnehmer zu verschiedenen Zeiten verschiedene Kopien betreten können, kann ein konsistentes Spielerlebnis nur gewährleistet werden, wenn die Kopien im Wesentlichen identisch sind. Aus diesem Grund wird bei sozialen virtuellen Welten eine Instantiierung in der Regel nicht eingesetzt.

3.1.1.3 Zonen

Eine weitere Möglichkeit der Verteilung von Teilnehmern auf mehrere Server ist die Unterteilung der virtuellen Welt in *Zonen*. Jeder Zone kann ein eigener Server zugewiesen werden, der für den Spielfluss in dieser Zone zuständig ist. Bei einer einfachen Implementierung muss ein Teilnehmer beim Überschreiten der Zonengrenze aktiv eine Verbindung mit dem für die neue Zone zuständigen Server aufbauen, was in der Regel mit einer gewissen Wartezeit verbunden ist. Ein nahtloser Zonenübergang ist nicht möglich. In der Praxis wird die Wartezeit entweder mit einem Ladebildschirm oder einer vorgefertigten Videosequenz überbrückt. Besteht die virtuelle Welt aus mehreren Planeten, Kontinenten oder Inseln, ist eine solche Zonenaufteilung üblich [152].

Eine transparente Zonenaufteilung ist möglich, wenn der Teilnehmer bereits bei Annäherung an die Zonengrenze eine Verbindung mit dem für die neue Zone zuständigen Server aufbaut. Erfolgt der Verbindungsaufbau, bevor Details der neuen Zone

für den Teilnehmer sichtbar werden, ist der Zonenübertritt für den Teilnehmer nicht mehr wahrnehmbar.

Bisherige virtuelle Welten arbeiten in der Regel mit statischen, vorab definierten Zonen. Eine dynamisch Anpassung der Zonen an die Lastsituation ist somit nicht möglich. Es kann also vorkommen, dass der für eine Zone zuständige Server ausgelastet oder überlastet ist, während die Server der anderen Zonen freie Ressourcen zur Verfügung haben [152]. Eine dynamische Aufteilung der Spielwelt, abhängig von der Verteilung der Teilnehmer in der virtuellen Welt, ist prinzipiell möglich [3, 78, 91, 117]. Entsprechende Forschungsergebnisse sind bislang jedoch nicht in kommerzielle Produkte eingeflossen.

3.1.2 Proxy-Architektur

Eine Erweiterung des Zonenkonzeptes ist eine *Proxy-Architektur*, bei der Proxies dynamisch für eine gewisse Zahl an Teilnehmern zuständig sind. Dabei wird für jeden Teilnehmer ein Einflussbereich berechnet und Ereignisse im Überschneidungsgebiet von Einflussbereichen betrachtet. Greifen alle Proxies auf eine gemeinsame Datenbank zu, können viele Teilnehmer gleichzeitig in einer nicht instanziierten Welt interagieren [13, 61]. Diese Architektur wird von dem MMOG „EVE Online“ verwendet [49]. Damit ist EVE Online inoffiziellen Statistiken nach in der Lage, mehr als 60.000 gleichzeitig aktive Spieler in einer Welt zu unterstützen [27, 104].

Einige Forschungsprojekte wie DoIT [65] oder ALVIC-NG [112] arbeiten an Middleware für virtuelle Welten, die ebenfalls eine Proxy-Architektur unterstützen. Auch von kommerziellen Anbietern werden Middleware-Lösungen mit Unterstützung einer solchen Architektur, wie Bigworld [20] oder das nicht mehr weitergeführte „Project Darkstar“ von Sun Microsystems [24], entwickelt.

3.1.3 Cloud-Computing

Cloud-Computing [6] ist ein aktives Forschungsfeld, das bereits in viele kommerzielle Anwendungen Einzug gehalten hat. Ein Vorteil der „Cloud“ ist es insbesondere, Rechenkapazitäten schnell und bedarfsangepasst bereitstellen zu können. Damit kann für Anbieter virtueller Welten der initiale Aufbau der nötigen Hardwareinfrastruktur und bei Wachstum der Teilnehmerzahl das Ausbauen der Kapazität vereinfacht werden. Weiterhin können Lastspitzen durch das kurzfristige Anmieten zusätzlicher Ressourcen aufgefangen werden. Wird Cloud-Computing mit dem dynamischen Aufteilen der Spielwelt verbunden, lassen sich Latenzen wie mit klassischen Client/Server-Systemen erreichen [132].

Dabei bleiben aber Aufwand und Kosten für die Grundlast, also die für die durchschnittliche Teilnehmerzahl benötigte Kapazität, gegenüber klassischen Client/Server-Systemen unverändert. Die mindestens lineare Abhängigkeit zwischen Teilnehmerzahl und benötigten Ressourcen kann durch Cloud-Computing nicht verbessert werden. Insbesondere bei virtuellen Welten mit global verteilten Teilnehmern, bei denen nur geringe tageszeitlichen Schwankungen der Teilnehmerzahlen auftreten, sind die Einsparmöglichkeiten somit gering.

3.1.4 Bewertung

Heute verfügbare kommerzielle virtuelle Welten nutzen in der Regel eine Kombination der Techniken Sharding, Instanziierung und Zonen, um die Skalierbarkeit der Welt zu verbessern. Das MMOG „World of Warcraft“ nutzt beispielsweise alle drei Techniken. Eine Ausnahme ist das MMOG „EVE Online“, das eine komplexe Proxy-Architektur einsetzt, um allen Teilnehmern eine gemeinsame, nicht aufgeteilte Welt bieten zu können. Cloud-Computing als relativ neues Konzept hat noch keinen Eingang in kommerzielle virtuelle Welten gefunden, kann es aber ermöglichen, die Hardwareinfrastruktur für Client/Server-basierte virtuelle Welten zu flexibilisieren.

Mit allen Ansätzen können prinzipiell beliebig viele Teilnehmer in der virtuellen Welt unterstützt werden. Dabei werden jedoch proportional zur Teilnehmerzahl mehr Server benötigt. Die maximale Teilnehmerzahl bleibt also stets durch die Ressourcen der verwendeten Server beschränkt. Das grundsätzliche Skalierungsproblem wird mithin nicht gelöst.

Weiterhin verursacht die für den Betrieb einer Client/Server-basierten virtuellen Welt benötigte Infrastruktur, also Kauf und Betrieb der benötigten Server sowie die benötigte Bandbreite, hohe Kosten. Das Client/Server-Prinzip ist somit für den Betrieb skalierbarer virtueller Welten nur bedingt geeignet.

3.2 P2P

Der Einsatz von P2P-Protokollen in virtuellen Welten ist bislang selten. Nur wenige virtuelle Welten meist nichtkommerzieller Anbieter verwenden diese Protokolle. Aufgrund der prinzipiell besseren Skalierungseigenschaften sind P2P-Protokolle für virtuelle Welten jedoch ein aktives Forschungsthema.

3.2.1 Klassifizierung von P2P-Protokollen für virtuelle Welten

Bisherige Protokolle für virtuelle Welten lassen sich zunächst in drei grundlegende Klassen einteilen. Die einfachsten Protokolle basieren auf einer *Vollvermaschung* der Teilnehmer. Sie zeigen einen quadratischen Aufwand bezüglich der Teilnehmerzahl, skalieren also nicht. Um den Aufwand zu senken, werden zwei Ansätze gewählt: *Zonenbasierte Protokolle* teilen die Spielwelt in verschiedene Zonen auf. Teilnehmer erhalten nur noch Nachrichten derjenigen anderen Teilnehmer, die sich in derselben Zone befinden. Bei *zonenlosen* Protokollen wird die Verteilung der Nachrichten über das Interessengebiet der Teilnehmer gesteuert.

Zonenbasierte Protokolle lassen sich weiter in drei Unterkategorien einteilen: In *Application Layer Multicast*- und *Supernode*-basierten Protokollen ist die Zonenaufteilung statisch. Die Verteilung der Nachrichten innerhalb der Zonen erfolgt dann über Application Layer Multicast beziehungsweise einen Supernode. Weiterhin gibt es Protokolle mit *dynamischen Zonen*. Hier wird die Zonengröße der Lastsituation angepasst. Innerhalb der Zonen kann dann die Nachrichtenverteilung über eine Vollvermaschung der Teilnehmer in der Zone geschehen. Weiterhin gibt es *hybride Protokolle*. Diese kombinieren einen zentralen Server mit P2P-Elementen.

Im Folgenden werden die Kategorien näher erläutert und beispielhaft einige Vertreter der Kategorien vorgestellt.

3.2.2 Vollvermaschte Protokolle

Die einfachste Realisierung von P2P für virtuelle Welten ist die Vollvermaschung aller Teilnehmer der virtuellen Welt. Jeder Teilnehmer sendet alle seine Ereignisnachrichten und Positionsmeldungen an alle anderen Teilnehmer [107]. Diese Art P2P-Protokoll wird beispielsweise von der virtuellen Welt „Open Cobalt“ [94] genutzt.

Da die Zahl der Verbindungen und damit die Zahl der Nachrichten in einem solchen Protokoll quadratisch mit der Teilnehmerzahl steigt, die Ressourcen jedoch nur linear, skaliert dieser Ansatz nicht. Es können nur wenige Teilnehmer in einer gemeinsamen Welt unterstützt werden. Durch eine Aufteilung der virtuellen Welt in Shards, wie sie auch bei Client/Server-basierten virtuellen Welten üblich ist, kann eine prinzipiell beliebige Zahl an Teilnehmern unterstützt werden. Da jedoch jeder Shard nur wenige Teilnehmer aufnehmen kann, sind die Interaktionsmöglichkeiten der Teilnehmer deutlich eingeschränkt. Vollvermaschte P2P-Protokolle sind also nicht für skalierbare virtuelle Welten geeignet.

3.2.3 Zonenbasierte Protokolle

Viele P2P-Protokolle für virtuelle Welten basieren auf einer Aufteilung der virtuellen Welt in Zonen. Diese sind vergleichbar mit den im Abschnitt 3.1.1.3 beschriebenen Zonen für Client/Server-basierte virtuelle Welten. Alle Teilnehmer, die Mitglied einer Zone sind, erhalten die Ereignisnachrichten und Positionsmitteilungen aller anderen Zonenmitglieder. Der Ressourcenbedarf steigt hier – zuzüglich eventuellem zusätzlichem Protokolloverhead – quadratisch mit der Zahl der Zonenmitglieder. Sind die Zonen so klein gewählt, dass nicht zu viele Teilnehmer Mitglied derselben Zone sind, werden die Skalierungsprobleme vollvermaschter Protokolle vermieden.

In der Regel können bei den vorgeschlagenen zonenbasierten Protokollen Teilnehmer Mitglied in mehreren Zonen werden, sodass ein transparenter Zonenwechsel möglich ist.

Die Verteilung der Nachrichten innerhalb der Zone ist protokollspezifisch. Grundsätzliche Möglichkeiten der Nachrichtenverteilung sind hier die Verwendung von *Application Layer Multicast*, die Verteilung durch *Supernodes* oder, in Verbindung mit *dynamischen Zonengrößen*, die Vollvermaschung innerhalb der Zone. Diese Möglichkeiten werden in den folgenden Abschnitten genauer erläutert.

3.2.3.1 ALM-basierte Protokolle

In ALM-basierten Protokollen wird die Nachrichtenverteilung innerhalb der Zonen durch ein Application Layer Multicast-Protokoll übernommen. Hierzu werden alle Zonenmitglieder Mitglied in einer Multicastgruppe. Prinzipiell kann jedes Multicast-Protokoll, das das Publizieren von Multicastnachrichten durch beliebige Gruppenmitglieder zulässt, für die Nachrichtenverteilung eingesetzt werden.

Das Protokoll SimMUD [79] setzt beispielsweise das auf dem KBR-Protokoll Pastry [121] basierende ALM-Protokoll Scribe [26] ein.[1] Skype4Games [137] setzt die durch das P2P-Voice-over-IP-Protokoll Skype [12] bereitgestellte Gruppenkommunikation

[1] Das Protokoll SimMUD wird in Abschnitt 5.1.1 genauer vorgestellt.

ein. Ahmed et al. [4] schlagen ein um Clustering erweitertes ALM für virtuelle Welten vor.

Ein Problem bei der Verwendung von ALM für die Nachrichtenverteilung ist, dass bei ALM-Protokollen eine Nachricht teilweise über viele Zwischenknoten geleitet wird, bis sie alle Empfänger erreicht hat. Dadurch ist es möglich, dass ALM-basierte Protokolle die von virtuellen Welten benötigten geringen Nachrichtenverzögerungen überschreiten.

Ein weiteres mögliches Problem ist, dass viele ALM-Protokolle keinen zuverlässigen Nachrichtenversand garantieren können. Gehen bei der Weiterleitung zu viele Nachrichten verloren, kann die von virtuellen Welten benötigte Nachbarschaftskenntnis nicht gewährleistet werden.

3.2.3.2 Supernode-basierte Protokolle

In Supernode-basierten Protokollen wird die Nachrichtenverteilung in einer Zone von einem Supernode übernommen. Dieser koordiniert in der Regel auch die Zonenmitgliedschaften und übernimmt sämtliche Verwaltungsaufgaben, die für die ihm zugewiesene Zone anfallen.

In einer einfachen Implementierung des Supernode-Prinzips werden alle Nachrichten von den Zonenmitgliedern an den Supernode versendet. Dieser wiederum leitet die Nachrichten direkt an alle Zonenmitglieder weiter. Dieser Ansatz wird beispielsweise vom Zoning Layer-Protokoll von Iimura [68] oder MOPAR von Yu und Vuong [154] verwendet. Die Zuordnung von Supernodes zu Zonen wird dort von einer DHT übernommen, in der auch der Zustand der Zone als Backup gespeichert wird. Bei dem ansonsten vergleichbaren Protokoll von Lee et al. [89] wird die Zonenaufteilung durch eine Baumstruktur verwaltet. Fiedler et al. [51] schlagen ein Publish-Subscribe-basiertes Protokoll vor. In allen diesen Ansätzen erfüllt der Supernode nahezu sämtliche Aufgaben, die für eine Zone einer Client/Server-basierten virtuellen Welt von einem Server übernommen werden[2]. Aus diesem Grund sind vergleichbare Protokolle anfällig für eine Überlastung der Supernodes. Befinden sich in einer Zone zu viele Teilnehmer, kann der Supernode seine Aufgaben nicht mehr voll erfüllen. Weitere Protokolle, die diesen Ansatz verfolgen, sind beispielsweise Mediator [48] und Solipsis2 [53], das Nachfolgeprotokoll des in Abschnitt 3.2.4 besprochenen zonenlosen Protokolls Solipsis.

Die Überlastung der Supernodes kann mit geeigneten Lastverteilmechanismen vermieden werden. Das von Yamamoto et. al 2005 vorgeschlagene Publish-Subscribe-basierte Protokoll [149] setzt hierfür einen Lastverteil-Baum ein, der vom Supernode gebildet wird, wenn die Zahl der Teilnehmer in der von ihm verwalteten Zone zu groß wird. In diesem Protokollvorschlag wird er dabei zusätzlich von einem zentralen Server unterstützt.[3]

Durch diese Lastverteilmechanismen können Supernode-basierte Protokolle dieselbe Skalierbarkeit erreichen wie ALM-basierte Protokolle. Allerdings erzeugen die Lastverteilmechanismen in der Regel eine zusätzliche Nachrichtenverzögerung, die gegebenenfalls über der für virtuelle Welten erforderlichen maximalen Nachrichtenverzögerung liegt.

[2] Vergleiche auch Abschnitt 3.1.1.3.

[3] Eine genauere Beschreibung des Protokolls findet sich in Abschnitt 5.1.2.

Ebenfalls auf Publish-Subscribe basiert das Protokoll Mercury von Bharambe et al. [19]. Bei diesem Protokoll kann das Wechseln einer Zone jedoch bei n Teilnehmern bis zu $O(n)$ Weiterleitungsschritte benötigen, sodass es nicht für große virtuelle Welten geeignet ist.

3.2.3.3 Ansätze mit dynamischen Zonen

Werden die Zonen nicht statisch aufgeteilt, sondern je nach Teilnehmerzahl in einer Region dynamisch gebildet, kann auf eine weitere Lastverteilung verzichtet werden. Da durch die dynamische Aufteilung sichergestellt werden kann, dass die Zahl der Teilnehmer einer Zone eine gewisse Zahl nicht überschreitet, können die Teilnehmer in einer Zone vollvermascht werden. Diesen Ansatz wählt beispielsweise das Protokoll N-Tree [58].[4] Jeder Zone ist ein Supernode zugeteilt, der die Mitglieder in dieser Zone verwaltet. Ein vergleichbarer Ansatz wird auch von Cover [98] gewählt.

Die Herausforderung bei Protokollen mit dynamischen Zonen ist es, die aktuelle Zonenaufteilung zu verwalten und das Wissen über die Aufteilung allen Teilnehmern konsistent zur Verfügung zu stellen. Gelingt dies nicht, kommt es bei Zonenaufteilungen, -zusammenlegungen oder -wechseln zu Fehlern bei der Zuordnung von Teilnehmern zu Zonen. Dadurch wird die Nachbarschaftskenntnis in der virtuellen Welt gefährdet.

Aufgrund dieser Schwierigkeit gibt es neben N-Tree und Cover nur wenige Vorschläge für P2P-Protokolle mit dynamischen Zonen für virtuelle Welten. Meist werden dynamische Zonen in Verbindung mit Client/Server-basierten Protokollen, oder für hybride P2P-Protokolle, bei denen viele Aufgaben von einem zentralen Server übernommen werden, vorgeschlagen.

3.2.4 Zonenlose Protokolle

In zonenlosen Protokollen findet keine Unterteilung der Spielwelt statt. Stattdessen werden die Verbindungen zwischen den Teilnehmern so aufgebaut, dass alle Ereignisnachrichten und Positionsmeldungen direkt vom Verursacher an alle interessierten Teilnehmer versendet werden. Hierzu wird das Interessengebiet der Teilnehmer betrachtet. Dies ist das Gebiet um die Position eines Teilnehmers, welches alle anderen Teilnehmer umfasst, mit denen der Teilnehmer interagieren kann. Teilnehmer, die sich im Interessengebiet befinden oder zu denen aus anderen Gründen eine Verbindung besteht, werden in zonenlosen Protokollen als *Nachbarn* bezeichnet.

Die Herausforderung bei zonenlosen Protokollen ist, dass alle Teilnehmer darüber informiert werden müssen, wenn ein anderer Teilnehmer in ihr Interessengebiet eintritt. Dies geschieht durch Benachrichtigungen durch die Nachbarn des Teilnehmers. Deswegen werden zonenlose Protokolle auch mit dem Begriff *Mutual Notification*-Protokolle bezeichnet.

Ein Vorreiter der zonenlosen Protokolle war das 2002 von Keller und Simon vorgeschlagene Protokoll Solipsis [77]. In Solipsis wird versucht, die Verbundenheit des Netzes zu garantieren, indem jeder Teilnehmer nach Möglichkeit immer so viele Nachbarn akquiriert, dass zwischen zwei Nachbarn nie mehr als ein 180° Winkel besteht. Dabei wird als Topologie der virtuellen Welt ein Torus vorausgesetzt. Trotz

[4] N-Tree wird in Abschnitt 5.1.3 ausführlicher beschrieben.

dieser Einschränkung kann die gewählte Struktur die Verbundenheit des Netzes und das Funktionieren der Nachbarsfindung nicht garantieren. Weiterentwicklungen des Protokolls setzten, wie in Abschnitt 3.2.3.2 beschrieben, auf eine zonenbasierte Struktur mit Supernodes [53]. Auch das von Kawahara et al. vorgeschlagene Protokoll [76] kann die Verbundenheit des Netzes nicht gewährleisten.

Bei pSense [122] wird der Netzzusammenhalt und die Nachbarsfindung von sogenannten Sensorknoten übernommen. Dies sind Teilnehmer, die sich außerhalb des Interessengebiets eines Teilnehmers befinden. Sie werden so gewählt, dass sie so gleichmäßig wie möglich um das Interessengebiet herum verteilt sind. Falls in Überlastsituationen ein Teilnehmer nicht allen seinen direkten Nachbarn Positionsmeldungen senden kann, erlaubt pSense eine mehrstufige Weiterleitung über die verbleibenden Nachbarn. Die Wahl des Sensorknoten in pSense kann jedoch insbesondere bei Gruppenbildung die Korrektheit der Nachbarsfindung und den Netzzusammenhalt nicht garantieren. Dieses Problem lässt sich nur verringern, indem jeder Teilnehmer die Zahl der Sensorknoten sehr hoch wählt. Da sich die Sensorknoten außerhalb des Interessengebiets befinden, erhöht dies den Bandbreitenoverhead erheblich. Weiterhin sieht pSense keine Backup-Mechanismen vor, sodass Teilnehmerausfälle den Netzzusammenhalt gefährden.

Ein besserer Ansatz der Nachbarsfindung wurde von VAST [66, 67] gewählt. In Vast werden die Nachbarschaftsbeziehungen durch die Bildung von Voronoi-Graphen über die Positionen der Teilnehmer bestimmt. Eine besondere Rolle nehmen die Grenznachbarn ein. Dies sind Nachbarn, deren Voronoi-Region von der Grenze des Interessengebiets geschnitten wird. Diese Nachbarn sind für die Nachbarsfindung zuständig. Eine genauere Beschreibung des Protokolls findet sich in Abschnitt 5.1.4. Ebenfalls auf Voronoi-Graphen basiert Nomad [116]. Einen ähnlichen, um Clustering erweiterten, Ansatz verfolgt ein auf den zu Voronoi-Graphen dualen Delauney-Graphen basierendes Protokoll von Varvello et al. [143].

Zonenlose Protokolle haben den Vorteil, dass alle Nachrichten ohne Zwischenknoten auf Overlay-Schicht an alle interessierten Teilnehmer versendet werden. Dadurch können sie eine geringe Nachrichtenverzögerung erreichen. Das Kernproblem von zonenlosen Protokollen ist die Nachbarsfindung. Ist sie nicht zuverlässig, sinkt die Nachbarschaftskenntnis in der virtuellen Welt. Bei bisherigen Protokollen ist die Nachbarsfindung oft aufwändig bezüglich der benötigten Bandbreite. Dadurch können Teilnehmer überlastet werden.

3.3 Hybride Protokolle

Hybride Protokolle verwenden einen zentralen Server, der über die Positionen aller Teilnehmer informiert ist. Das Verteilen von Ereignisnachrichten und Positionsmeldungen wird jedoch zu gewissen Teilen von den Teilnehmern selbst übernommen [107].

Bei peers@play [131] verbinden sich Teilnehmer direkt mit den Teilnehmern, die sich in ihrem Interessengebiet befinden. Übersteigt dieses ihre Netzkapazität, werden die Verbindungen zu den jeweils entferntesten Nachbarn fallengelassen. Der zentrale Server wird über alle Positionen informiert und weist die Teilnehmer an, welche Verbindungen aufzubauen und welche fallenzulassen sind. Dies ermöglicht eine verzögerungsarme Zustellung von Positionsmeldungen und Ereignisnachrichten. Zugleich

kann eine Überlastung der Teilnehmer ausgeschlossen werden. Gegenüber klassischen Client/Server-basierten virtuellen Welten ist die Skalierbarkeit verbessert, da die Teilnehmer einen Teil der Nachrichtenverteilung übernehmen.

Ein vergleichbares hybrides Protokoll ist Hyperverse [47]. Auch hier werden Verbindungen zwischen Teilnehmern durch einen zentralen Server, der über die Position aller Teilnehmer informiert ist, initiiert. Eine Besonderheit ist hier, dass sich die Aufgaben des Servers besonders gut auf einen Servercluster verteilen lassen.

Da der Server jedoch bei hybriden Protokollen weiterhin über alle Positionen informiert werden muss, steigt der Ressourcenbedarf des Servers mindestens linear mit der Teilnehmerzahl. Aus diesem Grund ist die Gesamtzahl der unterstützten Teilnehmer von der Serverkapazität begrenzt und die Skalierbarkeit eher mit Client/Server-basierten virtuellen Welten zu vergleichen. Deshalb werden hybride Protokolle in dieser Arbeit nicht weiter betrachtet.

3.4 Zusammenfassung

Die verschiedenen in diesem Kapitel vorgestellten Technologien sind in Tabelle 3.1 zusammengefasst. Client/Server-basierte virtuelle Welten können in der Regel gute Nachrichtenverzögerungen bieten. Durch die mindestens lineare Abhängigkeit zwischen Teilnehmerzahl und benötigter Serverkapazität sind sie jedoch nicht skalierbar.

Mit P2P-Protokollen können die Skalierungsprobleme der Client/Server-Protokolle überwunden werden. Während vollvermaschte Protokolle sich nicht für skalierbare virtuelle Welten eignen, können die anderen beschriebenen zonenbasierten und zonenlosen Ansätze eine gute Skalierbarkeit bieten, da in der Regel Bandbreiten- und Rechenbedarf nur von der Zahl der Zonenmitglieder oder Nachbarn, nicht jedoch von der Gesamtzahl der Teilnehmer abhängt.

Alle Ansätze werfen jedoch unterschiedliche Probleme in den Bereichen Nachrichtenverzögerung, Nachbarschaftskenntnis und Bandbreitenbedarf auf. Eine genauere Abschätzung der Eignung der jeweiligen Protokolle erfordert also eine weitere Evaluierung. Eine umfassende vergleichende Evaluierung ist bislang in der Literatur nicht zu finden. Deshalb wird in Kapitel 5 ein solcher Vergleich durchgeführt. Dabei wird aus jeder der hier vorgestellten Kategorien außer der der vollvermaschten Protokolle ein Protokoll ausgewählt und auf seine Leistung bezüglich der beschriebenen Anforderungen untersucht. Da das Ergebnis dieser Untersuchung zeigt, dass die bisherigen Protokolle die in Abschnitt 2.1.3 aufgeführten Anforderungen nicht voll erfüllen können, wird im weiteren Verlauf der Arbeit ein neues P2P-Protokoll für virtuelle Welten entwickelt.

Technologie			Bemerkungen	Latenz	Skalierbarkeit
Client/Server	Sharding Instanzen Zonen		Verbessert nicht die Skalierbarkeit, sondern nur die Verteilung der Teilnehmer auf die Server	⊕	⊖
	Proxy-Architektur		Ermöglicht eine gemeinsame Welt, verringert aber nicht die benötigte Serverkapazität	◯	⊖
	Cloud-Computing		Flexibilisiert die Hardwarebereitstellung, verringert aber nicht die benötigte Serverkapazität	⊕	⊖
	hybrid		Senkt die Serverlast, benötigte Serverkapazität wächst weiterhin linear mit Teilnehmerzahl	⊕	◯
P2P	vollvermascht		Quadratischer Aufwand bezüglich der Teilnehmerzahl	⊕	⊖
	zonenbasiert	ALM	Multicastbäume verursachen Nachrichtenverzögerungen	⊖	⊕
		Supernode	Lastverteilmaßnahmen erhöhen Nachrichtenverzögerung, aber ohne Lastverteilmaßnahmen nicht skalierbar	⊖	⊕
		dynamische Zonen	Nachbarschaftskenntnis nicht gewährleistet	⊕	⊕
	zonenlos		Möglicher Protokolloverhead durch Nachbarsfindung	⊕	⊕

Tabelle 3.1 Bewertung Technologien für virtuelle Welten.

4. Das Overlay-Simulationsrahmenwerk OverSim

Ein grundlegendes Problem bei der Erforschung von Overlay-Protokollen ist die Evaluierung existierender und neuer Protokolle. Da Testnetze für die Evaluierung dieser Protokolle nur eine begrenzte Größe haben, können wichtige Aspekte der Protokolle in diesen Netzen nicht ausreichend untersucht werden. Deswegen ist die Simulation der Protokolle ein wichtiger Baustein der Evaluierung. Da bisherige Simulationswerkzeuge für Overlay-Protokolle einige wichtige Anforderungen vernachlässigt haben, wurde im Rahmen dieser Arbeit in Zusammenarbeit mit Kollegen das Overlay-Simulationsrahmenwerk OverSim [15–17, 85] entwickelt. Dieses wird im Folgenden vorgestellt.

4.1 Anforderungen an Simulationswerkzeuge für Overlay-Netze

Bei der Entwicklung von OverSim wurden insbesondere die folgenden Anforderungen an Simulationswerkzeuge betrachtet:

- **Skalierbarkeit**: Da Overlay-Netze mehrere Millionen Teilnehmer umfassen können [129], sollte ein Simulationswerkzeug für Overlay-Netze Simulationen mit einer großen Zahl an Knoten unterstützen. Dabei müssen Speicherverbrauch und Simulationszeit mit üblicher Hardware handhabbar bleiben.

- **Flexibilität**: Ein Simulationswerkzeug für Overlay-Netze sollte Simulationen verschiedenster Overlay-Protokolle ermöglichen. Dabei sollten Parameter und Simulationsszenarien in einfacher Form konfiguriert werden können. Es sollte möglich sein, einzelne Teile der Simulationsumgebung, wie beispielsweise

die Netzabstraktion, transparent durch funktionell vergleichbare Module auszutauschen. Overlay-Protokolle sollten mit verschiedenen Anwendungen zusammenarbeiten und Anwendungen sollten die Dienste verschiedener Overlay-Protokolle in Anspruch nehmen können.

- **Realistische Netzmodellierung**: Für die Simulationen von Overlay-Netzen wird in der Regel das unterliegende Netz durch das Simulationswerkzeug modelliert. Dabei muss die Modellierung möglichst realistisch ausfallen, um gute Schlussfolgerungen aus den Simulationsergebnissen ziehen zu können. Insbesondere müssen realistische Bandbreiten, Netzverzögerungen und Paketverlustraten zwischen den Knoten simuliert werden. Zugleich muss die Netzmodellierung skalierbar bleiben, um eine große Zahl an Knoten unterstützen zu können. Idealerweise sollte das Netzmodell austauschbar sein. So kann, je nach konkreter Anforderung der jeweiligen Simulation, ein Netzmodell mit einer detaillierten Simulation des unterliegenden Netzwerks oder eine vereinfachte, aber dafür Ressourcen schonende, Modellierung ausgewählt werden.

- **Nutzer- und Anwendungsverhalten**: Bei der Evaluierung von Overlay-Protokollen sind das Verhalten des Nutzers der Protokolle sowie die Eigenschaften der Anwendung wichtige Einflussgrößen. Dabei ist insbesondere das Sitzungsverhalten der Nutzer, also die Verteilung der Sitzungszeiten der Teilnehmer, die Ankunftsrate neuer Teilnehmer und die Knotenausfallrate hervorzuheben. Aber auch anwendungsspezifisches Verhalten wie die Frequenz bestimmter Anfragen, die Häufigkeit der Interaktion zwischen verschiedenen Netzteilnehmern oder die Bewegung der Teilnehmer in der virtuellen Welt im Anwendungsszenario virtuelle Welten können ein Protokoll wesentlich beeinflussen. Ein Simulationswerkzeug für Overlay-Netze muss dieses Verhalten daher möglichst realistisch simulieren können. Dabei sollte auch mögliches böswilliges Verhalten der Nutzer berücksichtigt werden können. Besonders realistisches Verhalten lässt sich erreichen, wenn echte Anwendungen mit dem Simulationswerkzeug verbunden oder aufgezeichnete Sitzungen echter Nutzer im Simulator nachgespielt werden können.

- **Echtnetzanbindung**: Der für die Simulation entwickelte Code sollte auch in physischen Netzen einsetzbar sein. Dies ermöglicht es Forschern, die Evaluation der Overlay-Protokolle nicht nur in der Simulationsumgebung, sondern auch im Internet und Forschungsnetzen wie G-Lab [123, 136] oder PlanetLab [18, 109] durchzuführen. Zusätzlich kann so in Demonstratoren eine geringe Zahl an physischen Geräten durch simulierte Netzteilnehmer ergänzt werden. Dies lässt sich erreichen, indem das Simulationsrahmenwerk simulierte Nachrichten in standardkonforme IP-Datagramme umwandelt und so mit anderen Implementierungen des Protokolls über ein physisches Netzwerk kommuniziert.

- **Statistiken**: Ein Simulationswerkzeug für Overlay-Netze muss in der Lage sein, statistische Daten wie die benötigte Bandbreite pro Knoten, die durchschnittliche Nachrichtenverzögerung, den Anteil erfolgreicher oder nicht erfolgreicher Kommunikation oder andere protokollspezifische Messgrößen zu sammeln und zu aggregieren. Die sich daraus ergebenden Daten sollten in einer Form vorliegen, die das Auswerten und die Erstellung von Diagrammen möglichst vereinfacht.

- **Visualisierung**: Um die Interaktionen von Knoten in Overlay-Protokollen besser nachvollziehen und Fehler im Entwurf eines Protokolls oder Implementierungsfehler einfacher finden zu können, ist eine Visualisierung des Netzes hilfreich. Dabei sollten insbesondere Nachrichten und wichtige Beziehungen zwischen Knoten vom Simulationswerkzeug dargestellt werden können. Die Möglichkeit der Visualisierung der Topologie des unterliegenden Netzwerks ist wünschenswert.

- **Dokumentation und Erweiterbarkeit**: Um einem möglichst großen Kreis an Forschern die Benutzung des Simulationswerkzeuges zu ermöglichen, sollte eine ausführliche Anleitung vorliegen. Um die Entwicklung von neuen Protokollen oder Modulen für das Simulationswerkzeug zu vereinfachen, müssen alle vom Simulationswerkzeug angebotenen und verwendeten Schnittstellen dokumentiert sein.

4.2 Bisherige Simulationswerkzeuge

Es gibt verschiedene Simulationswerkzeuge für Overlay-Netze. Keines dieser Simulationswerkzeuge konnte sich jedoch in der Forschungsgemeinde als Standardwerkzeug durchsetzen. Eine Auswahl der zum Zeitpunkt des Beginns der Arbeiten an OverSim verbreiteten Simulationswerkzeuge soll hier kurz vorgestellt werden:

- **P2PSim**: P2PSim [90] ist einer der ersten P2P-Simulatoren, die zur allgemeinen Nutzung veröffentlicht wurden. P2PSim ist ein diskreter Ereignissimulator, der in C++ entwickelt wurde. Dabei beschränkt er sich auf die Simulation strukturierter Overlay-Netze, die einen KBR oder DHT-Dienst zur Verfügung stellen. P2PSim eignet sich nur bedingt für die Simulation anderer P2P-Protokolle wie unstrukturierte P2P-Protokolle oder P2P-Protokolle für virtuelle Welten. Da P2PSim laut Einschätzung seiner Autoren noch im „Alpha“-Stadium ist, es keine ausführliche Dokumentation gibt und das Projekt nicht mehr weiterentwickelt wird, ist es schwierig, P2PSim zu erweitern und mit Implementierungen eigener Protokolle zu ergänzen. Eine Visualisierung der Overlay-Topologie ist ebenso wie eine Interaktion mit physischen Netzen nicht möglich.

- **OverlayWeaver**: OverlayWeaver [125] ist ein in Java programmiertes Overlay-Entwicklungswerkzeug. Die Ziele von OverlayWeaver sind das einfache Entwickeln und Testen von KBR- und DHT-Protokollen. OverlayWeaver ist insbesondere auf das Testen der Protokolle in physischen Netzen ausgelegt. Der Nutzen als Simulator ist hingegen eingeschränkt. So wird kein realistisches Netzmodell für das unterliegende Netzwerk geboten. Die Messung von Statistiken ist nur bedingt möglich und die interaktive Visualisierung kann zwar die versendeten Nachrichten zeigen, nicht jedoch die tatsächliche Overlay-Topologie. Somit ist OverlayWeaver nur bedingt als Simulationswerkzeug für die Evaluierung von Overlay-Protokollen geeignet.

- **PlanetSim**: PlanetSim [57] ist ein Java-basiertes Overlay-Simulationsrahmenwerk. Durch seinen modularen Aufbau und seine ausführliche Dokumentation ist PlanetSim erweiterbar. Durch die Verwendung der CommonAPI [40] eignet

sich PlanetSim besonders für die Simulation von KBR- und DHT-Protokollen. Eine Unterstützung anderer Overlay-Protokolle ist nicht vorgesehen. Die Netzabstraktion ist erweiterbar, die vorliegenden Netzmodelle sind jedoch sehr einfach gehalten und erlauben keine realistische Simulation von Latenzen oder Bandbreiten. Als Simulationsergebnis wird im Wesentlichen die Netztopologie zum Ende des Simulationslaufes ausgegeben. Das Erstellen weitergehender Statistiken ist nur eingeschränkt möglich. Eine Echtzeit-Visualisierung der Netztopologie oder eine Anbindung des Simulators an physische Netze ist nicht vorgesehen.

- **PeerSim**: PeerSim [70] ist ein Simulationswerkzeug für Overlay-Netze. Ein Fokus bei PeerSim liegt in der Skalierbarkeit. Neben einer ereignisbasierten Simulation bietet PeerSim zusätzlich eine zyklusbasierte Simulation an, die die Simulation besonders großer Netze erlaubt. Durch einen modularen Aufbau ist PeerSim einfach erweiterbar. Das Netzmodell erlaubt die Simulation realistischer Latenzen zwischen Knoten durch die Verwendung von Messungen des King-Datensatzes [63]. Bandbreiten werden dabei jedoch nicht betrachtet. Eine Visualisierung der Overlay-Topologie ist nicht möglich, eine Echtnetzanbindung nicht vorgesehen.

Eine genauere Vorstellung dieser und weiterer Simulationswerkzeuge für P2P-Protokolle ist bei Naicken et al. zu finden [100]. Keines der von den Autoren untersuchten Simulationswerkzeuge konnte alle in Abschnitt 4.1 genannten Anforderungen erfüllen. Aus diesem Grunde wurde im Rahmen dieser Arbeit in Zusammenarbeit mit Kollegen das Overlay-Simulationsrahmenwerk OverSim entwickelt. OverSim wird mittlerweile international in vielen Forschungsgruppen eingesetzt. Seit der Veröffentlichung von OverSim sind noch weitere Simulationswerkzeuge für Overlay-Netze entstanden, wie beispielsweise PeerFaktSim.KOM [81]. Diese erreichten jedoch nicht die weite Verbreitung von OverSim.

4.3 Basis: OMNeT++

OverSim basiert auf der Methode der diskreten Ereignissimulation. Hierzu verwendet OverSim den Open-Source Ereignissimulator *OMNeT++* [140–142]. Neben der Verwaltung von Ereignissen bietet OMNeT++ eine grafische Benutzeroberfläche mit einer interaktiven Visualisierung der Simulation.

Die Basiseinheiten von OMNeT++ sind *Module* und *Nachrichten*. Module sind die aktiven Einheiten der Simulation. Dabei gibt es *einfache Module*, die direkt in C++ implementiert werden, und *zusammengesetzte Module*. Diese bestehen aus einer Menge an einfachen Modulen oder zusammengesetzten Modulen. Module werden in der OMNeT++-eigenen Deklarierungssprache NED definiert. Mit dieser Sprache lassen sich Modulparameter und im Fall von zusammengesetzten Modulen die Bestandteile des Moduls definieren.

Module können sich gegenseitig Nachrichten zusenden. Nachrichten werden in der OMNeT++-eigenen Deklarierungssprache MSG definiert. In dieser können die möglichen Inhalte der Nachricht deklariert werden. Dabei sind alle in C++ gültigen Datentypen zugelassen. Jede Nachricht enthält die Zustellzeit. Die Nachrichten werden

in OMNeT++ gemäß der Zustellzeit in einem Heap verwaltet. Die jeweils oberste Nachricht wird ihrem Empfänger zugestellt und dort bearbeitet.

Neben der direkten Zustellung von Nachrichten erlaubt OMNeT++ das Versenden von Nachrichten über *Verbindungen*. Hierzu können Module sogenannte *Gates* definieren. Diese Gates können dann mit Verbindungen unterschiedlicher Charakteristika wie Bandbreite, Verzögerung oder Paketverlustrate verbunden werden. Auf diese Weise können Module auf einfache Weise zu Netzwerken verbunden werden.

OMNeT++ enthält eine grafische Benutzeroberfläche, die Module, Verbindungen und Nachrichten visualisieren kann. In dieser Visualisierung lässt sich die Ausführung der Simulation ereignisgenau steuern. Zu jedem Zeitpunkt der Simulation lassen sich die Zustände aller Module und alle zum aktuellen Zeitpunkt in der Simulation vorhandenen Nachrichten betrachten. Um eine schnellere Simulation zu ermöglichen, lässt sich die Benutzeroberfläche auch deaktivieren.

Zu OMNeT++ sind verschiedene Erweiterungen veröffentlicht worden. Eine wichtige Erweiterung ist das INET-Framework [139, 148]. Dieses implementiert einen vollständigen TCP/IP-Stack für OMNeT++. OverSim verwendet, abhängig von der gewählten Netzmodellierung, die Implementierung der Transportschichtprotokolle TCP und UDP sowie das Vermittlungsschichtprotokoll IP aus dem INET-Framework.

4.4 Architektur

Um die für Simulationswerkzeuge erwünschte Flexibilität bieten zu können, ist OverSim modular aufgebaut. Die Architektur von OverSim ist in Abbildung 4.1 gezeigt.

Der Kern des Simulators besteht aus drei aufeinander aufbauenden Schichten, der *Underlay-Schicht*, der *Overlay-Schicht* und der *Anwendungsschicht*. Die Schichten sind durch Schnittstellen verbunden: Zwischen Underlay- und Overlay-Schicht liegt eine UDP-Schnittstelle. Zwischen Overlay- und Anwendungsschicht liegen spezifische Anwendungsschnittstellen für verschiedene Overlaytypen. Daneben gibt es den allen Schichten zugänglichen *GlobalObserver*, der Hilfsfunktionen wie Bootstrapping oder Statistikerstellung anbietet. Weiterhin gibt es die von Basisklassen angebotenen Funktionen RPC-Behandlung für die Overlay- und Anwendungsschicht sowie die Visualisierungsfunktion für Overlay- und Underlay-Schicht. Anwendungsschicht und Underlay-Schicht steht die Echnetzanbindung und die Sitzungsaufzeichnung zur Verfügung.

Die Unterste der drei Hauptschichten ist die *Underlay-Schicht*. Die wesentliche Aufgabe dieser Schicht ist die Abstraktion des unterliegenden Netzes. Hierbei können verschiedene Netzmodelle – verfügbar sind das *SimpleUnderlay*, das *ReaseUnderlay*, das *InetUnderlay* und das *SingleHostUnderlay* – transparent für die höheren Schichten ausgewählt werden. Die Netzmodelle bieten eine UDP-Schnittstelle an, welche die höheren Schichten zur Kommunikation zwischen Knoten nutzen können. Eine genaue Beschreibung der möglichen Netzmodelle findet sich in Abschnitt 4.5. Weiterhin wird in der Underlay-Schicht das Erzeugen und Entfernen von Knoten aus der Simulation gesteuert. Dies geschieht über Sitzungsmodelle, die in Abschnitt 4.8.1 näher vorgestellt werden.

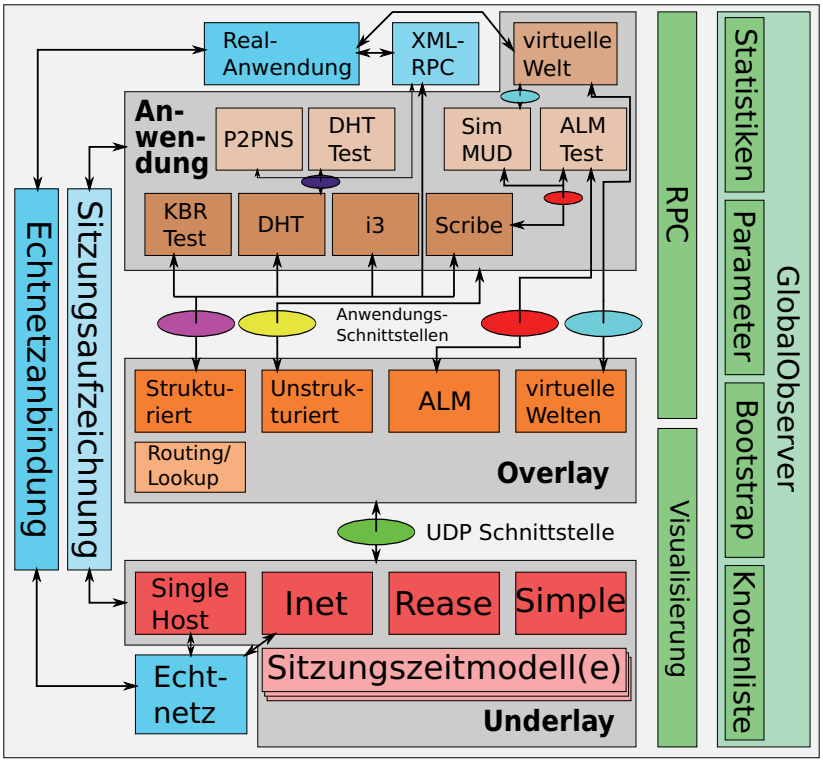


Abbildung 4.1 Die modulare Architektur von OverSim.

Oberhalb der Underlay-Schicht befindet sich die *Overlay-Schicht*. Die Overlay-Schicht enthält die in OverSim implementierten Overlay-Protokolle. Die von den Overlay-Protokollen bereitgestellten Schnittstellen hängen von der Art des Overlay-Protokolls ab. *Strukturierte Overlays*, die einen *KBR*-Dienst anbieten, besitzen eine *CommonAPI-Schnittstelle* [40]. *Application Layer Multicast*-Protokolle bieten eine spezielle *ALM-Schnittstelle*, Protokolle für virtuelle Welten eine *Schnittstelle für virtuelle Welten*. Unstrukturierte Overlay-Protokolle können eigene, für den speziellen Anwendungszweck geeignete Schnittstellen definieren. Auf die Funktionen der Overlay-Schicht wird in Abschnitt 4.6 genauer eingegangen.

Alle Protokolle, Dienste und Anwendungen, die Dienste eines Protokolles der Overlay-Schicht nutzen, finden sich in OverSim in der *Anwendungsschicht*. Da einige Protokolle der Anwendungsschicht Dienste für andere Anwendungen bereitstellen können, ist die Anwendungsschicht wiederum in Schichten unterteilt. So nutzt beispielsweise *Scribe* den KBR-Dienst der Overlay-Schicht, um einen Application Layer Multicast-Dienst zur Verfügung zu stellen. Dieser kann dann über die ALM-Schnittstelle von anderen Anwendungen genutzt werden. Durch die konsequente Verwendung definierter Schnittstellen ist es möglich, dass Anwendungen transparent Dienste aus der Overlay- oder Anwendungsschicht nutzen können. Dies ermöglicht beispielsweise den Vergleich von Protokollen, die einen speziellen Dienst bereits in der Overlay-Schicht bereitstellen, mit Protokollen, die diesen Dienst in der Anwendungsschicht erbringen. Die Anwendungsschicht von OverSim wird in Abschnitt 4.7 näher vorgestellt.

Neben diesen drei wesentlichen Schichten bietet OverSim schichtenübergreifende Module, die das Implementieren von Protokollen und das Auswerten von Simulationsergebnissen erleichtern. Hierzu gehört der *GlobalObserver*, der einen übergeordneten

Blick auf das simulierte Overlay-Netz hat. Dieser kann somit eine Liste aller Knoten erzeugen, die das *Bootstrapping* neu zur Simulation hinzukommender Knoten ermöglicht. Weiterhin können dort Messdaten der Knoten aggregiert werden oder Daten erhoben werden, die Wissen über die gesamte Netztopologie erfordern.

OverSim bietet Modulen der Anwendungs- und der Overlay-Schicht eine *RPC-Schnittstelle*. Diese ermöglicht den Modulen eine einfache Verwaltung von Zeitüberschreitungen beim Versand von Nachrichten und bei Bedarf die Durchführung von Sendewiederholungen. Hierdurch wird das Implementieren neuer Protokolle und Anwendungen in OverSim erheblich vereinfacht. In OverSim ist es möglich, RPCs sowohl direkt an andere Knoten zu versenden als auch durch Angabe eines Overlay-Schlüssels für den Versand die Routingmechanismen des verwendeten Overlay-Protokolls zu nutzen. Da die RPC-Funktionalität von fast allen Overlay-Protokollen und -Anwendungen genutzt wird, wird eine gute Vergleichbarkeit der Implementierungen ermöglicht.

Um die Netztopologie visualisieren zu können, verwendet OverSim die *tkenv*-Umgebung von OMNeT++. Diese ermöglicht die interaktive visuelle Darstellung von Beziehungen zwischen Knoten. In OverSim kann die Visualisierung von der Overlay-Schicht und der Underlay-Schicht verwendet werden. Erfolgt die Visualisierung durch die Overlay-Schicht, wird die Topologie des Overlay-Netzes gezeigt. Die Visualisierung ist hierbei protokollabhängig. Strukturierte, ringbasierte Overlay-Protokolle wie Chord können beispielsweise durch Anzeige der Verbindungen zu Vorgänger- und Nachfolgerknoten den durch die Knoten gebildeten Ring visualisieren; bei Application Layer Multicast-Protokollen wie Nice bietet sich die Visualisierung des Multicastbaumes an. Alternativ zur Visualisierung des Overlays kann in OverSim auch eine Visualisierung der Underlay-Schicht erfolgen. Hier kann die Topologie des Netzmodells visualisiert werden.

OverSim bietet eine Anbindung der Simulationsumgebung an phyische Netzwerke. Diese Netzanbindung ist an zwei Schichten des Simulators angebunden. Zunächst können zwei Underlay-Modelle, das SingleHostUnderlay und das InetUnderlay, mit dem Netz verbunden werden. Daten, die aus der Overlay-Schicht über die UDP-Schnittstelle versendet werden, können so an das Netz weitergegeben werden und Daten aus dem Netz können über die UDP-Schnittstelle an die Overlay-Schicht weitergeleitet werden. Zusätzlich kann die Anwendungsschicht mit dem Netz verbunden werden. Dies ermöglicht die Anbindung externer Applikationen, die so Anwendungen in OverSim steuern oder Dienste aus der Anwendungsschicht nutzen können. Die Echtnetzanbindung wird in Abschnitt 4.9 genauer beschrieben.

Üblicherweise wird in OverSim Nutzerverhalten mittels Wahrscheinlichkeitsverteilungen modelliert. Bei Bedarf gibt es jedoch auch die Möglichkeit, aufgezeichnete Sitzungen als Basis für die von OverSim erzeugten Ereignisse zu verwenden. Diese Sitzungsaufzeichnung kann in OverSim in der Underlay-Schicht und in der Anwendungsschicht genutzt werden. In der Underlay-Schicht werden anhand der aufgezeichneten Sitzung Knoten erzeugt oder aus dem Netz entfernt. Hierbei arbeitet die Sitzungsaufzeichnung als Sitzungsmodell (siehe auch Abschnitt 4.8.1.) Zusätzlich können Applikationsereignisse anhand der aufgezeichneten Sitzungen generiert werden (siehe hierzu Abschnitt 4.8.2.)

4.5 Underlay-Schicht: Netzabstraktion

In OverSim können verschiedene Modelle für die Abstraktion des unterliegenden Netzwerks gewählt werden. Das in OverSim am meisten verwendete Netzmodell ist das *SimpleUnderlay*-Modell. Dieses Model bietet geringen Speicherverbrauch, hohe Simulationsgeschwindigkeit und realistische Berechnungen von Latenzen. Ist ein höherer Grad an Realismus nötig, kann das *InetUnderlay* oder das davon abgeleitete *ReaseUnderlay* gewählt werden. Diese Modelle emulieren den vollständigen TCP/IP-Stack aller Teilnehmer sowie die Zugangsnetze und Router im unterliegenden Netzwerk. Ist eine Echtnetzanbindung gewünscht, kann das *SinglehostUnderlay* oder das *InetUnderlay* in Verbindung mit der TUN-Schnittstelle[1] verwendet werden. Im Folgenden werden die einzelnen Netzabstraktionsmodelle genauer vorgestellt.

4.5.1 SimpleUnderlay

Das SimpleUnderlay-Modell verbindet eine besonders Ressourcen schonende Simulation mit einem hohen Grad an Realismus. Hierzu werden Nachrichten zwischen Knoten direkt ausgetauscht. Die durch das unterliegende Netzwerk auftretenden Nachrichtenverzögerungen werden mittels synthetischer Koordinaten approximiert.

Für die Approximation werden den Knoten Koordinaten in einem mehrdimensionalen Euklidischen Koordinatensystem zugewiesen. Der Abstand der Knoten entspricht dann der Netzverzögerung zwischen diesen beiden Knoten multipliziert mit einer Konstante. Vergleichbare Methoden werden in der Forschung angewendet, um die erwartete Netzverzögerung zwischen zwei Knoten ohne vollständige Messungen abzuschätzen [103]. Die Knotenpositionen im SimpleUnderlay können entweder zufällig bestimmt werden oder aus einer Koordinaten-Datei entnommen werden. Für OverSim wurde eine Koordinaten-Datei erstellt, die die Knoten so positioniert, dass die resultierenden Verzögerungen den durch Studien wie CAIDA/SKITTER [95] ermittelten Verzögerungen zwischen Knoten im Internet entsprechen. So lassen sich effizient realistische Netzszenarien simulieren [86].

Zusätzlich zu der entfernungsbasierten Verzögerung wird jedem Knoten ein Zugangsnetz zugewiesen, welches eine statische Verzögerung verursacht. Weiterhin wird dem Knoten durch das Zugangsnetz eine maximale Bandbreite zugewiesen. Durch Paketgröße und Bandbreite ergibt sich so eine zusätzliche Nachrichtenverzögerung. Da die Nachrichtenverzögerungen im Internet üblicherweise gewissen Schwankungen unterliegen, kann noch eine weitere, von den vorigen Faktoren unabhängige Verzögerung gemäß einer Wahrscheinlichkeitsverteilung zu der gesamten Nachrichtenverzögerung addiert werden.

Insgesamt ergibt sich also die folgende Formel für die Ende-zu-Ende Nachrichtenverzögerung d_e eines Paketes P zwischen zwei Knoten A und B. Hierbei sei d_X die durch das Zugangsnetz von Knoten X verursachte Verzögerung, $\frac{l_p}{b_X}$ die sich aus der Paketlänge l_P und der maximalen Bandbreite b_X von Knoten X ergebende Nachrichtenverzögerung, $c \cdot \|A-B\|_2$ die sich aus dem euklidischen Abstand der Knoten A und B ergebende Verzögerung und d_j die sich durch zufällige Schwankungen ergebende Nachrichtenverzögerung:

$$d_e = d_A + \frac{l_P}{b_A} + c \cdot \|A - B\|_2 + d_B + \frac{l_P}{b_B} + d_j$$

[1] Die TUN-Schnittstelle (kurz für „Tunnels over TCP/IP“) ist eine virtuelle Netzwerkschnittstelle für Linux.

Zusätzlich zu der Ende-Zu-Ende-Verzögerung wird die Sendewarteschlange der Knoten emuliert. Hierzu wird für jedes Paket das Ende des Übertragungsvorgangs berechnet. Weitere Pakete werden dann erst nach diesem Zeitpunkt versendet. Sei t'_{tx} der Zeitpunkt der Beendigung des Übertragungsvorgangs des letzten Paketes und t_{now} die aktuelle Zeit, dann ergibt sich der Zeitpunkt der Beendigung des Übertragungsvorgangs t'_{tx} des aktuellen Paketes P als

$$t_{tx} = \max(t'_{tx}, t_{now}) + \frac{l_P}{b_A}$$

und die Ankunftszeit des Paketes t_{arrv} als

$$t_{arrv} = \max(t'_{tx}, t_{now}) + d_e$$

Zuletzt können mit dem SimpleUnderlay Paketverluste oder Übertragungsfehler simuliert werden. Hierzu werden den Knoten Bitfehlerraten zugewiesen. Die Wahrscheinlichkeit eines Bitfehlers in einem Paket P ergibt sich dann aus der Länge des Paketes l_P und der Bitfehlerrate des Senders p_A^{err} und der Bitfehlerrate des Empfängers p_B^{err}:

$$p^{err} = p_A^{err} * l_P + p_B^{err} * l_P$$

Mit diesen Berechnungen können alle wesentlichen Einflüsse des Netzwerks auf den Versand einer Nachricht zwischen zwei Knoten in nur einem Simulationsereignis approximiert werden. Hierdurch ergibt sich ein geringer Simulationsoverhead und damit die Möglichkeit, Netze mit einer großen Zahl von Knoten zu simulieren. Dabei bleibt das Netzmodell realistischer als das Netzmodell bisheriger Simulationswerkzeuge für Overlay-Netze.

4.5.2 InetUnderlay und ReaseUnderlay

Ist für eine Simulation eine genauere Modellierung des unterliegenden Netzes nötig, kann das *InetUnderlay* verwendet werden. Dieses simuliert bei allen Knoten einen vollständigen TCP/IP-Stack. Das unterliegende Netz wird dann durch eine Menge von Routern dargestellt. Beim Versenden einer Nachricht zwischen zwei Knoten wird der komplette Weg der Nachricht durch das unterliegende Netz über alle Zwischenrouter simuliert.

Hierzu werden alle am Overlay-Netz teilnehmenden Knoten einem Zugangsrouter zugewiesen. Die Verbindung zwischen den Knoten und dem Zugangsrouter erfolgt über eine OMNeT++-Verbindung, deren Charakteristik gängigen Zugangstechnologien für das Internet entspricht. Die Zugangsrouter sind dann über weitere Router miteinander verbunden. Einige hybride Overlay-Protokolle können über das Anbieten von Overlay-Diensten im Backbone oder Zugangsnetz unterstützt werden. Hierzu bietet OverSim die Möglichkeit, Zugangs- oder sonstige Router mit den entsprechenden Overlay-Protokollen auszustatten.

Eine Erweiterung des InetUnderlays ist das *ReaseUnderlay*. Während im InetUnderlay die Netztopologie weitgehend zufällig ist, nutzt das ReaseUnderlay den Topologiegenerator ReaSE [56], um realistische Netztopologien zu erzeugen. Mit den so erzeugten Topologien lassen sich Overlay-Protokolle unter Berücksichtigung aller wesentlichen Effekte des unterliegenden Netzes in Szenarien mit internetähnlicher

Netztopologie simulieren. Durch die vollständige Simulation des Weges aller Nachrichten durch das unterliegende Netz wird für Simulationen, die das Rease- oder InetUnderlay verwenden, mehr Rechenzeit benötigt als für Simulationen auf Basis des SimpleUnderlays.

4.5.3 Modelle für Echtnetzanbindung

Um Protokollimplementierungen an physische Netze anzubinden, kann das *SingleHostUnderlay* verwendet werden. Hier wird nur ein einzelner Knoten emuliert. Von diesem Knoten versendete Nachrichten werden an den Netzwerkstack des Betriebssystems weitergeleitet. Somit kann der emulierte Knoten über physiche Netzwerke kommunizieren.

Sollen statt eines einzelnen Knotens mehrere emulierte Knoten mit dem Netz verbunden werden, kann das *InetUnderlay* um einen *TunOutRouter* erweitert werden. Dieser Router kann mit dem Linux-eigenen *TUN-Interface* [82] verbunden werden. Nachrichten, die von außen an das Interface gesendet werden, werden vom TunOutRouter in die Simulationsumgebung eingespeist. Analog werden Nachrichten, die von innerhalb der Simulationsumgebung an den TunOutRouter gesendet werden, über das TUN-Interface an das Netzwerk weitergeleitet.

Eine genauere Beschreibung der Echtnetzanbindungsmöglichkeiten von OverSim findet sich in Abschnitt 4.9.

4.6 Overlay-Schicht

Alle Protokolle der Overlay-Schicht in OverSim sind von der Klasse *BaseOverlay* abgeleitet. Diese Klasse implementiert einige Funktionen, die von vielen Overlay-Protokollen benötigt werden und erleichtert so die Implementierung neuer Protokolle. So können Knoten über die Basisklasse die RPC-Schnittstelle verwenden, die eine vereinfachte Nachrichtenbehandlung mit der Verwaltung von Zeitüberschreitungen und Sendewiederholungen ermöglicht. Weiterhin können die Bootstrapping-Methoden von OverSim genutzt werden. Außerdem steht, insbesondere für strukturierte Overlays, eine Routing-Funktionalität mit Unterstützung für iteratives und rekursives Routing sowie Source-Routing zur Verfügung. Für strukturierte Overlays wird zusätzlich eine generische Lookup-Klasse angeboten. Zudem implementiert die Basisklasse Hilfsfunktionen wie die Verwaltung von Ping-Nachrichten und Latenzmessungen.

Die in OverSim verfügbaren Overlay-Protokolle lassen sich anhand der von ihnen angebotenen Schnittstelle in vier Kategorien unterteilen: In strukturierte Overlay-Protokolle, Application Layer Multicast-Protokolle, Protokolle für virtuelle Welten und unstrukturierte Overlay-Protokolle.

Strukturierte Overlay-Protokolle stellen den größten Teil der in OverSim verfügbaren Overlay-Protokolle dar. Diese bieten der Anwendungsschicht über die CommonAPI-Schnittstelle einen KBR-Dienst an. Zu den in OverSim implementierten strukturierten Overlay-Protokollen zählen Chord [130], Kademlia [96], Pastry [121], Bamboo [115], Broose [55] und Koorde [73].

Die *Application Layer Multicast*-Schnittstelle wird in der Overlay-Schicht in OverSim bislang nur vom Protokoll Nice [9, 85] angeboten. Von dritter Seite wurden jedoch weitere ALM-Protokolle in OverSim untersucht [44, 80].

Die Protokolle PubSubMMOG [149], Vast [67], N-Tree [58] und das im Rahmen dieser Arbeit entwickelte QuON bieten Modulen der Anwendungsschicht die *Schnittstelle für virtuelle Welten* an.

Als unstrukturiertes Protokoll ist GIA [29] in OverSim implementiert. Von dritter Seite wurden auch weitere unstrukturierte Protokolle wie Bittorrent [36, 75] implementiert.

4.7 Anwendungsschicht

Die Anwendungsschicht in OverSim fasst alle Protokolle, Dienste und Anwendungen zusammen, die auf die Dienste eines Protokolls der Overlay-Schicht zurückgreifen. Die Klasse *BaseApp* unterstützt Implementierungen beim Zugriff auf die Overlay-Schicht. Wie auch BaseOverlay bietet BaseApp die Verwendung der RPC-Schnittstelle mit der Verwaltung von Zeitüberschreitungen und Sendewiederholungen. Für Applikationen und Dienste, die auf einem KBR-Dienst aufbauen, bietet die BaseApp Zugriff auf die applikationsspezifischen Teile der CommonAPI-Schnittstelle.

Für jede Schnittstelle der Overlay-Schicht enthält OverSim eine Applikation, die zum Testen der Schnittstelle verwendet werden kann und die statistische Daten über die Leistungsfähigkeit der Implementierung der Schnittstelle sammelt. Viele dieser Applikationen lassen sich auch über eine externe Anwendung steuern (siehe auch Abschnitt 4.9).

In OverSim können Protokolle aus der Anwendungsschicht auch Dienste für andere Anwendungen bereitstellen. Dabei können auch solche Schnittstellen angeboten werden, die ansonsten von der Overlay-Schicht implementiert werden. So bietet beispielsweise das Protokoll Scribe [26], aufbauend auf einem KBR-Dienst eines Overlays, anderen Anwendungen eine ALM-Schnittstelle. Das Protokoll SimMUD [79] bietet, aufbauend auf einem ALM-Dienst, anderen Anwendungen eine Schnittstelle für virtuelle Welten. Ein weiterer wichtiger Dienst auf Anwendungsebene ist die *DHT*, die anderen Anwendungen eine verteilte Hashtabelle über die DHT-Schnittstelle der CommonAPI zur Verfügung stellt.

4.8 Modellierung des Nutzerverhaltens

Ein wesentlicher Teil eines Simulationswerkzeuges für Overlay-Netze ist die Möglichkeit, Nutzerverhalten realistisch zu simulieren. Dabei lassen sich verschiedene wichtige Aspekte des Nutzerverhaltens unterscheiden. Einige davon sollen im Folgenden aufgezeigt werden.

4.8.1 Sitzungsverhalten

Nutzer von Overlay-Netzen sind üblicherweise nicht konstant Teil des Netzes. Stattdessen nehmen sie nur zu bestimmten Zeiten oder nur über bestimmte Zeiträume am Netz teil. Dabei sind die Sitzungszeiten anwendungsabhängig und individuell verschieden. In OverSim wird dem Anwender die Möglichkeit gegeben, aus verschiedenen Sitzungsmodellen das für das jeweilige Szenario passende auszuwählen. Bei

Bedarf können in einer Simulation auch mehrere Modelle parallel verwendet werden. Dadurch wird die Definition verschiedener Knotenklassen mit unterschiedlichem Sitzungsverhalten ermöglicht.

Allen Sitzungszeitmodellen ist eine *Initialisierungsphase* gemein, in der die Simulation mit Knoten bevölkert wird. Da bei vielen Overlay-Protokollen das Overlay-Netz einen stabilen Zustand nicht oder nur nach sehr langer Zeit erreicht, wenn alle Knoten gleichzeitig versuchen, dem Netz beizutreten, werden die Knoten in der Initialisierungsphase nacheinander erzeugt. Die Initialisierungsphase ist beendet, wenn die gewünschte Anzahl Knoten erreicht ist. Nach der Initialisierungsphase kann eine *Übergangsphase* definiert werden, in der sich das Overlaynetz stabilisiert. Erst nach Ablauf dieser Phase beginnt die eigentliche *Messphase*, in der OverSim statistische Daten zu der Simulation sammelt.

Das einfachste Sitzungsmodell ist das *RandomChurn*-Modell. In diesem Modell wird eine Knotenbeitritts- und eine Knotenaustrittswahrscheinlichkeit konfiguriert. Nach einem konfigurierbaren *Churn-Intervall* wird dann mittels der Ziehung einer Zufallszahl gemäß der konfigurierten Wahrscheinlichkeiten entschieden, ob ein Knoten dem Netz beitritt oder ein Knoten das Netz verlässt. Ist die Beitrittswahrscheinlichkeit größer als die Austrittswahrscheinlichkeit, steigt die durchschnittliche Zahl an Knoten mit der Zeit, ist sie kleiner, sinkt die durchschnittliche Knotenzahl. Die Fluktuationsrate lässt sich über das Churn-Intervall steuern. Ein langes Intervall führt zu einer geringen Fluktuation, ein kurzes Intervall zu einer hohen Fluktuation. Hierzu wird nach Beendigung der Initialisierungsphase periodisch gemäß dem konfigurierten Churn-Interval die in Algorithmus 4.1 dargestellte Prozedur RandomChurn ausgeführt.

Vorbedingung: $p_a :=$ Knotenbeitrittswahrscheinlichkeit
Vorbedingung: $p_r :=$ Knotenaustrittswahrscheinlichkeit
procedure RandomChurn
  $p \leftarrow$Zufallszahl(Gleichverteilung(0,1))
  if $p \leq p_a$ **then**
    ErzeugeKnoten
  else if $p \leq p_a + p_r$ **then**
    ZerstöreKnoten
  end if
end procedure

Algorithmus 4.1 Sitzungsmodell: RandomChurn

Das *LifetimeChurn*-Modell verfolgt einen anderen Ansatz zur Steuerung des Sitzungsverhaltens. Hier kann die durchschnittliche Sitzungszeit der Knoten über einen dedizierten Parameter konfiguriert werden. Zusätzlich kann die Wahrscheinlichkeitsverteilung der Sitzungszeiten gewählt werden. Dabei stehen Weibullverteilung, Paretoverteilung oder Normalverteilung zur Auswahl. Auf der Basis der Verteilung und der durchschnittlichen Sitzungszeit wird für jeden zu der Simulation hinzukommenden Knoten eine konkrete Sitzungszeit ermittelt. Nach Ablauf der Sitzungszeit wird der Knoten aus der Simulation entfernt. Danach wird analog zur Sitzungszeit eine Zeit errechnet, nach welcher wieder ein neuer Knoten zu der Simulation hinzugefügt wird. So bleibt die Zahl der aktiven Knoten in der Simulation im Durchschnitt konstant. Dies ist durch die in Algorithmus 4.2 dargestellten Prozeduren KnotenHin-

zufügen und KnotenEntfernen dargestellt. In der Initialisierungsphase werden n Knoten mit der Prozedur KnotenHinzufügen erzeugt, wobei n der gewünschten durchschnittlichen Knotenzahl entspricht. Für eine ebenso große Zahl von Knoten wird ein Aufruf von KnotenHinzufügen nach einer durch die Zufallsverteilung bestimmten Zeit vorgemerkt. Da in der Initialisierungsphase die Erstellungszeitpunkte der Knoten jedoch nicht dem durch die Wahrscheinlichkeitsverteilung vorgegebenen Muster folgen, ist bei dem LifetimeChurm-Modell eine gewisse Einschwingzeit nötig, bis das beobachtete Sitzungsverhalten der vorgegebenen Verteilung entspricht. Dies kann durch eine entsprechend gewählte Übergangsphase erreicht werden.

Vorbedingung: $\bar{t}$:= Durchschnittliche Lebenszeit
Vorbedingung: $P(\bar{t})$:= Wahrscheinlichkeitsverteilung der Lebenszeit mit Erwartungswert $\bar{t}$
Vorbedingung: t_{NOW} := aktuelle Zeit
procedure KnotenHinzufügen(k)
$t \leftarrow$Zufallszahl($P(\bar{t})$)
ErzeugeKnoten(k)
Aufruf von KnotenEntfernen(k) zu Zeitpunkt $t_{NOW} + t$
end procedure
procedure KnotenEntfernen(k)
ZerstöreKnoten(k)
$t \leftarrow$Zufallszahl($P(\bar{t})$)
Aufruf von KnotenHinzufügen(k) zu Zeitpunkt $t_{NOW} + t$
end procedure

Algorithmus 4.2 Sitzungsmodell: LifetimeChurn

ParetoChurm ist ein Sitzungs- und Totzeitbasiertes Modell. Es basiert auf einer Veröffentlichung von Yao et al. [151]. In diesem Modell wird die Sitzungs- und Totzeit eines Knotens in zwei Schritten bestimmt. Zunächst wird anhand der durchschnittlichen Sitzungs- beziehungsweise Totzeit mittels einer Paretoverteilung für jeden möglichen Knoten eine durchschnittliche individuelle Sitzung- beziehungsweise Totzeit errechnet. Tritt der Knoten nun der Simulation bei, wird anhand der individuellen durchschnittlichen Sitzungszeit ebenfalls mit einer Paretoverteilung eine konkrete Sitzungszeit für diese Sitzung ermittelt. Nach Ablauf der Sitzung wird für den Knoten eine Totzeit berechnet. Dies geschieht analog zu der Sitzungszeitberechnung. Nach Ablauf der Totzeit tritt der Knoten der Simulation wieder bei. Yao et al. zeigen in [151], dass mit diesem Sitzungsmodell ein Sitzungsverhalten erreicht wird, das dem in vielen Overlay-Netzen beobachteten Sitzungsverhalten der Teilnehmer nahe kommt.

Algorithmus 4.3 zeigt die für die Implementierung des ParetoChurn-Modells wichtigen Prozeduren. Als Zufallsverteilung wird gemäß Yao et al. [151] die verschobene Paretoverteilung Pareto(α, β) verwendet. Als Wert für α wird gemäß der Veröffentlichung die Zahl 3 verwendet. Das β wird aus dem jeweils erwünschten Erwartungswert ermittelt. Zur Verbesserung der Lesbarkeit wird in dem Algorithmus der Aufruf der Zufallsverteilung als Pareto(α, E) dargestellt, wobei E den gewünschten Erwartungswert darstellt. Vor Beginn der Simulation werden mit der Prozedur PreInit die individuellen mittleren Lebenszeiten der Knoten berechnet. Dabei wird der Anteil der Lebenszeit an der Gesamtzeit berechnet und über einen Zufallswurf bestimmt,

ob der Knoten in der Initialisierungsphase aktiv oder inaktiv sein soll. Die Initialisierungsphase verläuft ähnlich zu dem LifetimeChurn-Modell. Für jeden als aktiv markierten Knoten wird KnotenHinzufügen aufgerufen; bei inaktiven Knoten wird nach einer aus der durchschnittlichen individuellen Totzeit zufällig erwürfelten Zeit ein Aufruf von KnotenHinzufügen vorgemerkt. In der Initialisierungsphase wird bei der Ermittlung der Lebens- und Totzeiten jedoch für den Parameter α der Paretoverteilung der Wert 2 genommen. Yao et al. zeigen, dass sich so die erwartete Restlebenszeit beziehungsweise Resttotzeit ermitteln lässt [151]. So ergeben sich auch in der Initialisierungsphase Lebens- und Totzeiten, die dem durch die Wahrscheinlichkeitsverteilung vorgegebenen Muster folgen; eine Einschwingzeit ist somit im Gegensatz zum LifetimeChurn-Modell nicht nötig.

Vorbedingung: $\bar{t}$:= Durchschnittliche Lebenszeit
Vorbedingung: $\bar{d}$:= Durchschnittliche Totzeit
Vorbedingung: $\bar{n}$:= Durchschnittliche Knotenanzahl
Vorbedingung: t_{NOW} := aktuelle Zeit
procedure PreInit
  $n \leftarrow 0$ ▷ Zahl aktiver Knoten
  $i \leftarrow 0$ ▷ Zähler aller Knoten
  while $n < \bar{n}$ **do**
    $t_i \leftarrow$Zufallszahl(Pareto$(3, \bar{t})$)
    $d_i \leftarrow$Zufallszahl(Pareto$(3, \bar{d})$)
    Anteil Lebenszeit := $\frac{t_i}{t_i + d_i}$
    if Zufallszahl(Gleichverteilung(0,1)) $\leq$ Anteil Lebenszeit **then**
      Markiere Knoten i für Initphase als aktiv
      $n \leftarrow n + 1$
    else
      Markiere Knoten i für Initphase als inaktiv
    end if
    $i \leftarrow i + 1$
  end while
end procedure
procedure KnotenHinzufügen(k)
  ErzeugeKnoten(k)
  $t \leftarrow$Zufallszahl(Pareto$(3, t_k)$)
  Aufruf von KnotenEntfernen(k) zu Zeitpunkt $t_{NOW} + t$
end procedure
procedure KnotenEntfernen(k)
  ZerstöreKnoten(k)
  $d \leftarrow$Zufallszahl(Pareto$(3, d_k)$)
  Aufruf von KnotenHinzufügen(k) zu Zeitpunkt $t_{NOW} + d$
end procedure

Algorithmus 4.3 Sitzungsmodell: ParetoChurn

Mit dem *TraceChurn*-Modell ist es in OverSim möglich, aufgezeichnete Sitzungen von Nutzern existierender Overlay-Netze als Basis für die Simulation zu verwenden. Hierzu werden die Zeitpunkte der Bei- und Austritte von Knoten in dem existierenden Overlay-Netz in einer Datei gespeichert. OverSim liest während der Simulation die Datei ein und fügt zu den entsprechenden Zeitpunkten Knoten in die Simula-

tion ein oder entfernt Knoten aus der Simulation. Dies ermöglicht die Verwendung besonders realistischer Sitzungszeiten in OverSim.

Eine weitere Verwendungsmöglichkeit von TraceChurn ist das Erzeugen von Sitzungszeiten zu speziellen Szenarien, für die keines der obigen Sitzungsmodelle geeignet ist. Für solche Szenarien kann eine Datei mit passenden Sitzungszeiten generiert und über TraceChurn als Basis für die Simulation verwendet werden.

Ein wichtiger Aspekt des Sitzungsverhaltens ist das Ausfallverhalten der Knoten. In Overlay-Netzen sind Knotenausfälle üblich. Diese können beispielsweise durch Programmabstürze oder Abbruch der Internetverbindung verursacht werden. In OverSim kann der Anteil an Sitzungen konfiguriert werden, der durch einen Ausfall statt durch eine geordnete Abmeldung beendet wird. Bei einer geordneten Abmeldung erhält der Knoten die Möglichkeit, andere Knoten über seinen bevorstehenden Netzaustritt zu informieren. Bei einem Ausfall wird hingegen ohne Vorwarnung das Empfangen und Senden von Nachrichten unterbunden.

4.8.2 Applikationsverhalten

Neben dem Sitzungsverhalten der Nutzer muss ein Simulationswerkzeug für Overlay-Netze die Nutzung der Anwendung simulieren. Beispiele für das zu simulierende Nutzungsverhalten sind die Frequenz der Nutzung bestimmter Dienste, die Anzahl und Größe der versendeten Anwendungsnachrichten oder – im Kontext virtueller Welten – das Bewegungsverhalten in der virtuellen Welt.

Dieses Verhalten ist anwendungsabhängig und muss somit von jeder Anwendung individuell implementiert werden. Dabei kann jedoch auf aufgezeichnete Sitzungen zurückgegriffen werden. In Verbindung mit dem oben beschriebenen TraceChurn-Modell können aufgezeichnete Nutzeraktionen an die Anwendungsimplementierung in OverSim weitergeleitet werden. Alternativ können in Verbindung mit der Echtnetzanbindung Aktionen eines Nutzers in Echtzeit in der Anwendung in OverSim verwendet werden.

Für Simulationen von virtuellen Welten bietet OverSim verschiedene Bewegungsmodelle. Das einfachste Bewegungsmodell ist das *RandomRoaming*. Dies ist ein einfaches *Random-Waypoint*-Bewegungsmodell mit konfigurierbarer Bewegungsgeschwindigkeit.

Das *GroupRoaming* erweitert dieses Modell um die Unterstützung von Gruppen. Das Modell ist vergleichbar mit dem in der Untersuchung von Ad Hoc-Netzwerken eingesetzten *Nomadic Community Mobility Model* [25]. Virtuelle Welten fördern oft die Kooperation zwischen Teilnehmern. Deshalb ist der Zusammenschluss von Teilnehmern zu Teilnehmergruppen üblich. Teilnehmer derartiger Gruppen bewegen sich in der virtuellen Welt nicht wie im Random-Waypoint-Modell unabhängig. Stattdessen setzt sich die Gruppe ein gemeinsames Ziel. Gruppenteilnehmer bewegen sich dann in Richtung des Ziels. Die Bewegung der Teilnehmer ist dabei aber nicht völlig gleichförmig. Vielmehr werden durch ein *Flocking-Modell* [114] Abweichungen der Teilnehmer von einer gradlinigen Bewegung erzeugt. In OverSim kann das GroupRoaming-Modell durch Angabe einer Gruppenmaximalgröße und der Bewegungsgeschwindigkeit der Teilnehmer konfiguriert werden.

In einigen virtuellen Welten gibt es sogenannte Hotspots. Dies sind Orte, an denen eine Vielzahl von Teilnehmern auf engem Raum beieinandersteht. Diese entstehen

beispielsweise bei lokal beschränkten Ereignissen in der virtuellen Welt oder an Orten, an denen für Teilnehmer besonders interessante Interaktionsmöglichkeiten geboten werden. Dies wird in OverSim mit dem *HotspotRoaming* simuliert. Hier kann eine Zahl an Hotspots, deren Positionen und Größen, eine minimale Verweilzeit am Hotspot sowie die Aufenthaltswahrscheinlichkeit konfiguriert werden.

Das *RealworldRoaming* ist für die Echtnetzanbindung zu verwenden und ermöglicht die Kopplung einer reellen Spieleanwendung für virtuelle Welten mit OverSim. Die Möglichkeiten der Echtnetzanbindung von OverSim werden im nächsten Abschnitt ausführlicher erläutert.

4.9 Echtnetzanbindung

Die Echtnetzanbindung in OverSim ermöglicht es, Overlay-Protokolle und Anwendungen nicht nur in einer Simulation zu evaluieren, sondern sie auch in Testnetzen wie G-Lab [123, 136] oder PlanetLab [18, 109] zu testen. Weiterhin kann durch die Anbindung externer Anwendungen eine Interaktion realer Nutzer mit der Simulation ermöglicht werden. Hierdurch lassen sich besonders realistische Testszenarien verwirklichen. Zusätzlich kann die Echtnetzanbindung genutzt werden, um Demonstrationen von Overlay-Systemen zu verbessern. So kann die in Demonstrationen üblicherweise beschränkte Anzahl physischer Geräte durch simulierte Netzteilnehmer ergänzt werden, um die Gesamtzahl der Knoten im System und damit die Realitätsnähe zu erhöhen.

Die Echtnetzanbindung wird in OverSim durch das Zusammenspiel verschiedener Module erreicht. Zunächst muss der standardmäßig verwendete Nachrichtenscheduler durch einen speziellen Echtzeitscheduler ausgetauscht werden, da in Simulationen die Simulationszeit normalerweise von der Echtzeit unabhängig ist. Dieser Scheduler ist auch für das Empfangen von Nachrichten vom externen Netzwerk zuständig. OverSim bietet zwei verschiedene Scheduler für die Echtnetzanbindung an: Der *udpOutScheduler* nutzt für die Kommunikation mit dem Netz die UDP-Schnittstelle des Betriebssystems, unterstützt jedoch nur die Netzanbindung eines einzelnen Knotens. Der *tunOutScheduler* ermöglicht über die Verwendung des Linux-TUN-Interfaces [82] die Anbindung ganzer simulierter Netze an das externe Netzwerk.

Um die vom Scheduler empfangenen Nachrichten in die Simulation einschleusen zu können, wird die Unterstützung der Underlay-Schicht benötigt. Auch hierzu bietet OverSim zwei Möglichkeiten. Das *SingleHostUnderlay* wird üblicherweise in Zusammenarbeit mit dem udpOutScheduler verwendet. Es simuliert einen einzelnen Knoten und leitet die vom Scheduler empfangenen Daten über die UDP-Schnittstelle an die Overlay-Schicht. Der *InetUnderlay* kann in Verbindung mit dem tunOutScheduler für die Echtnetzanbindung genutzt werden. Dazu wird in das Netzmodell um einen speziellen Router, den sogenannten *TunOutRouter*, erweitert. Dieser dient als Verbindung der simulierten Router zum externen Netzwerk. Vom Scheduler über das TUN-Interface empfangene Daten aus dem externen Netz werden über diesen Router in die Simulation eingespielt und über das simulierte Netz zu den Knoten geroutet. Umgekehrt werden Daten von Knoten an das externe Netz zum TunOutRouter geroutet und von dort über den tunOutScheduler an das externe Netz weitergeleitet.

Durch die Zusammenarbeit zwischen Scheduler und Anwendungsschicht in OverSim können externe Anwendungen an die Simulation angebunden werden. Hierzu verwaltet der Scheduler TCP-Verbindungen mit den externen Anwendungen und leitet

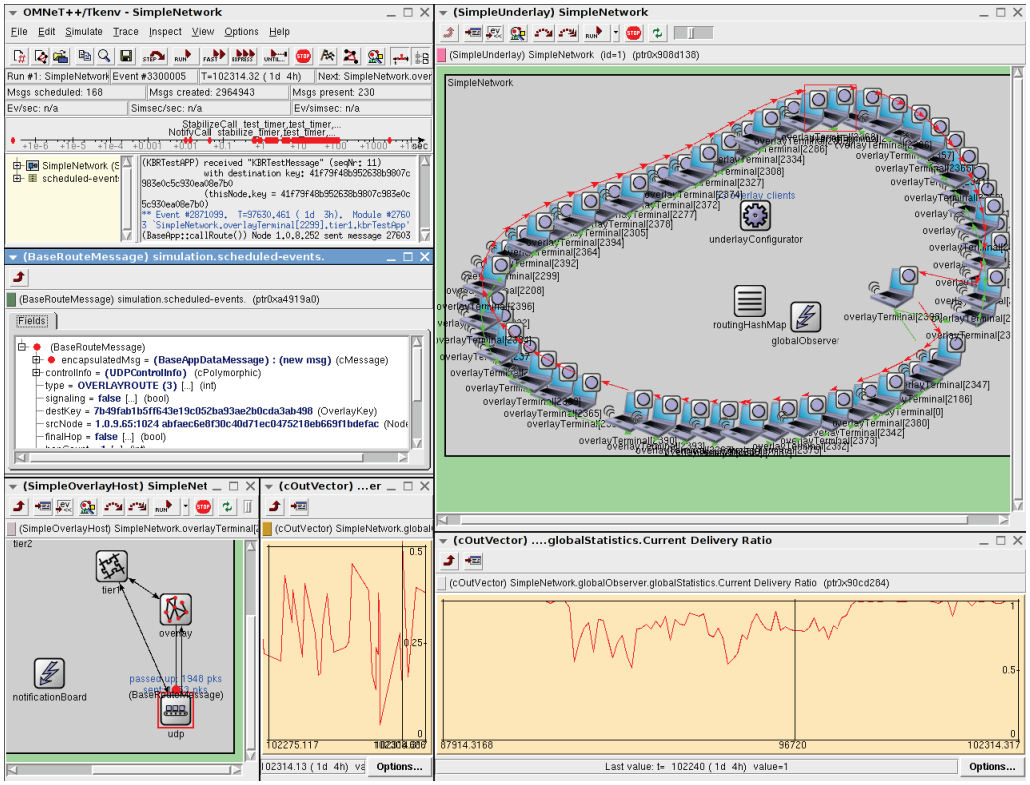


Abbildung 4.2 Die interaktive Benutzeroberfläche von OverSim.

die über die Verbindungen empfangenen Daten an die Anwendungsschicht weiter. Für einige Dienste wie DHT und KBR steht eine *XML-RPC-Schnittstelle* zur Verfügung. Über diese können alle Funktionen der jeweiligen Dienste genutzt werden. Zusätzlich können Anwendungen spezifische Schnittstellen für externe Anwendungen definieren. So bietet beispielsweise die OverSim-Anwendung für virtuelle Welten, der SimpleGameClient, den Bewegungsgenerator *RealworldRoaming*. Dieser bietet eine für virtuelle Welten geeignete Schnittstelle, sodass eine externe Spieleanwendung mit OverSim genutzt werden kann.

4.10 Interaktive Benutzeroberfläche

OverSim verwendet das von OMNeT++ angebotene *tkenv* als grafische Benutzeroberfläche. Diese bietet die Möglichkeit, interaktiv die Overlay-Topologie zu visualisieren, den Zustand der am Overlay beteiligten Knoten darzustellen, statistische Daten in Echtzeit auszugeben und die in der Ereigniswarteschlange befindlichen Nachrichten zu durchsuchen und anzuzeigen. Die grafische Oberfläche ist in Abbildung 4.2 dargestellt.

Über die Oberfläche kann der Ablauf der Simulation gesteuert werden. Es ist möglich, die Simulation zu beliebigen Zeitpunkten anzuhalten, Ereignisse in einzelnen Schritten abzuarbeiten, den Versand von Nachrichten animiert darzustellen oder die Ausgabe von Debug-Nachrichten ein- oder abzuschalten.

Die von der Benutzeroberfläche gebotenen Möglichkeiten erlauben Entwicklern ein besseres Verständnis des Protokollablaufs und eine einfache Suche nach Fehlern in der Protokollimplementierung oder im Protokollentwurf.

4.11 Evaluierung und Validierung

Um die Leistung und Skalierbarkeit von OverSim zu evaluieren, wurde eine Simulation mit einer steigenden Zahl an Knoten durchgeführt. Speicherverbrauch und Simulationsgeschwindigkeit wurden dann einer vergleichbaren Simulation des Simulators P2PSim gegenübergestellt. Das Simulationswerkzeug P2PSim wurde als Vergleichsobjekt gewählt, da es eines der meistverbreiteten bisherigen Simulationswerkzeuge ist.

In den Simulationen wurde das Protokoll Chord verwendet. Dabei wurden Chord-Stabilisierungsnachrichten alle 20 Sekunden versendet und ein Update der Fingertabelle alle 120 Sekunden durchgeführt. Um das Routing in Chord zu testen, versandten Knoten alle 10 Sekunden eine Nachricht an einen zufälligen Schlüssel.

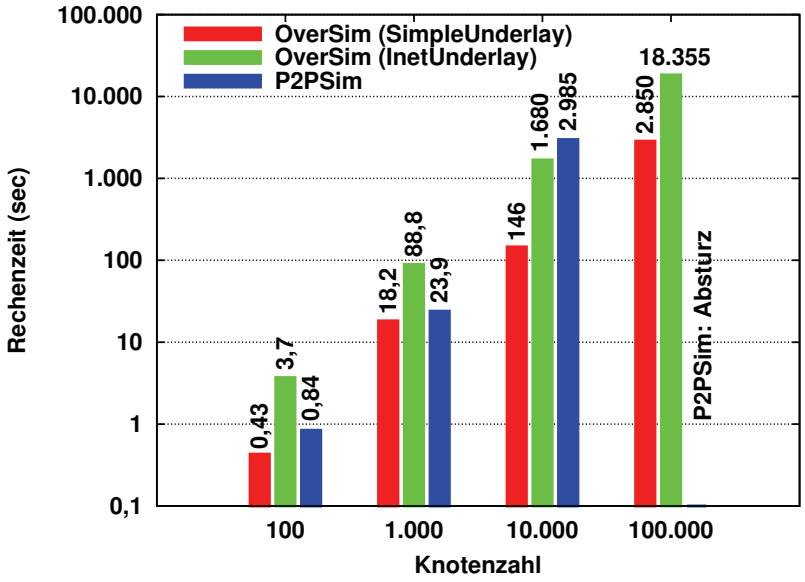
Die Simulationen wurden in OverSim mit dem SimpleUnderlay sowie dem InetUnderlay und in P2PSim mit identischen Parametern durchgeführt. Da P2PSim nur die Simulation konstanter Verzögerungen zwischen Knoten erlaubt, wurde, um die Vergleichbarkeit zu wahren, auch im SimpleUnderlay eine konstante Verzögerung zwischen den Knoten konfiguriert. Im InetUnderlay wurde eine einfache Topologie, bestehend aus 20 zufällig vernetzen Backbone-Routern und 20 mit diesen Routern verbundenen Zugangsroutern, verwendet.

Die Zahl der Knoten wurde von 100 über 1.000 und 10.000 auf 100.000 erhöht. Nach dem Beitritt der Knoten wurde das Overlay-Netz für eine Simulationszeit von 1.000 Sekunden simuliert. Dabei wurde die dafür benötigte CPU-Zeit sowie der Speicherverbrauch gemessen. Die Simulationen wurden auf einem Opteron 1,8 GHz mit 8 GB Hauptspeicher durchgeführt.

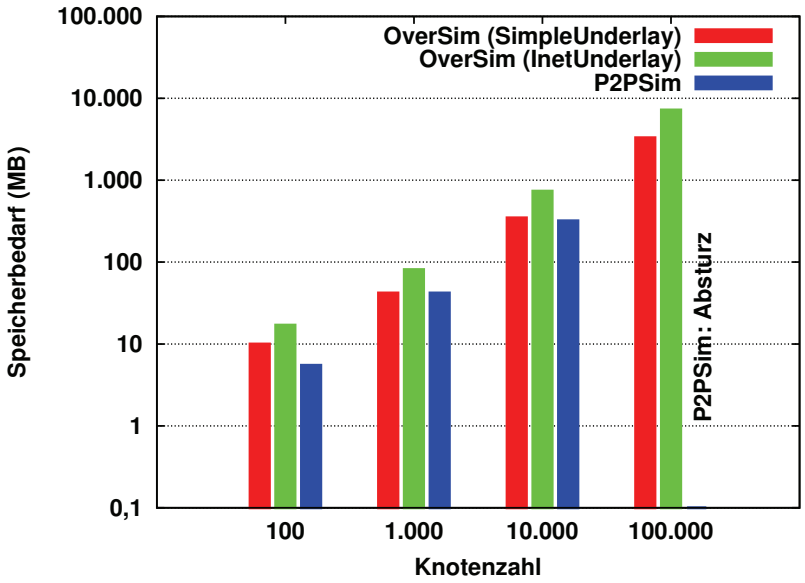
Abbildung 4.3 zeigt die Ergebnisse der Evaluierung. In Abbildung 4.3a ist mit einer logarithmischen Skala die CPU-Zeit aufgezeigt, die für die Simulation von 1.000 Sekunden bei der gegebenen Knotenzahl benötigt wurde. Die Simulationen mit weniger als 1.000 Knoten wurden von beiden Simulationswerkzeugen in wenigen Sekunden durchgeführt. Dabei benötigte OverSim mit Verwendung des SimpleUnderlays stets die geringste Rechenzeit. Da im InetUnderlay das unterliegende Netzwerk mit einem deutlich höheren Detailgrad simuliert wird, war die benötigte Rechenzeit entsprechend höher. Im evaluierten Szenario benötigte das InetUnderlay ungefähr die zehnfache Rechenzeit des SimpleUnderlays. P2PSim benötigte in den Szenarien mit 100 und 1.000 Knoten ungefähr die doppelte Rechenzeit im Vergleich zu OverSim mit dem SimpleUnderlay.

Bei der Simulation mit 10.000 Knoten benötigte OverSim mit dem SimpleUnderlay eine Rechenzeit von 146 Sekunden. OverSim mit dem InetUnderlay benötigte für die Simulation 28 Minuten und damit wieder ungefähr die zehnfache Zeit. P2PSim benötigte mit 50 Minuten deutlich länger als OverSim mit dem InetUnderlay und ungefähr 20 mal länger als das vom Detailgrad vergleichbare OverSim mit dem SimpleUnderlay.

Die Simulation mit 100.000 Knoten ließ sich mit P2PSim nicht durchführen. Nach mehreren Stunden Rechenzeit brach P2PSim die Simulation mit einem Programmabsturz ab. OverSim mit dem SimpleUnderlay beendete die Simulation in 47 Minuten, OverSim mit dem InetUnderlay in ungefähr 5 Stunden.



(a) Rechenzeit in Abhängigkeit der Knotenzahl.



(b) Speicherbedarf in Abhängigkeit der Knotenzahl.

Abbildung 4.3 Rechenzeit und Speicherverbrauch von OverSim und P2PSim.

Die Ergebnisse zeigen, dass OverSim im Gegensatz zu P2PSim skalierbar ist. Die Simulationszeiten mit OverSim steigen linear mit der Zahl der Ereignisse, die im simulierten Szenario aufgrund der Skalierungseigenschaften des simulierten Protokolls Chord bei $O(n \log(n))$ liegt. Dies gilt sowohl für das SimpleUnderlay als auch für das InetUnderlay, wobei letzteres aufgrund des höheren Detailgrades des Netzmodells ungefähr die zehnfache Laufzeit des SimpleUnderlays benötigt. Die Simulationszeit von P2PSim steigt hingegen deutlich überlinear. Bei 100.000 Knoten kommt es bei P2PSim sogar zu einem Programmabsturz.

Abbildung 4.3b zeigt den Speicherverbrauch der Simulationen. Dieser steigt bei beiden Simulatoren in etwa linear mit der Knotenzahl. Sowohl OverSim mit dem SimpleUnderlay als auch P2PSim belegen etwa 25 kB Hauptspeicher pro simulier-

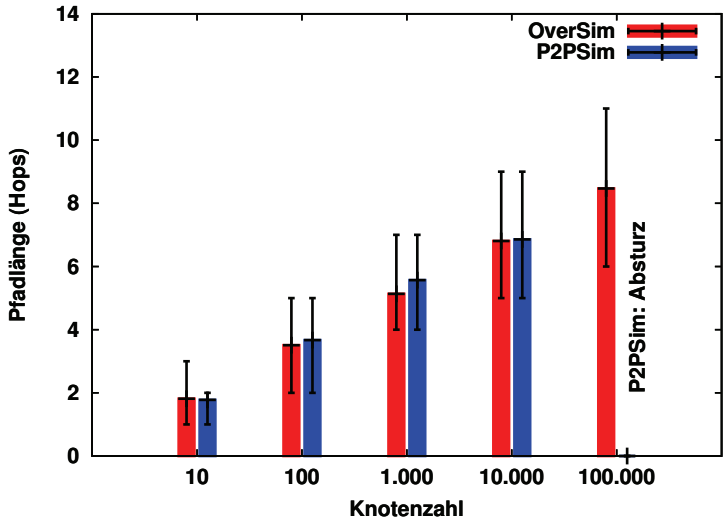


Abbildung 4.4 Durchschnittliche Pfadlänge in Abhängigkeit der Knotenzahl.

tem Knoten. Mit dem InetUnderlay benötigt OverSim ungefähr die doppelte Menge Speicher.

Um die Vergleichbarkeit der Simulationen zu belegen, wurden verschiedene Metriken gemessen. Exemplarisch ist in Abbildung 4.4 die durchschnittliche Pfadlänge in Abhängigkeit zur Knotenzahl zusammen mit dem 95%-Quantil der gemessenen Pfadlängen gezeigt. Die von OverSim und P2PSim ermittelten Pfadlängen wuchsen logarithmisch mit der Knotenzahl. Dies entspricht dem für Chord erwarteten Verhalten [130]. Die gemessenen Pfadlängen von OverSim und P2PSim unterscheiden sich nur geringfügig. Auch bei anderen Metriken, wie dem Bandbreitenbedarf für die Ringstabilisierung, waren die Messwerte der beiden Simulatoren vergleichbar.

4.12 Zusammenfassung

In diesem Kapitel wurde mit OverSim ein neues Simulationsrahmenwerk für Overlay-Netze vorgestellt. OverSim erfüllt wichtige Anforderungen, die von bisherigen Simulationswerkzeugen bislang vernachlässigt wurden. Insbesondere wurden die Anforderungen Skalierbarkeit, Flexibilität, realistische Netzmodellierung, realistisches Nutzer- und Anwendungsverhalten, Echtnetzanbindung, Erstellung aussagekräftiger Statistiken, Visualisierung und Dokumentation und Erweiterbarkeit betrachtet.

OverSim erfüllt diese Anforderungen besser als bisherige Simulationswerkzeuge. Aufgrund eines modularen Aufbaus bleibt OverSim flexibel; alle Module einer Simulation können transparent durch andere Module ersetzt werden. Insbesondere kann auch die Netzabstraktion durch verschiedene Modellierungen realisiert werden. Die verschiedenen Modellierungen unterscheiden sich durch den Detailgrad der Modellierung. Bereits mit dem einfachen *SimpleUnderlay* lässt sich das unterliegende Netz realistisch abstrahieren. Zugleich zeigt die Evaluierung, dass dieses Netzmodell skalierbare Simulationen ermöglicht.

Durch die von OverSim gebotenen Hilfs- und Basisklassen lassen sich neue Protokolle und Anwendungen einfach implementieren. Dabei hilft OverSim auch bei der

Erstellung von Statistiken oder der Visualisierung der Overlay-Topologie. Spezielle Modelle ermöglichen eine Anbindung von OverSim an reale Netzwerke.

Somit ist OverSim ein geeignetes Werkzeug für die Entwicklung und Evaluierung neuer P2P-Protkolle. Im weiteren Verlauf der Arbeit wird OverSim für die Simulation der vorgestellten Protokolle verwendet.

OverSim wurde als Open-Source-Software auf der Webseite www.oversim.org veröffentlicht. Seit der Veröffentlichung wird es von vielen nationalen und internationalen Forschungseinrichtungen zur Entwicklung und Evaluierung von Overlayprotokollen eingesetzt.

5. Analyse selektierter P2P-Protokolle für verteilte virtuelle Welten

In Kapitel 3 wurden bisherige Vorschläge für P2P-Protokolle für verteilte virtuelle Welten vorgestellt. Im Folgenden sollen nun einige dieser Protokolle genauer beschrieben werden. Bislang fehlt in der Literatur ein belastbarer Vergleich bisheriger Protokolle; die von den Autoren der einzelnen Protokolle ermittelten Ergebnisse sind in der Regel nicht unter einheitlichen Szenarien gewonnen worden und so für einen direkten Vergleich unbrauchbar. In diesem Kapitel soll deshalb die Leistungsfähigkeit der ausgewählten Protokolle evaluiert und verglichen werden.[1] Die Evaluierung zeigt, dass die Protokolle nicht alle der in Abschnitt 2.1.3 genannten Anforderungen virtueller Welten erfüllen können. Das Ziel der Evaluierung ist es, die Leistung bisheriger Protokolle zu vergleichen und Probleme der Protokolle zu identifizieren.

5.1 Protokollauswahl

In Kapitel 3 wurden bisherige Protokollvorschläge in zonenbasierte und zonenlose Protokolle unterteilt. Bei zonenbasierten Protokollen wurde zwischen Protokollen mit statischen und Protokollen mit dynamischen Zonen unterschieden. Protokolle mit statischen Zonen können wiederum *Application Layer Multicast*-basiert oder *Supernode*-basiert sein. Die relevanten Kategorien sind also:

- *Statische Zonen, ALM*
- *Statische Zonen, Supernode*
- *Dynamische Zonen*
- *Zonenlos*

[1] Eine vergleichbare Evaluierung wurde 2008 vom Autor dieser Arbeit veröffentlicht [83].

Aus jeder dieser Kategorien wurde ein Protokoll ausgewählt, das die für die jeweilige Kategorie typischen Eigenschaften deutlich zeigt oder das in der jeweiligen Kategorie als Stand der Technik gilt. Gewählt wurden:

- Statische Zonen, ALM: *SimMUD*. ALM-basierte Protokolle mit statischen Zonen unterscheiden sich im Wesentlichen durch die Auswahl des verwendeten ALM-Protokolls. SimMUD basiert auf dem ALM-Protokoll Scribe [26], welches als Stand der Technik von Key-based-Routing(KBR)-basierten ALM-Protokollen gilt.
- Statische Zonen, Supernode: *PubSubMMOG*. Bei Supernode-basierten Protokollen mit statischen Zonen liegt der wesentliche Unterschied in der Behandlung der Lastverteilung. In PubSubMMOG wird diese über einen Lastverteilbaum erreicht. Dies entspricht dem Stand der Technik dieser Protokolle.
- Dynamische Zonen: *N-Tree*. Die meisten Vorschläge für Protokolle für virtuelle Welten mit dynamischen Zonen basieren auf Serverclustern anstatt von P2P. N-Tree ist einer der wenigen P2P-basierten Vorschläge. Es verspricht besonders geringe Latenzen.
- Zonenlos: *Vast*. Vast ist einer der Vorreiter von zonenlosen Protokollen. Es ist Stand der Technik bei zonenlosen Protokollen für virtuelle Welten. Andere Vorschläge für zonenlose Protokolle basieren ebenso wie Vast auf Voronoi-Graphen oder den dazu dualen Delauney-Graphen.

Im Folgenden werden diese Protokolle näher vorgestellt und in verschiedenen Szenarien evaluiert.

5.1.1 SimMUD

SimMUD [79] ist ein zonenbasiertes P2P-Protokoll für verteilte virtuelle Welten. In SimMUD wird die virtuelle Welt in statische Zonen unterteilt. Für die Verteilung von Nachrichten wird *Application Layer Multicast* eingesetzt.

5.1.1.1 Funktionsweise

Als *Application Layer Multicast*-Protokoll wird bei SimMUD das auf dem Overlayprotokoll Pastry [121] aufsetzende Scribe [26] eingesetzt.

Pastry ist ein strukturiertes Overlayprotokoll, welches seinen Teilnehmern einen Key-Based-Routing-Dienst (kurz: KBR, siehe hierzu auch Abschnitt 2.2.3) zur Verfügung stellt. Jedem Teilnehmer in Pastry wird ein eindeutiger, zufälliger Schlüssel zugeordnet. Nachrichten können an beliebige Schlüssel adressiert werden und werden dann von Pastry zu dem Teilnehmer geroutet, dessen Schlüssel dem Zielschlüssel der Nachricht am nächsten ist. Bei einer Teilnehmerzahl n erfolgt die Zustellung in Pastry in $O(\log(n))$ Routingschritten.

Hierzu hält jeder Knoten in einem Pastry-Overlay eine Routingtabelle mit Adressen möglicher Weiterleitungsziele. Um die Routingtabelle zu bestimmen, wird der Schlüssel eines Teilnehmers in n Stellen zu b bit aufgeteilt. Die Routingtabelle enthält dann n Spalten. Die Spalte i enthält die Adressen von Knoten, deren Schlüssel

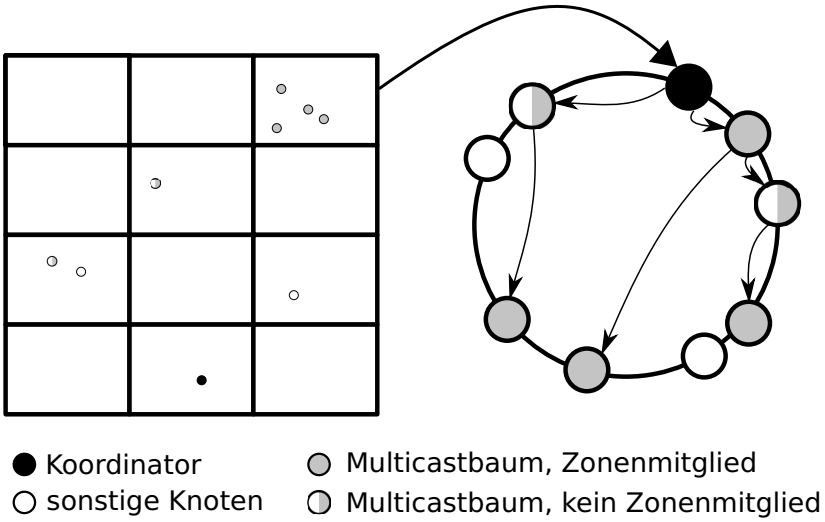


Abbildung 5.1 SimMUD: Zonenaufteilung der Spielwelt und Multicastbaum einer Zone.

ein gemeinsames Präfix der Länge i mit dem Teilnehmer haben und sich in der Stelle $i+1$ unterscheiden. Dabei wird jede Spalte so gefüllt, dass für jeden möglichen Wert der Stelle $i+1$ ein Knoten bekannt ist.

Weiterhin besitzt jeder Teilnehmer ein sogenanntes *Leaf-Set*, in dem die Knoten mit den dem Teilnehmer am nächsten liegenden Schlüsseln gespeichert sind. Erhält ein Knoten in Pastry eine Nachricht für einen Schlüssel, prüft er zunächst, ob er selbst oder einer der Knoten aus seinem Leaf-Set für diesen Schlüssel verantwortlich ist. Ist dies nicht der Fall, leitet er die Nachricht an den Knoten aus der Routingtabelle weiter, der dem Zielschlüssel am nächsten ist. So wird die Nachricht rekursiv bis zu ihrem Ziel weitergeleitet.

Scribe ist ein auf Pastry aufsetzendes Application-Layer-Multicast-Protokoll. Jede Multicast-Gruppe in Scribe hat einen Pastry-Schlüssel als Identifikator. Basierend auf diesem Identifikator wird nun ein Multicastbaum aufgespannt. Dazu wird von allen Mitgliedern der Gruppe ausgehend ein Paket zu dem Identifikator der Gruppe über Pastry geroutet. Die Zwischenknoten, die auf dem Weg der Pakete liegen, bilden dann zusammen mit den Gruppenmitgliedern den Multicastbaum. Die Tiefe des Multicastbaumes ergibt sich so aus der Pfadlänge in Pastry und beträgt somit bei n Teilnehmern $O(\log(n))$. Nachrichten, die an eine Multicastgruppe verteilt werden sollen, werden zunächst dem Wurzelknoten des Multicastbaumes geschickt und von dort aus über den Multicastbaum weitergeleitet.

Da in SimMUD die Zonenaufteilung statisch und vorab bekannt ist, kann für jede Zone eine eigene dauerhaft bestehende Multicast-Gruppe eingerichtet werden. Der Wurzelknoten der jeweiligen Multicastgruppe ist zugleich Koordinator der jeweiligen Zone. Abbildung 5.1 stellt diese beispielhaft dar: Gezeigt wird die Aufteilung der Spielwelt in 3 mal 4 Zonen. Dazu ist der Multicastbaum der rechten oberen Zone abgebildet. Die Wurzel des Multicastbaumes und damit der Koordinator der Zone ist derjenige Knoten, dessen Schlüssel dem Schlüssel der Multicastgruppe am nächsten ist. Der Multicastbaum wird aus den Mitgliedern der Zone gebildet sowie weiteren Knoten, die auf den Pastry-Routingpfaden der Teilnehmer zum Identifikator der Multicastzone liegen.

Betritt ein Teilnehmer die Zone, wird er Mitglied der Multicastgruppe. Über diese werden alle Ereignisse, die innerhalb der Zone auftreten, allen Teilnehmern innerhalb der Zone mitgeteilt. Dazu sendet der Verursacher des jeweiligen Ereignisses eine Nachricht an den Wurzelknoten der Gruppe. Von dort aus werden die Ereignisnachrichten über den Multicastbaum an die Mitglieder der Gruppe verteilt. Insbesondere sendet jeder Teilnehmer regelmäßig eine Positionsmeldung an die zu seiner Zone gehörige Multicastgruppe.

Um einer SimMUD-basierten virtuellen Welt beizutreten, muss ein Teilnehmer zunächst Mitglied im Pastry-Overlay werden. Daraufhin kann er der Scribe-Multicastgruppe der zu seiner gewünschten Position gehörigen Zone beitreten. Mit dem Versenden einer Positionsmeldung an diese Gruppe werden die bisherigen Mitglieder der Gruppe über den Beitritt in die virtuelle Welt informiert. Das Verlassen des Netzes erfolgt durch das Verlassen aller Multicastgruppen und dem anschließenden Verlassen des Pastry-Overlays.

Da Pastry und Scribe bereits den Ausfall von Knoten kompensieren können, benötigt SimMUD keine zusätzlichen Mechanismen, um bei Knotenausfall den Multicastbaum oder die Nachrichtenverteilung wiederherzustellen. Um den Ausfall eines Zonenkoordinators kompensieren zu können, muss für diese Aufgabe jedoch ein Backupknoten ernannt werden. Diese Aufgabe erfüllt der Knoten, der dem Schlüssel der jeweiligen Multicastgruppe am zweitnächsten ist. Fällt der Koordinator der Zone aus, ist dieser Knoten nun dem Schlüssel der Multicastgruppe für die Zone am nächsten und wird so automatisch neuer Koordinator. Die Kindknoten des ausgefallenen Knoten treten der Gruppe neu bei, sobald sie den Ausfall ihres Vaterknotens erkannt haben. So wird der Multicastbaum automatisch repariert.

5.1.1.2 Implementierung

Um SimMUD evaluieren zu können, wurde es in OverSim implementiert. Dabei wurde auf die in OverSim verfügbare Pastry- und Scribe-Implementierung zurückgegriffen.

Da es bei einer Aufteilung der Welt in Zonen regelmäßig dazu kommt, dass sich Teilnehmer nahe einer Zonengrenze befinden und somit deren jeweiliges Interessengebiet in die benachbarte Zone hinüberreicht, wurde es Teilnehmern ermöglicht, Mitglied in den Multicastgruppen mehrerer Zonen zu werden. Dies ist in dem Vorschlag von Knutsson et al. [79] als optional vorgesehen.

Neben der Behandlung von Ereignis- und Positionsmeldungen erlaubt SimMUD auch die Speicherung und Verwaltung von Zuständen von Objekten, die in der virtuellen Welt platziert werden können. Im Rahmen dieser Arbeit wird jedoch zunächst nur betrachtet, wie Ereignis- und Positionsmeldungen effizient in einem P2P-System an Teilnehmer einer virtuellen Welt verteilt werden können. Deswegen ist die Verwaltung von Objektzuständen hier nicht implementiert.

5.1.1.3 Aufwandsabschätzungen

Die Nachrichtenverzögerung der Positionsmeldungen und Ereignisnachrichten in SimMUD ist durch die Weiterleitung über den Multicastbaum von der Tiefe dieses Baumes abhängig. Diese ist aufgrund der Konstruktion des Baumes durch Scribe ausschließlich von der Gesamtzahl n der Teilnehmer der virtuellen Welt abhängig

und steigt mit $O(\log(n))$. Bei kleineren virtuellen Welten dürfte die daraus resultierende Nachrichtenverzögerung für übliche Interaktionen ausreichend bleiben, bei großen virtuellen Welten sind signifikante Verzögerungen zu erwarten.

Der Bandbreitenbedarf von SimMUD setzt sich aus dem Aufwand für den Versand von Bewegungsnachrichten, Durchführung von Gruppenwechseln und Management der Multicastbäume zusammen. Bei m Teilnehmern pro Zone müssen die Nachrichten von m Knoten über den Multicastbaum der Tiefe $O(\log(n))$ weitergeleitet werden. Es ergibt sich ein Gesamtaufwand von $O(m \log(n))$. Der mit dem Gruppenwechsel eines Teilnehmers einhergehende Beitritt einer neuen Multicastgruppe ist mit dem Versand von $O(\log(n))$ Nachrichten verbunden. Für die Wartung der Multicastbäume sind ebenfalls $O(\log(n))$ Nachrichten erforderlich. Insgesamt ist der Verwaltungsoverhead von SimMUD gering, der Overhead für den Versand der Positionsmeldungen ist akzeptabel.

Wesentliche Einflüsse auf die Nachbarschaftskenntnis von SimMUD ergeben sich aus der benötigten Zeit für den Gruppenwechsel sowie der Geschwindigkeit der Ausfallerkennung und Reparatur der Multicastbäume. Der Gruppenwechsel benötigt einen Zeitaufwand von $O(\log(n))$. Bei großen virtuellen Welten oder bei häufigem Gruppenwechsel können sich dadurch bereits Einschränkungen in der Nachbarschaftskenntnis ergeben. Die Ausfallerkennung in SimMUD ist an die unterliegenden Schichten Scribe und Pastry delegiert. Die durchschnittliche Erkennungszeit beträgt bei üblicher Parametrisierung dieser Protokolle ungefähr 3 Sekunden. Dies ist für virtuelle Welten relativ langsam. Da kein Backup-Mechanismus vorhanden ist, ist der Multicastbaum erst dann vollständig wiederhergestellt, wenn alle Kindknoten des ausgefallenen Knotens den Ausfall erkannt haben und den neuen Vaterknoten gefunden haben.

5.1.2 PubSubMMOG

Yamamoto et al. schlugen 2005 [149] ein Publish/Subscribe-basiertes Protokoll für virtuelle Welten vor. Da in diesem Vorschlag kein eigener Name für dieses Protokoll genannt wird, soll der Vorschlag hier als *PubSubMMOG* bezeichnet werden. Analog zu SimMUD wird in PubSubMMOG die virtuelle Welt in statische Zonen unterteilt. Die Verteilung von Nachrichten innerhalb einer Zone erfolgt über Publish/Subscribe.

5.1.2.1 Funktionsweise

Die Publish/Subscribe-Funktionalität wird in PubSubMMOG durch *Supernodes* bereitgestellt: Jede Zone bekommt einen solchen Knoten als Koordinator zugewiesen. Bei diesem können Teilnehmer, die sich in der jeweiligen Zone befinden, Ereignisnachrichten und Positionsmeldungen für diese Zone abonnieren sowie ihre Ereignisnachrichten und Positionsmeldungen publizieren. Der Koordinator aggregiert die Nachrichten für die Zone und leitet sie periodisch an alle Abonnenten weiter. Befindet sich ein Teilnehmer in der Nähe einer Zonengrenze, sodass mehrere Zonen innerhalb seines Interessengebiets liegen, kann er die Nachrichten aller für ihn interessanten Zonen abonnieren.

Wird die Zahl der Abonnenten zu groß, kann der Koordinator das Verteilen der Nachrichten an einen Lastverteilbaum delegieren. Hierzu wird ein balancierter k-ärer Baum durch sogenannte *Intermediate Nodes* gebildet. Die Tiefe des Lastverteilbaumes einer Zone beträgt dementsprechend bei m Zonenmitgliedern $O(\log_k(m))$. Die

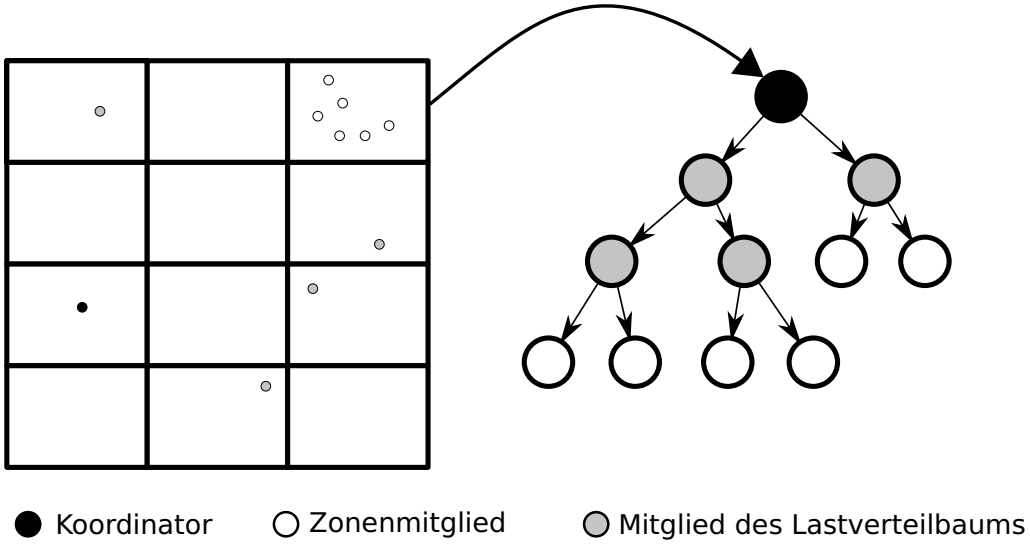


Abbildung 5.2 PubSubMMOG: Zonenaufteilung der Spielwelt und Lastverteilbaum einer Zone.

Abonnenten der Nachrichten bilden die Blätter dieses Baumes. Dies ist exemplarisch in Abbildung 5.2 gezeigt: Die Spielwelt ist in statische Zonen unterteilt. Für eine der Zonen ist der Lastverteilbaum mit dem Zonenkoordinator als Wurzel und den Mitgliedern der Zone als Blätter dargestellt.

Alle Aufgaben, also insbesondere die Aufgabe des Koordinators und des Intermediate Nodes, werden durch einen zentralen *Lobby-Server* vergeben. Dieser beobachtet die Lastsituation aller Teilnehmer und wählt für die Aufgaben jeweils Teilnehmer mit möglichst geringer Last aus.

Weiterhin versucht der Lobby-Server, die durch den Lastverteilbaum erzeugte Verzögerung beim Verteilen der Ereignisnachrichten zu minimieren. Hierzu kann er Intermediate Nodes, die eine hohe Latenz besitzen, durch andere Teilnehmer austauschen. Auf diese Weise wird der Lastverteilbaum kontinuierlich angepasst.

Um den Ausfall eines Koordinators möglichst schnell kompensieren zu können, wird für jeden Koordinator ein Backup-Knoten bestimmt. Dieser erhält alle Daten des Koordinators und kann so im Falle eines Ausfalls vom Lobby-Server unmittelbar als neuer Koordinator eingesetzt werden.

5.1.2.2 Implementierung

Um eine Evaluierung von PubSubMMOG zu ermöglichen, wurden Lobby-Server und P2P-Protokoll in OverSim implementiert. In Abweichung zu der Publikation von Yamamoto et al. [149] wurde die Last der Teilnehmer vom Lobby-Server nur abgeschätzt. Dies ist in einer Simulationsumgebung präzise möglich und hat somit nur einen geringen Einfluss auf das Protokoll.

Die Publikation von Yamamoto et al. sah vor, dass für die Aggregation der Ereignisnachrichten die Spielzeit in Runden aufgeteilt werden sollte. Der Koordinator sollte die erste Hälfte einer solchen Runde auf Nachrichten warten und dann die bislang empfangenen Nachrichten weiterverteilen. Beim Koordinator eintreffende Nachrichten, die noch zu einer früheren Runde gehörten, sollten ebenso verworfen werden

wie Nachrichtenlisten, die erst nach Ablauf der Runde bei den Teilnehmern eintrafen. Dies hat jedoch zur Folge, dass die Anzahl der Runden pro Sekunde durch die Nachrichtenverzögerung der Ereignisnachrichten, die sich in PubSubMMOG durch die Latenz der Teilnehmer und die Anzahl der Zwischenknoten im Lastverteilbaum ergibt, begrenzt wird. Bei realistischen Annahmen für diese Werte beträgt die durchschnittliche Nachrichtenverzögerung zwischen 300 und 350 ms[2]. So sind maximal zwei Runden pro Sekunde möglich, da ansonsten zu viele Nachrichten verworfen werden. Da in virtuellen Welten das Versenden von sechs Positionsmeldungen pro Sekunde üblich ist [30, 134], ist dies nicht ausreichend. Abweichend zur Publikation von Yamamoto et al. wurde in dieser Implementierung also das strikte Rundenkonzept durchbrochen: Der Koordinator aggregiert in der Implementierung für eine gewisse Zeit Nachrichten und leitet sie unabhängig von ihrem Alter periodisch weiter. Ebenso werden Nachrichtenlisten immer weitergeleitet und unter keinen Umständen verworfen. Die Änderungen erlauben eine kurze Rundenzeit ohne die Gefahr des Verwerfens von Nachrichten. Ohne diese Änderungen würde PubSubMMOG entweder eine deutlich höhere Verzögerung durch lange Rundenzeiten aufweisen oder aufgrund einer großen Zahl verworfener Nachrichten nur eine schlechte Nachbarskenntnis bieten können.

5.1.2.3 Aufwandsabschätzungen

Die Nachrichtenverzögerung in PubSubMMOG ergibt sich aus der Wartezeit für die Nachrichtenaggregation im Koordinator und der Weiterleitung der Nachrichten über den Lastverteilbaum. Da ein k-närer Lastverteilbaum in PubSubMMOG eine Tiefe von $O(\log_k(m))$ besitzt, beträgt die Verzögerung durch den Lastverteilbaum $O(\log_k(m))$. Befinden sich nur wenige Teilnehmer in einer Zone, ist die resultierende Nachrichtenverzögerung akzeptabel, doch bei Zonen mit vielen Teilnehmern kann die zu erwartende Verzögerung die Interaktionen zwischen Teilnehmern behindern.

Der Bandbreitenbedarf in PubSubMMOG setzt sich aus dem Bedarf für das Versenden der Positionsmeldungen, dem Zonenwechsel, der Kommunikation mit Backupknoten und der Wartung der Lastverteilbäume zusammen. Für das Verteilen einer Positionsmeldung an die m Mitglieder einer Zone über einen Lastverteilbaum der Tiefe $O(\log_k(m))$ ergibt sich ein Aufwand von $O(m \log_k(m))$. Gruppenwechsel können durch die Verwendung des zentralen Servers in $O(1)$ durchgeführt werden. Um dem Backupknoten alle Änderungen der Zonenmitgliedschaften zu kommunizieren, sind $O(m)$ Nachrichten nötig. Die Wartung der Lastverteilbäume erfordert $O(\log(m))$ Nachrichten. Insgesamt ist der Protokolloverhead für Verwaltungsaufgaben gering. Der Bandbreitenbedarf für die Verteilung der Positionsmeldungen ist akzeptabel.

Einflüsse auf die Nachbarschaftskenntnis könnten sich in PubSubMMOG durch die benötigte Zeit für den Zonenwechsel und die Ausfallerkennung ergeben. Da der Zonenwechsel in $O(1)$ durchgeführt werden kann und die Ausfallerkennung einen Knotenausfall mit regelmäßigen Heartbeat-Nachrichten schnell erkennen kann[3], ist eine sehr gute Nachbarschaftskenntnis zu erwarten.

[2] Vergleiche auch die Evaluierung der Nachrichtenverzögerung in Abschnitt 5.2.4.1.

[3] Bei Verwendung der vom Autor vorgeschlagenen Zeitintervalle ergibt sich eine durchschnittliche Erkennungszeit von einer Sekunde.

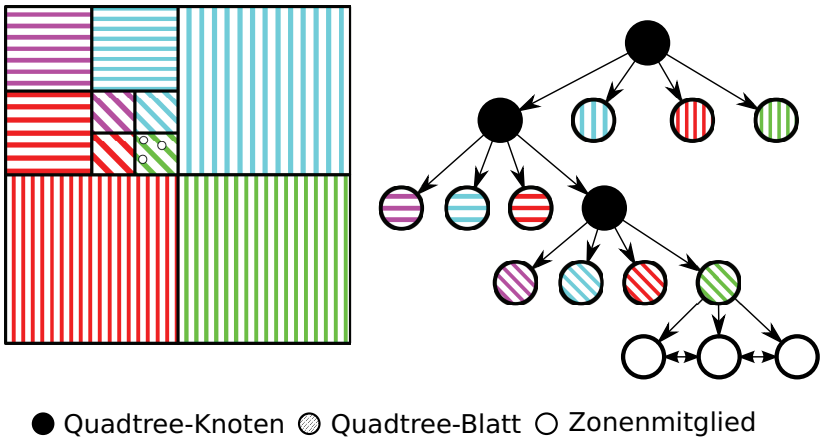


Abbildung 5.3 N-Tree: Aufteilung der Spielwelt in Zonen und dazugehöriger Quadtree.

Ein Nachteil von PubSubMMOG ist, dass der zentrale Lobby-Server über die Lastsituation und Aufgaben sämtlicher Teilnehmer informiert sein muss und für die gesamte Aufgabenverwaltung zuständig ist. Damit ist der Lobby-Server ein *single point of failure*: Fällt er aus, ist der Betrieb der virtuellen Welt nicht aufrecht zu erhalten. Bei großen virtuellen Welten kann die zentrale Verwaltung der Lastverteilbäume einen Leistungsengpass darstellen.

5.1.3 N-Tree

Das Protokoll *N-Tree* wurde von GaultierDickey et al. entwickelt [58]. Es basiert ebenso wie SimMUD und PubSubMMOG auf einer Unterteilung der virtuellen Welt in Zonen. Im Gegensatz zu diesen Protokollen erfolgt diese Aufteilung aber dynamisch abhängig von der Anzahl der Teilnehmer in den einzelnen Zonen.

5.1.3.1 Funktionsweise

Das Aufteilen der Spielwelt und das Verwalten der Zonen geschieht über einen *N-Tree*. Ein N-Tree ist ein n-ärer Baum, bei dem jeder Knoten entweder n Kindknoten hat oder keine Kindknoten. Knoten ohne Kindknoten werden *Blätter* genannt. Mit N-Trees lassen sich kartesische Räume unterteilen. Hierzu wird bei einem d-dimensionalen Raum ein 2^d-Tree verwendet. Die Wurzel des Baumes repräsentiert den gesamten Baum. Jede weitere Ebene des Baumes repräsentiert eine Unterteilung des Raumes entlang seiner d Achsen in jeweils 2 Teile.

Im Protokoll N-Tree wird die virtuelle Welt durch einen solchen n-ären Baum repräsentiert. Da im Rahmen dieser Arbeit zunächst 2-dimensionale virtuelle Welten betrachtet werden, ist der verwendete Baum ein 4-ärer N-Tree oder *Quadtree*. Der Wurzelknoten deckt hier die gesamte Spielwelt ab, ein Kindknoten deckt stets einen Quadranten der Fläche seines Vaterknotens ab. Die Blätter des Baumes bilden die feinste Unterteilung der Spielwelt. Dies sind die vom Protokoll für die Nachrichtenverteilung genutzten *Zonen*. In Abgrenzung davon wird die von den übergeordneten Knoten verwaltete Fläche als *Bereich* bezeichnet. In Abbildung 5.3 ist beispielhaft die Aufteilung der Spielwelt in Zonen und der dazugehörige Quadtree gezeigt.

Jeder Knoten im Baum wird durch einen Teilnehmer der virtuellen Welt verwaltet. Die Blätter des Quadtrees fungieren als Koordinatoren der ihrer Position im

Quadtree zugeordneten Zone. Sie verwalten eine Liste aller Teilnehmer, die sich in dieser Zone befinden. Alle Teilnehmer müssen somit Mitglied der zu ihrer Position zugehörigen Zone werden. Will ein Teilnehmer einer neuen Zone beitreten, sendet er eine Nachricht mit dem Beitrittswunsch an den Koordinator der entsprechenden Zone. Hierzu wird die Nachricht über den Quadtree weitergeleitet: Zunächst wird sie solange an den Vaterknoten weitergeleitet, bis der Verantwortungsbereich des Knotens die Zone umfasst. Von dort aus kann sie jeweils an den verantwortlichen Kindknoten für den Quadranten, der die Zielzone umfasst, weitergeleitet werden, bis der Koordinator der Zielzone erreicht ist. Da N-Trees nicht balanciert sein müssen, erfordert dies bei ungünstiger Verteilung der n Teilnehmer der virtuellen Welt $O(n)$ Weiterleitungsschritte. Bei Gleichverteilung der Teilnehmer in der virtuellen Welt werden $O(\log(n))$ Schritte benötigt.

Der Koordinator der Zone stellt sicher, dass alle Mitglieder seiner Zone stets die Adressen aller anderen Mitglieder der Zone kennen. Ereignisnachrichten und Positionsmeldungen, die die Zone betreffen, können dann direkt vom Verursacher an alle Mitglieder der Zone versendet werden. Befindet sich ein Teilnehmer am Rand einer Zone, sodass sein Interessengebiet in andere Zonen hereinragt, kann er zusätzlich Mitglied dieser benachbarten Zonen werden, sodass er über alle Ereignisse, die sein Interessengebiet betreffen könnten, informiert wird.

Überschreitet die Zahl der Teilnehmer in einer Zone einen Schwellenwert, wird die Zone in vier Unterzonen aufgeteilt. Hierzu wählt der Koordinator der Gruppe vier Teilnehmer aus der Zone aus, die jeweils Koordinator einer Unterzone werden. Danach werden alle Teilnehmer aus der Zone an die neuen Koordinatoren verwiesen. Der alte Koordinator ist danach nur noch für die Verwaltung des Knotens im Quadtree zuständig.

Stellt ein Knoten im Quadtree fest, dass die Gesamtzahl der Teilnehmer in seinen untergeordneten Zonen den Zonenaufteilungsschwellenwert um mehr als die Hälfte unterschreitet, werden diese untergeordneten Zonen zusammengelegt. Der Knoten teilt seinen Kindknoten mit, dass der Baum kollabiert wird. Daraufhin geben die Kindknoten die Verwaltung ihrer Zonen auf. Damit entsteht eine neue Zone, die das ursprüngliche Gebiet der Unterzonen umfasst.

Will ein Knoten das Netz verlassen, muss er seine Aufgaben an andere Teilnehmer abgeben. Dazu wählt er willkürlich ihm bekannte Teilnehmer aus, die ihn als Koordinator oder Verwalter eines Knotens im Quadtree ersetzen. Fällt ein Knoten aus, ohne seine Aufgaben ordnungsgemäß abgegeben zu haben, wird dies von den Vaterknoten bemerkt. Diese wählen dann einen Ersatz für den ausgefallen Knoten aus. Damit hierbei die Mitgliederlisten einer Zone nicht verloren gehen, werden diese in regelmäßigen Abständen von den Koordinatoren an ihre Vaterknoten im Quadtree versendet.

5.1.3.2 Implementierung

Um eine Evaluierung des Protokolls zu ermöglichen, wurde N-Tree in OverSim implementiert. In den anderen im Rahmen dieser Arbeit betrachteten Protokollen sind nur lokale Ereignisse vorgesehen, die sich nur auf einzelne Spieler auswirken. Um die Vergleichbarkeit zu gewährleisten, wurde die Implementierung von Ereignissen mit größerer Reichweite, die in der Veröffentlichung von GaultierDickey et al. [58] vorgesehen sind, unterlassen.

Weiterhin werden anders als bei dieser Veröffentlichung Teilnehmer nicht über eine zusätzliche DHT gefunden. Stattdessen erfolgt das Finden von Teilnehmern über das in OverSim integrierte *BootstrapOracle*. Hierdurch wird die Leistungsfähigkeit des Protokolls in keiner Weise berührt.

Da Teilnehmer in N-Tree Mitglied in allen Zonen werden, die von ihrem Interessengebiet teilweise abdeckt sind, kann es bei der dynamischen Aufteilung der Zonen zu Problemen kommen. Befinden sich in einer Zone viele Teilnehmer, wird die entsprechende Zone aufgeteilt. Ist nun das Interessengebiet der Teilnehmer größer als die neuen Zonen, werden sie sofort Mitglied in allen durch die Teilung entstandenen Zonen. Hierdurch sind in allen Zonen wiederum zu viele Teilnehmer, sodass sie erneut beliebig oft aufgeteilt werden. Dies würde bei hohen Teilnehmerdichten zu einer unendlichen Baumtiefe des N-Trees und somit zum Versagen des Protokolls führen. Um dies zu verhindern, wird in der Implementierung die Aufteilung von Zonen ausgesetzt, falls die Zonengröße die Größe der Interessengebiete unterschreiten würde.

Eine weitere Abweichung der Implementierung von der originalen Veröffentlichung ist das Ersetzen ausgefallener Teilnehmer. Bei der in der Veröffentlichung vorgeschlagenen Ausfallerkennung, bei der sowohl Vater- als auch Kindknoten eines ausgefallenen Teilnehmers diesen ersetzen können, kommt es regelmäßig zu ungeklärten Zuständigkeiten, wenn Vaterknoten sowie eine Zahl an Kindknoten unabhängig einen Ersatz für den ausgefallenen Knoten suchen. Dies würde die Konsistenz des N-Trees und damit die Nachbarschaftskenntnis bedrohen. Aus diesem Grund wird in der Implementierung der Ersatz ausgefallener Knoten stets vom Vaterknoten des betreffenden Knoten organisiert. Eine Ausnahme ist der Wurzelknoten des Quadtrees, da dieser nicht über einen Vaterknoten verfügt. Fällt der Wurzelknoten aus, wird der Ersatz vom für den ersten Quadranten zuständigen Kindknoten veranlasst.

5.1.3.3 Aufwandsabschätzung

Da in N-Tree die Positionsmeldungen direkt vom Verursacher an alle Teilnehmer der Zone versendet werden, ergibt sich eine Nachrichtenverzögerung von $O(1)$. Es sind also sehr gute Nachrichtenverzögerungen für Positionsmeldungen zu erwarten.

Der Bandbreitenbedarf für N-Tree setzt sich aus dem Bedarf für den Versand der Positionsmeldungen, dem Kommunikationsaufwand für das Backup der Mitgliederlisten einer Zone, dem Aufwand für das Management des N-Trees, insbesondere bei Zonenteilung oder -zusammenlegung, und dem Aufwand für den Zonenwechsel zusammen. Der Aufwand für den Versand der Positionsmeldungen ist mit $O(m)$ bei m Zonenmitgliedern gering, insbesondere da die Anzahl der Zonenmitglieder durch das Protokolldesign nach oben beschränkt wird, sofern die lokale Teilnehmerdichte nicht zu groß wird. Da für das Backup regelmäßig eine Liste mit allen Zonenteilnehmern an den Vaterknoten im N-Tree versendet werden muss, beträgt der Aufwand hierfür $O(m)$. Der Aufwand für den Zonenwechsel ist bei Gleichverteilung der n Teilnehmer in der virtuellen Welt $O(\log(n))$. Im unwahrscheinlichen Fall eines degenerierten N-Trees kann er aber $O(n)$ erreichen. Der Aufwand für das Warten des N-Trees, der Zonenaufteilung und der Zusammenlegung beträgt jeweils $O(m)$. Der Aufwand für die Verteilung der Positionsmeldungen und die Wartung des N-Trees ist relativ gering, bei ungünstigem Aufbau des N-Trees können Zonenwechsel jedoch relativ aufwändig sein.

Einflüsse auf die Nachbarschaftskenntnis ergeben sich in N-Tree aus der Zeit, die für den Zonenwechsel benötigt wird, und der Kompensation ausgefallener Knoten im N-Tree. Der Zonenwechsel benötigt bei Gleichverteilung der Teilnehmer in der virtuellen Welt $O(\log(n))$ Schritte, kann aber bei ungünstiger Struktur des N-Trees bis zu $O(n)$ Schritte benötigen. Dies stellt besonders bei häufigem Zonenwechsel ein deutliches Risiko für die Nachbarschaftskenntnis dar. Der Ausfall von Knoten im N-Tree ist ein weiteres mögliches Problem, da hierdurch der Erfolg des Zonenwechsels von Knoten verhindert werden kann. Je höher der ausgefallene Knoten im N-Tree liegt, desto höher ist die Zahl der potentiell betroffenen Teilnehmer. Insbesondere bei häufigem Zonenwechsel ist dementsprechend nur eine schlechte Nachbarschaftskenntnis zu erwarten.

5.1.4 Vast

Im Gegensatz zu den anderen hier beschriebenen Protokollen verzichtet *Vast* [67] vollständig auf eine Unterteilung der Spielwelt in Zonen. Stattdessen führt jeder Teilnehmer eine eigene Liste mit Nachbarn, die sich in ihrem Interessengebiet befinden, und kommuniziert direkt mit diesen. Damit gehört Vast zu den sogenannten *Mutual Notification*-Protokollen.

5.1.4.1 Funktionsweise

In Vast werden Positionsmeldungen und Ereignisnachrichten stets vom Verursacher an alle interessierten Teilnehmer weitergeleitet. Hierbei gelten alle Teilnehmer als interessiert, die sich innerhalb des Interessengebiets des Verursachers befinden. In Vast ist dieses Interessengebiet als ein Kreis mit einem fixen Radius definiert. Teilnehmer innerhalb dieses Gebietes werden auch als *direkte Nachbarn* bezeichnet. Jeder Teilnehmer hat eine direkte Verbindung zu allen seinen direkten Nachbarn.

Da sich durch die Bewegung der Teilnehmer in der Spielwelt die Menge der direkten Nachbarn eines Teilnehmers fortlaufend ändert, ist ein Mechanismus notwendig, Teilnehmer über neu in ihr Interessengebiet eintretende Teilnehmer zu informieren. Dazu bildet jeder Teilnehmer ein Voronoi-Diagramm aus den Positionen der ihm bekannten Teilnehmern. Aus diesem Voronoi-Diagramm werden nun gewisse Nachbarschaftsbeziehungen definiert: *Grenznachbarn* sind Nachbarn, deren Voronoi-Region von der Grenze des Interessengebiets geschnitten wird. *Hüllennachbarn* sind Nachbarn, deren Voronoi-Regionen direkt neben der eigenen Voronoi-Region liegt. In Abbildung 5.4 sind diese Nachbarschaftsbeziehungen beispielhaft aus Sicht eines Spielers dargestellt.

Grenz- und Hüllennachbarn haben in Vast die Aufgabe, Teilnehmer über neue Nachbarn zu informieren. Wenn ein Teilnehmer eine Positionsmeldung an einen Grenznachbarn sendet, prüft dieser, ob der Absender durch eine Bewegung nun Nachbar eines seiner Hüllennachbarn geworden ist. Ist dies der Fall, wird der Absender über dieses Ereignis informiert und kann nun eine Verbindung mit seinem neuen Nachbarn aufbauen.

5.1.4.2 Implementierung

Vast wurde wie die anderen in diesem Kapitel beschriebenen Protokolle in OverSim implementiert. In verschiedenen Veröffentlichungen zu Vast [66, 67] werden teils

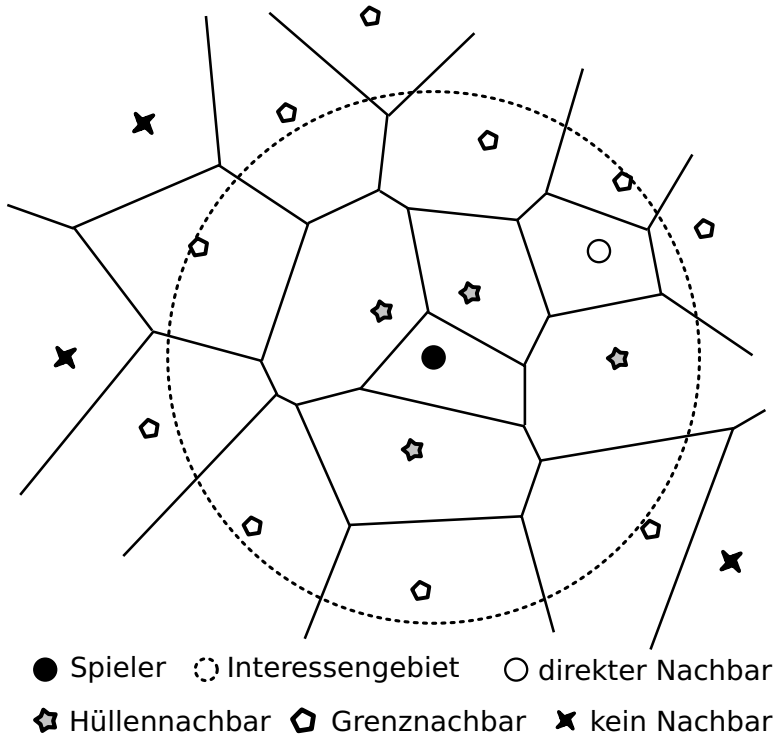


Abbildung 5.4 Vast: Voronoi-Diagramm über die Positionen der Spieler und sich daraus ergebende Nachbarschaftsbeziehungen.

widersprüchliche Definitionen für die Nachbarschaftsbeziehungen gegeben. In der Implementierung wurden die in der Veröffentlichung „Scalable peer-to-peer networked virtual environment“ [67] gegebenen Definitionen verwendet, da bei der alternativen Definition, bei ungünstiger Positionierung der Teilnehmer in der virtuellen Welt, nicht garantiert werden kann, dass alle Teilnehmer ausreichend Hüllennachbarn besitzen. Weiterhin weicht das Verhalten beim Verbindungsaufbau zu neuen Nachbarn in den Veröffentlichungen ab. In „VON: a scalable peer-to-peer network for virtual environments“ [66] ist ein zusätzlicher Überprüfungsschritt auf zusätzliche neue Nachbarn enthalten, der die Nachbarschaftskenntnis steigern kann. Dieser ist in der Implementierung enthalten.

In Vast kann es durch Teilnehmerausfall oder Nachrichtenverzögerungen zu inkonsistenten Voronoi-Diagrammen kommen. Dies hat schlimmstenfalls zur Folge, dass ein Vast-Netz in Teilnetze zerfällt. Teilnehmer aus verschiedenen Teilnetzen können dann nicht mehr miteinander interagieren. Aus diesem Grund wurde von Jiang et al. ein Backup-Mechanismus entwickelt [71], der den Zerfall des Netzes verhindern soll. Diese Erweiterung ist in der Implementierung enthalten. Der Backup-Mechanismus sieht vor, dass Teilnehmer, die eine unterdurchschnittliche Zahl an Nachbarn besitzen, ihre Nachbarslisten mit ihren Nachbarn auszutauschen. Durch die so gewonnenen zusätzlichen Nachbarschaftsbeziehungen kann auch bei Teilnehmerausfall der Netzzusammenhalt gesichert werden. Allerdings erfordert dieser Backup-Mechanismus für die betroffenen Teilnehmer bei m Nachbarn einen Bandbreitenaufwand von $O(m^2)$.

5.1.4.3 Aufwandsabschätzung

Vast versendet alle Positionsmeldungen direkt vom Verursacher an alle interessierten Nachbarn, Positionsmeldungen werden also in $O(1)$ zugestellt. Demzufolge sind sehr geringe Nachrichtenverzögerungen zu erwarten.

Der Bandbreitenbedarf setzt sich in Vast aus dem Aufwand für die Verteilung der Positionsmeldungen, dem Aufwand für das Benachrichtigen neuer Nachbarn und dem Aufwand für den Backup-Mechanismus zusammen. Bei m Nachbarn beträgt der Aufwand für das Versenden der Positionsmeldungen $O(m)$. Der Aufwand für die Benachrichtigung über neue Nachbarn hängt vom Anteil der Grenznachbarn an der gesamten Nachbarschaft zusammen. Bei g Grenznachbarn beträgt der Aufwand $O(gm)$. Bei ungünstiger Verteilung der Knoten im Voronoi-Graphen können alle Nachbarn zugleich auch Grenznachbarn sein. In diesen Fällen ist der Aufwand für die Nachbarschaftsfindung $O(m^2)$. Der Aufwand für den Backup-Mechanismus beträgt für Teilnehmer, die aufgrund ihrer unterdurchschnittlichen Zahl an Nachbarn Teil des Mechanismus werden $O(m^2)$, der Anteil an diesen Knoten ist jedoch stark von der konkreten Verteilung der Teilnehmer in der virtuellen Welt abhängig. Insgesamt ist der Aufwand für das Versenden der Positionsmeldungen also verhältnismäßig gering; der Aufwand für die Nachbarschaftsfindung und für den Backup-Mechanismus kann jedoch in ungünstigen Fällen sehr hoch werden.

Die Nachbarschaftskenntnis in Vast ist im Wesentlichen von der Güte der Nachbarschaftsfindung abhängig. Da sich in Vast die Nachbarschaftsbeziehungen implizit aus der Positionen der Knoten im zugrunde liegenden Voronoi-Graphen ergebenen, ist es insbesondere problematisch, wenn die Voronoi-Graphen benachbarter Teilnehmer nicht konsistent sind. Inkonsistenzen können hier insbesondere durch Knotenausfälle und Nachrichtenverzögerungen entstehen. Da bei hoher lokaler Teilnehmerdichte die einzelnen Regionen im Voronoi-Graphen kleiner werden und bei kleineren Regionen bereits kleinere Fehler zu Inkonsistenzen führen, ist in diesen Situationen eine Verschlechterung der Nachbarschaftskenntnis zu erwarten.

5.1.5 Vergleich der Aufwandsabschätzungen

Die Aufwandabschätzungen der Protokolle sind in Tabelle 5.1 zusammengefasst. Dabei steht n für die Gesamtzahl der Teilnehmer in der virtuellen Welt, m für die Zahl der Teilnehmer in einer Zone oder – im Fall von Vast – für die Zahl der Nachbarn eines Teilnehmers. Im Fall von PubSubMMOG steht k für die Breite des verwendeten Lastverteilbaumes.

Tabelle 5.1a zeigt den zeitlichen Aufwand für den Versand von Positionsmeldungen und Ereignisnachrichten. Ein hoher Aufwand führt zu signifikanter Verzögerung der Nachrichten und damit zu einer Beeinträchtigung der Interaktionsmöglichkeiten der Teilnehmer. Bei SimMUD und PubSubMMOG müssen die Nachrichten über einen Multicastbaum beziehungsweise Lastverteilbaum weitergeleitet werden. Die Tiefe dieses Baumes ist bei SimMUD von der Gesamtzahl der Teilnehmer und bei PubSubMMOG von der Zahl der Teilnehmer in der jeweiligen Zone abhängig. Dies führt bei großen virtuellen Welten beziehungsweise bei vollen Zonen zu einer hohen Nachrichtenverzögerung. Bei N-Tree und Vast werden die Nachrichten ohne Zwischenknoten im P2P-Netz direkt an die Empfänger gesendet. Hieraus resultiert eine geringe Nachrichtenverzögerung.

Protokoll	Aufwand	Bewertung
SimMUD	$O(\log(n))$	⊖
PubSubMMOG	$O(\log_k(m))$	⊖
N-Tree	$O(1)$	⊕
Vast	$O(1)$	⊕

(a) Nachrichtenverzögerung.

Protokoll	Pos. Meldungen	Zonenwechsel	Backup	Verwaltung	Bew.
SimMUD	$O(m\log(n))$	$O(\log(n))$	–	$O(\log(n))$	⊕
PubSubMMOG	$O(m\log(m))$	$O(1)$	$O(m)$	$O(\log(m))$	⊕
N-Tree	$O(m)$	$O(n)$	$O(m)$	$O(m)$	◯
Vast	$O(m)$	–	$O(m^2)$	$O(m^2)$	⊖

(b) Bandbreitenbedarf.

Protokoll	Zonenwechsel	Ausfallerkennung	Weitere Einflüsse	Bew.
SimMUD	$O(\log(n))$	langsam	Kein Backup-Mechanismus	⊖
PubSubMMOG	$O(1)$	schnell	Zentraler Server ist single point of failure	⊕
N-Tree	$O(n)$	langsam	Ausfall nahe Wurzel des N-Trees verhindert Zonenwechsel	⊖
Vast	–	schnell	Inkonsistente Voronoi-Graphen stören Nachbarsfindung	◯

(c) Nachbarschaftskenntnis.

Tabelle 5.1 Bewertung des Aufwands der Protokolle.

In Tabelle 5.1b ist der für den Bandbreitenbedarf benötigte Aufwand dargestellt. SimMUD benötigt nur maximal logarithmischen Aufwand für alle Aufgaben, mit Ausnahme der Verteilung der Positionsnachrichten, die einen leicht überlinearen Aufwand aufweist. Insgesamt ergibt sich so ein geringer erwarteter Bandbreitenbedarf. Der von PubSubMMOG benötigte Aufwand ist dem von SimMUD sehr ähnlich, nur der Aufwand für das Backup ist linear statt logarithmisch. Da das Backup nur relativ selten ausgeführt wird, hat dies keinen großen Einfluss auf den zu erwarteten Bandbreitenbedarf. N-Tree benötigt für alle Aufgaben linearen Aufwand. Der Mehraufwand in den Verwaltungsaufgaben im Verhältnis zu SimMUD und PubSubMMOG wird durch den leicht geringeren Bedarf für die Verteilung der Positionsmeldungen teilweise ausgeglichen, sodass sich insgesamt ein befriedigender Bandbreitenbedarf ergibt. Vast hingegen weist für die Nachbarschaftsfindung und für das Backup quadratischen Aufwand auf, sodass in Szenarien mit einer hohen durchschnittlichen Anzahl an Nachbarn ein sehr hoher Bandbreitenbedarf zu erwarten ist.

Tabelle 5.1c zeigt die möglichen Einflüsse auf die Nachbarschaftskenntnis. SimMUD benötigt logarithmische Zeit abhängig von der Gesamtzahl der Teilnehmer für den Zonenwechsel. In Verbindung mit der langsamen, auf Scribe zurückgreifenden Ausfallerkennung ergeben sich in Szenarien mit häufigen Zonenwechseln deutliche Risiken für die Nachbarschaftskenntnis. In PubSubMMOG kann der Zonenwechsel aufgrund des zentralen Servers in konstanter Zeit durchgeführt werden. Zusammen mit der schnellen Ausfallerkennung ist eine sehr gute Nachbarschaftskenntnis zu erwarten. N-Tree benötigt in ungünstigen Fällen lineare Zeit für den Zonenwechsel. Weiterhin ist die Ausfallerkennung relativ langsam, und der Ausfall eines Knotens an ungünstiger Stelle im N-Tree kann den Zonenwechsel einer großen Zahl von Teilnehmern behindern. Somit kann – besonders in Szenarien mit häufigen Zonenwechseln – nur eine schlechte Nachbarschaftskenntnis erreicht werden. Vast besitzt keine Zonen, weswegen ein Zonenwechsel hier irrelevant ist. Einflüsse auf die Nachbarschaftskenntnis ergeben sich im Wesentlichen aus der Nachbarsfindung. Diese setzt voraus, dass die Voronoi-Graphen benachbarter Knoten konsistent sind. Bei inkonsistenten Voronoi-Graphen kann die Nachbarschaftskenntnis beeinträchtigt werden. Aufgrund der schnellen Ausfallerkennung und des aufwändigen Backup-Mechanismus ist insgesamt eine akzeptable Nachbarschaftskenntnis zu erwarten.

Die Bewertungen gelten zunächst nur für die untersuchten Protokolle. In vielen Bereichen sind die Aufwände der einzelnen Protokolle jedoch typisch für die jeweilige Protokollkategorie, sodass sie teilweise verallgemeinert werden können. Dies trifft insbesondere für die Überlegungen zur Nachrichtenverzögerung der Positionsmeldungen und Ereignisnachrichten zu. So werden in ALM-basierten Protokollen diese Nachrichten konzeptbedingt über einen Multicastbaum oder -Mesh weitergeleitet. Dies hat bei allen üblichen ALM-Protokollen mindestens logarithmischen Aufwand [64, 153]. Supernode-basierte Protokolle mit statischen Zonen müssen eine Lastverteilung implementierten, um bei Zonen mit vielen Mitgliedern den Supernode nicht zu überlasten. Diese Lastverteilung wird in der Regel über eine Baumstruktur durchgeführt, sodass auch hier ein logarithmischer Aufwand üblich ist. Protokolle mit dynamischen Zonen oder zonenlose Protokolle können auf eine Lastverteilung verzichten und somit die Verteilung der Positionsmeldungen und Ereignisnachrichten mit konstantem Aufwand durchführen.

Der Bandbreitenaufwand der Protokolle ist nicht so einfach zu verallgemeinern. Bei Protokollen mit statischen Zonen sind aufgrund der Multicastbäume oder Lastver-

teilbäume für die Verteilung der Positionsnachrichten leicht überlineare Aufwände nicht zu vermeiden. Da die übrigen Aufwände bei SimMUD und PubSubMMOG jeweils unter dem Aufwand für die Verteilung der Positionsnachrichten liegen, ist zu vermuten, dass andere Protokolle dieser Klassen nicht wesentlich effizienter sind. Der lineare Aufwand von N-Tree für diverse Verwaltungsaufgaben ließe sich möglicherweise unterbieten. Eine deutliche Steigerung der Effizienz ist dadurch aber nur bedingt zu erwarten. Bei Vast hingegen ist der quadratische Aufwand für Nachbarschaftsfindung und Backup problematisch. Dieser ließe sich durch einen besseren Protokollentwurf verringern. Die meisten bisherigen zonenlosen Protokolle basieren jedoch ebenso wie Vast auf Voronoi-Graphen oder den dazu dualen Delauney-Graphen, sodass in diesen Protokollen ähnliche Ergebnisse wie für Vast zu erwarten sind.

Der für die Nachbarschaftskenntnis relevante Aufwand der Protokolle ist ebenfalls bedingt zu verallgemeinern. Ohne einen zentralen Server ist bei zonenbasierten Protokollen ein logarithmischer Aufwand für den Zonenwechsel schwer zu vermeiden, sodass SimMUD in dieser Hinsicht typisch für ein Protokoll mit statischen Zonen ist. Durch eine verbesserte Ausfallerkennung und die Implementierung eines Backup-Mechanismus könnte die Nachbarschaftskenntnis allerdings gesteigert werden. PubSubMMOG profitiert bei der Nachbarschaftskenntnis von einem zentralen Server. Dieser ermöglicht einen konstanten Aufwand für den Zonenwechsel und kann bei Knotenausfällen für geeigneten Ersatz sorgen. Dabei stellt er allerdings einen single point of failure dar. Der lineare Aufwand beim Zonenwechsel ist für Protokolle mit dynamischen Zonen nicht unüblich, da die Zonen oft in einer Baumstruktur verwaltet werden und ein balancierter Baum in der Regel nicht garantiert werden kann. Eine andere Verwaltung der Zonen ist möglich, es wurden bislang aber noch keine serverlosen Protokolle mit einer effizienteren Zonenverwaltung vorgeschlagen. Auch wenn die Ausfallerkennung von Protokollen mit dynamischen Zonen schneller sein könnte als bei N-Tree, ist so eine wesentliche Verbesserung der Nachbarschaftskenntnis schwierig. Bei zonenlosen Protokollen sind die für die Nachbarschaftskenntnis relevanten Faktoren nur schwer zu verallgemeinern, da diese stark von der verwendeten Protokollstruktur abhängen. Da andere zonenlose Protokolle auf denselben oder äquivalenten Datenstrukturen basieren wie Vast, sind für diese Protokolle ähnliche Ergebnisse zu erwarten.

Zusammengefasst lassen die Aufwandsabschätzungen vermuten, dass Protokolle mit statischen Zonen nicht in allen Situationen die gewünschte niedrige Latenz bieten können. Protokolle mit dynamischen Zonen können eine niedrige Latenz erreichen, aber dabei nicht die nötige Nachbarschaftskenntnis gewährleisten. Zonenlose Protokolle können sowohl eine geringe Latenz als auch eine gute Nachbarschaftskenntnis bieten. Vast birgt dabei jedoch das Risiko eines übermäßigen Bandbreitenbedarfs. Dieses Risiko ist aber nicht systematisch für zonenlose Protokolle.

5.2 Evaluierung

Um die in der Aufwandsabschätzung getroffenen Überlegungen experimentell zu überprüfen, wurden die ausgewählten Protokolle in OverSim in verschiedenen Szenarien mit wechselnden Teilnehmerdichten evaluiert. Dabei wurden Latenz, Bandbreitenbedarf und Nachbarschaftskenntnis gemessen.

5.2.1 Metriken

Wie in Abschnitt 2.1.3 erläutert, ist eines der wichtigsten Kriterien für die Beurteilung von Protokollen für virtuelle Welten die *Nachrichtenverzögerung*. Eine flüssige Interaktion ist in virtuellen Welten nur möglich, wenn Ereignisnachrichten um nicht mehr als 200 ms verzögert werden [126]. Beobachten Teilnehmer eine Nachrichtenverzögerung von mehr als 250 ms, beträgt die Sitzungszeit in einer virtuellen Welt nur noch ein Viertel der durchschnittlichen Sitzungszeit [31]. Erreichen Ereignisnachrichten ihr Ziel mit Verzögerungen von mehr als 500 ms, wird dies von Nutzern virtueller Welten in der Regel nicht toleriert [35].

In den Simulationen wurde gemessen, welche Zeit durchschnittlich zwischen dem Absenden einer Ereignisnachricht und dem Empfang der Nachricht durch andere Teilnehmer vergeht. Diese Zeit wird im Wesentlichen durch die netzwerkbedingte durchschnittliche Latenz zwischen zwei Teilnehmern sowie die durchschnittliche Anzahl an Zwischenknoten, die an der Weiterleitung der Nachricht beteiligt sind, beeinflusst.

Ein weiteres wichtiges Kriterium ist der Bandbreitenbedarf. Nutzer haben nur eine gewisse Bandbreite zur Verfügung. Bei üblichen ADSL-Endkundenverträgen beträgt die verfügbare Bandbreite in Senderichtung beispielsweise zwischen 384 kBit/s und 1 MBit/s, die verfügbare Bandbreite in Empfangsrichtung beträgt üblicherweise zwischen 1 MBit/s und 16 MBit/s. Der Bandbreitenbedarf des Protokolls darf diese nicht überschreiten. In der Simulation wurde der durchschnittliche Bandbreitenbedarf der Teilnehmer in Senderichtung gemessen, da dieser aufgrund der asymmetrischen Datenraten bei üblichen Endkundenverträgen in P2P-Protokollen in der Regel zuerst ausgelastet wird. Bei den Messungen in den Simulationen waren die Unterschiede der benötigten Bandbreite in Sende- und Empfangsrichtung nur gering.

Die dritte wichtige Metrik ist die Nachbarschaftskenntnis. Um eine sinnvolle Interaktion zwischen Teilnehmern zu ermöglichen, müssen alle Teilnehmer über die Existenz aller Teilnehmer innerhalb ihres Interessengebiets informiert sein. Um dies zu messen, wird in den Simulationen periodisch überprüft, wie viele Teilnehmer zu dem jeweiligen Zeitpunkt über alle ihre Nachbarn informiert sind. Geteilt durch die Gesamtteilnehmerzahl ergibt sich so ein prozentuales Maß für die Nachbarschaftskenntnis. Ein Wert von 100% bedeutet eine perfekte Nachbarschaftskenntnis. Liegt ein Messwert deutlich unter 100%, bedeutet dies, dass viele Teilnehmer nicht alle Nachbarn kennen und somit die Interaktionsmöglichkeiten vieler Teilnehmer eingeschränkt sind.

5.2.2 Einflussgröße: Teilnehmerdichte

Eine wesentliche Einflussgröße auf die oben genannten Metriken bei Protokollen für verteilte virtuelle Welten ist die Teilnehmerdichte. Hohe Teilnehmerdichten führen zu einer höheren Anzahl an Teilnehmern innerhalb des Interessengebiets und somit in der Regel zu einem höheren Bandbreitenbedarf. Zusätzlich steigt je nach verwendetem Protokoll auch der Wartungsaufwand für die vom Protokoll verwendeten Strukturen.

Dabei können die *durchschnittliche Teilnehmerdichte* und die *lokale Teilnehmerdichte* unterschiedliche Einflüsse auf das Protokoll ausüben. Eine höhere *durchschnittliche* Teilnehmerdichte wird durch eine größere Anzahl an Teilnehmern bei gleicher

Größe der virtuellen Welt oder Verkleinerung der Spielweltgröße bei gleichbleibender Teilnehmerzahl erreicht. Dies führt zu einer höheren Anzahl an Teilnehmern innerhalb des Interessengebiets und somit in der Regel zu einem höheren Bandbreitenbedarf. Weiterhin müssen bei zonenbasierten Protokollen mit statischen Zonen kleinere Zonengrößen gewählt werden, um eine vergleichbare Last auf die Koordinatoren zu erreichen. Bei dynamischen Zonengrößen führt eine höhere Teilnehmerdichte automatisch zu kleineren Zonengrößen. Durch die kleineren Zonen kommt es zu häufigeren Zonenübertritten der Teilnehmer. Dadurch können der Bandbreitenbedarf und die Nachbarschaftskenntnis beeinflusst werden.

Eine höhere *lokale* Teilnehmerdichte entsteht dadurch, dass sich Teilnehmer zu Gruppen zusammenschließen. Diese Gruppen verfolgen ein gemeinsames Ziel und bewegen sich somit gemeinsam in der virtuellen Welt. Durch die Gruppenbildung ergibt sich eine Ungleichverteilung der Teilnehmer in der virtuellen Welt: An Orten, an denen sich Gruppen aufhalten, ist die Teilnehmerdichte lokal hoch, während sie in den Zwischenräumen niedrig ist. Schließen sich Teilnehmer zu größeren Gruppen zusammen, sinkt die Gesamtzahl der Gruppen in der virtuellen Welt; der Anteil der Spielwelt ohne Teilnehmer steigt ebenso wie die Teilnehmerdichte innerhalb der Gruppen. Größere Gruppen führen also zu einer größeren Ungleichverteilung der Teilnehmer in der virtuellen Welt. Durch diese Ungleichverteilung ergibt sich in der Regel ein höherer Bandbreitenbedarf; abhängig vom Protokoll kann es auch zu erhöhten Latenzen oder verringerter Nachbarschaftskenntnis kommen.

5.2.3 Szenarien

Für die Evaluierung wurde in OverSim die Applikation *SimpleGameClient* implementiert. Diese simuliert eine einfache virtuelle Welt. In dieser virtuellen Welt bewegen sich die Teilnehmer in einem quadratischen Spielfeld. Weitergehende Interaktionsmöglichkeiten wie Textchat werden zunächst nicht weiter betrachtet. Teilnehmer der virtuellen Welt bewegen sich gemäß eines konfigurierbaren Bewegungsmodells durch die Spielwelt und senden dabei Positionsmeldungen an das unterliegende P2P-Protokoll. Dieses verteilt die Positionsmeldungen gemäß des jeweiligen Protokolls an die Teilnehmer in derselben Zone oder an die Teilnehmer innerhalb des Interessengebiets.

In der so simulierten Welt bewegten sich im Mittel 500 Teilnehmer. Die Sitzungszeiten der Teilnehmer wurden anhand des in Abschnitt 4.8.1 vorgestellten *Pareto-Churn*-Modells bestimmt. Die von diesem Modell bestimmten Sitzungszeiten folgen einer *heavy Tailed*-Verteilung: Viele Sitzungen sind sehr kurz, einige wenige Sitzungen hingegen sehr lang. Dabei steigt der Erwartungswert der Restsitzungszeit mit der Dauer der bisherigen Sitzung. Dieses Verhalten entspricht dem in Studien ermittelten Verhalten von Nutzern virtueller Welten [30, 110, 133]. In den verschiedenen Studien wurden jeweils unterschiedliche durchschnittliche Sitzungszeiten ermittelt. Die Werte schwanken dabei zwischen 54 Minuten [133] und 2,8 Stunden [134]. In der Evaluierung wurde eine mittlere Sitzungszeit von 100 Minuten gewählt [30]. Nach Ablauf der Sitzungszeit wurde zufällig entschieden, ob der Teilnehmer das Netz geordnet verlässt (*graceful leave*) oder ohne Vorwarnung ausfällt (*fail-stop*).

Das Interessengebiet der Spieler in diesem Szenario ist ein Kreis mit einem Radius von 50 m um die Position des Spielers. Spieler bewegen sich mit einer Geschwindigkeit von 5 m/s. Dies entspricht gängigen Geschwindigkeiten in MMOGs wie beispielsweise

„World of Warcraft“. Dabei werden 6 Bewegungsnachrichten pro Sekunde generiert. Auch dies entspricht realistischen Nachrichtenfrequenzen für MMOGs [30, 134].

Um verschiedene durchschnittliche Spielerdichten zu erhalten, wurden die Simulationen bei gleichbleibender durchschnittlicher Teilnehmerzahl mit 3 verschiedenen Spielfeldgrößen von 10.000 m mal 10.000 m, 5.000 m mal 5.000 m und 1.000 m mal 1.000 m durchgeführt. Bei durchschnittlich 500 Teilnehmern ergeben sich somit Teilnehmerdichten von 5, 20 und 500 Teilnehmern/km^2. Bei zonenbasierten Protokollen mit statischen Zonen – SimMUD und PubSubMMOG – wird das Spielfeld in jeweils 10 mal 10 Zonen unterteilt. Es ergeben sich also, abhängig von der Spielfeldgröße, Zonen von 1.000 m mal 1.000 m, 500 m mal 500 m und 100 m mal 100 m.

Um Variationen der lokalen Teilnehmerdichte zu erlauben wurde das Bewegungsmodell *GroupRoaming* des SimpleGameClients gewählt. GroupRoaming basiert auf dem *Random Waypoint*-Modell. Anstatt sich wie in einem klassischen Random Waypoint-Modell unabhängig von den anderen Teilnehmern ein individuelles Ziel zu suchen, bilden die Teilnehmer in GroupRoaming Gruppen mit konfigurierbarer Maximalgröße. Das Bewegungsmodell ist vergleichbar dem in der Evaluierung von Ad-Hoc-Netzwerken eingesetzten *Nomadic Community Mobility Model* [25]. Teilnehmer, die sich zu einer Gruppe zusammengeschlossen haben, wählen sich ein gemeinsames Ziel in der Spielwelt. Die Spieler bewegen sich dann in Richtung dieses Zieles. Sobald das Ziel von einem Mitglied der Gruppe erreicht wird, wird ein neues Ziel gewählt. Um eine allzu gleichförmige Bewegung zu vermeiden, wird ein *Flocking*-Algorithmus [114] angewendet, der die Bewegung in Abhängigkeit benachbarter Spieler variiert. In der Evaluierung wurden maximale Gruppengrößen von $1, 5, 10, 20, 40$ und 60 gewählt. Bei einer Gruppengrößen von 1 verhält sich GroupRoaming identisch zu einem *Random Waypoint*-Modell.

Als Modellierung des unterliegenden Netzwerks wurde das *SimpleUnderlay*-Modell gewählt. Dieses erlaubt die Simulation realitätsnaher Internet-Szenarien. Hierzu werden die Netzknoten in einem Koordinatensystem so platziert, dass die Abstände zwischen den Knoten proportional zu den Latenzen sind, die sich in Messungen zwischen Internetknoten [95] ergeben. In der Simulation können dann die Latenzen zwischen zwei Knoten effizient aus deren Entfernung in diesem Koordinatensystem bestimmt werden. Das SimpleUnderlay wird in Abschnitt 4.5.1 genauer beschrieben. Aufbauend auf diesem Modell wurden die oben vorgestellten Protokolle SimMUD, PubSubMMOG, N-Tree und Vast simuliert. Alle Simulationen wurden mit 30 verschiedenen Seeds durchgeführt. Die abgebildeten Ergebnisse sind die Mittelwerte dieser Durchläufe zusammen mit dem 99%-Konfidenzintervall. Die Simulationsparameter sind in Tablle 5.2 zusammengefasst.

5.2.4 Ergebnisse

In den folgenden Abschnitten werden die Evaluierungsergebnisse dargestellt. Dabei werden die Metriken Nachrichtenverzögerung, Bandbreitenbedarf und Nachbarschaftskenntnis betrachtet.

5.2.4.1 Nachrichtenverzögerung

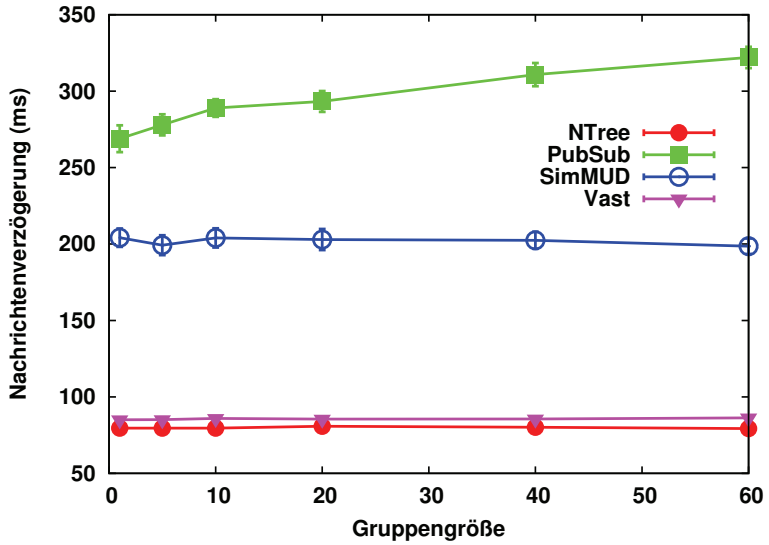
Die Abbildung 5.5 zeigt die Nachrichtenverzögerung der Protokolle für verschiedene Gruppengrößen bei unterschiedlichen Spielfeldgrößen.

Parameter	Wert
Simulationszeit	2 Stunden nach Initialisierungsphase
Anzahl Seeds	30
Teilnehmerzahl	500
Spielfeldgröße	1 km*1 km, 5 km*5 km, 10 km*10 km
Zonen (bei statischen Zonen)	10*10
Bewegungsmodell	GroupRoaming
max. Gruppengröße	1, 5, 10, 20, 40, 60
Sitzungsmodell	ParetoChurn
Sitzungszeit	100 Minuten
Graceful-leave	50%
Interessengebietsradius	50 m
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Netzmodell	SimpleUnderlay

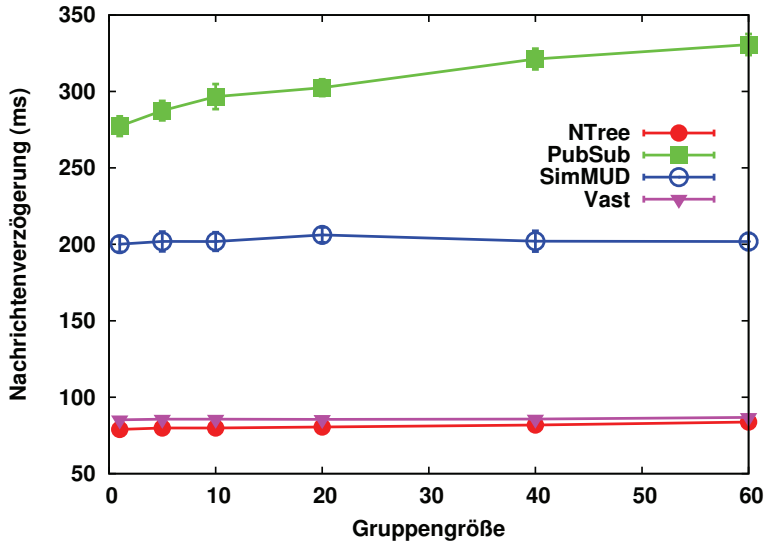
Tabelle 5.2 Simulationsparameter.

In Abbildung 5.5a sind die Verzögerungen bei einer Spielfeldgröße von 10.000 m mal 10.000 m zu sehen. Die niedrigsten Verzögerungen zeigen Vast und N-Tree mit ungefähr 90 ms unabhängig von der Gruppengröße. Dieses Ergebnis entspricht den Erwartungen, da in Vast und N-Tree alle Nachrichten mit einem Aufwand von $O(1)$ unmittelbar von den Absendern zu den Empfängern gesendet werden. Die bei SimMUD beobachteten Verzögerungen sind ebenfalls unabhängig von der Gruppengröße. Sie liegen bei ungefähr 200 ms. Auch dieses Ergebnis entspricht den Erwartungen: Die von SimMUD verwendeten Multicastbäume sind im Wesentlichen vom Overlay-Routing in Pastry, dem verwendeten KBR-Protokoll, festgelegt. Der erwartete Aufwand ist bei n Teilnehmern in der virtuellen Welt $O(\log(n))$ und somit unabhängig von der Gruppengröße. Die Verzögerungen von PubSubMMOG steigen hingegen mit steigender Gruppengröße. Bei einer Gruppengröße von 1 betragen die Verzögerungen ungefähr 270 ms, bei einer Gruppengröße von 10 erreichen sie 290 ms und bei einer Gruppengröße von 60 betragen sie ungefähr 320 ms. Bei PubSubMMOG nimmt die Tiefe des Lastverteilbaumes bei steigender Anzahl an Spielern in einer Zone zu; der Aufwand beträgt bei m Teilnehmern in einer Zone $O(\log(m))$. Bei steigender Gruppengröße hält sich der größte Teil der Spieler in Regionen mit vielen Spielern auf. Eine steigende Nachrichtenverzögerung bei steigender Gruppengröße entspricht also den Erwartungen.

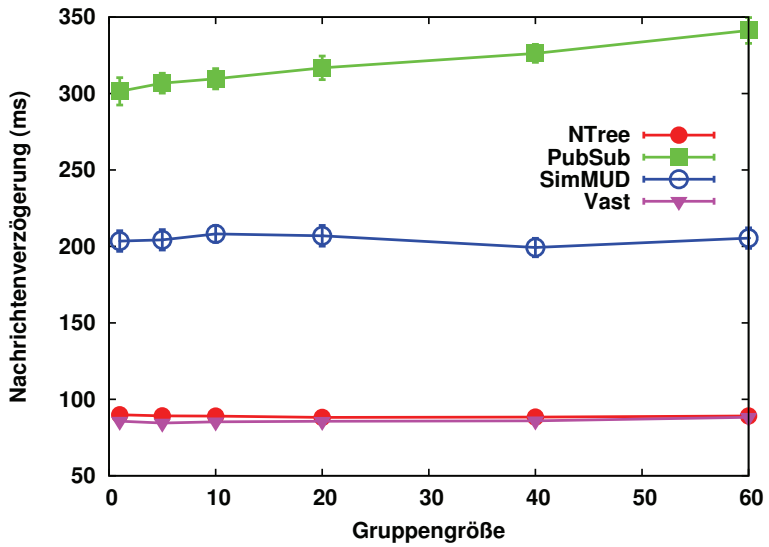
Abbildung 5.5b zeigt die Verzögerungen bei der nächst kleineren Spielfeldgröße von 5.000 m mal 5.000 m. Im Vergleich zu dem größeren Spielfeld ergeben sich nur geringe Änderungen. Vast und N-Tree zeigen, wie bei der Spielfeldgröße 10.000 m mal 10.000 m, eine Nachrichtenverzögerung von 90 ms. Auch SimMUD zeigt mit 200 ms dieselben Verzögerungen wie bei der Spielfeldgröße 10.000 m mal 10.000 m. PubSubMMOG benötigt zum Zustellen der Positionsmeldungen wenige Millisekunden mehr als bei dem größeren Spielfeld und liegt zwischen 275 ms und 330 ms. Bei einer höheren Teilnehmerdichte werden die Lastverteilbäume in PubSubMMOG im Durchschnitt tiefer. Die Veränderungen zu der größeren Spielfeldgröße entsprechen also den Erwartungen. Bei den anderen Protokollen ist die Zahl der Zwischenknoten und somit die Latenz unabhängig von der Teilnehmerdichte.



(a) Nachrichtenverzögerung bei Spielfeldgröße 10.000 m mal 10.000 m.



(b) Nachrichtenverzögerung bei Spielfeldgröße 5.000 m mal 5.000 m.



(c) Nachrichtenverzögerung bei Spielfeldgröße 1.000 m mal 1.000 m.

Abbildung 5.5 Evaluierungsergebnisse: Nachrichtenverzögerung in Abhängigkeit der Gruppengröße.

Aus diesem Grund unterscheiden sich auch die in Abbildung 5.5c gezeigten Verzögerungen bei der Spielfeldgröße 1.000 m mal 1.000 m nur geringfügig von den Ergebnissen der anderen Spielfeldgrößen. Vast, N-Tree und SimMUD liegen wie bei den beiden anderen Szenarien bei 90 ms beziehungsweise 200 ms. Die Nachrichtenverzögerung von PubSubMMOG hingegen steigt wiederum gegenüber den vorangehenden Szenarien. Sie beträgt 300 ms bei einer Gruppengröße von 1 und 340 ms bei einer Gruppengröße von 60.

Zusammenfassend können Vast und N-Tree in allen Szenarien hervorragende Latenzen von ungefähr 90 ms bieten. Die Latenzen von SimMUD liegen mit durchschnittlich 200 ms im guten Bereich, können jedoch bei virtuellen Welten mit hohem Interaktionsgrad bereits spürbar werden. Die Latenzen von PubSubMMOG sind nur bei geringer Teilnehmerdichte akzeptabel. Bei hoher Spielerdichte, insbesondere bei großer Gruppengröße, werden mit 340 ms Werte erreicht, die eine Interaktion zwischen den Teilnehmern bereits merklich beeinträchtigen können.

5.2.4.2 Bandbreitenbedarf

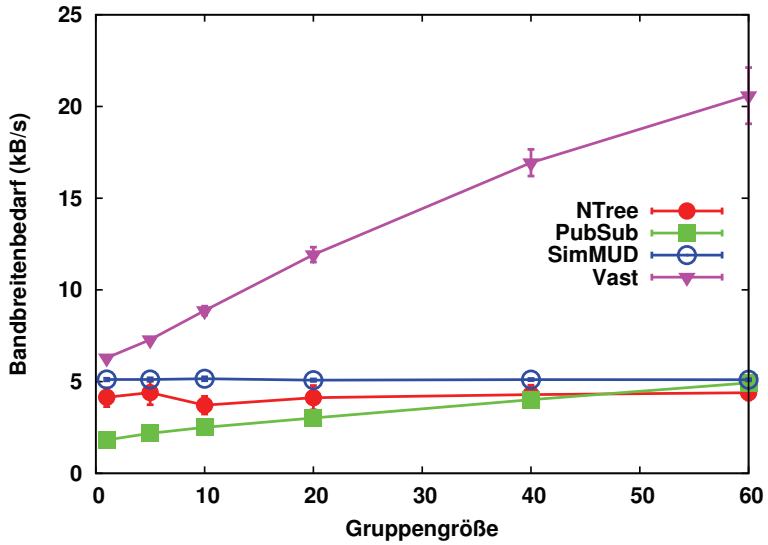
Abbildung 5.6 zeigt den durchschnittlichen Bandbreitenbedarf der Protokolle in den verschiedenen Szenarien. Abbildung 5.6a zeigt den Bandbreitenbedarf der Protokolle bei einer Spielfeldgröße von 10.000 m mal 10.000 m. Bei Gruppengrößen kleiner als 60 zeigt PubSubMMOG den geringsten durchschnittlichen Bandbreitenbedarf. Bei einer Gruppengröße von 1 beträgt er nur 1.8 kByte/s. Bei größeren Gruppengrößen steigt der Bandbreitenbedarf von PubSubMMOG an, bei einer Gruppengröße von 60 erreicht er knapp 5 kByte/s. Der Anstieg lässt sich durch die verwendeten Lastverteilbäume erklären. Die Lastverteilbäume haben bei m Teilnehmern in einer Zone einen Aufwand von $O(\log(m))$. Bei höherer Gruppengröße und somit höherer lokaler Teilnehmerdichte muss öfter ein tiefer Lastverteilbaum eingesetzt werden, der Bandbreitenbedarf steigt dementsprechend.

N-Tree zeigt ebenfalls einen geringen Bandbreitenbedarf. In diesem Szenario ist nur eine geringe Korrelation zwischen Gruppengröße und Bandbreitenbedarf zu erkennen. Der Bandbreitenbedarf bei einer Gruppengröße von 1 liegt bei 4,2 kByte/s und bei einer Gruppengröße von 60 bei 4,3 kByte/s. In diesem Szenario dominiert die Verteilung der Ereignisnachrichten den Bandbreitenbedarf mit ungefähr 80% des Gesamtbedarfes. Die Zahl der Teilnehmer pro Zone ist in diesem Szenario relativ unabhängig von der maximalen Gruppengröße: Zwar steigt bei größeren Gruppen die Teilnehmerdichte, dies wird in diesem Szenario aber durch die Verkleinerung der Zonen ausgeglichen[4]. Der minimale Bandbreitenbedarf wird in dem Szenario bei der Gruppengröße 10 erreicht. Hier werden nur 3,6 kByte/s benötigt. Aufgrund der Wahl der Schwellenwerte in N-Tree wird bei dieser Gruppengröße die minimale Zahl von Teilnehmern pro Zone erreicht[5].

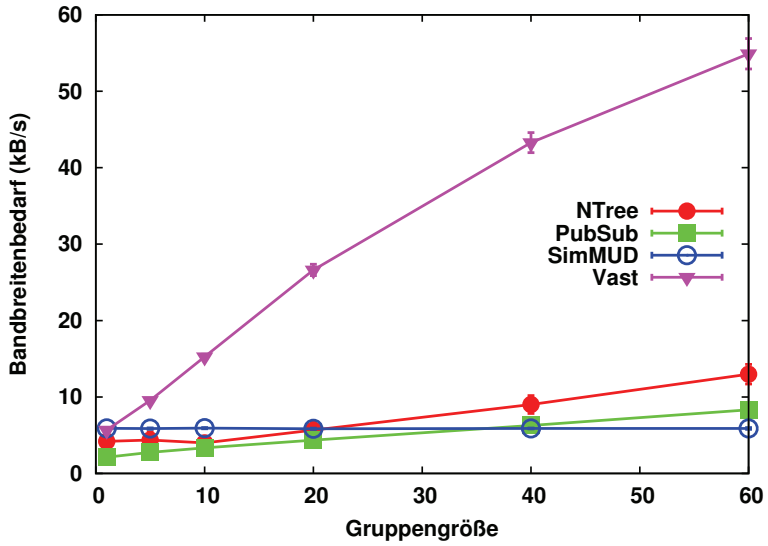
Von den untersuchten Protokollen ist SimMUD das einzige, bei dem der Bandbreitenbedarf nicht von der Gruppengröße abhängt. Bei der Spielfeldgröße 10.000 m mal 10.000 m beträgt dieser 5,1 kByte/s. Da bei statischen Zonen die Anzahl der Teilnehmer pro Zone unabhängig von der Gruppengröße ist, ändert sich die durchschnittliche Gesamtzahl an Empfängern für Positionsmeldungen nicht mit der Gruppengröße. Da

[4]Siehe hierzu die Darstellung der Zonengrößen in N-Tree in Abbildung A.1a in Anhang A.

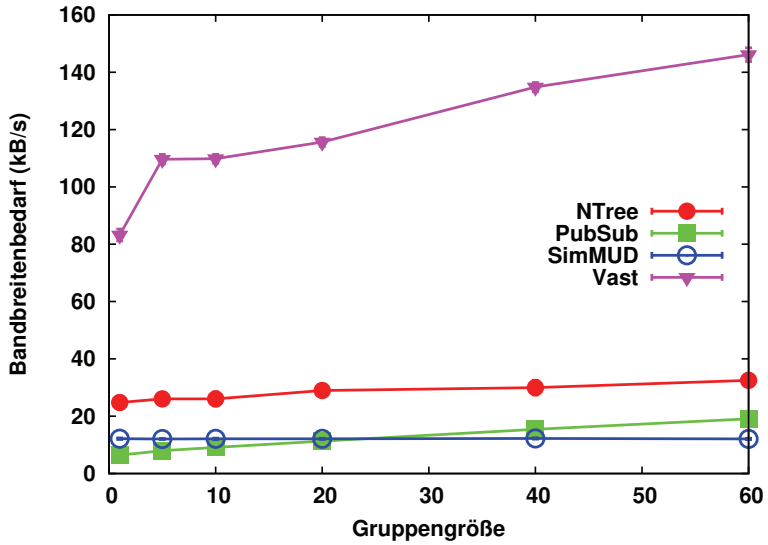
[5]Siehe hierzu die Darstellung der Zahl der Teilnehmer pro Zone in N-Tree in Abbildung A.1b in Anhang A.



(a) Bandbreitenbedarf bei Spielfeldgröße 10.000 m mal 10.000 m.



(b) Bandbreitenbedarf bei Spielfeldgröße 5.000 m mal 5.000 m.



(c) Bandbreitenbedarf bei Spielfeldgröße 1.000 m mal 1.000 m.

Abbildung 5.6 Evaluierungsergebnisse: Bandbreitenbedarf in Abhängigkeit der Gruppengröße.

der Multicastbaum in SimMUD ausschließlich vom unterliegenden KBR-Protokoll festgelegt wird, ändert auch hier die Gruppengröße nichts an der Gesamtzahl der benötigten Nachrichten. Somit ergibt sich in SimMUD ein von der Gruppengröße unabhängiger Bandbreitenbedarf.

Der Bandbreitenbedarf von Vast ist der höchste der untersuchten Protokolle. Zusätzlich steigt er sehr viel stärker als bei den anderen Protokollen mit steigender Gruppengröße. Bei einer Gruppengröße von 1 beträgt der Bandbreitenbedarf knapp 6,3 kByte/s und steigt auf ungefähr 20,5 kByte/s bei einer Gruppengröße von 60. Der Anstieg liegt vor allem darin begründet, dass Teilnehmer bei Vast Verbindungen zu allen Nachbarn aufnehmen müssen und die durchschnittliche Anzahl an Nachbarn mit steigender Gruppengröße zunimmt. Zusätzliche Faktoren sind die Nachbarsfindung und der verwendete Backup-Mechanismus, deren Aufwand bei m Nachbarn bei $O(m^2)$ liegt und somit bei hoher Teilnehmerdichte zu einem deutlichen Mehraufwand führt.

In Abbildung 5.6b wird der Bandbreitenbedarf bei einer Spielfeldgröße von 5.000 m mal 5.000 m gezeigt. Bei allen evaluierten Protokollen ist die benötigte Bandbreite aufgrund der höheren Teilnehmerdichte höher als bei der Spielfeldgröße 10.000 m mal 10.000 m. PubSubMMOG benötigt bei kleinen Gruppengrößen wieder die kleinste Bandbreite unter den evaluierten Protokollen. Bei einer Gruppengröße von 1 liegt sie bei 2,1 kByte/s. Bei einer Gruppengröße von 60 steigt sie auf 8,3 kByte/s. Wie bei der größeren Spielfeldgröße entspricht auch hier der Anstieg den Erwartungen.

Der Bandbreitenbedarf von N-Tree liegt leicht oberhalb des Bandbreitenbedarfs von PubSubMMOG. Bei einer Gruppengröße von 1 liegt der Bandbreitenbedarf bei 4,2 kByte/s. Er steigt bei einer Gruppengröße von 60 auf 13,0 kByte/s. Die Steigerung lässt sich damit erklären, dass hier die durchschnittliche Zahl der Teilnehmer pro Zone steigt[6]. Dies tritt, anders als bei der Spielfeldgröße 10.000 m mal 10.000 m, in diesem Szenario auf, weil die Zonen in diesem Szenario bereits so klein werden, dass bei einem signifikanten Teil der Teilnehmer das Interessengebiet bis in die benachbarte Zone reicht und der Teilnehmer somit Mitglied in mehreren Zonen werden muss[7]. Wie im vorigen Szenario liegt das Minimum des Bandbreitenbedarfs mit 4,0 kByte/s bei der Gruppengröße 10, wo auch hier die minimale Zahl an Teilnehmern pro Zone erreicht wird.

SimMUD benötigt nur wenig mehr Bandbreite als bei der Spielfeldgröße 10.000 m mal 10.000 m. Der Bandbreitenbedarf liegt unabhängig von der Gruppengröße bei ungefähr 5,9 kByte/s. Durch die geringere Zonengröße kann somit die höhere Teilnehmerdichte im Vergleich zu der Spielfeldgröße 10.000 m mal 10.000 m in SimMUD ausgeglichen werden.

Vast benötigt teilweise deutlich mehr Bandbreite. Bei einer Gruppengröße von 1 liegt der Bandbreitenbedarf bei 5,7 kByte/s und damit noch auf vergleichbarer Höhe zu dem Szenario mit der Spielfeldgröße 10.000 m mal 10.000 m. Er steigt jedoch deutlich mit steigender Gruppengröße und erreicht bei einer Gruppengröße von 60 ungefähr 55 kByte/s. Der Anstieg lässt sich auch hier mit der höheren Anzahl an Nachbarn und dem hohen Aufwand für die Nachbarsfindung und den Backup-Mechanismus erklären.

[6]Siehe hierzu die Darstellung der Zahl der Teilnehmer pro Zone in N-Tree in Abbildung A.1b in Anhang A.

[7]Siehe hierzu die Darstellung der Zonengrößen in N-Tree in Abbildung A.1a in Anhang A.

Abbildung 5.6c zeigt den Bandbreitenbedarf bei einer Spielfeldgröße von 1.000 m mal 1.000 m. Auch hier ist bei allen Protokollen ein höherer Bandbreitenbedarf im Verhältnis zu den anderen Szenarien zu verzeichnen. Bei geringen Gruppengrößen benötigt wiederum PubSubMMOG die geringste Bandbreite. Bei einer Gruppengröße von 1 zeigt PubSubMMOG einen Bandbreitenbedarf von 6,4 kByte/s. Bei einer Gruppengröße von 60 steigt der Bedarf auf 19,1 kByte/s.

N-Tree benötigt deutlich mehr Bandbreite als bei den größeren Spielfeldgrößen. Damit liegt es nun oberhalb aller anderen untersuchten Protokolle mit Ausnahme von Vast. Bei einer Gruppengröße von 1 benötigt es 24,8 kByte/s. Bei einer Gruppengröße von 60 steigt der Bandbreitenbedarf auf 32 kByte/s. Der Grund für die starke Steigerung im Vergleich zu den größeren Spielfeldgrößen ist, dass bei den in diesem Szenario auftretenden hohen Teilnehmerdichten die Zonen nicht mehr hinreichend aufgeteilt werden können. Die durchschnittliche von den Teilnehmern beobachtete Zonengröße liegt unabhängig von der Gruppengröße bei 75 m. Da bei einer weiteren Aufteilung die Zonengröße unter die Interessengebietsgröße sinken würde, wird, wie in Abschnitt 5.1.3.2 beschrieben, die Zonenaufteilung ausgesetzt. Damit ist die Anzahl der Teilnehmer pro Zone deutlich höher als bei den größeren Spielfeldgrößen, der Aufwand für die Verteilung der Positionsmeldungen steigt entsprechend.

Der Bandbreitenbedarf von SimMUD ist, wie bei den anderen Szenarien, von der Gruppengröße unabhängig. Er beträgt 12,1 kByte/s. Die Steigerung im Verhältnis zu den anderen Szenarien erklärt sich hier dadurch, dass bei den hier verwendeten kleineren Zonen Teilnehmer öfter Zonen wechseln und öfter Mitglied in mehreren Zonen sind, sodass Positionsmeldungen im Durchschnitt an mehr Empfänger zu versenden sind[8].

Vast zeigt in diesem Szenario einen stark erhöhten Bandbreitenbedarf. Bei einer Gruppengröße von 1 beträgt er 83,2 kByte/s und steigt bei einer Gruppengröße von 60 auf 146 kByte/s. In diesem Szenario ist der Bandbreitenbedarf von Vast durch den Aufwand für die Nachbarsfindung und den Backup-Mechanismus dominiert[9]. Der Aufwand für den Backup-Mechanismus setzt sich aus der Zahl der als Backup-Knoten fungierenden Teilnehmer sowie der Zahl der Nachbarn dieser Teilnehmer zusammen. Aufgrund der hohen Teilnehmerdichte erreicht die Zahl der Backup-Knoten bereits bei einer kleinen Gruppengröße ein Maximum, sodass bei größeren Gruppen der Aufwand für den Backup-Mechanismus nur noch linear mit der Zahl der Nachbarn steigt. Der Aufwand für die Nachbarsfindung hängt unter anderem von der Zahl der Grenznachbarn eines Teilnehmers ab. Bei der hohen Teilnehmerdichte in diesem Szenario sind die Grenznachbarn oft nicht Mitglieder der eigenen Teilnehmergruppe, sodass nach einer starken initialen Steigung von der Gruppengröße 1 auf die Gruppengröße 5 die Gruppengröße einen geringeren Einfluss auf die Zahl der Grenznachbarn und somit den Bandbreitenbedarf hat als in Szenarien mit geringerer Teilnehmerdichte.

Zusammenfassend kann man festgehalten, dass der Bandbreitenbedarf von SimMUD und PubSubMMOG in allen Szenarien mit unter 20 kByte/s relativ gering ist. Alle heute üblichen Breitbandanschlüsse können diese Bandbreite liefern. Der Bandbreitenbedarf von N-Tree ist bei hoher Teilnehmerdichte mit bis zu 32 kByte/s grö-

[8] Siehe hierzu die Darstellung der Zahl der Zonenbeitritte in Abbildung A.2 in Anhang A.

[9] Siehe hierzu die Darstellung des Bandbreitenbedarfs für Nachbarsfindung und Backup-Mechanismus in Vast in Abbildung A.3 in Anhang A.

ßer. Gängige DSL-Anschlüsse besitzen mindestens eine maximale Bandbreite von 386 kBit/s und können demnach die Bandbreitenanforderungen von N-Tree erfüllen, könnten aber bei Lastspitzen ihre Leistungsgrenze erreichen. Der Bandbreitenbedarf von Vast ist hingegen signifikant zu groß. Der maximale Durchschnittsbedarf von 146 kByte/s liegt deutlich über den Möglichkeiten gängiger Internetanbindungen. Zum Vergleich beträgt der Bandbreitenbedarf des Spiels „World of Warcraft“ in einem vergleichbaren Szenario 25 kByte/s [133].

5.2.4.3 Nachbarschaftskenntnis

Als dritte Metrik wurde die durchschnittliche Nachbarschaftskenntnis der Protokolle evaluiert. Abbildung 5.7 zeigt die dazugehörenden Ergebnisse. In Abbildung 5.7a ist die Nachbarschaftskenntnis bei einer Spielfeldgröße von 10.000 m mal 10.000 m dargestellt. Die beste Nachbarschaftskenntnis kann PubSubMMOG aufweisen: Diese liegt durchgängig bei über 99,99%. Die von zentraler Stelle verwalteten Koordinatoren und Lastverteilbäume können also durchgängig für alle Teilnehmer die Nachbarschaftskenntnis sichern. Auch ein Ausfall der betreffenden Teilnehmer kann ohne Einbußen für das Netz korrigiert werden.

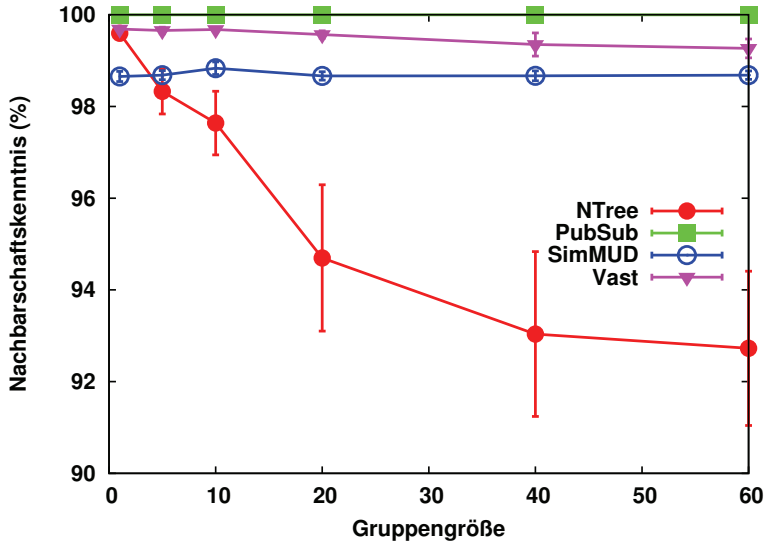
Vast erreicht in diesem Szenario die zweitbeste Nachbarschaftskenntnis. Diese fällt leicht bei steigender Gruppengröße. Bei einer Gruppengröße von 1 werden 99,7% erreicht, bei einer Gruppengröße von 60 noch 99,3%. Der Grund für diesen Trend ist, dass bei der Begegnung größerer Gruppen in kurzer Zeit viele Nachbarschaftsbeziehungen wechseln, sodass schon bei geringen Abweichungen der lokalen Voronoi-Diagramme nicht alle neuen Nachbarschaftsbeziehung korrekt gemeldet werden.

Die von SimMUD erreichte Nachbarschaftskenntnis ist im Wesentlichen unabhängig von der Gruppengröße und beträgt zwischen 98,6% und 98,8%. Eine Ursache für die im Verhältnis zu PubSubMMOG und Vast geringere Nachbarschaftskenntnis liegt in der langsameren Kompensation von Teilnehmerausfällen: Fallen Knoten des Multicastbaumes aus, ist die Weiterleitung von Nachrichten gestört, bis die Knoten durch das unterliegende KBR-Protokoll Pastry ersetzt werden.

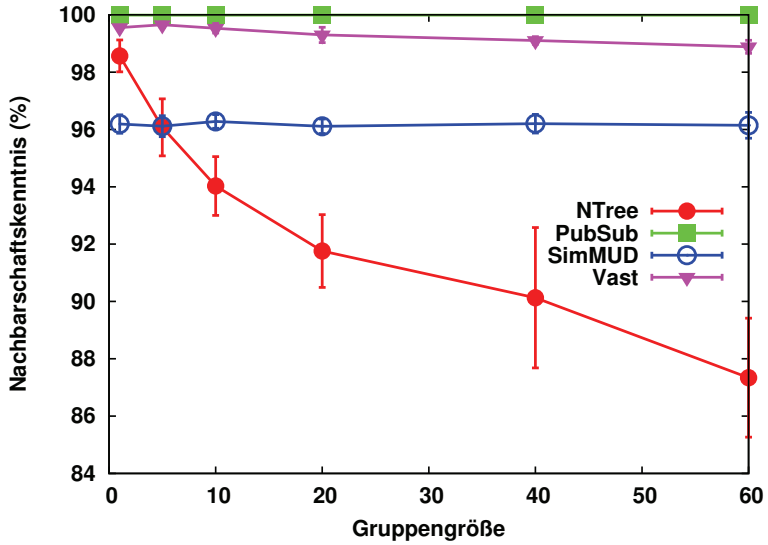
N-Tree erreicht insgesamt die geringste Nachbarschaftskenntnis. Diese ist zudem sehr stark von der Gruppengröße abhängig. Werden bei einer Gruppengröße von 1 noch gute 99,6% erreicht, fällt dieser Wert bei einer Gruppengröße von 60 auf nur noch 92,7%. Erreicht die Teilnehmerzahl einer Zone in N-Tree einen Schwellenwert, wird die Zone in kleinere Zonen unterteilt. Da dies eine gewisse Zeit dauert, kann es in der Zwischenzeit zu fehlenden Nachbarschaftsbeziehungen kommen. Betritt eine große Gruppe nun eine Zone, ist die Wahrscheinlichkeit einer solchen Teilung groß.

Abbildung 5.7b zeigt die Nachbarschaftskenntnis bei einer Spielfeldgröße von 5.000 m mal 5.000 m. Wie beim Szenario mit dem größeren Spielfeld erreicht PubSubMMOG die besten Werte und bleibt stets zwischen 99,98% und 99,99%. Auch Vast erreicht im Wesentlichen die Werte des obigen Szenarios und bleibt zwischen 99,7% und 98,9%.

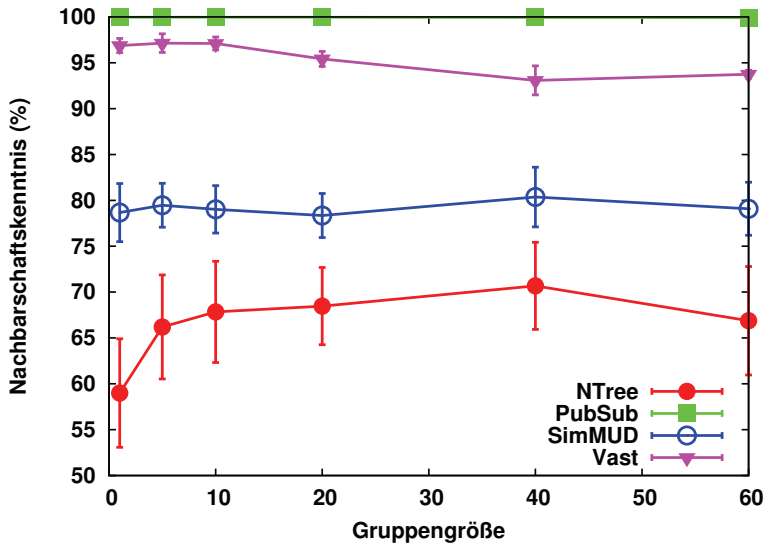
Die Nachbarschaftskenntnis von SimMUD liegt hingegen niedriger als im ersten Szenario. Es werden nur noch Werte zwischen 96,3% und 96,1% erreicht. Ein Grund hierfür sind die häufigeren Zonenübertritte der Teilnehmer aufgrund der kleineren Zonen. Die häufigen Zonenübertritte verstärken das Problem der langsamen Reparatur der Multicastbäume: Ist der Multicastbaum beim Eintritt eines Teilnehmers



(a) Nachbarschaftskenntnis bei Spielfeldgröße 10.000 m mal 10.000 m.



(b) Nachbarschaftskenntnis bei Spielfeldgröße 5.000 m mal 5.000 m.



(c) Nachbarschaftskenntnis bei Spielfeldgröße 1.000 m mal 1.000 m.

Abbildung 5.7 Evaluierungsergebnisse: Nachbarschaftskenntnis in Abhängigkeit der Gruppengröße.

in eine Zone defekt, wird der nicht erreichbare Teil des Multicastbaumes nicht über den Eintritt eines neuen Teilnehmers informiert.

Auch die Werte für N-Tree sind deutlich schlechter als bei einer Spielfeldgröße von 10.000 m mal 10.000 m. Bei einer Gruppengröße von 1 wird noch eine Nachbarschaftskenntnis von 98,6% erreicht, bei einer Gruppengröße von 60 beträgt der Wert nur noch 87,3%. Auch hier verstärkt der häufigere Zonenwechsel der Teilnehmer die im ersten Szenario beschriebenen Probleme.

In Abbildung 5.7c ist die Nachbarschaftskenntnis bei einer Spielfeldgröße von 1.000 m mal 1.000 m abgebildet. Bei fast allen Protokollen sind die Werte deutlich schlechter als bei den anderen Szenarien. Die Ausnahme ist hier PubSubMMOG. Es erreicht mit Werten zwischen 99,91% und 99,96% fast die guten Werte der größeren Spielfelder.

Vast erreicht in diesem Szenario nur noch Werte zwischen 96,8% bei kleinen Gruppen und 93,1% bei großen Gruppen. Aufgrund der hohen Teilnehmerdichte in diesem Szenario führen bereits kleine Abweichungen der Positionen zu großen Unterschieden in den Voronoi-Diagrammen. Dies kann dazu führen, dass Teilnehmer nicht oder nicht rechtzeitig über neue Nachbarn informiert werden.

Die von SimMUD erreichte Nachbarschaftskenntnis schwankt in diesem Szenario zwischen 78% und 80%. Hier verstärken sich die bereits vorher beschriebenen Probleme aufgrund der im Vergleich zum vorigen Szenario nochmals deutlich erhöhten Teilnehmerdichte. Bei einer Nachbarschaftskenntnis von 80% sind die Interaktionsmöglichkeiten der Teilnehmer bereits deutlich eingeschränkt.

Bei N-Tree ist, anders als bei den vorigen Szenarien die direkte Korrelation zwischen Gruppengröße und Nachbarschaftskenntnis nicht mehr gegeben. Bereits bei einer Gruppengröße von 1 ist das Minimum von 59% erreicht. Von dort steigt der Wert bis auf 71% bei einer Gruppengröße von 40, um dann bei einer Gruppengröße von 60 auf 67% zu fallen. Dabei fällt auch die deutlich stärkere Streuung der Ergebnisse auf. Im Wesentlichen zeigen die Ergebnisse, dass N-Tree bei hoher Teilnehmerdichte keine sinnvolle Weiterleitung der Positionsmeldungen mehr leisten kann. Durch die häufigen Zonenwechsel der Teilnehmer sowie die häufige Teilung und Zusammenlegung der Zonen ist der verwendete Quadtree über größere Zeiträume nicht mehr in einem konsistenten Zustand, sodass eine sinnvolle Information der Teilnehmer über Nachbarn nicht mehr erreicht werden kann.

Zusammenfassend kann PubSubMMOG in allen Szenarien eine sehr gute Nachbarschaftskenntnis erreichen. Vast weist in vielen Szenarien akzeptable Werte auf. Bei großen Teilnehmerdichten erreicht die Nachbarschaftskenntnis bei Vast nur noch 93%; dies kann die Interaktionsmöglichkeiten einiger Teilnehmer bereits deutlich einschränken. SimMUD und N-Tree zeigen nur bei geringen Teilnehmerdichten akzeptable Werte, zusätzlich zeigt N-Tree bei größeren Gruppen deutliche Einbußen bei der Nachbarschaftskenntnis. Bei hohen Teilnehmerdichten werden bei SimMUD minimale Werte von 78%, bei N-Tree von unter 60% gemessen. Dies ist in virtuellen Welten in der Regel nicht mehr hinnehmbar.

5.2.5 Schlussfolgerungen

SimMUD bietet akzeptable Latenzen bei einem geringen Bandbreitenbedarf. Allerdings sinkt bei hoher Teilnehmerdichte die Nachbarschaftskenntnis auf inakzeptable

Werte. Die Ursache für dieses Problem ist, dass SimMUD auf das unterliegende ALM-Protokoll für die Ausfallerkennung und Reparatur der Multicastbäume angewiesen ist. Übliche ALM-Protokolle sind in der Regel jedoch nicht für die in virtuellen Welten häufigen schnellen Gruppenwechsel optimiert. Da zudem der Zonenwechsel in SimMUD bei n Teilnehmern in der virtuellen Welt mit einem Aufwand von $O(\log(n))$ relativ lang dauert, kann bei häufigen Zonenwechseln keine ausreichende Nachbarschaftskenntnis erreicht werden.

PubSubMMOG kann bei geringem Bandbreitenbedarf eine sehr gute Nachbarschaftskenntnis erreichen. Dafür benötigt PubSubMMOG allerdings einen zentralen Server. Dieser ist nicht nur in den Login der Teilnehmer involviert, sondern muss durchgängig über den Lastzustand aller Teilnehmer informiert sein und aktiv Koordinatoren und Lastverteilbäume bestimmen und verwalten. Dadurch stellt er einen *single point of failure* dar: Ein Ausfall des Servers hat einen Ausfall der gesamten virtuellen Welt zur Folge. Ein weiterer Nachteil von PubSubMMOG sind die Latenzen. Der zeitliche Aufwand für die Verteilung von Ereignisnachrichten und Positionsmitteilungen liegt in PubSubMMOG bei m Teilnehmern in einer Zone bei $O(\log(m))$. Insbesondere beim Auftreten großer Teilnehmergruppen sind die Latenzen für schnelle Interaktionen zu hoch. Diese Latenzen lassen sich bei zonenbasierten Protokollen mit statischen Zonen und Supernodes nur schwer vermeiden, da bei hoher Teilnehmerzahl in der Zone ein Lastverteilbaum geschaffen werden muss, der unvermeidlich Ereignisnachrichten und Positionsmeldungen zusätzlich verzögert.

N-Tree bietet hervorragende Latenzen bei geringem Bandbreitenbedarf. Dies ist ein grundsätzlicher Vorteil zonenbasierter Protokolle mit dynamischen Zonen: Da die Zonen lastabhängig verkleinert werden, wird der Bandbreitenbedarf beschränkt. Dadurch kann auf weitere Lastverteilmechanismen verzichtet werden, sodass die Nachrichtenverzögerungen gering gehalten werden können. Die Verwaltung der Zonen ist allerdings fehleranfällig. Insbesondere bei hoher Teilnehmerdichte müssen Zonen häufig aufgeteilt oder zusammengelegt werden. Dies wirkt sich negativ auf die Nachbarschaftskenntnis aus. Zudem ist der Aufwand für den Zonenwechsel bei n Teilnehmern der virtuellen Welt mit $O(n))$ sehr hoch. N-Tree kann deswegen bei hoher Teilnehmerdichte keine akzeptable Nachbarschaftskenntnis erreichen.

Vast kann hervorragende Latenzen bieten. Dies ist ein wesentlicher Vorteil von Mutual Notification Protokollen: Alle Positionsmeldungen und Ereignisnachrichten werden direkt vom Verursacher an die interessierten Teilnehmer zugestellt. Es gibt keine Zwischenknoten auf der Protokollebene, die zusätzliche Latenzen verursachen könnten. Die Nachbarschaftskenntnis bei Vast bleibt dabei akzeptabel. Ein spezifischer Nachteil von Vast ist jedoch der Bandbreitenbedarf. Dieser liegt hauptsächlich im Aufwand für die von Vast verwendeten Backup-Mechanismen begründet: Bei m Nachbarn besitzen diese einen Aufwand von $O(m^2)$. Bei größeren Teilnehmerdichten benötigt Vast deutlich mehr Bandbreite als durchschnittliche Nutzer zur Verfügung haben.

5.3 Zusammenfassung

In diesem Kapitel wurde eine Auswahl an P2P-Protokollen für virtuelle Welten vorgestellt. Es wurden Aufwandabschätzungen bezüglich Latenz, Bandbreitenbedarf und Nachbarschaftskenntnis durchgeführt. Abschließend wurden die Protokolle in einer Simulation evaluiert.

Die Ergebnisse der Evaluierung bestätigen die Vermutungen, die aufgrund der Aufwandsabschätzung in Abschnitt 5.1.5 getroffen wurden. Es zeigt sich, dass keines der untersuchten Protokolle gleichzeitig eine gute Latenz, eine gute Nachbarschaftskenntnis und einen akzeptablen Bandbreitenbedarf bieten kann.

Wie in Abschnitt 5.1.5 dargelegt, sind viele der Aufwände der Protokolle systematisch für die jeweilige Protokollkategorie. In Protokollen mit statischen Zonen fällt für die Verteilung von Positionsmeldungen und Ereignisnachrichten aufgrund der Weiterleitung der Nachrichten über Lastverteil- oder Multicastbäume in der Regel ein mindestens logarithmischer zeitlicher Aufwand an. Die hierdurch erzeugten Latenzen sind in vielen Szenarien zu hoch. In Protokollen mit dynamischen Zonen ist es aufgrund des in der Regel linearen Aufwandes für den Zonenwechsel schwierig, eine ausreichende Nachbarschaftskenntnis sicherzustellen. Zonenlose Protokolle können sowohl eine gute Nachrichtenverzögerung als auch eine gute Nachbarschaftskenntnis bieten. Das hier untersuchte Protokoll Vast besitzt jedoch einen zu hohen Bandbreitenbedarf. Andere zonenlose Protokolle sind Vast relativ ähnlich, sodass hier vergleichbare Ergebnisse zu erwarten sind.

Um alle hier untersuchten Anforderungen virtueller Welten ausreichend zu unterstützen, ist also die Entwicklung eines neuen Protokolls nötig. Aufgrund der in diesem Kapitel gewonnenen Erkenntnisse ist es am wahrscheinlichsten, dass sich die Anforderungen mit einem zonenlosen *MutualNotification*-Protokoll erfüllen lassen. Aus diesem Grund verwendet das im Rahmen dieser Arbeit entwickelte Protokoll *QuON* keine Unterteilung der Spielwelt in Zonen. QuON verbindet die geringe Latenz von Mutual Notification-Protokollen mit einer guten Nachbarschaftskenntnis und einem deutlich geringerem Bandbreitenbedarf.

6. Ein Overlay-Protokoll für verteilte virtuelle Welten: QuON

Die im vorigen Kapitel vorgestellten Ergebnisse zeigen, dass Peer-to-Peer-Protokolle sich grundsätzlich dafür eignen, die Anforderungen verteilter virtueller Welten zu erfüllen. Bisherige Protokolle schaffen es jedoch nicht, eine geringe Latenz mit einer ausreichenden Nachbarschaftskenntnis und einem akzeptablen Bandbreitenbedarf zu verbinden. In dieser Arbeit wird das „Quadrantenbasierte Overlay-Netz“ QuON [8][1] als ein neues Peer-to-Peer-Protokoll für verteilte virtuelle Welten vorgestellt. QuON bietet die Latenzvorteile bisheriger Mutual-Notification-Protokolle, erreicht im Gegensatz zu diesen aber eine bessere Nachbarschaftskenntnis und einen deutlich geringeren Bandbreitenbedarf. In diesem Kapitel wird die Funktionsweise von QuON im Detail erläutert.

6.1 Grundidee

Die Ergebnisse der Untersuchung bisheriger Protokolle in Kapitel 5 zeigen, dass zonenlose *Mutual-Notification-Protokolle*, bei denen die Teilnehmer Nachrichten über Positionsänderungen oder Spielereignisse ohne Zwischenknoten direkt an alle interessierten Teilnehmer verteilen, deutliche Latenzvorteile gegenüber anderen Protokollen bieten. Aus diesem Grund wird dieses Konzept für QuON übernommen.

Dafür ist es zunächst nötig, aus der Menge aller Teilnehmer der virtuellen Welt die Menge der interessierten Teilnehmer zu bestimmen. Da in virtuellen Welten alle Aktionen von Spielern nur lokalen Einfluss haben, befinden sich alle an den Aktionen eines bestimmten Teilnehmers interessierten Teilnehmer in einem gewissen Umkreis um diesen Teilnehmer. Dieses Gebiet wird als Interessengebiet definiert. Alle Teilnehmer, die sich innerhalb des Interessengebiets aufhalten, werden als *direkte Nachbarn*

[1] In der dort veröffentlichten Version basierte QuON noch auf Quad-Trees für die Wahl von Verbindungsnachbarn. In dieser Arbeit wird eine optimierte Version von QuON vorgestellt, die auf einer einfacheren, quadrantenbasierten Auswahl basiert.

bezeichnet. Damit alle direkten Nachbarn über alle Aktionen eines Teilnehmers informiert werden, muss dieser eine Verbindung zu allen diesen Nachbarn halten. Über diese Verbindung werden dann Nachrichten über alle vom Teilnehmer ausgehenden Ereignisse versendet. Insbesondere werden regelmäßig Positionsmeldungen versendet.

Somit sind alle Teilnehmer über die Positionen aller direkten Nachbarn, zu denen eine Verbindung besteht, informiert. Die Herausforderung ist nun, dass sich die Spieler in der virtuellen Welt bewegen. Somit können zu nicht vorhersagbaren Zeitpunkten neue Teilnehmer in die Interessengebiete anderer Teilnehmer eintreten. Da zwischen den neu eintretenden Teilnehmern und den Teilnehmern, zu denen das Interessengebiet gehört, zu diesem Zeitpunkt keine Verbindung besteht, müssen sie von dritten Teilnehmern über dieses Ereignis informiert werden.

Diese Aufgabe kann zunächst von den direkten Nachbarn übernommen werden. Dazu prüfen diese beim Empfang einer Positionsmeldung eines Teilnehmers A, ob er durch eine Bewegung in das Interessengebiet eines anderen Teilnehmers B eingetreten ist. Ist dies der Fall, wird A über dieses Ereignis informiert. Dieses Verfahren wird in Abschnitt 6.6.1 näher erläutert.

Dieses Verfahren ist jedoch nicht ausreichend, um ein zusammenhängendes Netz zu gewährleisten. Um den Netzzusammenhalt zu sichern, werden in QuON die sogenannten *Verbindungsnachbarn* eingeführt. Um diese zu bestimmen, teilt jeder Teilnehmer die virtuelle Welt bezüglich seiner Position in vier Quadranten auf. In jedem dieser Quadranten wird der nächste Teilnehmer zum Verbindungsnachbarn bestimmt. Die Verbindungsnachbarn werden ausschließlich aufgrund lokalen Wissens bestimmt und explizit über ihre Zuständigkeit informiert. Dabei ist kein konsistentes Wissen über die Positionen aller Nachbarn nötig. Die in QuON verwendete Wahl der Verbindungsnachbarn ermöglicht eine Selbst-Stabilisierung: Wählt ein Teilnehmer aufgrund fehlenden lokalen Wissens einen nicht-optimalen Verbindungsnachbarn, wird er in QuON iterativ auf bessere Verbindungsnachbarn hingewiesen, bis der optimale Verbindungsnachbar gefunden ist. Die Bestimmung von Verbindungsnachbarn wird in Abschnitt 6.4.2 näher erläutert.

Die Verwendung explizit bestimmter Verbindungsnachbarn stellt einen wesentlichen Unterschied zwischen QuON und bisherigen Mutual-Notification-Protokollen wie Vast dar. Diese versuchen, mit Hilfe von implizit aus Voronoi-Graphen oder damit verwandten Datenstrukturen bestimmten Beziehungen zwischen Teilnehmern, den Netzzusammenhalt zu sichern. Diese Vorgehensweise setzt jedoch voraus, dass das Voronoi-Diagramm über die von den Teilnehmern angenommenen Positionen mit der entsprechenden Datenstruktur der Nachbarn übereinstimmt. Wird diese Annahme jedoch verletzt, kann der Netzzusammenhalt nicht mehr gewährleistet werden. In der Praxis kommt es aufgrund von Nachrichtenverzögerungen oder Teilnehmerausfällen regelmäßig zu Inkonsistenzen der Datenstrukturen. Hier liegt eine wesentliche Ursache für die bei Mutual-Notification-Protokollen beobachteten Probleme [7].

In QuON sind Verbindungsnachbarbeziehungen meist symmetrisch. Es kann jedoch auch vorkommen, dass ein Teilnehmer A Verbindungsnachbar von B ist, B jedoch nicht Verbindungsnachbar von A. In diesem Fall wird die *temporäre Nachbarschaftsbeziehung* eingeführt: Ist A Verbindungsnachbar von B, B jedoch kein Verbindungsnachbar von A, so wird B temporärer Nachbar von A. Temporäre Nachbarn sind

hinsichtlich ihrer Aufgaben Verbindungsnachbarn gleichgestellt. Eine genauere Erläuterung der temporären Nachbarschaftsbeziehung findet sich in Abschnitt 6.4.3.

Mit Verbindungsnachbarn und temporären Nachbarn werden regelmäßig Nachrichten über die eigene Position sowie die Position der anderen Verbindungsnachbarn und temporären Nachbarn ausgetauscht. Mit Hilfe dieser Nachbarschaftsbeziehungen kann der Netzzusammenhalt gesichert werden. Zusätzlich können mit diesen Nachbarschaftsbeziehungen auch solche neuen Nachbarn gefunden werden, die unbemerkt von direkten Nachbarn in das Interessengebiet eines Spielers eintreten. Dies wird in Abschnitt 6.6.2 genauer ausgeführt.

Die in QuON verwendete Auswahl der Verbindungsnachbarn verringert den Protokolloverhead im Vergleich zu bisherigen Protokollen. Bei hoher Teilnehmerdichte sind Verbindungsnachbarn in der Regel auch direkte Nachbarn, zu denen ohnehin eine Verbindung aufgebaut werden muss. Somit müssen im Regelfall keine zusätzlichen Verbindungen aufgebaut werden oder zusätzliche Nachrichten versendet werden. Bei geringer Spielerdichte ist der Bandbreitenbedarf von Mutual-Notification-Protokollen aufgrund der geringen durchschnittlichen Zahl an Nachbarn relativ gering. Trotzdem sollte auch in diesen Situationen der Protokolloverhead begrenzt sein. Bei QuON benötigt ein Teilnehmer in jedem Quadranten einen Verbindungsnachbarn; die Anzahl an Verbindungsnachbarn ist so auf vier begrenzt. Aufgrund der *temporären Nachbarschaftsbeziehung* können kurzzeitig eine größere Anzahl an Verbindungen gehalten werden. Dies tritt jedoch nur selten auf, wie in der Evaluierung im Kapitel 8 gezeigt wird.

Der typische Protokollablauf eines Teilnehmers von QuON sieht wie folgt aus: Jeder Teilnehmer empfängt kontinuierlich Positionsmeldungen anderer Teilnehmer. Beim Empfang einer solchen Meldung wird überprüft, ob der Absender seit seiner letzten Bewegungsmeldung neu in das Interessengebiet eines anderen Teilnehmers eingetreten ist. Ist dies der Fall, wird er über dieses Ereignis informiert. Weiterhin wird der Absender der Positionsmeldung in einer Liste der bekannten Teilnehmer gespeichert. Periodisch werden die auf dieser Liste gespeicherten Teilnehmer gemäß der oben beschriebenen Nachbarschaftsbeziehungen klassifiziert. Teilnehmer, zu denen nach der Klassifizierung keine Nachbarschaftsbeziehung besteht, werden im Anschluss aus der Liste entfernt. Danach wird die aktuelle Position an alle verbleibenden Nachbarn versendet.

6.2 Mögliche Metriken in QuON

Da in QuON die Verbindungen zwischen den Teilnehmern von ihren Positionen in der virtuellen Welt und ihrem Abstand zueinander abhängen, hat die Wahl der Abstandsmetrik einen Einfluss auf das Protokoll. Üblicherweise wird in virtuellen Welten mit dem Euklidischen Abstand gearbeitet. Dies ist auch der für den Spieler relevante Abstand. Einige Eigenschaften von QuON lassen sich jedoch nur bei Verwendung der Maximumsnorm beweisen. Im zweidimensionalen Raum ist der Abstand zweier Punkte $a = (x_a, y_a)$ und $b = (x_b, y_b)$ unter der Maximumsmetrik definiert als $d(a, b) = \max(x_a - x_b, y_a - y_b)$.

Bei der von QuON verwendeten Quadrantenstruktur lässt sich bei Verwendung der Maximumsnorm zeigen, dass bei nur einem Verbindungsnachbarn pro Quadranten der Netzzusammenhalt und eine perfekte Nachbarschaftskenntnis garantiert werden

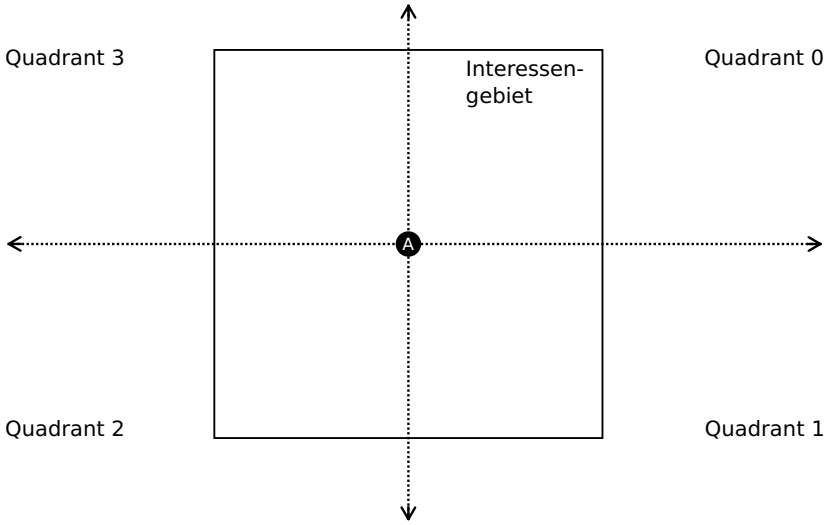


Abbildung 6.1 Die Aufteilung der virtuellen Welt in Quadranten und das Interessengebiet eines Teilnehmers.

kann.[2] Bei Verwendung der Maximumsnorm vergrößert sich jedoch die Fläche des Interessengebiets eines Teilnehmers. Hieraus resultiert eine größere durchschnittliche Zahl an direkten Nachbarn und somit ein größerer durchschnittlicher Bandbreitenbedarf. Im Folgenden wird in diesem Kapitel, wenn nicht anders erwähnt, von der Verwendung der Maximumsmetrik ausgegangen; alle Angaben zu Abständen und Interessengebietsradien sind bezüglich dieser Metrik zu verstehen.

Um die benötigte Bandbreite zu reduzieren, kann die Verwendung der Euklidischen Abstandsmetrik in Betracht gezogen werden. Dabei können bei ungünstiger Positionierung der Teilnehmer in der virtuellen Welt Einbußen in der Nachbarschaftskenntnis nicht ausgeschlossen werden. Die Evaluierung zeigt jedoch, dass die Einbußen in der Regel zu vernachlässigen sind. Genauere Analysen, wie sich die Wahl der Metrik auf Bandbreitenbedarf und Netzkonsistenz auswirkt, finden sich in Abschnitt 8.1.4.

6.3 Definitionen

Bevor in den folgenden Abschnitten die Funktionen von QuON genauer betrachtet werden, sollen zunächst einige Begriffe definiert werden: Bereits in Kapitel 2 wurde der Begriff des *Interessengebiets* eingeführt. Das Interessengebiet ist die Region um die aktuelle Position eines Teilnehmers, auf die er gemäß den Regeln der virtuellen Welt Einfluss ausüben kann. Alle Teilnehmer, mit denen er unmittelbar interagieren kann, befinden sich innerhalb dieses Interessengebiets. In QuON wird das Interessengebiet vereinfacht definiert als eine bezüglich der gewählten Metrik kreisförmige Region um die aktuelle Position des Spielers mit einem gegebenen Radius. Dieser Radius ist so gewählt, dass das Interessengebiet alle Teilnehmer umfasst, die gemäß den Regeln der virtuellen Welt mit dem jeweiligen Teilnehmer interagieren können. Es ist anzumerken, dass bei Verwendung der Maximumsnorm das Interessengebiet eine quadratische Form besitzt. Abbildung 6.1 zeigt das Interessengebiet eines Teilnehmers. Der Radius des Interessengebiets eines Teilnehmers t soll im Folgenden als r_t^{AoI} bezeichnet werden.

[2] Ein formeller Beweis hierfür findet sich in Anhang B.1 und B.2.

In den folgenden Abschnitten werden die Protokollabläufe von QuON im Pseudocode dargestellt. Da die Algorithmen von jedem Teilnehmer jeweils mit lokalem Wissen angewendet werden, sind alle in den Algorithmen verwendeten Variablen und Mengen jeweils als lokales Wissen des ausführenden Teilnehmers zu verstehen. Dieser wird in den Algorithmen als A bezeichnet. Weiterhin werden folgende Notationen festgelegt:

- p: Bezeichnet die aktuelle Position des lokalen Teilnehmers (A).
- p_t: Bezeichnet die aktuelle Position eines Teilnehmers t.
- $p_t^{'}$: Bezeichnet die vorige Position eines Teilnehmers t.
- Kalligrafisch gesetzte Großbuchstaben wie $\mathcal{T}$ bezeichnen jeweils Mengen von Teilnehmern. In einer Implementierung des Protokolls können diese Mengen in einer Baumstruktur gespeichert werden.
- $\mathcal{P}_{\mathcal{X}}$: Bezeichnet die Menge der Positionen der Teilnehmer aus Menge $\mathcal{X}$. In einer Implementierung können in den Baumstrukturen, in denen Mengen von Teilnehmern gespeichert sind, zugleich auch deren Positionen gespeichert werden.
- send MESSAGE$(x, [y]) \mapsto t$: Bezeichnet das Senden einer Nachricht MESSAGE mit dem Inhalt x und optionalem Inhalt y an Teilnehmer t.
- Q_t^i: Bezeichnet die Teilnehmer in Quadrant i bezüglich der Quadrantenaufteilung von Teilnehmer t.

In den Entwurfsentscheidungen wird die Verwendung einer *Quadrantenstruktur* festgelegt. Hierbei teilt jeder Teilnehmer, ausgehend von seiner Position, die virtuelle Welt parallel zu den Koordinatenachsen in vier Flächen auf. Diese Quadranten werden wie in Abbildung 6.1 gezeigt mit Quadrant 0 bis Quadrant 3 bezeichnet. Für einen Teilnehmer A mit der Position $p_A = (x_A, y_A)$ lässt sich der Quadrant eines Teilnehmers t wie in Algorithmus 6.1 gezeigt bestimmen.

Vorbedingung: $p_t := (x_t, y_t)$: Position von Teilnehmer t
Nachbedingung: q: Quadrant von t
procedure BESTIMMEQUADRANT(t)
if $x_A < x_t$ und $y_A < y_t$ **then**
$q = 0$
else if $x_A < x_t$ und $y_A \geq y_t$ **then**
$q = 1$
else if $x_A \geq x_t$ und $y_A \geq y_t$ **then**
$q = 2$
else if $x_A \geq x_t$ und $y_A < y_t$ **then**
$q = 3$
end if
return q
end procedure

Algorithmus 6.1 Bestimmung des Quadranten

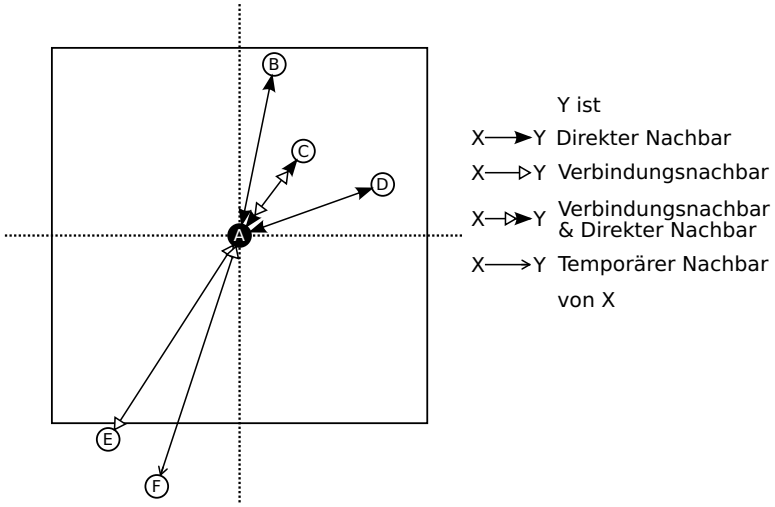


Abbildung 6.2 Mögliche Nachbarschaftsbeziehungen in QuON.

6.4 Nachbarschaftsbeziehungen

Ein wichtiger Schritt in QuON ist die Klassifizierung der Teilnehmer in verschiedene Nachbarschaftsbeziehungen. In QuON wird zwischen *direkten Nachbarn*, *Verbindungsnachbarn* und *temporären Nachbarn* unterschieden. Die Beziehungen sind beispielhaft in Abbildung 6.2 dargestellt. Direkte Nachbarn sind alle Teilnehmer, die sich innerhalb des Interessengebiets eines Teilnehmers befinden. Verbindungsnachbarn werden zur Sicherung des Netzzusammenhalts und zum Finden neuer Nachbarn benötigt. Temporäre Nachbarn sind kurzlebige Nachbarschaftsbeziehungen, die aufgrund möglicher Asymmetrie der Verbindungsnachbarschaftsbeziehung benötigt werden. Im Folgenden soll nun ausführlicher auf die einzelnen Nachbarschaftsbeziehungen eingegangen werden.

6.4.1 Direkte Nachbarn

Ein Spieler hat eine *direkte Nachbarschaftsbeziehung* zu allen Spielern, die sich innerhalb seines *Interessengebiets* befinden. Die Menge $\mathcal{N}$ der direkten Nachbarn bestimmt sich also aus der Menge aller Teilnehmer $\mathcal{T}$ wie folgt:

$$\mathcal{N} := \{t \in \mathcal{T} : d(p, p_t) \leq r^{AoI}\}$$

Da sich direkte Nachbarn somit im unmittelbaren Einflussbereich eines Teilnehmers befinden, müssen sie über alle Spielereignisse informiert werden. Besonders wichtig sind hier die Positionsmeldungen, da eine Bewegung der Nachbarn in einer virtuellen Welt möglichst unmittelbar grafisch dargestellt werden sollte. Jeder Spieler muss also regelmäßig seine aktuelle Position an alle direkten Nachbarn melden. In typischen Szenarien müssen diese Positionsmeldungen mehrmals pro Sekunde versendet werden. Die Bestimmung der direkten Nachbarn aus der Liste aller bekannten Teilnehmer ist in Algorithmus 6.2 beschrieben. Mit diesem Algorithmus werden alle direkten Nachbarn in $O(|\mathcal{T}|)$ bestimmt.

Direkte Nachbarn sind jedoch nicht nur aufgrund der direkten Interaktionsmöglichkeit wichtig. In QuON haben sie zusätzlich die Aufgabe, Teilnehmer zu informieren wenn sie in das Interessengebiet eines anderen Teilnehmers eintreten. In Abschnitt 6.6.1 wird diese Aufgabe genauer beschrieben.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Nachbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A
procedure BestimmeDirekteNachbarn($\mathcal{T}$)
$\mathcal{N} \leftarrow \emptyset$
for all $t \in \mathcal{T}$ **do**
if $d(p, p_t) \leq r^{AoI}$ **then**
$\mathcal{N} \leftarrow \mathcal{N} \cup \{t\}$
end if
end for
return $\mathcal{N}$
end procedure

Algorithmus 6.2 Bestimmung direkter Nachbarn

6.4.2 Verbindungsnachbarn

In virtuellen Welten bilden sich oft Spielergruppen. Eine Spielergruppe ist eine Menge von Teilnehmern, die in der virtuellen Welt über einen gewissen Zeitraum miteinander interagieren und daher in diesem Zeitraum räumlich nahe beieinander stehen. Diese Spielergruppen werden üblicherweise mit direkten Nachbarschaftsbeziehungen zusammenhängend verbunden. Zwischen verschiedenen Gruppen oder einzelnen Teilnehmern, die größere Abstände zu anderen Teilnehmern halten, besteht jedoch keine solche Nachbarschaftsbeziehung. Somit würde ein Overlaynetz, welches ausschließlich auf direkte Nachbarschaftsbeziehungen aufgebaut ist, in untereinander nicht verbundene Komponenten zerfallen. In QuON wird das Zerfallen des Overlays durch *Verbindungsnachbarn* verhindert.

Die wesentliche Aufgabe der Verbindungsnachbarn ist folglich, Teilnehmer oder Teilnehmergruppen zu verbinden, zwischen denen in dem durch direkte Nachbarschaftsbeziehungen gebildeten Graphen kein Weg gefunden werden kann. Zweckmäßigerweise werden diese Verbindungen von Teilnehmern am Rand der betreffenden Gruppen gebildet, da diese im Falle eines Aufeinandertreffens der Gruppen zuerst mit der jeweils anderen Gruppe in Kontakt kommen. Teilnehmer, die sich im Inneren der Gruppen befinden, können dann von außenliegenden Teilnehmern benachrichtigt werden, sollten sich die Spielergruppen so nahe kommen, dass sie in Interessengebiete von Teilnehmern der jeweils anderen Gruppe eintreten.

In QuON wurde hierzu der folgende Ansatz gewählt: Zur Bestimmung von Verbindungsnachbarn teilt jeder Spieler die virtuelle Welt an seiner Position in vier Quadranten auf. In jedem nicht-leeren Quadranten benötigt er einen Verbindungsnachbarn. Hierzu wählt er in jedem Quadranten den Spieler aus, der sich am nächsten zu seiner Position befindet. Es gilt also für die Menge der Verbindungsnachbarn $\mathcal{V}$:

$$\mathcal{V} := \bigcup_{i=0}^{3} \{t \in Q^i : \forall t' \in Q^i : d(p, p_t) \leq d(p, p_{t'})\}$$

Bei hoher Spielerdichte wird es sich, sofern sich der Teilnehmer nicht am äußersten Rand einer Spielergruppe befindet, um einen Spieler handeln, der gleichzeitig auch ein direkter Nachbar ist. Die Bestimmung der Verbindungsnachbarn wird in Algorithmus 6.3 dargestellt. Die Laufzeit des Algorithmus ist $O(|\mathcal{T}|)$.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Nachbedingung: $\mathcal{V} := \{v_0, v_1, v_2, v_3\}$: Menge der Verbindungsnachbarn von A

```
procedure BestimmeVerbindungsnachbarn(T)
    v_0, v_1, v_2, v_3 ← undefiniert
    for all t ∈ T do
        i ← BestimmeQuadrant(t)
        if (v_i undefiniert) ∨ (d(p, p_t) < d(p, p_{v_i})) then
            v_i ← t
        end if
    end for
    return V
end procedure
```

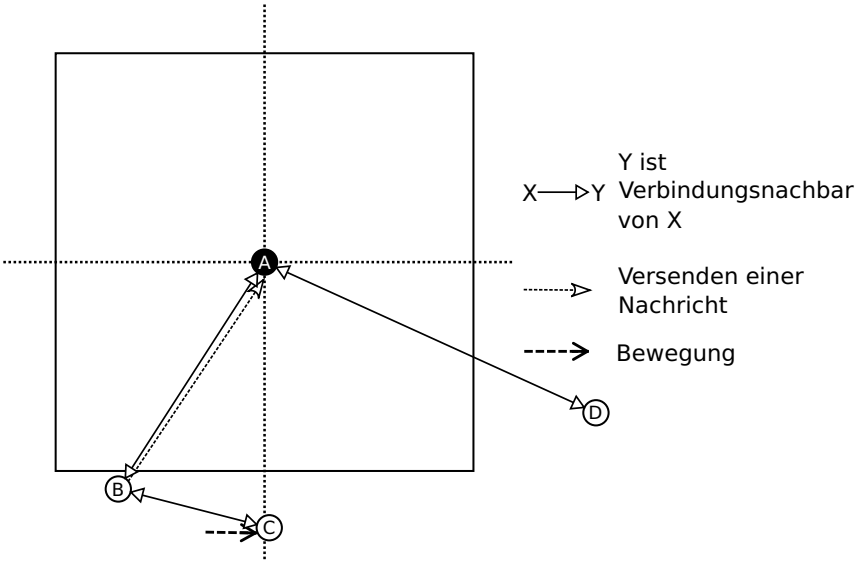
Algorithmus 6.3 Bestimmung der Verbindungsnachbarn

Da sich die Positionen der Spieler und somit die Stellung zueinander fortlaufend ändert, muss der Verbindungsnachbar nach jeder eigenen Bewegung neu bestimmt werden. Nur so kann gewährleistet bleiben, dass immer der jeweils nächste Spieler in jedem Quadranten Verbindungsnachbar ist. Spieler, die als Verbindungsnachbarn fungieren, informieren sich regelmäßig über ihre Positionen sowie über die Positionen ihrer anderen Verbindungsnachbarn. Dadurch werden Spieler über potentiell bessere Verbindungsnachbarn informiert. Abbildung 6.3a verdeutlicht dies. In der Abbildung sind die Positionen von vier Teilnehmern A bis D in der virtuellen Welt sowie die Verbindungsnachbarschaftsbeziehungen zwischen diesen Teilnehmern dargestellt. Teilnehmer B ist ein Verbindungsnachbar von A und informiert diesen deshalb über alle seine weiteren Verbindungsnachbarn. C, ein Verbindungsnachbar von B, hat sich vor kurzem über die Quadrantengrenze bewegt und befindet sich nun aus Sicht von A in Quadrant 1 (Q_A^1). Durch die Mitteilungen von B erfährt A davon. Da C in Quadrant 1 für A ein besserer Verbindungsnachbar ist als der bisherige Verbindungsnachbar D, baut A im nächsten Schritt eine Verbindung zu C auf und lässt die Nachbarschaftsbeziehung zu D fallen. Dies ist in Abbildung 6.3b gezeigt. Die sich hierbei ergebenden Nachbarschaftsbeziehungen sind nicht mehr symmetrisch. Um die Symmetrie wiederherzustellen, wird die im nächsten Abschnitt beschriebene temporäre Nachbarschaftsbeziehung benötigt.

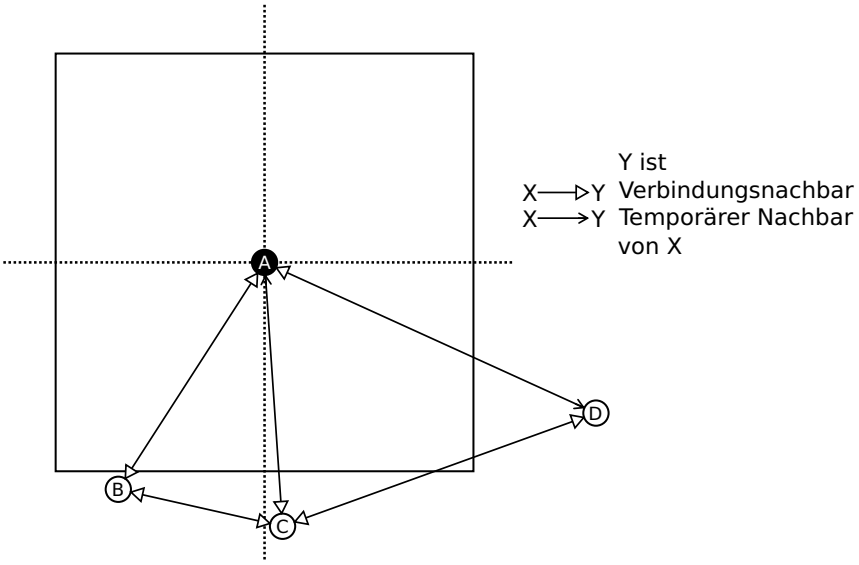
6.4.3 Temporäre Nachbarn

Wie im vorangegangenen Abschnitt deutlich wurde, sind Verbindungsnachbarbeziehungen nicht in jedem Fall symmetrisch. Es kann somit vorkommen, dass ein Spieler A Verbindungsnachbar von Spieler B ist, aber Spieler B weder direkter Nachbar noch Verbindungsnachbar von Spieler A.

Spieler B ist allerdings auf Informationen von Spieler A angewiesen, um seine Aufgabe als Verbindungsnachbar gegenüber seinen anderen Nachbarn erfüllen zu können. Um die Nachbarschaftsbeziehungen symmetrisch zu halten, muss daher eine weitere Nachbarschaftsbeziehung eingeführt werden: Die *temporäre Nachbarschaftsbeziehung.* Diese wird wie folgt definiert: Ist ein Teilnehmer A Verbindungsnachbar eines Teilnehmers B und ist B nicht Verbindungsnachbar von A, so ist B temporärer Nachbar von A. Um diese Nachbarschaftsbeziehung aufrechtzuerhalten, muss Spieler B als Initiator der Nachbarschaftsbeziehung regelmäßig eine entsprechende Anfrage an Spieler A senden. Sobald A keine Anfragen mehr von Spieler B erhält, wird



(a) C tritt in Quadrant 1 von A ein.



(b) Wechsel der Verbindungsnachbarn von A in Quadrant 1.

Abbildung 6.3 Nachbarschaftsbeziehungen zwischen vier Teilnehmern A bis D.

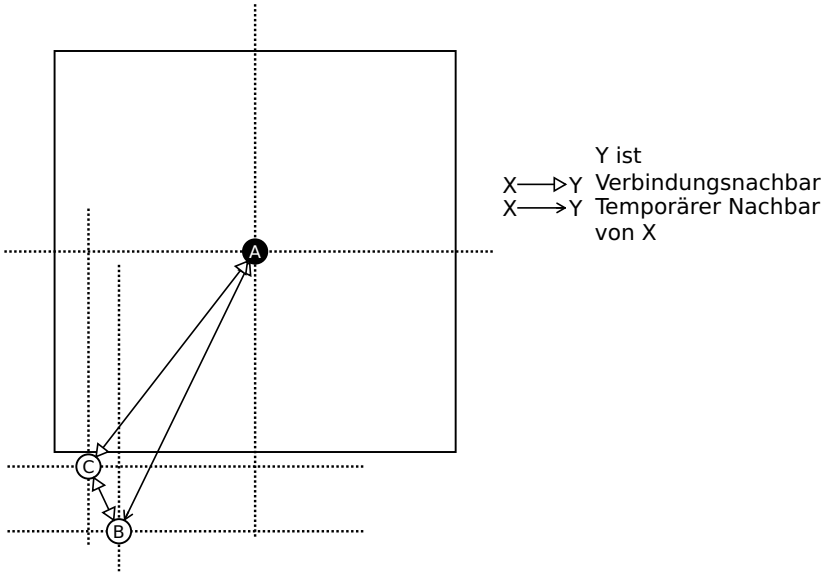


Abbildung 6.4 Nachbarschaftbeziehungen zweier Teilnehmer B und C zu einem Teilnehmer A.

die Nachbarschaftsbeziehung fallen gelassen. Algorithmus 6.4 beschreibt die Bestimmung der temporären Nachbarn. Wie die anderen Algorithmen zur Klassifizierung der Nachbarn hat auch dieser Algorithmus eine Laufzeit von $O(|\mathcal{T}|)$.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Vorbedingung: $\mathcal{V}$: Menge der Verbindungsnachbarn von A
Nachbedingung: $\mathcal{S}$: Menge der temporären Nachbarn von A

```
procedure BestimmeTemporäreNachbarn(T, V)
    S ← ∅
    for all t ∈ T do
        if {Verbindungsnachbaranfrage von t erhalten} ∧ t ∉ V then
            S ← S ∪ {t}
        end if
    end for
    return S
end procedure
```

Algorithmus 6.4 Bestimmung der temporären Nachbarn

Ein einfaches Beispiel ist in Abbildung 6.4 dargestellt: Da sich in dem oberen rechten Quadranten von Teilnehmer B (Q_B^0) nur Teilnehmer A befindet, ist dieser der Verbindungsnachbar von Teilnehmer B. Im unteren linken Quadranten von Teilnehmer A (Q_A^2) befinden sich jedoch zwei Teilnehmer, B und C. C ist näher an A als B, also ist Spieler C Verbindungsnachbar von A. Um die Beziehung zwischen A und B symmetrisch zu machen, wird B temporärer Nachbar von A.

Temporäre Nachbarn tauschen mit Verbindungsnachbarn dieselben Daten aus wie Verbindungsnachbarn untereinander. Damit sind sie funktional äquivalent zu Verbindungsnachbarn. Die Unterschiede zu Verbindungsnachbarn sind die explizite Anforderung dieser Nachbarschaftsbeziehung durch den Initiator der Verbindung sowie

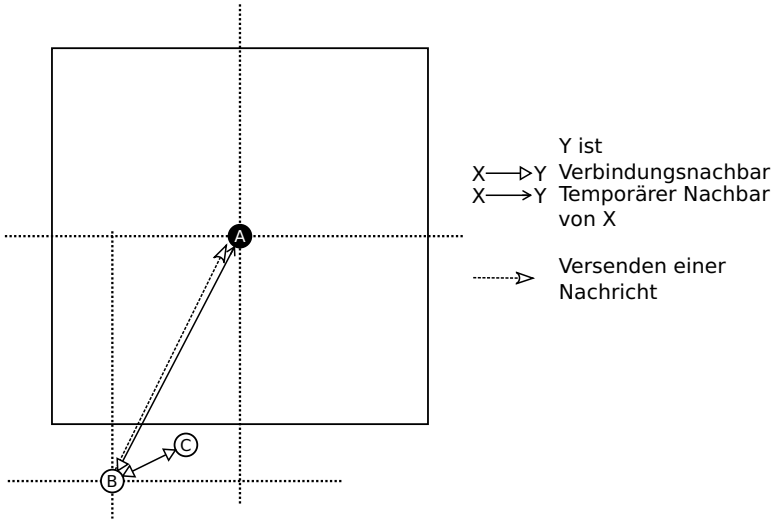


Abbildung 6.5 Verbindungs- und temporäre Nachbarschaftsbeziehungen zwischen drei Teilnehmern A bis C.

die Notwendigkeit, die Beziehung durch den Initiator in jeder Positionsmeldung erneut zu bestätigen.

Eine Situation, in der die temporäre Nachbarschaftsbeziehung besonders wichtig ist, ist in Bild 6.5 gezeigt: Kommt es aufgrund von Paketverlusten oder Knotenausfällen bei einem Teilnehmer A zu fehlender Information, kann es vorkommen, dass er einen nicht-optimalen Verbindungsnachbarn B in einem Quadranten hat. Dann ist A oft nicht Verbindungsnachbar von B. Aufgrund der temporären Nachbarschaftsbeziehung bekommt A trotzdem von B die Adressen und Positionen von dessen Verbindungsnachbarn zugesendet. Mit diesen Informationen kann A iterativ bessere Verbindungsnachbarn finden, bis er schließlich den optimalen Verbindungsnachbarn gefunden hat. Somit werden Fehler, die aufgrund von Paketverlusten oder Knotenausfällen in der Overlaystruktur auftreten können, selbsttätig korrigiert.

Wie der Beweis in Anhang B.1 zeigt, kann hierdurch gewährleistet werden, dass jeder Teilnehmer in jedem Quadranten stets den nächsten Nachbarn finden und als Verbindungsnachbarn wählen kann. Die Beweisidee ist, zu zeigen, dass, wenn durch die Bewegung eines Teilnehmers dieser zu einem Verbindungsnachbarn eines anderen Teilnehmers werden soll, dieser stets von einem seiner Verbindungsnachbarn oder temporären Nachbarn darüber informiert werden kann.

Im Zusammenspiel mit der in Abschnitt 6.6.2 beschriebenen Methode zum Finden neuer Nachbarn können Verbindungsnachbarn und temporäre Nachbarn den Zusammenhalt des Overlays sichern.

6.5 Positionsmeldungen

Die in Abschnitt 6.4 beschriebene Nachbarklassifizierung benötigt die Positionen der betreffenden Teilnehmer in der virtuellen Welt. Um diese aktuell zu halten, müssen alle Teilnehmer regelmäßig ihre aktuelle Position an alle benachbarten Teilnehmer senden. Die Frequenz des Aussendens kann von verschiedenen Faktoren wie beispielsweise der maximalen Bewegungsgeschwindigkeit in der virtuellen Welt abhängen. In

heutigen virtuellen Welten sind Frequenzen von ungefähr sechs Positionsaktualisierungen pro Sekunde üblich [30].

Das Senden der Positionsmeldungen läuft in QuON entsprechend Algorithmus 6.5 ab: Zunächst werden die bekannten Teilnehmer gemäß der in den Abschnitten 6.4.1 bis 6.4.3 beschriebenen Kategorien klassifiziert. Daraufhin wird die aktuelle Position an alle direkten Nachbarn versendet. An Verbindungsnachbarn und temporäre Nachbarn werden zusätzlich die Adressen und Positionen aller Verbindungsnachbarn und temporären Nachbarn gesendet. Im Anschluss werden alle Teilnehmer aus der Liste der bekannten Teilnehmer gelöscht, zu denen keine Nachbarschaftsbeziehung besteht. Die Laufzeit und die Zahl der im Rahmen dieses Algorithmus versendeten Nachrichten beträgt $O(|\mathcal{N}| + |\mathcal{V}| + |\mathcal{S}|)$.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
  procedure SendePositionsmeldungen($\mathcal{T}$)
    $\mathcal{N} \leftarrow$ BestimmeDirekteNachbarn($\mathcal{T}$)
    $\mathcal{V} \leftarrow$ BestimmeVerbindungsnachbarn($\mathcal{T}$)
    $\mathcal{S} \leftarrow$ BestimmeTemporäreNachbarn($\mathcal{T}, \mathcal{V}$)
    /* Position an alle Nachbarn senden. */
    for all $n \in \mathcal{N}$ **do**
      send MOVE$(A, p, r^{AoI}) \mapsto n$
    end for
    /* An Verbindungs- und temporäre Nachbarn zusätzlich $\mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}$ senden, an Verbindungsnachbarn: Verbindungsnachbarstatus melden. */
    for all $t \in \mathcal{V}$ **do**
      send MOVE$(A, p, r^{AoI}, \mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}$,Verbindungsnachbaranfrage$) \mapsto t$
    end for
    for all $t \in \mathcal{S}$ **do**
      send MOVE$(A, p, r^{AoI}, \mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}) \mapsto t$
    end for
    $\mathcal{T} \leftarrow \mathcal{N} \cup \mathcal{V} \cup \mathcal{S}$
  end procedure

Algorithmus 6.5 Versenden von Positionsmeldungen

Beim Empfang einer Positionsmeldung verfährt ein Teilnehmer wie in Algorithmus 6.6 beschrieben: Zunächst werden die Adresse des Absenders der Positionsmeldung in der Liste der bekannten Teilnehmer gespeichert und die Position des Absenders aktualisiert. Enthält die Nachricht noch die Positionen weiterer Teilnehmer[3], wird überprüft, ob sich diese bereits in der Liste der bekannten Teilnehmer befinden. Falls dies nicht der Fall ist, werden sie in die Liste aufgenommen und ihre Position wird gespeichert. Sind sie bereits in der Liste der bekannten Teilnehmer enthalten, wird die gesendete Position ignoriert. Da von Teilnehmern, die sich auf der Liste befinden, eine regelmäßige Positionsmeldung erwartet wird, sind Informationen von Dritten in der Regel veraltet. Durch das Ignorieren der Positionsmeldungen für bereits bekannte Dritte wird somit verhindert, dass die Position von Teilnehmern durch veraltete Informationen verfälscht wird. Die Laufzeit dieses Algorithmus beträgt $O(\log|\mathcal{T}| \cdot (|\mathcal{V}_m| + |\mathcal{S}_m|))$. Im Anschluss an die hier beschriebene Aktualisierung

[3]Dies ist der Fall, wenn der Empfängerknoten Verbindungsnachbar oder temporärer Nachbar des Absenders ist.

der Positionen wird die in den nächsten Abschnitten beschriebene Nachbarschaftsfindung durchgeführt.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmern
Vorbedingung: MOVE$(m, p_m, r_m^{AoI}, [\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}])$: Positionsmeldung von Teilnehmer m
procedure BehandlePositionsmeldung($\mathcal{T}$,MOVE$(m, p_m, r_m^{AoI}$, $[\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}]$))
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{m\}$
  AktualisierePosition(m)
  for all $t \in \mathcal{V}_m \cup \mathcal{S}_m$ **do**
    if $t \notin \mathcal{T}$ **then**
      AktualisierePosition(m)
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
    end if
  end for
end procedure

Algorithmus 6.6 Empfang einer Positionsmeldung

6.6 Finden neuer Nachbarn

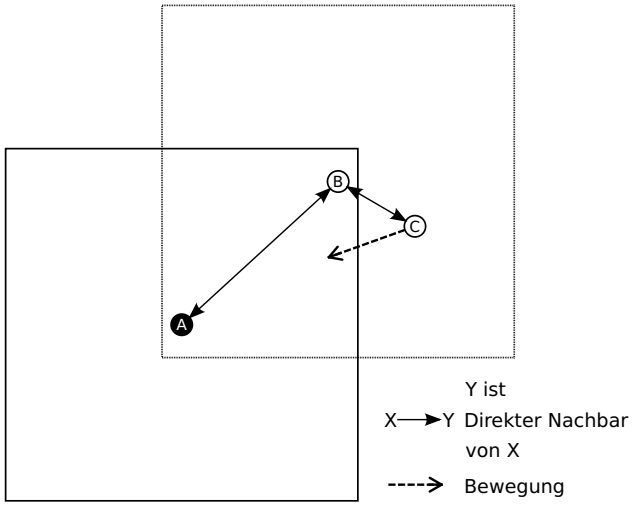
Da Teilnehmer in QuON stets die Positionen aller anderen Teilnehmer in ihrem Interessengebiet kennen müssen, ist es wichtig, dass jeder Teilnehmer stets über neu in sein Interessengebiet eintretende Teilnehmer informiert wird. In den folgenden Abschnitten wird erläutert, wie dieses Ziel mit Hilfe der in Abschnitt 6.4 vorgestellten Nachbarschaftsbeziehungen erreicht werden kann.

6.6.1 Finden über direkte Nachbarn

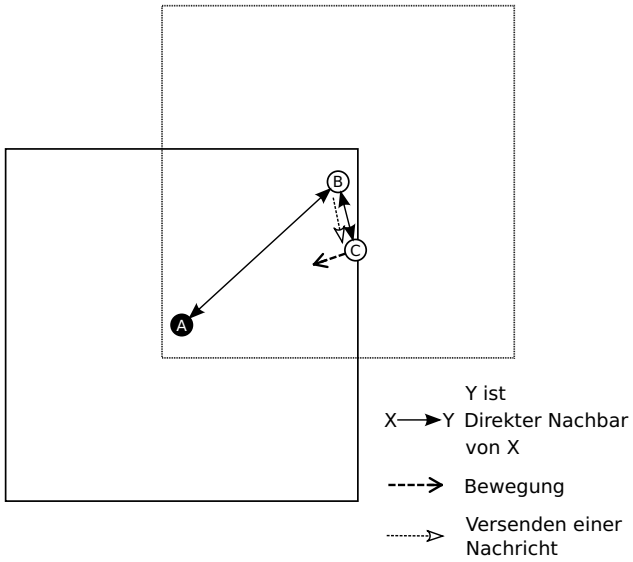
Zunächst soll betrachtet werden, wie das Finden neuer Nachbarn durch direkte Nachbarn funktioniert. Abbildung 6.6 zeigt eine mögliche Situation in einer virtuellen Welt, in der ein Spieler C in das Interessengebiet des Spielers A eintritt.

Für die Nachbarschaftsfindung überprüfen alle Teilnehmer nach dem Empfang einer Positionsmeldung, ob der jeweilige Teilnehmer durch seine Bewegung in das Interessengebiet eines anderen Teilnehmers eingetreten ist. Um dies überprüfen zu können, wird in den Positionsmeldungen neben der Position auch die Größe des Interessengebiets mitgesendet. Stellt ein Teilnehmer fest, dass einer seiner benachbarten Teilnehmer in das Interessengebiet eines anderen Teilnehmers eingetreten ist, informiert er den sich bewegenden Nachbarn über dieses Ereignis. Dieser kann nun eine Verbindung zu seinem neuen Nachbarn aufbauen.

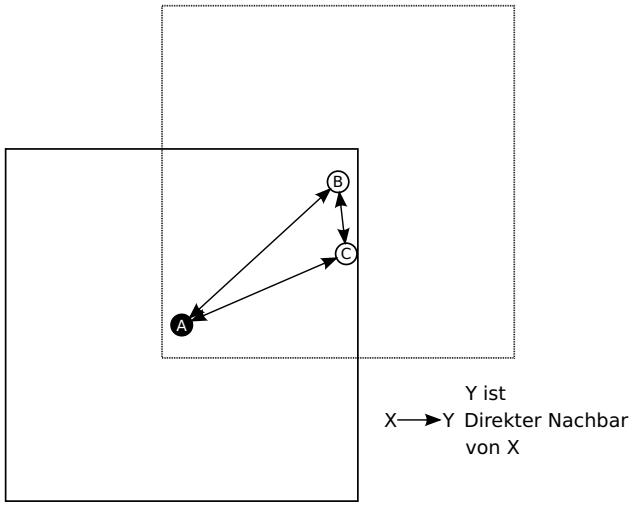
Sofern ein Teilnehmer in einem Quadranten einen direkten Nachbarn hat, deckt das Interessengebiet dieses direkten Nachbarn bei Verwendung der Maximumsnorm den entsprechenden Quadranten des Interessengebiets des Teilnehmers vollständig ab. Somit befindet sich ein neuer Nachbar, der in diesem Quadranten in das Interessengebiets des Teilnehmers eintreten will, zu diesem Zeitpunkt im Interessengebiet des direkten Nachbarn. Es besteht also eine Nachbarschaftsbeziehung zwischen dem direkten Nachbarn und dem neuen Nachbarn, und der neue Nachbar sendet dem



(a) Spieler C nähert sich dem Interessengebiet von A.



(b) Spieler C ist in das Interessengebiet von A eingetreten.



(c) Nachbarschaftsbeziehungen nach Abschluss der Bewegung von C.

Abbildung 6.6 Ein neuer Nachbar tritt in das Interessengebiet von A ein.

direkten Nachbarn regelmäßig Positionsmeldungen. Dadurch kann dieser den neuen Nachbarn über den Eintritt in das Interessengebiet des Teilnehmers informieren. Dieses Verfahren ist in Algorithmus 6.7 dargestellt. Der Algorithmus hat eine Laufzeit von $O(|\mathcal{N}|)$. Der resultierende Bandbreitenbedarf beträgt ebenfalls $O(|\mathcal{N}|)$.

Vorbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A
Vorbedingung: MOVE(m, p_m, r_m^{AoI}): Positionsmeldung von Teilnehmer m
Vorbedingung: $p_m^{'}$: Vorige Position von Teilnehmer m
procedure BENACHRICHTIGEÜBERNEUENACHBARN($\mathcal{N}$,MOVE(m, p_m, r_m^{AoI}))
for all $n \in \mathcal{N}$ **do**
if $d(p_n, p_m) \leq r_m^{AoI} \wedge d(p_n, p_m^{'}) > r_m^{AoI}$ **then**
send NEW_NEIGHBOR$(n, p_n) \mapsto m$
end if
end for
end procedure

Algorithmus 6.7 Nachbarschaftsfindung über direkte Nachbarn

Die Mechanismen zum Finden neuer Nachbarn werden im Folgenden an einem Beispiel dargestellt. Ein Teilnehmer A hat einen Nachbarn B. Ein weiterer Teilnehmer, Teilnehmer C, will in das Interessengebiet von A eintreten. Da sich Teilnehmer C im selben Quadranten wie B befindet, folgt bei Verwendung der Maximumsnorm, dass C ein direkter Nachbar von Teilnehmer B sein muss, wenn er in das Interessengebiet von Teilnehmer A eintritt. Diese Nachbarschaftsbeziehungen sind in Abbildung 6.6a dargestellt.

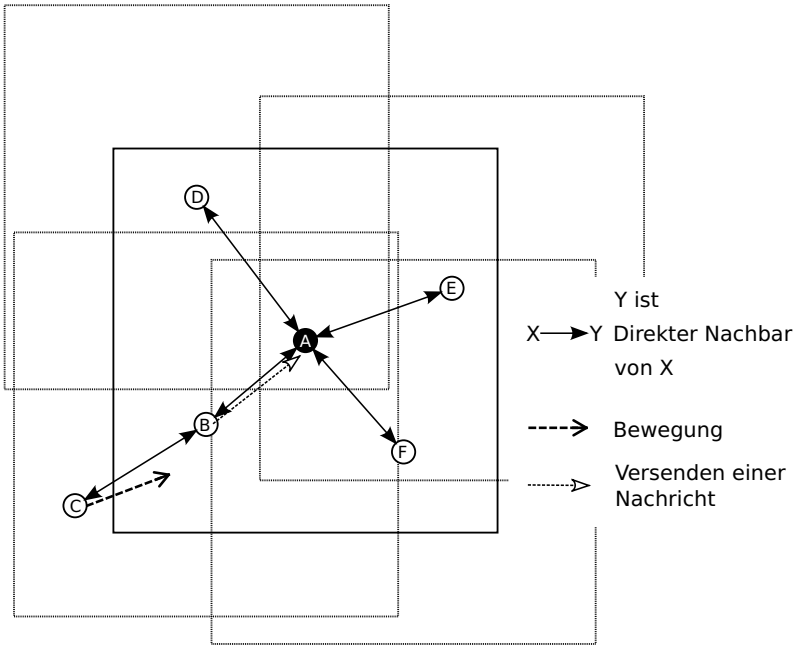
Sobald Teilnehmer C in das Interessengebiet von A eintritt und seine neue Position in einer Positionsmeldung an seine Nachbarn versendet, wird dies von Teilnehmer B festgestellt. Daraufhin informiert Teilnehmer B, wie in Abbildung 6.6b dargestellt, Teilnehmer C.

Daraufhin kann Spieler A Spieler C kontaktieren. Es entsteht eine neue direkte Nachbarschaftsbeziehung zwischen den beiden Spielern. Dies ist in Abbildung 6.6c gezeigt.

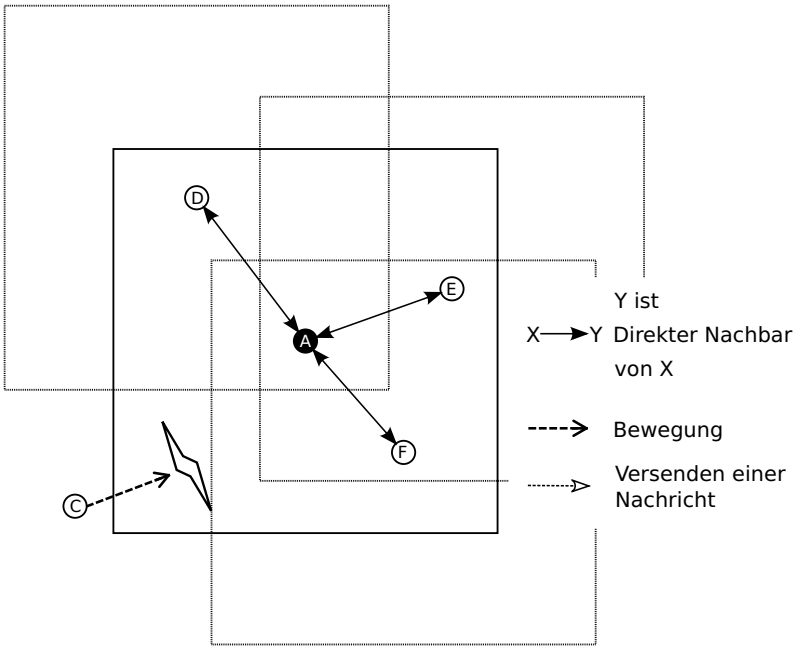
Mit Hilfe der direkten Nachbarn können Teilnehmer so über die meisten neuen Nachbarn informiert werden. Eine vollständige Information ist allerdings nur gewährleistet, wenn das Interessengebiet eines Teilnehmers vollständig von den Interessengebieten anderer Teilnehmer überdeckt wird. Bei der Verwendung der Maximumsmetrik ist die Überdeckung des Interessengebiets gewährleistet, wenn sich in jedem Quadranten mindestens ein anderer Teilnehmer innerhalb des Interessengebiets befindet. Dies ist beispielhaft in Abbildung 6.7a dargestellt: Ein Spieler C, der in das Interessengebiet von Spieler A eintreten will, muss vorher das Interessengebiet eines anderen Teilnehmers B durchqueren.

6.6.2 Finden über Verbindungsnachbarn

Besonders bei geringeren Teilnehmerdichten oder bei ungleichmäßiger Verteilung der Teilnehmer in der virtuellen Welt ist das Interessengebiet einiger Teilnehmer nicht vollständig von den Interessengebieten anderer Teilnehmer überdeckt. In diesem Fall reichen die Informationen der direkten Nachbarn nicht aus, um diesen Teilnehmer



(a) Das Interessengebiet von Teilnehmer A wird vollständig von den Interessengebieten seiner direkten Nachbarn überdeckt.



(b) Das Interessengebiet von Teilnehmer A wird nicht vollständig von den Interessengebieten seiner direkten Nachbarn überdeckt.

Abbildung 6.7 Überdeckung der Interessengebiete mehrerer Teilnehmer.

über alle neuen Nachbarn zu informieren. In Abbildung 6.7b ist eine Situation abgebildet, in der das Interessengebiet eines Teilnehmers A in Quadrant 3 nicht vollständig überdeckt ist. Hier kann ein Teilnehmer C unbemerkt von direkten Nachbarn in das Interessengebiet von A eintreten.

In diesem Fall muss die Benachrichtigung des neuen Nachbarn auf einem anderen Weg erfolgen. In QuON geschieht dies mit Hilfe der Verbindungsnachbarn: Durch den in den Abschnitten 6.4.2 und 6.4.3 beschriebenen Auswahlmechanismus wird gewährleistet, dass in jedem Quadranten stets der nächste Teilnehmer bekannt ist und als Verbindungsnachbar gewählt wird. Kommt also ein Teilnehmer näher als der aktuelle Verbindungsnachbar, wird dieser über den Verbindungsnachbaralgorithmus als neuer Verbindungsnachbar gewählt. Existiert in einem Quadranten kein direkter Nachbar, der nach dem in Abschnitt 6.6.1 beschriebenen Verfahren über neue Nachbarn informieren kann, muss der aktuelle Verbindungsnachbar außerhalb des Interessengebiets sein. Somit ist jeder Teilnehmer, der in das Interessengebiet eintreten will, näher als der aktuelle Verbindungsnachbar. Der neue Nachbar wird somit zum neuen Verbindungsnachbarn gewählt, bevor er in das Interessengebiet eintritt.

Abbildung 6.8a zeigt dazu ein Beispiel. Ein Spieler C bewegt sich in Richtung eines Spielers A. In dem Quadranten, aus dem Spieler C in das Interessengebiet von Spieler A eintreten wird, hat Teilnehmer A keinen direkten Nachbarn. B ist der Verbindungsnachbar von A in dem betreffenden Quadranten. C ist ein Nachbar von Spieler B.

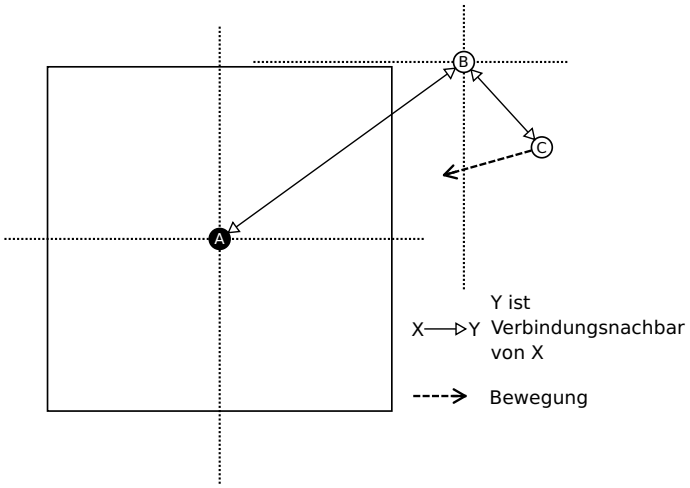
In Abbildung 6.8b hat sich Teilnehmer C nun so weit bewegt, dass näher zu A ist als der alte Verbindungsnachbar B. Aufgrund der in Abschnitt 6.4.2 beschriebene Methode zur Aktualisierung der Verbindungsnachbarn informiert B nun A und C darüber, dass C ein neuer Verbindungsnachbarkandidat für A ist.

Somit wird, wie in Abbildung 6.8c gezeigt, Teilnehmer C Verbindungsnachbar von A bevor er in dessen Interessengebiet eintritt.

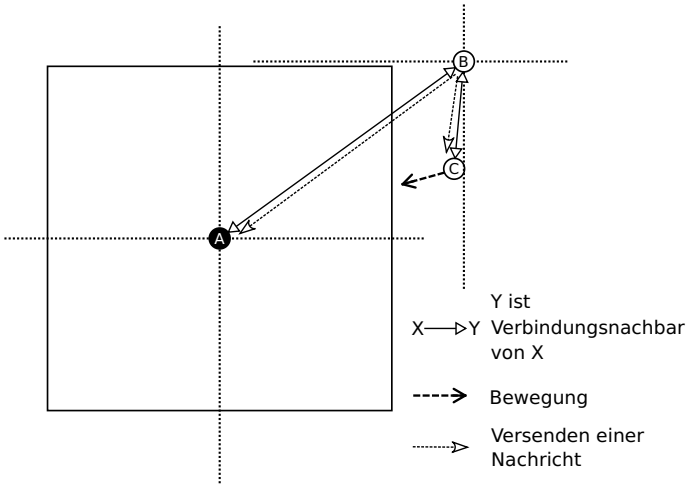
Bei Verwendung der Maximumsmetrik lässt sich beweisen, dass der Verbindungsnachbarmechanismus in jedem Fall das Finden neuer Nachbarn gewährleistet. Die Beweisidee ist es, zu zeigen, dass jeder Teilnehmer, der sich dem Interessengebiet eines anderen Teilnehmers nähert, mindestens einen Nachbarn haben muss, der auch Nachbar des anderen Teilnehmers ist. Der vollständige Beweis findet sich in Anhang B.2.

6.6.3 Alternative Nachbarsfindung mit geringerem Bandbreitenbedarf

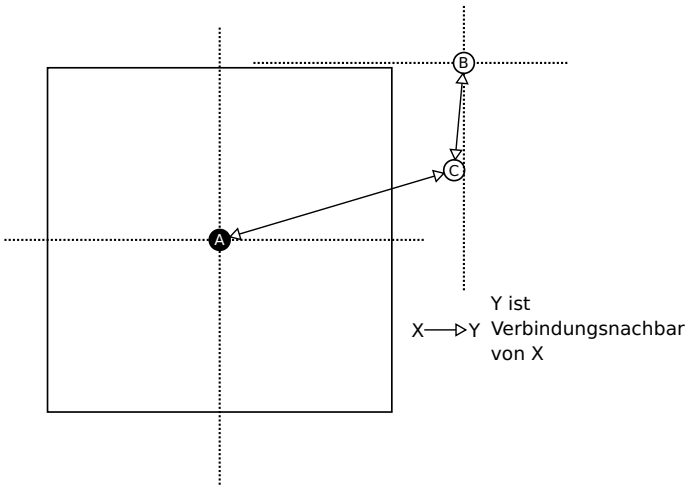
Steht den Teilnehmern einer virtuellen Welt nur wenig Bandbreite zur Verfügung, kann die Nachbarschaftsfindung modifiziert werden, um den Bandbreitenbedarf von QuON zu senken. Diese Modifizierung sieht vor, dass nur noch die Verbindungsnachbarn für das Finden neuer Nachbarn zuständig sind. Ist der Verbindungsnachbar außerhalb des Interessengebiets eines Teilnehmers, verhält er sich unverändert wie im obigen Abschnitt 6.6.2 beschrieben. Die in Abschnitt 6.6.1 beschriebene Nachbarsfindung durch direkte Nachbarn wird jedoch nur noch von Verbindungsnachbarn, die sich innerhalb des Interessengebiets eines Teilnehmers befinden, durchgeführt. Falls ein Teilnehmer in das Interessengebiet eines anderen Teilnehmer eintritt, wird



(a) Ein Teilnehmer C bewegt sich in Richtung des Interessengebiets von A.



(b) C ist nun näher an A als B. B informiert A und C.



(c) C wird neuer Verbindungsnachbar von A.

Abbildung 6.8 Ein neuer Nachbar nähert sich dem Interessengebiet von A.

also nur noch genau eine Benachrichtigung über diese Ereignis verschickt. Dieses Verfahren ist in Algorithmus 6.8 dargestellt. Dieser Algorithmus hat eine Laufzeit von $O(|\mathcal{N}|)$.

Vorbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A
Vorbedingung: $\mathcal{V}$: Menge der Verbindungsnachbarn von A
Vorbedingung: $\mathcal{S}$: Menge der temporären Nachbarn von A
Vorbedingung: MOVE(m, p_m, r_m^{AoI}): Positionsmeldung von Teilnehmer m
Vorbedingung: $p_m^{'}$: Vorige Position von Teilnehmer m
procedure BENACHRICHTIGEÜBERNEUENACHBARN2(MOVE(m, p_m, r_m^{AoI}), $\mathcal{N}, \mathcal{V}, \mathcal{S}$)
  if $m \in \mathcal{V} \cup \mathcal{S}$ **then**
    for all $n \in \mathcal{N}$ **do**
      if $d(p_n, p_m) \leq r_m^{AoI} \wedge d(p_n, p_m^{'}) > r_m^{AoI}$ **then**
        send NEW_NEIGHBOR$(n, p_n) \mapsto m$
      end if
    end for
  end if
end procedure

Algorithmus 6.8 Alternative Nachbarschaftsfindung, nur über Verbindungsnachbarn

Bei Verwendung der Maximumsmetrik überdecken Teilnehmer, die sich innerhalb des Interessengebiets eines anderen Teilnehmers befinden, dieses innerhalb ihres Quadranten vollständig. Befindet sich also der Verbindungsnachbar innerhalb des Interessengebiets, kann er alle Teilnehmer, die in dem jeweiligen Quadranten in das Interessengebiet eintreten, über dieses Ereignis informieren. Ist der Verbindungsnachbar außerhalb des Interessengebiets, greift die übliche in Abschnitt 6.6.2 beschriebene Nachbarsfindung über Verbindungsnachbarn. Somit ist dieses Verfahren prinzipiell ausreichend, um jeden Teilnehmer über alle neuen Nachbarn zu informieren. Da weniger Nachbarschaftsbenachrichtigungen verschickt werden als bei der unmodifizierten Nachbarsfindung, sinkt der Bandbreitenbedarf. Allerdings geht hierbei auch durch fehlende Redundanz ein Teil der Robustheit des Protokolls verloren: Kommt es zu Paketverlusten oder längeren Paketverzögerungen, werden Teilnehmer in einigen Fällen nicht oder zu spät über neue Nachbarn informiert. Eine genauere Evaluierung dieses Verfahrens findet sich in Abschnitt 8.1.5.

6.7 Teilnehmerbeitritt und Verlassen des Overlays

In den vorigen Abschnitten wurde der Protokollablauf für Spieler beschrieben, die bereits Teil eines QuON-Netzes sind. In diesem Abschnitt soll nun erläutert werden, wie Teilnehmer einem QuON-Netz beitreten können und wie das Verlassen des QuON-Netzes gehandhabt wird. Dabei werden auch die Einflüsse von Knotenausfällen betrachtet.

6.7.1 Knotenbeitritt

Ein Teilnehmer, welcher der virtuellen Welt beitreten will, benötigt hierzu die Adresse eines beliebigen Mitglieds des Overlaynetzes. Diese kann über verschiedene in der

Literatur beschriebene Verfahren bezogen werden [37, 42]. An diese Adresse kann nun ein *JOIN-Request* versendet werden. In dieser Nachricht wird die gewünschte Eintrittsposition in der virtuellen Welt vermerkt. Der Empfänger eines solchen JOIN-Requests leitet diesen nun an denjenigen seiner Nachbarn weiter, der sich am nächsten an der gewünschten Position befindet. Auf diese Weise wird die Nachricht *greedy* weitergeleitet, bis der Teilnehmer erreicht ist, der sich am nächsten zu der gewünschten Position befindet. Dieser Teilnehmer beantwortet nun den JOIN-Request des beitretenden Teilnehmers mit einer vollständigen Liste seiner Nachbarn. Der beitretende Teilnehmer kann nun diese Nachbarn kontaktieren und sich so in das Overlay einbinden. Die Behandlung einer Teilnehmerbeitrittsanfrage ist in Algorithmus 6.9 dargestellt.

Vorbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A
Vorbedingung: $\mathcal{V}$: Menge der Verbindungsnachbarn von A
Vorbedingung: $\mathcal{S}$: Menge der temporären Nachbarn von A
Vorbedingung: JOIN(B, p_B): JOIN-Request von Teilnehmer B
procedure BehandleBeitrittsanfrage(JOIN(B, p_B),$\mathcal{N}, \mathcal{V}, \mathcal{S}$)
$n \leftarrow A$
for all $t \in \mathcal{N} \cup \mathcal{V} \cup \mathcal{S}$ **do**
if $d(p_B, p_t) < d(p_B, p_n)$ **then**
$n \leftarrow t$
end if
end for
if $n \neq A$ **then**
send JOIN$(B, p_B) \mapsto n$
else
send JOIN_ACK$(\mathcal{N}, \mathcal{P}_\mathcal{N}, \mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}) \mapsto B$
end if
end procedure

Algorithmus 6.9 Behandlung einer Teilnehmerbeitrittsanfrage

Die Laufzeit des Algorithmus beträgt $O(|\mathcal{N}| + |\mathcal{V}| + |\mathcal{S}|)$. Im ungünstigsten Fall werden für das beschriebene Verfahren bei n Teilnehmern der virtuellen Welt $O(n)$ Nachrichten benötigt. Unter Annahme der Gleichverteilung der Teilnehmer in der virtuellen Welt sind $O(\sqrt{n})$ Nachrichten nötig.

Eine schnellere Anmeldung ist möglich, wenn der initiale Join-Request an einen Teilnehmer gesendet wird, der bereits nahe der gewünschten Position in der virtuellen Welt ist. Hierzu müssen die Bootstrap-Mechanismen erweitert werden. Eine Möglichkeit besteht darin, die ungefähre Position aller Teilnehmer zusammen mit ihrer Adresse in einem *Location Cache* zu speichern. Dies kann sowohl an zentraler Stelle, beispielsweise einem *Login Server*, als auch verteilt, beispielsweise in einer *verteilten Hash-Tabelle* [130], geschehen. Um unnötigen Datenverkehr zu vermeiden, sollte die Positionsangabe nur aktualisiert werden, wenn sich ein Teilnehmer zu weit von der zuletzt veröffentlichen Position entfernt hat. Bei der in Kapitel 7 vorgestellten Erweiterung zur Absicherung des Protokolls gegen böswillige Teilnehmer kann die dort zu Autorisierungszwecken verwendete Autorisierungsstelle einfach un einen *Location Cache* erweitert werden. Eine Evaluierung der verschiedenen Teilnehmerbeitrittsvarianten findet sich in Abschnitt 8.1.6.

6.7.2 Geordnetes Verlassen des Overlays

Verlässt ein Teilnehmer das Overlaynetz, informiert er alle benachbarten Teilnehmer. In der hierzu verwendeten *LEAVE*-Nachricht listet er alle seine momentanen Nachbarn auf. Die Empfänger einer solchen Nachricht entfernen den Absender aus ihrer Nachbarsliste und überprüfen, ob durch diese Aktion einer der in der LEAVE-Nachricht aufgeführten Teilnehmer nun in einer Nachbarschaftsbeziehung zu ihnen steht. Damit kann Inkonsistenzen in der Netzstruktur vorgebeugt werden.

Um das geordnete Abmelden vom Netz zu forcieren, schaffen viele virtuelle Welten Anreize, die das geordnete Abmelden belohnen. In „World of Warcraft“ wird beispielsweise Spielern, die sich in Gaststätten der virtuellen Welt ausloggen, ein Bonus auf die erworbenen Erfahrungspunkte gewährt.

6.7.3 Behandlung von Teilnehmerausfällen

Trotz dieser Maßnahmen kann jedoch nicht garantiert werden, dass alle Teilnehmer beim Verlassen des Netzes ordnungsgemäß eine LEAVE-Nachricht versenden: In Peer-to-Peer-Netzen muss damit gerechnet werden, dass ein Teil der Teilnehmer ausfällt, ohne diesen Ausfall kommunizieren zu können. Ursachen dafür können beispielsweise Computerabstürze oder Abbrüche der Internetverbindung sein. Empfängt ein Teilnehmer von einem anderen Teilnehmer für eine gewisse Zeitspanne, dem sogenannten *Alive-Timeout,* keine Nachrichten mehr, geht er davon aus, dass dieser Teilnehmer ausgefallen ist. In diesem Fall entfernt er den Teilnehmer aus seiner Nachbarsliste.

Weiterhin muss das Wiedereinfügen der auf diese Weise als ausgefallen erkannten Teilnehmer in die Nachbarsliste verhindert werden. Ein Wiedereinfügen kann dadurch erfolgen, dass andere Teilnehmer, die den Ausfall noch nicht bemerkt haben, veraltete Informationen weiterleiten. Beim Empfang einer solchen Information wird der ausgefallene Teilnehmer durch die in Algorithmus 6.6 beschriebene Behandlung von Positionsmeldungen wieder in die Nachbarsliste aufgenommen, auch wenn er zuvor als ausgefallen erkannt und aus der Nachbarsliste entfernt wurde. Eine einfache Lösung für dieses Problem ist die Einführung einer Liste gelöschter Teilnehmer. Informationen über Teilnehmer, die auf dieser Liste aufgeführt sind, werden ignoriert. Es muss allerdings ermöglicht werden, dass Teilnehmer, die aufgrund einer zeitweiligen Unterbrechung ihrer Netzanbindung als gelöscht markiert wurden, wieder in das Overlaynetz eingegliedert werden können. Dies lässt sich erreichen, indem man Teilnehmer, deren Position sich seit ihrer Eintragung auf die Liste geändert hat, von der Löschliste entfernt und wieder in die Nachbarsliste aufnimmt. Die um diesen Mechanismus erweiterte Behandlung empfangener Bewegungsmeldungen ist in Algorithmus 6.10 dargestellt. Die Laufzeit des Algorithmus beträgt $O((\log|\mathcal{N}| + \log|\mathcal{T}| + \log|\mathcal{L}|) \cdot (|\mathcal{V}_m| + |\mathcal{S}_m|))$.

6.7.4 Backup-Nachbarn

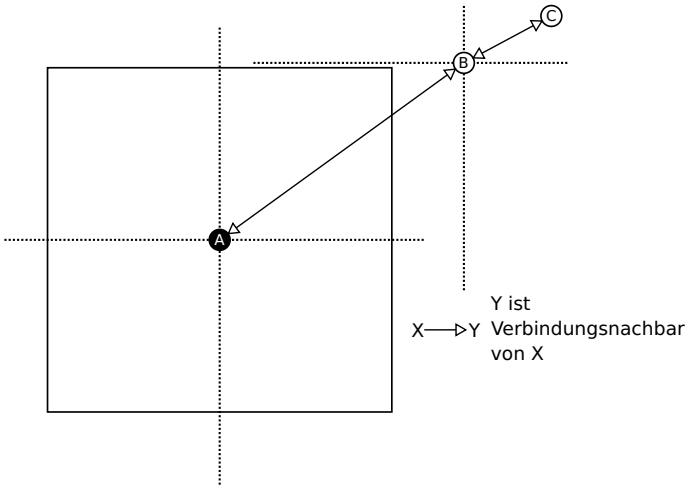
Das Erkennen ausgefallener Teilnehmer ist jedoch allein nicht ausreichend, um den Netzzusammenhalt zu gewährleisten. Insbesondere kann der Ausfall eines Verbindungsnachbarn dazu führen, dass Teilnehmer oder Teilnehmergruppen vom Netz getrennt werden und die Kontaktmöglichkeit zu anderen Teilnehmern verlieren. Dies ist in Abbildung 6.9 gezeigt: In Abbildung 6.9a sind die Nachbarschaftsbeziehungen

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Vorbedingung: $\mathcal{L}$: Menge der von A als gelöscht markierten Nachbarn
Vorbedingung: $\forall l \in \mathcal{L}$: $p_l^{'}$ Zuletzt bekannte Position von l
Vorbedingung: MOVE$(m, p_m, r_m^{AoI}, [\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}])$: Positionsmeldung von Teilnehmer m
procedure BehandlePositionsmeldung($\mathcal{T}, \mathcal{L}$,MOVE$(m, p_m, r_m^{AoI}$, $[\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}])$)
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{m\}$
  AktualisierePosition(m)
  for all $t \in \mathcal{V}_m \cup \mathcal{S}_m$ **do**
    if $t \notin \mathcal{T} \wedge (t \notin \mathcal{L} \vee p_t \neq p_t^{'})$ **then**
      AktualisierePosition(m)
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
      $\mathcal{N} \leftarrow \mathcal{N} \setminus \{t\}$
    end if
  end for
end procedure

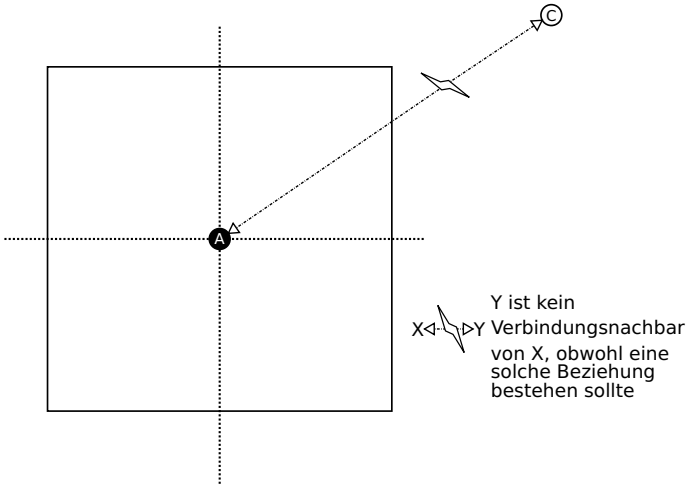
Algorithmus 6.10 Empfang einer Positionsmeldung

zwischen drei Teilnehmern dargestellt. Die Teilnehmer A und C haben jeweils eine Nachbarschaftsbeziehung mit Teilnehmer B. Fällt dieser nun, wie in Abbildung 6.9b gezeigt, aus, verlieren die beiden anderen Teilnehmer die Möglichkeit, eine Verbindung zueinander aufzubauen, da sie die Adresse des jeweils anderen Teilnehmers nicht kennen und kein weiterer Teilnehmer existiert, der die beiden Teilnehmer miteinander bekannt machen könnte. Wenn nun einer dieser Teilnehmer in das Interessengebiet des anderen eintritt, kommt es zu einem Verlust der *Nachbarschaftskenntnis*: Nach den Regeln der virtuellen Welt müssten die Teilnehmer Nachbarn werden. Da sie jedoch nicht über die Position des jeweils anderen Teilnehmers informiert werden, wird die erforderliche Nachbarschaftsbeziehung nicht aufgebaut.

Um den Ausfall von Verbindungsnachbarn zu kompensieren, kann ein einfaches Backup-Schema angewendet werden. Hierzu führt jeder Teilnehmer eine Liste mit möglichen Ersatzkandidaten für die Rolle des Verbindungsnachbarn. In jedem Quadranten werden die Adressen der nächsten Teilnehmer gespeichert. Der nächste Nachbar in jedem Quadranten wird, wie gewohnt, Verbindungsnachbar. Eine durch den Parameter n_b bestimmte Anzahl an Teilnehmern werden zu *Backup-Nachbarn*. Die Bestimmung dieser Nachbarn geschieht in Verbindung mit der Bestimmung der Verbindungsnachbarn. Algorithmus 6.11 zeigt die Bestimmung der Verbindungs- und Backup-Nachbarn. Bei diesem Algorithmus wird die Menge der bekannten Teilnehmer durchlaufen. Für jeden bekannten Teilnehmer wird festgestellt, in welchem Quadranten sich der Teilnehmer befindet. Falls in diesem Quadranten noch kein Verbindungsnachbarkandidat existiert, wird der Teilnehmer zum aktuellen Verbindungsnachbarkandidaten. Existiert bereits ein Verbindungsnachbarkandidat, wird geprüft, ob dieser weiter entfernt ist als der aktuell geprüfte Teilnehmer. Ist dies der Fall, ersetzt der Teilnehmer den Verbindungsnachbarkandidaten. Der alte Verbindungsnachbarkandidat wird der Liste der Backup-Kandidaten zugefügt. Ist der Teilnehmer zwar weiter weg als der aktuelle Verbindungsnachbarkandidat, aber näher als ein Element der Liste der Backup-Nachbarn oder sind in der Liste der Backup-Nachbarn weniger als n_b Teilnehmer verzeichnet, wird er der Liste der Backup-Nachbarn zu-



(a) Nachbarschaftsbeziehungen zwischen drei Teilnehmern.



(b) Ausfall von Teilnehmer B.

Abbildung 6.9 Verlust des Netzzusammenhaltes durch Teilnehmerausfall.

gefügt. Enthält die Liste nun mehr als n_b Backup-Nachbarn für den betreffenden Quadranten wird der am weitesten entfernte Backup-Nachbar aus der Liste entfernt.

Nachdem die komplette Menge der bekannten Teilnehmer durchlaufen ist, sind die nächsten Teilnehmer in jedem Quadranten Verbindungsnachbarkandidaten. Die Liste der Backup-Nachbarn enthält in jedem Quadranten die n_b nächsten Teilnehmer, sofern in den Quadranten genügend Teilnehmer bekannt sind. Die Laufzeit des Algorithmus ist $O(|\mathcal{T}| \cdot /n_b)$. Dieser Algorithmus wird nun anstelle der in Algorithmus 6.3 beschriebenen Auswahl der Verbindungsnachbarn eingesetzt.

Jeder Teilnehmer kontaktiert in regelmäßigen Abständen die mit Hilfe des hier beschriebenen Algorithmus bestimmten Backup-Nachbarn und erfragt ihre aktuelle Position sowie die Positionen und Adressen ihrer jeweiligen Verbindungsnachbarn. Mit den hieraus gewonnenen Informationen lässt sich im Fall eines Teilnehmerausfalls ein neuer Verbindungsnachbar für den betreffenden Quadranten finden, sofern in dem betreffenden Quadranten ein Backup-Nachbar bekannt war. Da die Zahl der Backup-Nachbarn konstant ist, ist der zusätzliche Bandbreitenaufwand für diese Backup-Methode $O(1)$.

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Nachbedingung: $\mathcal{V} := \{v_0, v_1, v_2, v_3\}$: Menge der Verbindungsnachbarn von A
Nachbedingung: $\mathcal{B}$: Menge der Backup-Nachbarn von A

procedure BestimmeVerbindungsUndBackupNachbarn($\mathcal{T}$)
  $v_0, v_1, v_2, v_3 \leftarrow$ undefiniert
  for all $t \in \mathcal{T}$ **do**
    $i \leftarrow$ BestimmeQuadrant(t)
    if v_i undefiniert **then**
      $v_i \leftarrow t$
    else if $d(p, p_t) < d(p, p_{v_i})$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{v_i\}$
      $v_i \leftarrow t$
    else if $|\mathcal{B} \cap Q_A^i| < n_b \vee \exists b \in \mathcal{B} \cap Q_A^i : d(p, p_t) < d(p, p_b)$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{t\}$
    end if
    if $|\mathcal{B} \cap Q_A^i| > n_b$ **then**
      $s := s \in \mathcal{B} \cap Q_A^i : \forall b \in \mathcal{B} \cap Q_A^i : d(p, p_s) \geq d(p, p_b)$
      $\mathcal{B} \leftarrow \mathcal{B} \setminus \{s\}$
    end if
  end for
  return $\mathcal{V}, \mathcal{B}$
end procedure

Algorithmus 6.11 Bestimmen der Backup-Nachbarn

6.7.4.1 Evaluierung

Um das Backup-Schema zu evaluieren und gute Werte für den Parameter n_b zu bestimmen, wurden in OverSim Simulationen durchgeführt. Hierbei wurde das in Abschnitt 4.5.1 erläuterte SimpleUnderlay verwendet, um das unterliegende Netzwerk zu abstrahieren. Es wurde ein einfaches Szenario betrachtet, in dem sich Teilnehmer in einer quadratischen virtuellen Welt bewegen. Die simulierte virtuelle Welt hatte eine Größe von 1.000 m mal 1.000 m. In dieser Welt bewegten sich im Mittel 500 Teilnehmer mit einer Geschwindigkeit von $5m/s$. Die Teilnehmer bildeten Gruppen mit einer Größe von bis zu 40 Teilnehmern. Teilnehmer in einer Gruppe bewegten sich immer gemeinsam zu einem zufällig gewählten Ziel; bei Erreichen des Ziels wurde ein neues Ziel gewählt. Die Sitzungszeiten der Teilnehmer wurden anhand des in Abschnitt 4.8.1 vorgestellten *ParetoChurn*-Modells bestimmt. Die mittlere Sitzungszeit betrug 100 Minuten. Nach Ablauf der Sitzungszeit wurde zufällig entschieden, ob der Teilnehmer das Netz geordnet verlässt (*graceful leave*) oder ohne Vorwarnung ausfällt (*fail-stop*). Um Latenzeffekte auszuschließen, wurde der im Kapitel 6.8 beschriebene Interessengebietspuffer mit einem Radius von 5 m eingesetzt. Die Wahrscheinlichkeit eines geordneten Abmeldens wurde in verschiedenen Simulationsläufen auf 0%, 50%, 75%, 90% oder 100% gesetzt. Die Anzahl n_b an Backup-Nachbarn pro Quadrant wurde von eins bis drei variiert. Diese Backup-Nachbarn wurden alle fünf Sekunden kontaktiert. Zum Vergleich wurden Simulationen ohne den Backup-Mechanismus durchgeführt. Die simulierte Zeit betrug zwei Stunden, dabei wurde jeder Simulationslauf mit 30 verschiedenen Seeds wiederholt. Tabelle 6.1 fasst die Simulationsparameter zusammen.

Parameter	Wert
Simulationszeit	2 Stunden nach Initialisierungsphase
Anzahl Seeds	30
Teilnehmerzahl	500
Spielfeldgröße	1 km*1 km
Bewegungsmodell	GroupRoaming
max. Gruppengröße	5
Sitzungsmodell	ParetoChurn
Sitzungszeit	100 Minuten
Graceful-leave	0%, 50%, 75%, 90%, 100%
Interessengebietsradius	50 m
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Backupnachbarn	0,1,2,3/Quadrant
Interessengebietspuffer	5 m
Vektornorm	euklidische Norm
Netzmodell	SimpleUnderlay

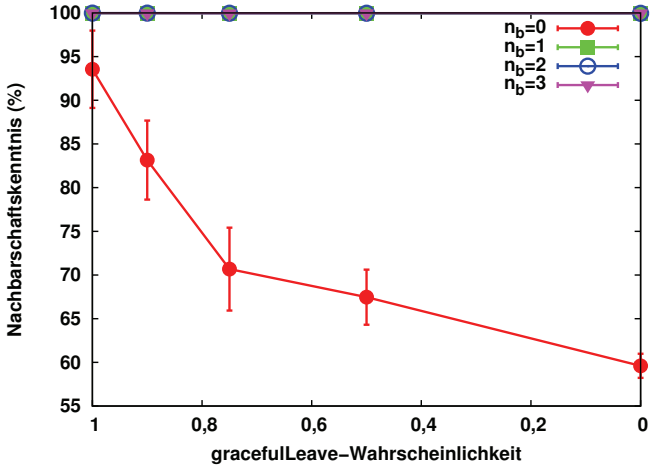
Tabelle 6.1 Simulationsparameter.

In den Simulationen wurde die Nachbarschaftskenntnis und die durchschnittlich benötigte Bandbreite gemessen. Die Nachbarschaftskenntnis drückt aus, welcher Anteil an Teilnehmern alle anderen Teilnehmer kennt, die sich in seinem Interessengebiet befinden. Die Simulationsergebnisse sind in Abbildung 6.10 dargestellt.

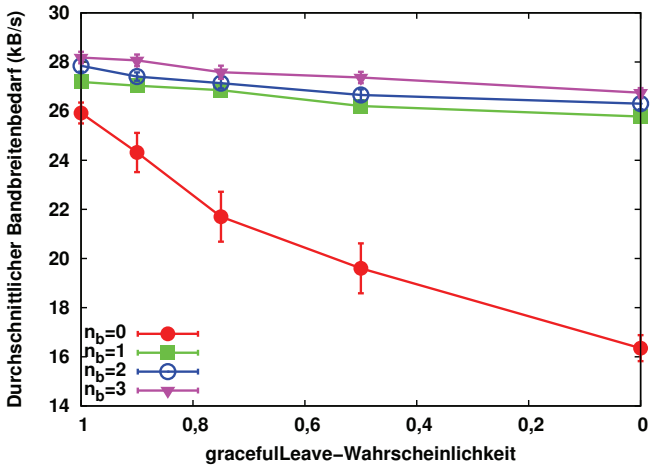
Abbildung 6.10a zeigt die Nachbarschaftskenntnis in Abhängigkeit von der *graceful leave*-Wahrscheinlichkeit. Deutlich ist zu sehen, dass die Nachbarschaftskenntnis bei QuON ohne Backup-Mechanismus bei einem höheren Anteil an *fail-stop* ausfallenden Teilnehmer schnell fällt. Wenn sich 100% der Teilnehmer ordnungsgemäß abmelden, liegt der Wert bei knapp 94%. Wird jedoch die Sitzung aller Teilnehmer durch einen Ausfall beendet, sodass kein Teilnehmer sich geordnet abmelden kann, beträgt er nur noch ungefähr 60%. Hier zeigt sich, dass der Ausfall von Verbindungsnachbarn wie erwartet zu Einbußen in der Nachbarschaftskenntnis führt. Sind jedoch die oben beschriebenen Backup-Mechanismen aktiviert, bleibt die Nachbarschaftskenntnis bei nahezu 100%. Die Backup-Mechanismen können also den Ausfall von Verbindungsnachbarn wirksam kompensieren.

Dabei erweist sich bereits ein Backup-Nachbar pro Quadrant als sehr effektiv. Diese Konfiguration erreicht eine Nachbarschaftskenntnis von 99.97% bei 100% *graceful leave* und 99.93% bei 0% *graceful leave*. Bei zwei Backup-Nachbarn pro Quadrant werden mit 99.98% bei 100% *graceful leave* und 99.96% bei 0% *graceful leave* leicht bessere Werte erreicht. Bei drei Backup-Nachbarn pro Quadranten bleibt die Nachbarschaftskenntnis zwischen 99.97% und 99.98%. Somit zeigt sich, dass eine höhere Anzahl an Backup-Nachbarn die Nachbarschaftskenntnis verbessert und den Einfluss der *graceful leave*-Wahrscheinlichkeit auf die Nachbarschaftskenntnis verringert.

Der durchschnittliche Bandbreitenbedarf in Abhängigkeit der *graceful leave*-Wahrscheinlichkeit ist in Abbildung 6.10b dargestellt. Dabei fällt der Bandbreitenbedarf stets mit sinkender *graceful leave*-Wahrscheinlichkeit; bei einem Backup-Nachbarn pro Quadranten beispielsweise von 27 kByte/s auf 26 kByte/s. Dies ist damit zu begründen, dass bei geringerer *graceful leave*-Wahrscheinlichkeit eine geringere Anzahl



(a) Nachbarschaftskenntnis in Abhängigkeit der *graceful leave*-Wahrscheinlichkeit.



(b) Durchschnittlicher Bandbreitenbedarf in Abhängigkeit der *graceful leave*-Wahrscheinlichkeit.

Abbildung 6.10 Evaluierungsergebisse des Backup-Mechanismus.

der relativ großen LEAVE-Nachrichten versendetet werden. Ohne Verwendung der Backup-Nachbarn sinkt der Bandbreitenbedarf noch deutlich stärker. Ursache dafür ist die fallende Nachbarschaftskenntnis: Wenn ein Teilnehmer Nachbarn nicht kennt, werden auch keine Positionsmeldungen ausgetauscht. Somit sinkt der Bandbreitenbedarf bei fallender Nachbarschaftskenntnis deutlich. Eine sinnvolle Abschätzung des Bandbreitenmehrbedarfs bei Verwendung von Backup-Nachbarn ist also nur bei einer *graceful leave*-Wahrscheinlichkeit von 100% gegeben, da nur dann QuON ohne Backup-Mechanismen eine akzeptable Nachbarschaftskenntnis erreicht. Hier benötigt QuON ohne Backup-Nachbarn eine durchschnittliche Bandbreite von 26 kByte/s. Mit einem Backup-Nachbarn werden knapp 27 kByte/s benötigt, mit zwei Backup-Nachbarn 27,7 kByte/s und mit drei Backup-Nachbarn 28,2 kByte/s. Dabei ist der Abstand zwischen den Varianten mit Backup-Nachbarn über verschiedene *graceful leave*-Wahrscheinlichkeiten relativ konstant.

Die Evaluierung zeigt, dass die Einführung von Backup-Nachbarn wichtig für QuON ist. Ohne Backup-Nachbarn fällt die Nachbarschaftskenntnis bei Teilnehmerausfällen schnell auf inakzeptable Werte: Bereits bei einer *graceful leave*-Wahrscheinlichkeit von 75% fehlen 30% der Teilnehmer Informationen über mindestens einen ihrer Nach-

barn. Mit Backup-Nachbarn bleibt die Nachbarschaftskenntnis in den betrachteten Szenarien immer bei über 99,93%. Eine Erhöhung der Zahl der Backup-Nachbarn kann diesen Wert weiter verbessern. Dabei steigt die benötigte durchschnittliche Bandbreite leicht an. Mit einem Backup-Nachbarn pro Quadrant benötigt QuON knapp 4% mehr Bandbreite als ohne Backup-Mechanismen. Ein Teil dieses Bandbreitenmehrbedarfs wird durch die bessere Nachbarschaftskenntnis verursacht, sodass die Erhöhung der Zahl der Backup-Nachbarn auf zwei oder drei pro Quadrant den Bandbreitenbarf jeweils nur um weitere 2% steigert. Backup-Nachbarn sind also eine effiziente Lösung, um Probleme durch ausfallende Teilnehmer in QuON zu vermeiden. Durch den Einsatz von mehreren Backup-Nachbarn pro Quadranten kann die Nachbarschaftskenntnis noch einmal verbessert werden.

6.8 Verbesserung der Nachbarschaftskenntnis bei Nachrichtenverzögerung

Das Interessengebiet eines Teilnehmers wurde im Rahmen dieser Arbeit so definiert, dass sich alle Teilnehmer, mit denen nach den Regeln der virtuellen Welt direkt interagiert werden kann, im Interessengebiet befinden. In QuON wird mit Teilnehmern innerhalb des Interessengebiets im Rahmen der direkten Nachbarschaftsbeziehung eine Verbindung aufgebaut. Zur Bestimmung der Nachbarschaftsbeziehung wird die Position des anderen Teilnehmers mit dem eigenen Interessengebiet verglichen.

Allerdings werden Positionsmeldungen nur in bestimmten Intervallen ausgesendet. Zusätzlich werden sie durch das unterliegende Netzwerk verzögert. Aus diesem Grund ist das Wissen über die Position eines anderen Teilnehmers stets mit einer gewissen Unschärfe versehen. Die Unschärfe lässt sich wie folgt abschätzen: Sei $\bar{l}$ die durchschnittliche Latenz zwischen dem Absender A einer Positionsmeldung und dessen Empfänger B, f die Versendefrequenz der Positionsmeldungen und $\bar{v}$ die durchschnittliche Bewegungsgeschwindigkeit in der virtuellen Welt. Dann ist die Abweichung der Position, an der B den Aufenthaltsort von A vermutet und der tatsächlichen Position von A die durchschnittliche Zeit zum Versenden der nächsten Positionsmeldung $1/2f$ zuzüglich der durchschnittlichen Latenz $\bar{l}$ multipliziert mit der durchschnittlichen Geschwindigkeit $\bar{v}$.

$$\bar{d} := (\frac{1}{2f} + \bar{l}) \cdot \bar{v}$$

Analog ist die maximale Abweichung bei einer maximalen Latenz l_{MAX} und einer maximalen Bewegungsgeschwindigkeit v_{MAX} die maximale Zeit zum Versenden der nächsten Positionsmeldung $1/f$ zuzüglich der maximalen Latenz l_{MAX} multipliziert mit der maximalen Geschwindigkeit v_{MAX}.

$$d_{MAX} := (\frac{1}{f} + l_{MAX}) \cdot v_{MAX}$$

In Abbildung 6.11 wird verdeutlicht, wie hierdurch die Nachbarschaftskenntnis beeinträchtigt werden kann: Teilnehmer A vermutet einen Teilnehmer B außerhalb seines Interessengebiets, tatsächlich befindet sich dieser jedoch innerhalb des Interessengebiets von A. Dieses Problem kann vermindert werden, wenn die oben beschriebene Unschärfe bei der Nachbarklassifikation und der Benachrichtigung neuer Nachbarn

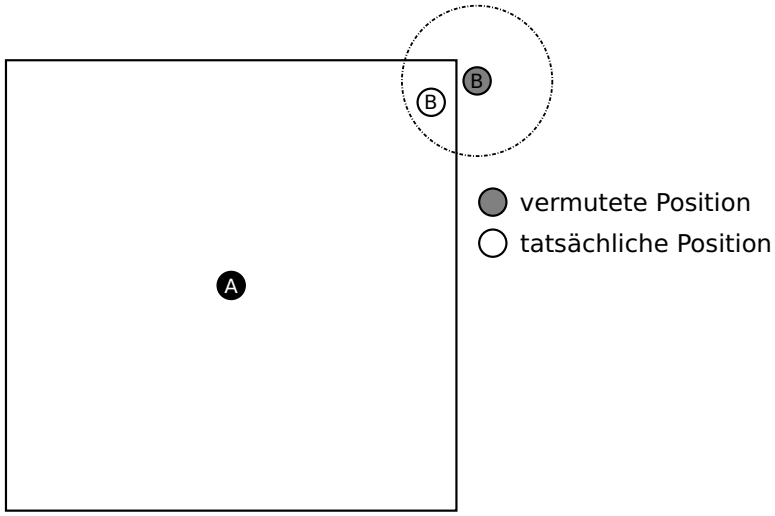


Abbildung 6.11 Zwei Teilnehmer A und B sowie die von A vermutete Position von B.

berücksichtigt wird. Hierzu kann eine Pufferzone um das Interessengebiet bestimmt werden. Teilnehmer, deren vermutete Position innerhalb der Pufferzone liegt, werden dann als direkte Nachbarn klassifiziert. Algorithmus 6.12 zeigt die derart modifizierte Klassifizierung direkter Nachbarn. Analog wird bei der Benachrichtigung neuer Nachbarn dann über Nachbarn informiert, wenn diese in die Pufferzone eintreten. Dieses ist in Algorithmus 6.13 gezeigt. Die Größe der Pufferzone kann im einfachsten Fall auf Basis einer Vorabschätzung über die auftretenden Latenzen und konfigurierten Wissens über maximale Geschwindigkeit und Positionsmeldefrequenz bestimmt werden. Ein größerer Puffer führt hierbei zu einer höheren Nachbarschaftskenntnis, erhöht jedoch auch die benötigte Bandbreite aufgrund der zusätzlich aufgebauten Nachbarschaftsbeziehungen.

Vorbedingung: $\mathcal{T}$: Menge der Teilnehmer A bekannten Teilnehmer
Vorbedingung: r^{Buf}: Größe des Interessengebietspuffers
Nachbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A

```
procedure BestimmeDirekteNachbarn(T)
    N ← ∅
    for all t ∈ T do
        if d(p, p_t) ≤ r^AoI + r^Buf then
            N ← N ∪ {t}
        end if
    end for
    return N
end procedure
```

Algorithmus 6.12 Bestimmung direkter Nachbarn mit Interessengebietspuffer

6.8.1 Evaluierung

Um die Wirksamkeit des Interessengebietspuffers zu evaluieren, wurden Simulationen in OverSim durchgeführt. Die Simulationsparameter entsprechen denen der in Abschnitt 6.7.4.1 beschriebenen Evaluierung der Backup-Nachbarn:

Vorbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A
Vorbedingung: r^{Buf}: Größe des Interessengebietspuffers
Vorbedingung: MOVE(m, p_m, r_m^{AoI}): Positionsmeldung von Teilnehmer m
Vorbedingung: $p_m^{'}$: Vorige Position von Teilnehmer m
procedure BENACHRICHTIGEÜBERNEUENACHBARN($\mathcal{N}$,MOVE(m, p_m, r_m^{AoI}))
for all $n \in \mathcal{N}$ **do**
if $d(p_n, p_m) \leq r_m^{AoI} + r^{Buf} \wedge d(p_n, p_m^{'}) > r_m^{AoI} + r^{Buf}$ **then**
send NEW_NEIGHBOR$(n, p_n) \mapsto m$
end if
end for
end procedure

Algorithmus 6.13 Nachbarschaftsfindung über direkte Nachbarn mit Interessengebietspuffer

Die simulierte virtuelle Welt hatte eine Größe von 1.000 m mal 1.000 m. In dieser Welt bewegten sich im Mittel 500 Teilnehmer mit einer Geschwindigkeit von $5\,m/s$. Die Teilnehmer bildeten Gruppen mit einer Größe von bis zu 40 Teilnehmern. Die Sitzungszeiten der Teilnehmer wurden anhand des in Abschnitt 4.8.1 vorgestellten *ParetoChurn*-Modells bestimmt. Die mittlere Sitzungszeit betrug 100 Minuten. Positionsmeldungen wurden sechs mal pro Sekunde versendet. Anders als bei der Untersuchung der Backup-Nachbarn wurde hier die graceful-leave-Wahrscheinlichkeit auf 100% festgesetzt, sodass sich alle Teilnehmer nach dem Sitzungsende geordnet abmeldeten. Da sich gezeigt hat, dass Backup-Nachbarn auch in diesem Fall für die Wahrung der Nachbarschaftskenntnis von Vorteil sind, wurde n_b auf eins gesetzt, sodass jeder Teilnehmer pro Quadrant einen Backup-Nachbarn auswählte. Der Radius des Interessengebietspuffers wurde in 1m-Schritten zwischen 0 m und 5 m variiert. In den Simulationen wurde die Nachbarschaftskenntnis und die durchschnittlich benötigte Bandbreite gemessen. Da sich bei den gewählten Parametern ein $\bar{d}$ von unter 0,5 m ergibt, ist bereits bei einem Interessengebietspuffer von 1m eine deutliche Verbesserung der Nachbarschaftskenntnis zu erwarten. Die weiteren Rahmenbedingungen, wie die Modellierung des unterliegenden Netzwerks, die Simulationszeit von zwei Stunden sowie die Wiederholung der Simulationsläufe mit 30 verschiedenen Seeds wurde von der in Abschnitt 6.7.4.1 beschriebenen Evaluierung der Backup-Nachbarn übernommen. Die Simulationsparameter sind in Tabelle 6.2 gezeigt. Die Simulationsergebnisse sind in Abbildung 6.12 gezeigt.

Abbildung 6.12a zeigt die Nachbarschaftskenntnis in Abhängigkeit von der Größe des Interessengebietspuffers. Ohne Interessengebietspuffer liegt die durchschnittliche Nachbarschaftskenntnis bei 96,5%. Erwartungsgemäß steigt sie bei einer Vergrößerung des Puffers deutlich. Bei einer Interessengebietspuffergröße von 1 m beträgt sie bereits 99,3%, bei 2 m 99,8%. Bei Puffergrößen zwischen 3 m und 5 m ist die Nachbarschaftskenntnis größer als 99,93%. Ein Interessengebietspuffer von 1 m kann wie erwartet bereits einen Großteil der durch Latenz verursachten Einbußen der Nachbarschaftskenntnis beheben. Eine weitere Vergrößerung bis auf 3 m kann die Nachbarschaftskenntnis noch weiter steigern, da hier zusätzlich Teilnehmer mit überdurchschnittlichen Latenzen von der Pufferzone erfasst werden. Größere Puffer als 3 m bringen nur noch eine sehr kleine weitere Verbesserung.

Parameter	Wert
Simulationszeit	2 Stunden nach Initialisierungsphase
Anzahl Seeds	30
Teilnehmerzahl	500
Spielfeldgröße	1 km*1 km
Bewegungsmodell	GroupRoaming
max. Gruppengröße	40
Sitzungsmodell	ParetoChurn
Sitzungszeit	100 Minuten
Graceful-leave	100%
Interessengebietsradius	50 m
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Backupnachbarn	1/Quadrant
Interessengebietspuffer	0 m, 1 m, 2 m, 3 m, 4 m, 5 m
Vektornorm	euklidische Norm
Netzmodell	SimpleUnderlay

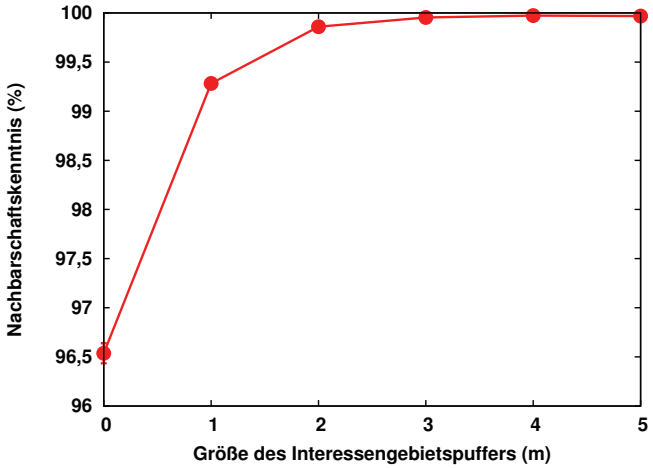
Tabelle 6.2 Simulationsparameter.

Der durchschnittliche Bandbreitenbedarf in Abhängigkeit der Größe des Interessengebietspuffers ist in Abbildung 6.12b dargestellt. Erwartungsgemäß steigt die durchschnittlich benötigte Bandbreite mit steigender Puffergröße. So liegt die durchschnittliche Bandbreite ohne Puffer bei 26,2 kByte/s, bei einer Puffergröße von 1 m bei 26,5 kByte/s und bei einer Puffergröße von 5 m bei 27,1 kByte/s. Dabei steigt der Bandbreitenbedarf sublinear mit der Interessenspuffergröße. Bei der Puffergröße 4 m war der Bandbreitenbedarf etwas geringer als es nach den anderen Ergbnissen zu erwarten wäre. Eine mögliche Erklärung ist, dass bei der gewählten Parametrisierung aufgrund der Verteilung und Größe der Spielergruppen die Zahl der Nachbarn im Vergleich zur Puffergröße 3 m weniger als erwartet anstieg. Da die Abweichung mit weniger als 1% relativ gering war, wurden keine weiteren Untersuchungen dieser Abweichung durchgeführt.

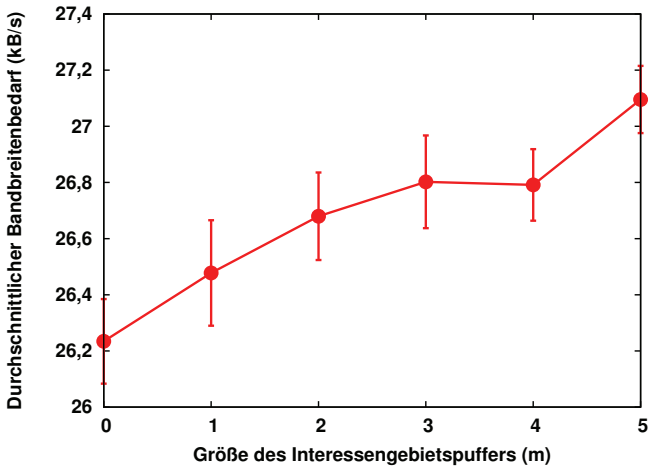
Die Evaluierung zeigt, dass der Interessengebietspuffer eine wirksame und effiziente Methode ist, latenzbedingte Einbußen der Nachbarschaftskenntnis zu vermeiden. Im betrachteten Szenario lässt sich mit einem Interessengebietspuffer die Nachbarschaftskenntnis von 96,5% auf über 99,92% steigern. Durch die Verwendung des Puffers erhöht sich die durchschnittlich benötigte Bandbreite um weniger als 2,5% von 26,2 kByte/s auf 26,8 kByte/s.

6.9 Dynamische Interessengebietsgröße

Ein mögliches Problem bei Overlayprotokollen für MMOGs und virtuelle Welten sind sogenannte *Hotspots*. Dies sind Bereiche, in denen sich eine Vielzahl an Teilnehmern auf sehr engem Raum aufhält. Virtuelle Welten fördern dieses Verhalten in der Regel, da oft die Interaktion innerhalb und zwischen Gruppen einen wesentlichen Teil der Welt ausmacht. Da normalerweise Spieler Ereignisnachrichten an alle anderen Spieler in ihrem Interessengebiet senden, kann bei Hotspots eine Überlastung der Teilnehmer aufgrund des daraus resultierenden Bandbreitenbedarfs nicht ausgeschlossen werden.



(a) Nachbarschaftskenntnis in Abhängigkeit der Größe des Interessengebietspuffers.



(b) Durchschnittlicher Bandbreitenbedarf in Abhängigkeit des Interessengebietspuffers.

Abbildung 6.12 Evaluierungsergebnisse des Interessengebietspuffers.

Eine mögliche Lösung dieses Problems ist das dynamische Anpassen der Größe des Interessengebiets an die Lastsituation. Droht eine Überlast, können die Teilnehmer ihr Interessengebiet dynamisch verkleinern. Das Anpassen des Interessengebiets wurde bereits in der Literatur vorgeschlagen, beispielsweise für Vast [66]. Allerdings fehlte bislang eine Untersuchung geeigneter Algorithmen: Eine solche dynamische Anpassung der Interessengebiete kann die Nachbarschaftskenntnis negativ beeinflussen: Wenn das Interessengebiet vergrößert oder verkleinert wird, ändern sich die Nachbarschaftsbeziehungen. Daraus resultiert, dass Verbindungen zwischen Teilnehmern auf- oder abgebaut werden müssen. In der dazu benötigten Zeitspanne kann es passieren, dass ein Teilnehmer nicht alle seine aktuellen Nachbarn kennt. Deswegen ist es wichtig, einen Algorithmus für die Adaption der Interessengebiete zu finden, der einen möglichst geringen negativen Einfluss auf die Nachbarschaftskenntnis hat.

Mögliche Adaptionsstrategien lassen sich in zwei Kategorien einteilen. *Lokale Strategien* berücksichtigen nur das Wissen eines einzelnen Teilnehmers, wie beispielsweise die aktuell benötigte Bandbreite oder die Anzahl an Nachbarn innerhalb des Interessengebiets. *Verteilte Strategien* berücksichtigen zusätzlich Daten von benachbarten Teilnehmern [84].

6.9.1 Erforderliche Anpassungen des Grundprotokolls

Zunächst muss eine geeignete Metrik zur Bestimmung der aktuellen Lastsituation ausgewählt werden. Eine offensichtliche Wahl ist hier die aktuell benötigte Bandbreite in Verbindung mit einem Bandbreitenschwellenwert. Sobald dieser Schwellenwert überschritten wird, sollte die Adaption das Interessengebiet verkleinern. Eine weitere mögliche Metrik ist die Anzahl der Nachbarn. Die Anzahl der Nachbarn ist in QuON stark korreliert mit der benötigten Bandbreite. Da die Zahl der Nachbarn einfacher zu bestimmen ist als die momentan benötigte Bandbreite, wird in der Implementierung und Evaluierung diese Zahl als Metrik verwendet. Die vorgeschlagenen Mechanismen lassen sich jedoch ebenso mit der benötigten Bandbreite als Metrik nutzen.

Um die Adaption der Interessengebietsgröße in QuON zu integrieren, betrachtet jeder Teilnehmer nach jedem Aussenden einer Positionsmeldung seine Lastsituation gemäß der gewählten Metrik. Übersteigt diese den Schwellenwert, wird das Interessengebiet verkleinert. Dabei kann optional eine minimale Interessengebietsgröße gewählt werden, die nicht unterschritten werden soll. Liegt die Zahl der Nachbarn oder die benötigte Bandbreite unter dem Schwellenwert, kann die Interessengebietsgröße wieder vergrößert werden. Oberhalb einer maximalen Interessengebietsgröße wird die Vergößerung ausgesetzt. Da in einer virtuellen Welt die Teilnehmer möglichst selten in eine Überlastsituation geraten sollten, ist dieser maximale Wert für die Interessengebietsgröße zugleich die reguläre Interessengebietsgröße.

Insbesondere bei der Wahl der Zahl der Nachbarn als Metrik könnte das Interessengebiet stets derart verkleinert werden, dass die resultierende Anzahl an Nachbarn genau dem Schwellenwert entspricht. Ebenso könnte beim Vergrößern anhand der Teilnehmerdichte geschätzt werden, um wie viel das Interessengebiet vergrößert werden müsste, um dem Schwellenwert möglichst nahe zu kommen. Diese Adaptionsstrategie würde jedoch die Nachbarschaftskenntnis erheblich beeinträchtigen, da diese Strategie zu stark und sprunghaft schwankenden Interessengebietsgrößen führt. Insbesondere beim Vergrößern des Interessengebiets benötigt das Finden der neuen Nachbarn und das Aufbauen einer Verbindung zu diesen eine gewisse Zeit. Erfolgt das Verändern der Interessengebietsgrößen zu sprunghaft, kommt es zu einer hohen Zahl an Teilnehmern, die noch nicht alle ihre neuen Nachbarn kennen. Aus diesem Grund ist diese einfache Strategie nicht sinnvoll. Bessere Adaptionsstrategien werden in Abschnitt 6.9.2 erläutert.

Bei adaptiver Interessengebietsgröße muss die Bestimmung der direkten Nachbarn geändert werden. Bei konstanter Interessengebietsgröße ist diese Nachbarschaftsbeziehung automatisch symmetrisch[4]: Befindet sich ein Teilnehmer A im Interessengebiet eines Teilnehmers B, so liegt auch B im Interessengebiet von A. Da bei adaptiver Interessengebietsgröße nicht mehr gegeben ist, dass die Interessengebiete von A und B dieselbe Größe haben, gilt dies nicht mehr. Somit kann sich ein Teilnehmer B im Interessengebiet eines Teilnehmers A befinden, ohne dass sich A im Interessengebiet von B befindet. Dies ist in Abbildung 6.13 dargestellt. Nach der in Abschnitt 6.4.1 definierten direkten Nachbarschaftsbeziehung wäre nun B direkter Nachbar von A, aber A kein direkter Nachbar von B. Die Symmetrie lässt sich wiederherstellen, wenn

[4]Latenzbedingte Verletzungen dieser Symmetrie werden durch den in Abschnitt 6.8 beschriebenen Interessengebietspuffer vermieden.

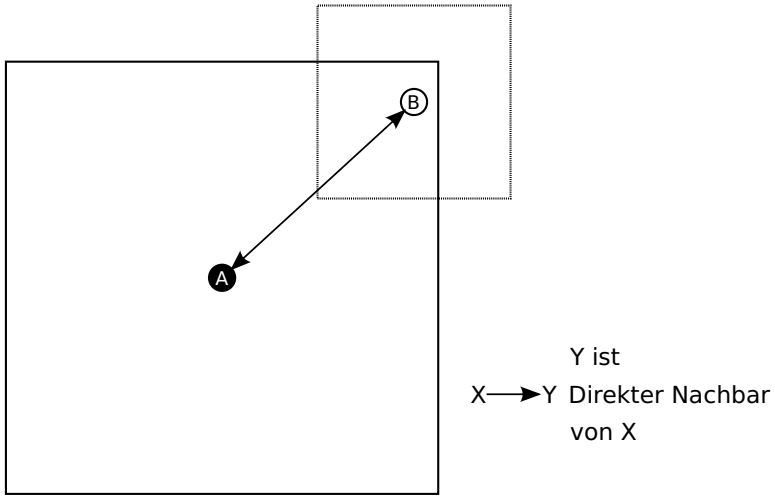


Abbildung 6.13 Zwei Teilnehmer A und B mit unterschiedlich großen Interessengebieten.

die Definition der direkten Nachbarschaftsbeziehung erweitert wird: Ein Teilnehmer A ist nun direkter Nachbar eines Teilnehmers B, wenn sich A im Interessengebiet von B befindet oder sich B im Interessengebiet von A befindet. Algorithmus 6.14 zeigt die derart angepasste Bestimmung der direkten Nachbarn unter Berücksichtigung des in Abschnitt 6.8 eingeführten Interessengebietspuffers.

Vorbedingung: $\mathcal{T}$: Menge der Teilnehmer A bekannten Teilnehmer
Nachbedingung: $\mathcal{N}$: Menge der direkten Nachbarn von A

```
procedure BestimmeDirekteNachbarn(T)
    N ← ∅
    for all t ∈ T do
        if d(p, p_t) ≤ max(r^AoI + r^Buf, r_t^AoI + r^Buf) then
            N ← N ∪ {t}
        end if
    end for
    return N
end procedure
```

Algorithmus 6.14 Bestimmung direkter Nachbarn mit adaptivem Interessengebiet

6.9.2 Adaptionsstrategien

Im folgenden werden Strategien zur Adaption der Interessengebietsgröße besprochen. Dabei wird zunächst auf lokale Strategien eingegangen, die nur lokal vorliegende Information verwenden. Danach wird eine verteilte Strategie vorgestellt, die zusätzlich Informationen anderer Teilnehmer in die Adaption mit einbezieht.

6.9.2.1 Lokale Strategien

Verwendet man eine lokale Strategie, passt ein Knoten die Größe seines Interessengebiets anhand lokaler Informationen an. Hier wird die Anzahl an Verbindungen zu Nachbarn als Metrik verwendet. Diese korreliert stark mit der benötigten Bandbreite und ist in jedem Knoten einfach zu ermitteln.

Nach jeder Bewegung vergleicht ein Knoten die Anzahl an Verbindungen mit einem vorkonfigurierten Schwellenwert. Überschreitet die Zahl der Verbindungen diesen Schwellenwert, wird das Interessengebiet verkleinert. Wird der Schwellenwert unterschritten, wird das Interessengebiet vergrößert. Im Folgenden werden zwei Strategien vorgestellt, mit denen diese Adaption durchgeführt werden kann:

- **Lineare Adaption:** Die derzeitige Größe des Interessengebiets wird proportional zur Differenz zwischen dem Verbindungsschwellenwert und der tatsächlichen Anzahl an Verbindungen vergrößert beziehungsweise verkleinert. Dieser Vorgang wird mit einem *Sensitivitätsparameter* s_l gewichtet: Bei einer hohen Sensitivität sind die Veränderungen der Größe des Interessengebiets größer als bei einer geringen Sensitivität.

  Sei r^{AoI} der Radius des Interessengebiets, n die Anzahl an Nachbarn und n_{MAX} der Verbindungsschwellenwert. Dann gilt:

  $$r^{AoI} \leftarrow r^{AoI} - (n - n_{MAX}) \cdot s_l$$

  Der Sensitivitätsparameter kann zusätzlich normalisiert werden. Hierzu wird er mit der Differenz aus maximalem r^{AoI}_{REG} und minimalem r^{AoI}_{MIN} Interessengebietsradius multipliziert und durch den Verbindungsschwellenwert dividiert:

  $$r^{AoI} \leftarrow r^{AoI} - \frac{(r^{AoI}_{REG} - r^{AoI}_{MIN})(n - n_{MAX})}{n_{MAX}} \cdot s_l$$

  In diesem Fall bleiben die sinnvollen Werte für s_l zwischen $s_l = 0$ für keine Adaption und $s_l = 1$ für sofortiges Setzen des Interessengebietsradius auf die minimale beziehungsweise maximale Größe bei Über- beziehungsweise Unterschreiten des Verbindungsschwellenwertes.

- **Multiplikative Adaption:** Die alte Größe des Interessengebiets wird mit dem Quotienten aus Verbindungsschwellenwert und der Anzahl an Nachbarn multipliziert. Diese Adaption kann wie folgt mit einem Sensitivitätsparameter gewichtet werden:

  $$A_{i+1} = r^{AoI} \leftarrow r^{AoI} \cdot (1 - s_l) + \frac{n_{MAX}}{n} \cdot s_l$$

  Im Gegensatz zur linearen Adaption ist die multiplikative Adaption von der Größe des Interessengebiets abhängig: Wenn das Interessengebiet klein ist, verläuft die Adaption in kleineren Schritten; ist das Interessengebiet groß, werden auch die Adaptionsschritte größer.

6.9.2.2 Verteilte Strategien

Bei einer verteilten Strategie greift ein Teilnehmer auf Informationen von anderen Teilnehmern zurück, um sein Interessengebiet anzupassen. Diese Informationen können beispielsweise die Interessengebietsgröße, die Anzahl der Nachbarn oder der aktuelle Bandbreitenbedarf sein. Die Verwendung verteilter Strategien hat den Vorteil, dass Interessengebietsgrößen angepasst werden können, bevor ein Teilnehmer einen

Hotspot erreicht. Dadurch wird der Übergang in den Hotspot sanfter. Hiermit hat die Verwendung verteilter Information einen dämpfenden Einfluss auf die Adaption. Dies kann sich positiv auf die Nachbarschaftskenntnis in QuON auswirken.

Die hier vorgeschlagene Strategie basiert darauf, die Größe des Interessengebiets zwischen den Teilnehmern zu kommunizieren. Diese müssen zur Bestimmung der direkten Nachbarn ohnehin versendet werden, sodass kein zusätzlicher Overhead entsteht. Das Mittel der Größe des Interessengebiets aller Nachbarn wird dann zur Anpassung des eigenen Interessengebiets genutzt. Parametrisiert wird die Adaption mit einem *Sensitivitätsparameter* s_v. Sei $\bar{r}^{AoI}$ die mittlere Größe des Interessengebiets der Nachbarn. Dann erfolgt die Anpassung wie folgt:

$$r^{AoI} \leftarrow r^{AoI} \cdot (1 - s_v) + \bar{r}^{AoI} \cdot s_v$$

Bei einem hohen Wert für s_v werden also die Interessengebietsgrößen der benachbarten Teilnehmer stärker gewichtet. Diese Strategie muss mit einer lokalen Strategie kombiniert werden, da sonst die Interessengebietsgrößen aller Teilnehmer konstant auf dem Maximalwert blieben. Sowohl die lineare als auch die multiplikative Adaption können hierzu genutzt werden. Die Kombination von lokaler und verteilter Adaption ist in Algorithmus 6.15 dargestellt.

Vorbedingung: r^{AoI}_{REG}: Regulärer Radius des Interessengebiets
Vorbedingung: r^{AoI}_{MIN}: Minimaler Radius des Interessengebiets
Vorbedingung: r^{AoI}: Aktueller Radius des Interessengebiets
Vorbedingung: $\bar{r}^{AoI}$: Durchschnitt des Radius des Interessengebiets aller Nachbarn
Vorbedingung: $n := |\mathcal{N} \cup \mathcal{V} \cup \mathcal{S}|$: Momentane Anzahl an Nachbarn
Vorbedingung: n_{MAX}: Schwellenwert, maximal erwünschte Anzahl an Nachbarn
Vorbedingung: s_l: Sensitivität der lokalen Adaption
Vorbedingung: s_v: Sensitivität der verteilten Adaption
Nachbedingung: r^{AoI}: Neuer Radius des Interessengebiets

procedure VERTEILTEADAPTIONAOI($r^{AoI}_{REG}, r^{AoI}_{MIN}, r^{AoI}, \bar{r}^{AoI}, n, s_l, s_v$)
  if lineare lokale Adaption **then**
    $r^{AoI} \leftarrow r^{AoI} - \frac{(r^{AoI}_{REG} - r^{AoI}_{MIN})(n - n_{MAX})}{n_{MAX}} \cdot s_l$ ▷ Lokale lineare Adaption
  else if $n > 0$ **then**
    $r^{AoI} \leftarrow r^{AoI} \cdot (1 - s_l) + n_{MAX}/n \cdot s_l$ ▷ Lokale multiplikative Adaption
  end if
  $r^{AoI} \leftarrow r^{AoI} \cdot (1 - s_v) + \bar{r}^{AoI} \cdot s_v$ ▷ Verteilte Adaption
  if $r^{AoI} > r^{AoI}_{REG}$ **then**
    $r^{AoI} \leftarrow r^{AoI}_{REG}$
  else if $r^{AoI} < r^{AoI}_{MIN}$ **then**
    $r^{AoI} \leftarrow r^{AoI}_{MIN}$
  end if
  return r^{AoI}
end procedure

Algorithmus 6.15 Adaption der Interessengebietsgröße

6.9.3 Evaluierung

Um die für QuON optimale Adaptionsstrategie zu finden, wurden die in den vorigen Abschnitten erwähnten Strategien mit verschiedenen Werten für die bei der Adap-

tion verwendeten Sensitivitätsparameter mittels Simulation evaluiert. Hierzu wurde wie bei der in Abschnitt 6.7.4.1 beschriebenen Evaluierung OverSim verwendet. Die Konfiguration der Underlay-Parameter wurde beibehalten.

Da hier isoliert der Einfluss der Adaptionsstrategien auf den Bandbreitenbedarf und die Nachbarschaftskenntnis untersucht werden sollte, wurde die Knotenfluktuation in diesen Simulationen unterbunden. Alle Teilnehmer blieben über den gesamten Simulationsverlauf Teil des Netzes. Weiterhin wurde die Simulationszeit auf 30 Minuten verringert (Evaluierung „Verlauf der Interessengebietsgröße:“ 9 Minuten) und dafür die Anzahl der Wiederholungen auf 60 verschiedene Seeds (Evaluierung „Verlauf der Interessengebietsgröße:“ 500 Seeds) erhöht. Zusätzlich wurde das Bewegungsmodell der Teilnehmer angepasst. 250 Teilnehmer bewegen sich auf einem Spielfeld der Größe 1.000 m mal 1.000 m. Sie bewegen sich gemäß eines *Hotspot-Random-Waypoint-Modells*. Erreicht ein Teilnehmer sein aktuelles Ziel, wählt er sich ein neues. Mit einer 50-prozentigen Wahrscheinlichkeit liegt dieses Ziel im Hotspot, einer kreisförmigen Region mit 25 m Radius in der Mitte des Spielfeldes. Spieler, die den Hotspot betreten, halten sich dort für mindestens 20 Sekunden auf. Durch dieses Bewegungsmodell befinden sich im Durchschnitt 100 Spieler in der Hotspot-Region; dort ist die Spielerdichte also sehr hoch. Außerhalb des Hotspots ist die Spielerdichte hingegen relativ niedrig.

Der reguläre Radius der Interessengebiete der Spieler wurde auf 50 m gesetzt. Wurde der Verbindungsschwellenwert von 20 Verbindungen überschritten, wurde dieser Radius gemäß der verschiedenen Adaptionsstrategien verringert. Wurde dieser Schwellenwert unterschritten, wurde das Interessengebiet wiederum gemäß der Adaptionsstrategie bis auf maximal 50 m vergrößert. Zusätzlich wurde ein Minimalradius der Interessengebiete von 1 m festgelegt, unterhalb dessen die Adaption ausgesetzt wurde.

Es wurde die lineare und die multiplikative Adaption sowohl alleine als auch in Kombination mit der verteilten Adaption untersucht. Bei der verteilten Adaption wurde je eine Strategie mit hohem Einfluss der Nachbarn (Parameter s_v=0,7) und eine mit geringerem Einfluss der Nachbarn (Parameter s_v=0,3) gewählt. Für den Sensitivitätsparameter der lokalen Strategien wurden Werte zwischen s_l=0,01 und s_l=0,7 verwendet. Die Simulationsparameter sind in Tabelle 6.3 zusammengefasst. Die Simulationsergebnisse sind in Abbildung 6.14 dargestellt.

Um den Einfluss der verschiedenen Adaptionsstrategien auf die Interessengebietsgröße zu untersuchen, wurde zunächst der Verlauf der Interessengebietsgröße eines Teilnehmers bei der Durchquerung des Hotspots aufgezeichnet. Hierzu startete jeweils ein Teilnehmer bei der Simulation zum Zeitpunkt 400 vom linken Rand des Spielfeldes und durchquerte das Feld mit konstanter Geschwindigkeit. Zum Zeitpunkt 480 trat er in den Hotspot ein. Das Zentrum des Hotspots wurde zum Zeitpunkt 500 erreicht, zum Zeitpunkt 520 wurde der Hotspot verlassen. Zum Zeitpunkt 550 wurde die Simulation beendet. Die Ergebnisse dieser Messung finden sich in Abbildung 6.14a, wobei exemplarisch 0,01 als Wert für den Sensitivitätsparameter gewählt wurde. Es ist zu sehen, dass bei den lokalen Strategien die Adaption der Interessengebietsgröße erst unmittelbar beim Eintritt in den Hotspot erfolgt. Die minimale Interessengebietsgröße wird erst kurz nach dem Durchqueren des Zentrums des Hotspots erreicht. Bei den verteilten Strategien wird die Interessengebietsgröße bereits vor dem Eintritt in den Hotspot verringert. Die minimale Interessengebiets-

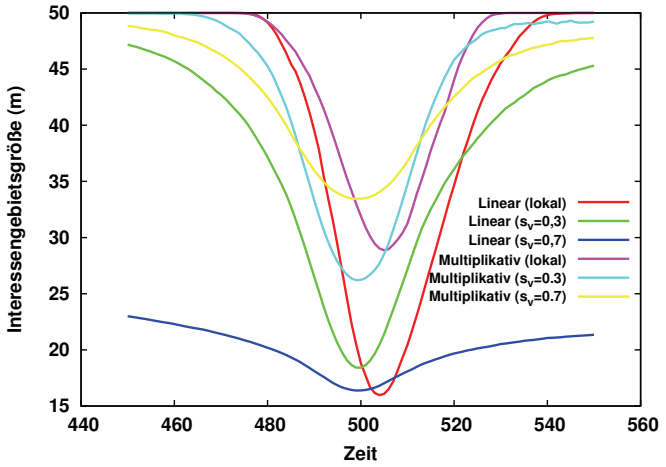
Parameter	Wert
Simulationszeit	30 Minuten nach Initialisierungsphase
Anzahl Seeds	60
Teilnehmerzahl	250
Spielfeldgröße	1 km*1 km
Bewegungsmodell	HotspotRoaming
Hotspot-Wahrscheinlichkeit	50%
Hotspot-Größe	Radius: 25 m
Hotspot-Verweilzeit	20 Sekunden
Sitzungsmodell	NoChurn
Interessengebietsradius	50 m
min. Interessengebietsradius	1 m
Adaption	linear, multiplikativ
Sensitivität s_l	0,01, 0,05, 0,1, 0,2, 0,3, 0,5, 0,7
Sensitivität s_v	0 (lokale Strategie), 0,3, 0,7
Verbindungsschwellenwert	20
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Backupnachbarn	1/Quadrant
Interessengebietspuffer	5 m
Vektornorm	euklidische Norm
Netzmodell	SimpleUnderlay

Tabelle 6.3 Simulationsparameter.

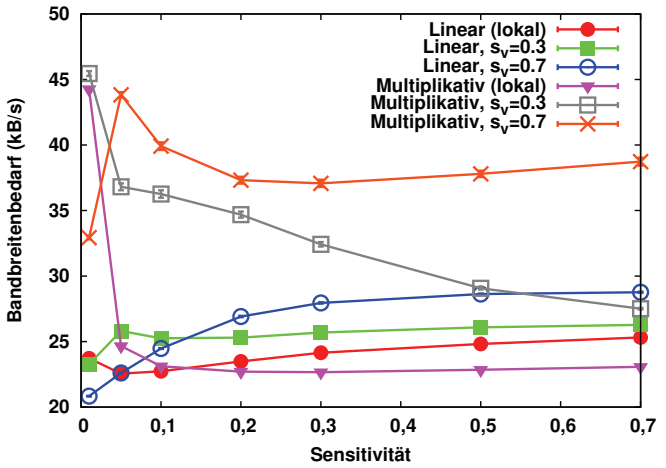
größe wird genau zum Zeitpunkt 500 erreicht, wenn der Teilnehmer das Zentrum des Hotspots erreicht.

Erwartungsgemäß wird bei der Verwendung linearer Strategien ein kleineres Interessengebiet erreicht als bei multiplikativen Strategien, da diese bei kleinen Interessengebietsradien zusätzlich dämpfend wirken. Besonders zu beachten ist der Verlauf der Interessengebietsgröße bei der linearen Strategie mit hohem Einfluss der Nachbarn (s_v=0,7). Hier wird das Interessengebiet bereits lange vor Erreichen des Hotspots verkleinert. Analog wird nach Verlassen des Hotspots das Interessengebiet nur geringfügig vergrößert. Eine solche Degeneration der Interessengebietsgrößen kann auftreten, wenn die lokale Sensitivität s_l (hier: s_l=0,01) im Verhältnis zum Einfluss der Interessengebietsgrößen benachbarter Knoten (hier: s_v=0,7) zu klein gewählt ist. Dadurch bekommen die im Hotspot befindlichen Teilnehmer einen zu starken Einfluss auf die Interessengebietsgrößen von Teilnehmern, die sich in größerer Entfernung zum Hotspot aufhalten. So wird das Interessengebiet in der gesamten virtuellen Welt verkleinert.

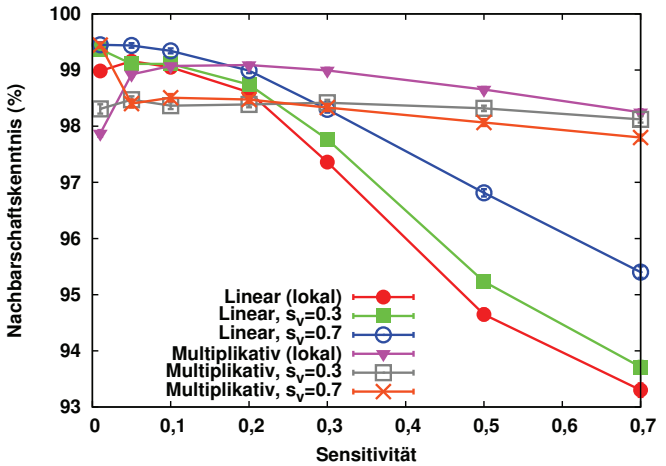
Entscheidender als das Verhalten der Adaptionsstrategien beim Durchqueren von Hotspots ist das Verhalten bei Teilnehmern, die sich für längere Zeit in den Hotspots aufhalten, da dort die Wahrscheinlichkeit einer Überlastsituation am höchsten ist. Besonders wichtig ist hier, in wie weit die Adaption den Bandbreitenbedarf innerhalb der Hotspots senken kann. Hierzu wurde der durchschnittliche Bandbreitenbedarf der Teilnehmer gemessen, die sich innerhalb des Hotspots befanden. Durch die Beschränkung der Bandbreitenmessung auf diese Teilnehmer lässt sich der Bandbreiteneinsparungseffekt der dynamischen Interessengebietsverkleinerung



(a) Verlauf der Interessengebietsgröße eines Teilnehmers beim Queren eines Hotspots.



(b) Durchschnittlicher Bandbreitenbedarf der Teilnehmer innerhalb des Hotspots in Abhängigkeit der Adaptionsstrategie.



(c) Durchschnittliche Nachbarschaftskenntnis in Abhängigkeit der Adaptionsstrategie.

Abbildung 6.14 Evaluierungsergebnisse der Adaption der Interessengebietsgröße.

zeigen. Abbildung 6.14b zeigt den Bandbreitenbedarf der Teilnehmer innerhalb des Hotspots abhängig von der gewählten Adaptionsstrategie.

Die Ergebnisse zeigen, dass die Kombination aus verteilter Adaption mit einer multiplikativen lokalen Adaptionsstrategie deutlich mehr Bandbreite benötigt als andere Strategien. So benötigt die multiplikative verteilte Strategie mit s_v=0,7 abhängig von der lokalen Sensitivität s_l zwischen 33 kByte/s für s_l=0,01 und 45 kByte/s für s_l=0,05. Hierbei ist der Bandbreitenbedarf für s_l=0,01 deutlich niedriger als die anderen Meßwerte der multiplikativen verteilten Strategie mit s_v=0,7. Der Grund hierfür liegt darin, dass in dieser Konfiguration die durchschnittliche Interessengebietsgröße aufgrund des zu geringen Einflusses der lokalen Adaption degeneriert und auch außerhalb des Hotspots nur noch 14 m beträgt. Der Effekt tritt bei dieser Konfiguration nur auf, wenn Teilnehmer längere Zeit im Hotspot-Gebiet verbleiben, und ist deswegen bei der Evaluierung „Verlauf der Interessengebietsgröße“ aufgrund der dort untersuchten kurzen Durchquerung des Hotspots nicht zu erkennen.

Die multiplikative verteilte Strategie mit s_v=0,3 benötigt bei s_l=0,01 mit über 45 kByte/s die höchste gemessene Bandbreite. Bei s_l=0,7 sinkt der Bandbreitenbedarf bis auf 27 kByte/s. Die multiplikative lokale Strategie benötigt bei s_l=0,01 mit fast 45 kByte/s ebenfalls eine sehr hohe Bandbreite. Bei höheren Werten für den Sensitivitätsparameter fällt dieser Wert jedoch deutlich. Ist s_l größer als 0,05, werden maximal 25 kByte/s benötigt, bei s_l=0,7 sinkt die benötigte Bandbreite bis auf gute 23 kByte/s.

Auch bei den linearen Strategien benötigt die lokale Strategie weniger Bandbreite als die verteilten Strategien. Hier sind die Unterschiede allerdings nicht so groß wie bei den multiplikativen Strategien. Die lokale Strategie benötigt zwischen 23 kByte/s bei s_l=0,05 und 25 kByte/s bei s_l=0,7. Die verteilte Strategie mit s_v=0,3 benötigt zwischen 23 kByte/s bei s_l=0,01 und 27 kByte/s bei s_l=0,7. Bei der verteilten linearen Strategie liegt der Bandbreitenbedarf bei s_l=0,01 bei 21 kByte/s. Hier ist zu beachten, dass bei dieser Konfiguration die Interessengebietsgröße wie oben beschrieben auch außerhalb des Hotspots deutlich verkleinert wird; diese Strategie ist mit diesen Parametern somit nicht für einen Einsatz in der Praxis geeignet. Bei einer lokalen Sensitivität von 0,05 liegt der Bandbreitenbedarf bei 23 kByte/s, mit größeren Werten für den Sensitivitätsparameter steigt der Bandbreitenbedarf auf 28 kByte/s für s_l=0,7. Zum Vergleich liegt die benötigte Bandbreite bei Verzicht auf den Einsatz dynamischer Interessengebiete bei 91 kByte/s.

Die Simulationsergebnisse zeigen, dass die dynamische Anpassung der Interessengebietsgröße an die Lastsituation eine wirksame Methode ist, um den Bandbreitenbedarf in Hotspot-Regionen zu verringern. Erwartungsgemäß benötigen die lokalen Strategien am wenigsten Bandbreite, da bei verteilten Strategien durch die Einbeziehung der Lastsituation der Nachbarn die Interessengebiete am Rande der Hotspot-Region im Vergleich zu rein lokalen Strategien etwas vergrößert werden. Mit linearen verteilten Strategien lassen sich dennoch mit Einsparungen von mehr als 70% sehr gute Werte erreichen.

Abbildung 6.14c zeigt die Nachbarschaftskenntnis der Strategien in Abhängigkeit des lokalen Sensitivitätsparameters. Dabei sinkt in der Regel die Nachbarschaftskenntnis bei steigender Sensitivität. Dies ist zu erwarten, da eine höhere Sensitivität zu größeren Sprüngen der Interessengebietsgrößen führt.

Besonders ausgeprägt ist dies bei den linearen Strategien. Bei der lokalen linearen Strategie wird bei einer lokalen Sensitivität s_l=0,05 eine maximale Nachbarschaftskenntnis von über 99,1% erreicht. Bei höheren Sensitivitätswerten fällt diese stetig

auf 93,3% bei einer Sensitivität s_l=0,07. Die verteilten linearen Strategien liegen in der Nachbarschaftskenntnis bei gegebener Sensitivität stets über den Werten für die lokale Strategie. Die verteilte lineare Strategie mit s_v=0,3 erreicht bei einer Sensitivität s_l=0,01 eine Nachbarschaftskenntnis von 99,4%. Bei einer Sensitivität von 0,7 fällt die Nachbarschaftskenntnis bis auf 93,7%. Die verteilte lineare Strategie mit s_v=0,7 erreicht die besten Werte für die Nachbarschaftskenntnis. So wird bei s_l=0,01 eine Nachbarschaftskenntnis von 99,5% erreicht, bei einer lokalen Sensitivität von 0,7 beträgt die Nachbarschaftskenntnis noch 95,4%.

Bei den multiplikativen Strategien ist die Korrelation zwischen Sensitivität und Nachbarschaftskenntnis nicht so deutlich ausgeprägt. Bei der multiplikativen lokalen Strategie liegt das Minimum der Nachbarschaftskenntnis bei einer Sensitivität s_l=0,01. Hier wird eine Nachbarschaftskenntnis von 97,9% erreicht. Von diesem Punkt steigt die Nachbarschaftskenntnis bis auf 99,1% bei einer Sensitivität s_l=0,2. Von dort sinkt sie bis auf 98,3% bei s_l=0,7. Die multiplikativen verteilten Strategien liegen im Regelfall unter diesen Werten. Die Nachbarschaftskenntnis variiert hier zwischen 98,5% und 97,8%. Ein Ausreißer ist die multiplikative verteilte Strategie mit s_v=0,7 und s_l=0,01. Hier liegt die Nachbarschaftskenntnis bei fast 99,5%. Dieser Ausreißer erklärt sich dadurch, dass in dieser Konfiguration die Adaptionsschritte deutlich kleiner sind als bei allen anderen Konfigurationen. Aufgrund des relativ hohen Bandbreitenbedarfs dieser Konfiguration ist sie jedoch nur bedingt für einen Praxiseinsatz geeignet.

Die Simulationsergebnisse zeigen, dass bei Verwendung dynamischer Interessengebietsgrößen eine gute Nachbarschaftskenntnis gewahrt werden kann. Bei geeigneter Wahl der Parameter ist eine Nachbarschaftskenntnis von 99,5% möglich. Ohne die Verwendung dynamischer Interessengebietsgrößen lag die Nachbarschaftskenntnis im untersuchten Szenario bei 99,9%. Bei der Evaluierung konnte die verteilte lineare Strategie mit s_v=0,7 die beste Nachbarschaftskenntnis vorweisen. Auch mit den rein lokalen Strategien ließ sich eine Nachbarschaftskenntnis von über 99% erreichen, beispielsweise mit der linearen Strategie bei s_l=0,05 oder mit der multiplikativen Strategie bei s_l=0,2. Die verteilten multiplikativen Strategien konnten nicht überzeugen.

Betrachtet man das Verhältnis von Nachbarschaftskenntnis und Bandbreite, ist die verteilte lineare Strategie mit s_v=0,7 und kleinen Werten für den lokalen Sensitivitätsparameter s_l am überzeugendsten. So kann bei einer Wahl von s_l=0,05 eine Nachbarschaftskenntnis von 99,4% bei einer Bandbreite von 23 kByte/s erreicht werden, bei s_l=0,1 noch eine Nachbarschaftskenntnis von 99,3% bei einem Bandbreitenbedarf von 25 kByte/s.

Da es, wie oben beschrieben, bei kleinen Werten für s_l und großen für s_v dazu kommen kann, dass abhängig vom Szenario die Interessengebietsgrößen außerhalb des Hotspots deutlich verkleinert werden, ist auch die Wahl rein lokaler Strategien in Betracht zu ziehen. Hierbei sind die lineare Strategie mit s_l=0,05 und die multiplikative Strategie mit s_l=0,2 annähernd gleichwertig. Die hiermit erreichte Nachbarschaftskenntnis liegt bei 99,1% und der Bandbreitenbedarf bei 23 kByte/s.

Verglichen mit den Messwerten bei Verzicht auf dynamische Interessengebietsgrößen lässt sich mit den hier beschriebenen Strategien die durchschnittliche Bandbreite in Hotspots um mehr als 75% reduzieren. Dabei muss eine Einbuße in der Nachbarschaftskenntnis von 0,8 Prozentpunkten in Kauf genommen werden. Somit sind

dynamische Interessengebietsgrößen eine effektive Möglichkeit, Überlastsituationen in Hotspot-Szenarien zu vermeiden.

6.10 Aufwandsabschätzung

In Abschnitt 5.1.5 wird der Aufwand bisheriger P2P-Protokolle für virtuelle Welten abgeschätzt. Um einen Vergleich mit diesen Abschätzungen zu ermöglichen, werden hier die Aufwände bezüglich Nachrichtenverzögerung, Bandbreitenbedarf und Nachbarschaftskenntnis der einzelnen Funktionen von QuON zusammengefasst.

In QuON werden alle Positionsmeldungen direkt vom Verursacher zu allen interessierten Nachbarn gesendet. Die Zustellung von Positionsmeldungen erfolgt also in $O(1)$. Demzufolge sind sehr geringe Nachrichtenverzögerungen zu erwarten.

Der Bandbreitenbedarf von QuON setzt sich aus dem Aufwand für die Verteilung der Positionsmeldungen, dem Aufwand für den Backup-Mechanismus und dem sonstigen Verwaltungsaufwand zusammen. Letzterer ist die Summe des Aufwands für die Nachbarschaftsfindung und des Aufwands für das Aufrechterhalten der Verbindung zu Verbindungsnachbarn und temporären Nachbarn. Bei m Nachbarn beträgt der Aufwand für das Versenden der Positionsmeldungen wie in Abschnitt 6.5 erläutert $O(m)$. Die Zahl der Backup-Nachbarn pro Quadrant ist wie in Abschnitt 6.7.4 beschrieben konstant, der Aufwand für den Backup-Mechanismus ist somit $O(1)$. Insgesamt ist der Aufwand für die Positionsmeldungen und Backup gering.

Der Aufwand für die Nachbarschaftsfindung beträgt gemäß Abschnitt 6.6.1 $O(m)$. Da die Zahl der Verbindungsnachbarn in QuON konstant ist, beträgt der Aufwand für die Kommunikation mit den Verbindungsnachbarn $O(1)$. In pathologischen Fällen können alle Nachbarn eines Teilnehmers temporäre Nachbarn sein. In diesem Fall beträgt der Aufwand für die Verbindungen zu den temporären Nachbarn $O(m)$. Im Normalfall liegt die Zahl der temporären Nachbarn jedoch deutlich unter der Zahl der Verbindungsnachbarn. Der Aufwand für die Verwaltung des Netzes beträgt so insgesamt $O(m + m + 1) = O(m)$. Im Vergleich zu bisherigen Protokollen ist dies ein durchschnittlicher Wert.

Die Nachbarschaftskenntnis von QuON wird im Wesentlichen von der Güte der Nachbarschaftsfindung bestimmt. Ohne Teilnehmerausfälle, Paketverluste und Paketverzögerung kann QuON 100% Nachbarschaftskenntnis gewährleisten (Beweis siehe Anhang B.2). Besonders der Ausfall von Verbindungsnachbarn kann jedoch die Nachbarschaftsfindung kurzfristig beeinträchtigen, bis über den Backup-Mechanismus ein neuer Verbindungsnachbar gefunden worden ist. Diese Zeit ist gemäß Abschnitt 6.7.4 konstant, also $O(1)$. Bei Gruppenbildung kann der Ausfall eines Verbindungsnachbarn mehrere Teilnehmer betreffen und so das Risiko einer eingeschränkten Nachbarschaftskenntnis vergrößern. Andererseits haben bei hoher Teilnehmerdichte Teilnehmer mehr direkte Nachbarn, die in der Nachbarsfindung aushelfen können. Dies mildert das Risiko wieder ab. Zudem können bei hoher Teilnehmerdichte ausgefallene Verbindungsnachbarn besser durch den Backup-Mechanismus ersetzt werden. Insgesamt ist so eine sehr gute Nachbarschaftskenntnis zu erwarten.

In Tabelle 6.4 wird der Aufwand von QuON mit den in Kapitel 5 untersuchten bisherigen Protokollen verglichen[5]. Als einziges der betrachteten Protokolle kann

[5]Für eine Erläuterung des Aufwands der bisherigen Protokolle siehe Abschnitt 5.1.5.

QuON eine gute Nachrichtenverzögerung mit einer guten Nachbarschaftskenntnis verbinden.

Protokoll	Aufwand	Bewertung
QuON	$O(1)$	$\oplus$
SimMUD	$O(\log(n))$	$\ominus$
PubSubMMOG	$O(\log_k(m))$	$\ominus$
N-Tree	$O(1)$	$\oplus$
Vast	$O(1)$	$\oplus$

(a) Nachrichtenverzögerung.

Protokoll	Pos. Meldungen	Zonenwechsel	Backup	Verwaltung	Bew.
QuON	$O(m)$	–	$O(1)$	$O(m)$	$\bigcirc$
SimMUD	$O(m\log(n))$	$O(\log(n))$	–	$O(\log(n))$	$\oplus$
PubSubMMOG	$O(m\log(m))$	$O(1)$	$O(m)$	$O(\log(m))$	$\oplus$
N-Tree	$O(m)$	$O(n)$	$O(m)$	$O(m)$	$\bigcirc$
Vast	$O(m)$	–	$O(m^2)$	$O(m^2)$	$\ominus$

(b) Bandbreitenbedarf.

Protokoll	Zonenwechsel	Ausfallerkennung	Weitere Einflüsse	Bew.
QuON	–	schnell	Ausfall von Verbindungsnachbarn kann für kurze Zeit Nachbarsfindung stören	$\oplus$
SimMUD	$O(\log(n))$	langsam	Kein Backup-Mechanismus	$\ominus$
PubSubMMOG	$O(1)$	schnell	Zentraler Server ist single point of failure	$\oplus$
N-Tree	$O(n)$	langsam	Ausfall nahe Wurzel des N-Trees verhindert Zonenwechsel	$\ominus$
Vast	–	schnell	Inkonsistente Voronoi-Graphen stören Nachbarsfindung	$\bigcirc$

(c) Nachbarschaftskenntnis.

Tabelle 6.4 Aufwandsabschätzung und Bewertung des Aufwands von QuON in Vergleich mit bisherigen Protokollen.

6.11 Zusammenfassung des Protokollablaufs

In diesem Kapitel wurden die einzelnen Protokollbausteine von QuON erläutert. Zum Abschluss soll nun der komplette Protokollablauf aus Sicht eines Teilnehmers eines QuON-Netzes zusammengefasst werden. Initial ist der Teilnehmer nicht mit dem

QuON-Netz verbunden. Um dem Netz beizutreten, sendet er einen *JOIN-Request* mit seiner gewünschten Position in der virtuellen Welt an einen beliebigen Teilnehmer, der bereits Teil des Netzes ist. Dieser JOIN-Request wird solange gemäß Algorithmus 6.9 im Netz weitergeleitet, bis der zuständige Teilnehmer gefunden ist. Dieser beantwortet den *JOIN-Request* mit einem *JOIN_ACK*. In diesem ist eine Liste mit potentiellen Nachbarn enthalten. Hiermit kann der beitretende Teilnehmer seine Liste bekannter Teilnehmer $\mathcal{T}$ initialisieren. Damit ist er dem Netz beigetreten.

Jeder Teilnehmer, der Teil eines QuON-Netzes ist, klassifiziert regelmäßig – in üblichen Szenarien mehrmals pro Sekunde – alle ihm bekannten Teilnehmer aus der Menge $\mathcal{T}$ nach den in Abschnitt 6.4 beschriebenen Nachbarschaftskriterien: Zunächst werden die direkten Nachbarn (siehe Abschnitt 6.4.1) unter Berücksichtigung des in Abschnitt 6.8 vorgestellten Interessengebietspuffers gemäß Algorithmus 6.14 oder bei Nichtverwendung dynamischer Interessengebiete nach 6.12 bestimmt und in der Menge $\mathcal{N}$ gespeichert. Weiterhin werden die Verbindungsnachbarn $\mathcal{V}$ (siehe Abschnitt 6.4.2) gemäß Algorithmus 6.11 bestimmt. Dabei werden gleichzeitig auch die in Abschnitt 6.7.4 vorgestellten Backup-Nachbarn in der Menge $\mathcal{B}$ gesichert. Zum Abschluss der Klassifizierung werden die temporären Nachbarn $\mathcal{S}$ (siehe Abschnitt 6.4.3) gemäß Algorithmus 6.4 bestimmt.

Im Anschluss an die Klassifizierung werden alle Teilnehmer, die nicht als Teilnehmer klassifiziert wurden, aus der Liste der bekannten Teilnehmer $\mathcal{T}$ entfernt. Bei Bedarf kann daraufhin die in Abschnitt 6.9 beschriebene Interessengebietsadaption nach Algorithmus 6.15 durchgeführt werden. Im Anschluss daran wird die aktuelle Position gemäß Algorithmus 6.5 an alle Nachbarn versendet. Dabei erhalten Verbindungsnachbarn und temporäre Nachbarn zusätzlich Adressen und Positionen der anderen Verbindungsnachbarn und temporären Nachbarn zugesendet; Verbindungsnachbarn werden in der versendeten Nachricht über ihren Status als Verbindungsnachbar informiert. In längeren Intervallen – etwa alle fünf Sekunden – kontaktiert jeder Teilnehmer die Backup-Nachbarn aus seiner Menge $\mathcal{B}$ und fordert von diesen ihre aktuelle Position sowie die ihrer Verbindungs- und temporären Nachbarn an.

Jeder Teilnehmer eines QuON-Netzes empfängt regelmäßig Positionsnachrichten seiner Nachbarn. Beim Empfang einer solchen Nachricht wird der Absender gemäß Algorithmus 6.10 in die Liste der bekannten Teilnehmer $\mathcal{T}$ gespeichert. Weiterhin wird überprüft, ob der Absender aufgrund seiner Bewegung mindestens einen neuen Nachbarn gewonnen hat. Ist dies der Fall, wird die Bewegungsnachricht gemäß Algorithmus 6.13 mit einer Liste der neuen Nachbarn beantwortet. Ist der Empfänger der Positionsmeldung ein Verbindungsnachbar oder temporärer Nachbar des Senders, enthält die Positionsmeldung zusätzlich die Liste der Verbindungs- und temporären Nachbarn des Absenders. Der Empfänger fügt diese, sofern sie ihm nicht bereits bekannt sind, ebenfalls seiner Liste der bekannten Teilnehmer $\mathcal{T}$ hinzu.

Empfängt ein Teilnehmer von einem seiner Nachbarn eine gewisse Zeit keine Positionsmeldungen, geht er davon aus, dass der Nachbar ausgefallen ist. Dieser wird nun in einer Löschliste $\mathcal{L}$ gespeichert und aus der Liste der bekannten Teilnehmer $\mathcal{T}$ gelöscht. Informationen über Teilnehmer, die in der Löschliste gespeichert sind, werden ignoriert, es sei denn, der Absender einer solchen Information ist der vermeintlich ausgefallene Teilnehmer selber oder in dieser Nachricht wird eine von der in der Löschliste gespeicherten Position abweichende Position genannt.

Will ein Teilnehmer das QuON-Netz verlassen, versendet er eine *LEAVE-Notification* an alle bekannten Teilnehmer. Diese Nachricht enthält seine vollständige Liste bekannter Teilnehmer. Die Empfänger der Nachricht fügen alle Teilnehmer aus dieser Liste ihrer lokale Liste der bekannten Nachbarn hinzu. In Algorithmus 6.16 ist der hier beschriebene Protokollablauf dargestellt.

Vorbedingung: x: Beliebiger Teilnehmer eines QuON-Netzes
procedure QuON(x)
send JOIN$(A, p_A) \mapsto x$
while JOIN_ACK$(\mathcal{N}_B, \mathcal{P}_{\mathcal{N}_B}, \mathcal{V}_B, \mathcal{P}_{\mathcal{V}_B}, \mathcal{S}_B, \mathcal{P}_{\mathcal{S}_B})$ noch nicht empfangen **do**
wait(Wartezeit)
send JOIN$(A, p_A) \mapsto B$
end while
$\mathcal{T} \leftarrow \mathcal{N}_B \cup \mathcal{P}_{\mathcal{N}_B} \cup \mathcal{V}_B \cup \mathcal{P}_{\mathcal{V}_B} \cup \mathcal{S}_B \cup \mathcal{P}_{\mathcal{S}_B}$
$\mathcal{L} \leftarrow \emptyset$
repeat
$\mathcal{N} \leftarrow$ BestimmeDirekteNachbarn($\mathcal{T}$) ▷ Alg. 6.14
$\mathcal{V}, \mathcal{B} \leftarrow$ BestimmeVerbindungsUndBackupNachbarn($\mathcal{T}$) ▷ Alg. 6.11
$\mathcal{S} \leftarrow$ BestimmeTemporäreNachbarn($\mathcal{T}, \mathcal{V}$) ▷ Alg. 6.4
if Zeit seit letzer Positionsmeldung $\geq 1/$ Sendefrequenz **then**
if Dynamische Interessengebiete **then**
AdaptionAoI($r^{AoI}_{REG}, r^{AoI}_{MIN}, r^{AoI}, \bar{r}^{AoI}, n, s_l, s_v$) ▷ Alg. 6.15
end if
SendePositionsmeldungen($\mathcal{T}$) ▷ Alg. 6.5
end if
if Zeit seit letzer Anfrage an Backup-Nachbarn $\geq 1/$ Backupfrequenz **then**
for all $b \in \mathcal{B}$ **do**
send Backupanfrage$(A) \mapsto b$
end for
end if
if Backupanfrage(b) empfangen **then**
send MOVE$(A, p, r^{AoI}, \mathcal{V}, \mathcal{P}_{\mathcal{V}}, \mathcal{S}, \mathcal{P}_{\mathcal{S}}) \mapsto b$
end if
if MOVE$(m, p_m, r^{AoI}_m, [\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}])$ empfangen **then** ▷ Alg. 6.10
BehandlePositionsmeldung($\mathcal{T}, \mathcal{L}$,MOVE$(m, p_m, r^{AoI}_m, [\mathcal{V}_m, \mathcal{P}_{\mathcal{V}_m}, \mathcal{S}_m, \mathcal{P}_{\mathcal{S}_m}])$)
if $m \in \mathcal{N}$ **then** ▷ Alg. 6.13
BenachrichtigeÜberNeueNachbarn($\mathcal{N}$,MOVE(m, p_m, r^{AoI}_m))
end if
end if
if JOIN(n, p_n) empfangen **then**
BehandleBeitrittsanfrage(JOIN(n, p_n),$\mathcal{N}, \mathcal{V}, \mathcal{S}$) ▷ Alg. 6.9
end if
if LEAVE$(o, \mathcal{T}_o, \mathcal{P}_{\mathcal{T}_o})$ empfangen **then**
for all $t \in \mathcal{T}_o : t \notin \mathcal{T}$ **do**
AktualisierePosition(t)
$\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
end for
end if
if Zeit seit Nachricht eines Teilnehmers p empfangen > Alive-Timeout **then**
$\mathcal{T} \leftarrow \mathcal{T} \setminus \{p\}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{p\}$
end if
until Benutzer will QuON verlassen
for all $t \in \mathcal{T}$ **do**
send LEAVE$(A, \mathcal{T}, \mathcal{P}_{\mathcal{T}}) \mapsto t$
end for
end procedure

Algorithmus 6.16 Der Protokollablauf von QuON

7. Cheating

Ein verbreitetes Problem in virtuellen Welten ist das sogenannte *Cheating*, Englisch für „Mogeln“. In vielen bisherigen Protokollvorschlägen für virtuelle Welten wurde das Problem des Cheatings ignoriert. Andere, auf die Abwehr von Cheating fokussierte Protokollvorschläge waren oftmals nicht skalierbar oder führten zu inakzeptablen Latenzen. In diesem Kapitel wird eine Cheating-Erkennung vorgestellt, die nicht mit negativen Einflüssen auf Skalierbarkeit oder Latenz verbunden ist. Die Mechanismen werden im Kontext des in Kapitel 6 vorgestellten Protokolls QuON vorgestellt, lassen sich aber auch mit anderen P2P-Protokollen für virtuelle Welten kombinieren.

7.1 Cheating in virtuellen Welten

Cheating ist seit Einführung der Online-Spiele ein Problem für ehrliche Spieler und Spielebetreiber. Insbesondere im Kontext von P2P-basierten virtuellen Welten stellt Cheating ein aktuelles Forschungsproblem dar. In diesem Kapitel wird das Problem „Cheating“ genauer vorgestellt.

7.1.1 Problembeschreibung

Cheating ist üblicherweise definiert als das regelwidrige Erlangen von Vorteilen von Spielern eines Spiels [74, 146, 150]. Dabei können von dem cheatenden Spieler verschiedene Angriffswege gewählt werden. Mögliche Angriffswege sind beispielsweise das Ausnutzen von Fehlern in der Spielsoftware, das Verändern der Spielsoftware um Gegner schwächer oder die eigene Spielfigur stärker zu machen oder die Steuerung der Spielfigur mit Automatisierungsprogrammen. Gemein ist diesen Methoden nur, dass dem Spieler das Erreichen von Zielen im Spiel vereinfacht wird und dass das Ausnutzen dieser Möglichkeiten gegen die Regeln verstößt, die der Anbieter oder Betreiber des Spiels vorgesehen hat.

Das Phänomen des Cheatings konnte bereits vor der größeren Verbreitung von Online-Spielen in Einspieler-Spielen beobachtet werden. Beispielsweise modifizierten

Spieler Spiele derart, dass die Spielfigur nicht getötet werden konnte oder dass die Spielfigur beliebig oft nach ihrem Tod wiederbelebt werden konnte [38]. Hierbei diente das Cheaten im Wesentlichen der Vereinfachung des Spiels für den Spieler. Da in Einspieler-Spielen die Folgen des Cheatings auf den cheatenden Spieler selber beschränkt bleiben, entsteht aus dieser Art des Cheatings kein Schaden. Einige Hersteller lieferten ihre Spiele sogar mit versteckten Funktionen aus, die dem Spieler die Vorteile bringen, die ansonsten nur durch Cheating erreicht werden können. Beispielsweise lassen sich in vielen Spielen des Herstellers Konami durch das Drücken der Tasten „Oben Oben Unten Unten Links Rechts Links Rechts B A“ Vorteile für die Spielfigur, wie zusätzliche Ausrüstung oder höhere Spielstärke der Spielfigur, erlangen [38].

Bei Mehrspieler-Spielen sind die Auswirkungen von Cheating jedoch nicht mehr auf den Cheater selbst begrenzt, sondern für alle Mitspieler des Spiels spürbar. Bei kompetitiven Spielen verhindert oder erschwert Cheaten beispielsweise das Gewinnen nicht-cheatender Gegenspieler. Gerade bei Online-Spielen, bei denen nicht nur befreundete Personen, sondern auch einander unbekannte Menschen in einem Spiel aufeinander treffen können, kann Cheating den Spielspaß nicht-cheatender Spieler deutlich einschränken. Muss ein Spieler damit rechnen, dass er beim online Spielen regelmäßig auf cheatende Gegenspieler trifft, kann ihn das vom Spielen abschrecken und so einen finanziellen Schaden beim Hersteller oder Betreiber des Spiels verursachen.

In virtuellen Welten, insbesondere in MMOGs, ist dieses Problem besonders ausgeprägt. MMOGs sind üblicherweise nur online zu spielen, und Spieler interagieren regelmäßig mit ihnen fremden Spielern. Wird ein Spieler bei Interaktionen regelmäßig durch Cheater übervorteilt, kann er die Motivation verlieren, weiter zu spielen. Da MMOGs oft durch monatliche Zahlungen der Spieler finanziert werden, ist der Verlust von Spielern aufgrund von cheatenden Teilnehmern ein großes finanzielles Risiko für den Anbieter dieser Spiele.

7.1.2 Klassifizierung von Cheating

Aufgrund der unscharfen Definition von Cheating und der Vielfalt an Cheats ist eine vollständige Aufzählung aller bekannten konkreten Arten des Cheatings, wie sie beispielsweise Robles et al. versucht haben [119], schwierig. Derartige Aufzählungen werden schnell durch Entwicklung neuer Cheats obsolet. Erfolgversprechender sind Ansätze, die Cheats systematisch zu klassifizieren. Hierbei sind verschiedene Ansätze möglich. Neumann et al. schlagen vor, Cheats analog zu der Klassifikation von Angriffen auf Kryptosysteme nach der durch den Cheat kompromittierten Eigenschaft zu klassifizieren [102]. Dabei werden die drei Eigenschaften *Vertraulichkeit*, *Integrität* und *Verfügbarkeit* betrachtet. Diese Klassifizierung deckt zwar alle wesentlichen Cheatmöglichkeiten ab, ist aber für die Entwicklung von Gegenmaßnahmen oder Erkennungsstrategien wenig hilfreich: Angriffe auf die einzelnen Eigenschaften können auf vielen verschiedenen Ebenen stattfinden, sodass Cheats aus einer der gegebenen Kategorien nur wenige Gemeinsamkeiten aufweisen müssen.

Eine deutlich ausführlichere Klassifizierung wird von Yan und Randell vorgeschlagen [150]. In dieser Klassifizierung werden Cheats nach drei Dimensionen unterteilt:

- **Angegriffenes Ziel:** Welches Ziel wird von dem Cheat angegriffen?

  - Das Spielsystem
  - Die unterliegende Infrastruktur
  - Einzelne Spieler
  - Administratoren des Spiels

- **Konsequenz:** Welcher Schaden entsteht durch den Cheat?
  - Verletzung der Fairness
  - Annahme einer falschen Identität
  - Verletzung der Integrität des Spielsystems
  - Dienstblockade
  - Diebstahl von Informationen oder (virtuellen) Besitztümern

- **Angreifer:** Wer ist der Angreifer?
  - Ein einzelner Spieler
  - Eine Gruppe von Spielern
  - Ein Administrator oder Mitglied der Betreiberfirma
  - Ein Administrator in Zusammenarbeit mit Spielern

Dabei ist die Zuordnung von Cheats zu den einzelnen Klassen nicht eindeutig; so kann beispielsweise ein Cheat durch den Diebstahl von Informationen die Fairness des Spiels verletzen. Eine weitere Schwäche der Klassifikation ist, dass einige Kategorien auf nahezu alle üblichen Cheats zutreffen: Da das Ziel eines Cheats ein regelwidriger Vorteil für den Cheater ist, verletzt jeder Cheat in der Regel auch die Fairness.

Webb und Soh klassifizieren Cheats eindimensional nach der Schicht, auf der der Angriff erfolgt [146]. Dabei unterscheiden sie zwischen folgenden Schichten:

- **Spielebene:** Cheats, die ohne Modifikation oder externe Einflüsse innerhalb der Spielsoftware auftreten.
- **Anwendungsebene:** Cheats, die durch Modifikation der Spielsoftware oder durch Interaktion der Spielsoftware mit externen Programmen auftreten.
- **Protokollebene:** Cheats, die durch das Verändern, Einfügen oder Unterdrücken von Paketen in die Kommunikation des Spiels auftreten.
- **Infrastrukturebene:** Cheats, die auf Modifikationen oder Angriffe auf die dem Spiel unterliegende Software oder Hardware aufbauen.

Diese Klassifikation ermöglicht eine deutlich bessere Differenzierung der Angriffsziele von Cheats.

Eine weitere Klassifikation von Cheating ist die *Häufigkeit* von regelwidrigen Aktionen des Cheaters. Es lassen sich so *sporadische Cheats* und *permanente Cheats*

unterscheiden. Bei sporadischen Cheats verhält sich der Cheater die meiste Zeit regelkonform und bricht nur gelegentlich die Regeln. Sporadische Cheats lohnen sich für den Cheater nur, wenn der einzelne Regelbruch einen deutlichen Vorteil bringen kann. Bei permanentem Cheating sind die meisten der Aktionen des Cheaters regelwidrig. Die Auswirkungen dieser Cheats sind in der Regel gering, durch die permanente Anwendung bringen sie jedoch insgesamt einen Vorteil für den Cheater.

Ein Beispiel für sporadisches Cheating ist das Ignorieren von Hindernissen in der Spielwelt: Die meiste Zeit bewegt sich der Cheater normal in der Welt. Um Zeit zu sparen, durchquert er jedoch von Zeit zu Zeit ein eigentlich nicht passierbares Hindernis. Ein Beispiel für permanentes Cheating ist die Erhöhung der Maximalgeschwindigkeit der Spielfigur. Hier erhöht der Cheater die zwischen zwei Positionsmeldungen maximal zurücklegbare Distanz. Damit dies einen Vorteil für den Cheater bringt, muss er dieses Verhalten über einen längeren Zeitraum aufrecht erhalten. In der Folge ist der größte Teil seiner Positionsmeldungen nicht regelkonform.

Eine spezielle Klasse von Cheats sind *Angriffe auf Cheat-Erkennungsmaßnahmen.* Implementiert eine virtuelle Welt Maßnahmen, mit denen ein Cheating der Teilnehmer erkannt oder verhindert werden soll, können Cheater versuchen, die Maßnahmen anzugreifen oder zu umgehen. Der Angriff auf diese Maßnahmen bringt dem Cheater zunächst keinen direkten Vorteil, ermöglicht aber gegebenenfalls, weitere Cheats unerkannt durchzuführen, die ansonsten von der Cheat-Erkennung bemerkt oder verhindert worden wären.

7.1.3 Cheating in P2P-Systemen

In virtuellen Welten auf der Basis von P2P-Protokollen gibt es aufgrund der fehlenden zentralen Kontrollinstanz mehr Cheatmöglichkeiten als in klassischen Client/Server-basierten virtuellen Welten. Diese betreffen die Protokoll- und Anwendungsebene der obigen Klassifizierung. Der Grund ist hier, dass in P2P-basierten virtuellen Welten alle Berechnungen von den Teilnehmern durchgeführt werden. Dadurch ist es einfach möglich, durch Veränderung der Spielsoftware Ereignisse zu erzeugen, die nicht den Regeln des Spiels entsprechen. Da eine zentrale Kontrolle nicht stattfinden kann, können Spieler diese Ereignisse ungehindert publizieren. Auf Protokollebene findet ebenfalls keine zentrale Kontrolle der versendeten Nachrichten statt. So lassen sich durch manipulierte Nachrichten Spielzustände irregulär verändern oder durch das Unterdrücken von Nachrichten oder das Einschleusen gefälschter Nachrichten bei anderen Spielern falsche Spielzustände erzeugen.

In dieser Arbeit sollen vorrangig diese neuen, durch das Verwenden von P2P-Protokollen verursachten Cheatmöglichkeiten betrachtet werden. Das Ziel ist es, ein Sicherheitsniveau zu erreichen, das zu klassischen Client/Server-basierten virtuellen Welten äquivalent ist.

Auch in klassischen virtuellen Welten kann Cheating nicht vollständig verhindert werden. Theoretisch wäre zwar eine vollständige Überprüfung aller von Spielern durchgeführten Eingaben und eine vollständige Berechnung der Auswirkungen dieser Eingaben in dem zentralen Spielserver möglich. Der dazu nötige Rechenaufwand wäre jedoch sehr hoch, sodass diese Methode von keiner aktuellen virtuellen Welt eingesetzt wird. Stattdessen wird ein Teil der Berechnungen durch die Spielsoftware der Teilnehmer durchgeführt und die Ergebnisse vom Server übernommen. Dadurch

werden Cheats auf der Anwendungsebene möglich: Durch Veränderung der Spielsoftware können diese Berechnungen verändert werden, sodass cheatende Spieler beispielsweise Hindernisse in der Spielwelt ignorieren oder sich schneller als vorgesehen in der Spielwelt bewegen können.

Um das Cheaten für Spieler unattraktiv zu machen, werden üblicherweise die Aktionen der Spieler, entweder aufgrund einer zufälligen Stichprobe oder aufgrund von Anschuldigungen anderer Spieler, auf Plausibilität geprüft. Fällt dabei ein Regelbruch eines Spielers auf, wird er für sein Fehlverhalten durch den Anbieter des Spiels bestraft. Je nach Schwere des Vorfalls kann eine solche Strafe beispielsweise eine zeitlich begrenzte Spielsperre oder der permanente Ausschluss aus dem Spiel sein.[1] Neben der Entfernung von böswilligen Spielern aus dem Spiel besitzen diese Strafen eine abschreckende Wirkung auf die verbleibenden Spieler, sodass bei konsequenter Anwendung dieser Strafen virtuelle Welten größtenteils frei von Cheatern bleiben können.

Die Verwendung der Kombination aus Cheat-Erkennung und Bestrafung bietet sich auch in P2P-basierten virtuelle Welten an. Im Gegensatz zu einer im Protokoll integrierten Cheat-Verhinderung ist eine Cheat-Erkennung ohne zusätzliche Nachrichtenverzögerung möglich. Um diese Mechanismen in P2P-basierten virtuellen Welten einsetzen zu können, müssen zwei grundlegende Funktionen bereitgestellt werden: Für das Erkennen von Cheats ist eine *verteilte Cheat-Erkennung* nötig; um die Bestrafung von Cheatern durchführen zu können, wird eine *Authentifizierung und Autorisierung* benötigt.

7.2 Related Work

Bei einem großen Teil der P2P-Protokollvorschläge für virtuelle Welten wird nicht näher auf das Problems des Cheatings eingegangen [58, 67, 79, 149]. In diesen Protokollen ist es für böswillige Spieler in der Regel einfach, unbehelligt Cheats auf der Anwendungs- oder Protokollebene durchzuführen.

Einige Arbeiten versuchen, bestimmte Cheats durch das Protokolldesign zu verhindern. Eine größere Zahl von Vorschlägen betrachtet das sogenannte *Latency-Cheating*. Bei dieser Art des Cheatens versucht der Angreifer, seine eigenen Ereignisnachrichten zu verzögern, bis er die Ereignisnachrichten seiner Gegenspieler erhalten hat. Er kann dann auf diese Ereignisse reagieren, indem er seine eigenen Nachrichten rückdatiert. Im *Lockstep*-Protokoll von Baughman et al. [14] wird dies verhindert, indem die Spielzeit in Runden unterteilt wird. Jeder Spieler muss sich zum Ende einer Runde mittels eines kryptographischen Hashwertes auf eine Aktion festlegen. Wenn sich alle Spieler festgelegt haben, werden die Aktionen der Spieler veröffentlicht. Dieser Ansatz hat jedoch das Problem, dass zunächst auf alle Hashwerte gewartet werden muss, bis die Aktionen bekannt gegeben werden können. Somit ergibt sich durch das Protokoll eine zusätzliche Verzögerung in der Höhe der maximalen Ende-zu-Ende-Verzögerung zwischen den Teilnehmern. Dies ist in der Regel für Spiele mit hohem Interaktionsgrad nicht akzeptabel. Zusätzlich wird eine Angriffsmöglichkeit geschaffen: Verzögert ein einzelner Spieler bewusst seine Hashwerte, müssen alle anderen Teilnehmer des Spiels warten. Somit kann ein einzelner böswilliger Spieler das Spiel-Erlebnis für alle anderen Teilnehmer der virtuellen Welt beeinträchtigen.

[1] Beispielsweise sperrte der Spielehersteller Blizzard am 20.04.2010 über 320.000 Accounts den Zugang zu ihren Online-Spielen für mindestens 30 Tage [5, 21].

Um dieses Problem zu mildern, wird von Baughman et al. *Asynchronous Synchronisation* vorgeschlagen [14]. In dieser Erweiterung des Lockstep-Protokolls wird nur auf die Hashwerte von Spielern gewartet, die einen Einfluss auf das Interessengebiet eines Teilnehmers haben könnten. Somit wird der Einfluss von Spielern mit hoher Latenz auf das Interessengebiet eingegrenzt, das grundsätzliche Problem der zusätzlichen Wartezeit wird jedoch nicht gelöst.

Cronin et al. erweitern den Lockstep-Ansatz um eine *Sliding Pipeline* [39]. Dabei werden mehrere lokale Ereignisse erzeugt und deren Hashwerte verschickt, bevor auf die Nachrichten der anderen Teilnehmer gewartet wird. Die Maximalzahl der vorab versendeten Hashwerte richtet sich nach der Latenz der Spieler. Bei diesem Ansatz wird der lokale Spielfluss nicht mehr durch das Warten auf Nachrichten anderer Spieler verzögert. Die Klartext-Ereignisnachrichten werden jedoch weiterhin erst versendet, wenn die Hashwerte der anderen Spieler empfangen wurden. Somit erreichen die Klartext-Ereignisnachrichten andere Spieler wie bei Lockstep erst nach einer großen Verzögerung. Weiterhin können Spieler durch Vortäuschen einer hohen Latenz gegnerischen Spielern den eigenen Hashwert vorenthalten, bis die Klartextnachricht der vorigen Runde des gegnerischen Spielers eingetroffen ist. Somit können von dem Sliding Pipeline-Protokoll nicht alle Arten des Latency-Cheating verhindert werden.

Das Protokoll *NEO* [59] von GauthierDickey et al. versucht, die Vorteile einer Sliding Pipeline beizubehalten, ohne dabei wie bei dem im vorigen Abschnitt vorgeschlagenen Protokoll Einbußen bei der Verhinderung von Latency-Cheating hinnehmen zu müssen. Hierzu teilt es die Spielzeit in Runden fester Länge ein. In jeder Runde sendet jeder Teilnehmer seine Ereignisnachricht verschlüsselt an die anderen Teilnehmer. Bei einer Pipeline-Länge von l veröffentlicht er den Schlüssel zu dieser Nachricht l Runden später.[2] Um die im klassischen Sliding Pipeline-Protokoll bestehenden Cheatmöglichkeiten unmöglich zu machen, stimmen die Teilnehmer nach dem Ende der Runde ab, welche Ereignisnachrichten beachtet oder ignoriert werden sollen. Dabei werden alle Nachrichten beachtet, die bei einem vorab festgelegten Anteil der Teilnehmer rechtzeitig eingetroffen sind, alle anderen Nachrichten werden ignoriert. Somit können Teilnehmer ihre Nachrichten nicht mehr künstlich verzögern. Allerdings wird es Teilnehmern mit zu hoher Latenz unmöglich gemacht, an dem Spiel teilzunehmen. Wie bei den bislang beschriebenen Protokollen ergibt sich in NEO eine Verzögerung der Ereignisnachrichten. Aufgrund der Aufteilung der Spielzeit in Runden beträgt die Verzögerung das Doppelte der Rundenlänge.

Um Inkonsistenzen zu vermeiden, können Teilnehmer in NEO ihre Spielstände vergleichen. Sind diese nicht identisch, werden die zurückliegenden Nachrichten verglichen. Wird dabei festgestellt, dass von einem Teilnehmer in einer Runde unterschiedliche Nachrichten an unterschiedliche Empfänger versendet worden sind, wird dies als Cheatversuch gewertet. Um dieses Verfahren sicher durchführen zu können, müssen alle Ereignisnachrichten vom Absender signiert und vom Empfänger für eine

[2] Kryptographisch ist der Ansatz von Lockstep und Sliding Pipeline, zuerst einen Hashwert des Ereignisses zu versenden und danach das Ereignis im Klartext zu publizieren, besser. Die Sicherheit des Verfahrens beruht auf der Annahme, dass es nicht möglich ist, seine Aktion nach Publizierung der ersten Nachricht zu ändern. Bei kryptographisch sicheren Hashfunktionen ist diese Eigenschaft gegeben. Bei verschlüsselten Nachrichten ist jedoch eine Kollisionsfreiheit nicht notwendig. Es kann also möglich sein, dass bei geschickter Auswahl der Nachricht verschiedene Schlüssel generiert werden können, die zu verschiedenen gültigen Ereignisnachrichten führen.

gewisse Zeit gespeichert werden. Dies fordert einen nicht unerheblichen Rechen- und Speicherbedarf bei den Teilnehmern.

Insgesamt kann in Frage gestellt werden, wie sinnvoll das Verhindern von Latency-Cheating in der Praxis ist. Latency-Cheating setzt voraus, dass alle Ereignisnachrichten der Spieler in eine eindeutige Reihenfolge gebracht werden müssen, die durch die vorgebliche Sendezeit bestimmt wird. Zusätzlich geht Latency-Cheating davon aus, dass bei diesen Ereignissen die Kausalität der Spielereignisse vollständig gewahrt bleibt. Dies ist jedoch in virtuellen Welten üblicherweise nicht der Fall. Um die verschiedene Latenz der einzelnen Spieler auszugleichen, erlauben die meisten Spiele eine unscharfe Wirkung der Ereignisse. So kann beispielsweise ein Angriff einen Spieler treffen, der den Zielort des Angriffs vor kurzer Zeit bereits verlassen hat. In der virtuellen Welt „World of Warcraft“ ist es möglich, dass ein Duell zweier Spieler als unentschieden gewertet wird, obwohl bei vollständiger Ordnung der Spielereignisse einer von beiden Spielern zuerst alle Lebenspunkte verloren haben müsste, und somit ein eindeutiger Gewinner festzustellen gewesen wäre. Unter diesen Bedingungen sind die durch Latency-Cheating erzielbaren Vorteile vernachlässigbar. Die durch die Vermeidung dieser Cheats benötigte zusätzliche Latenz ist dem geringen Nutzen nicht angemessen.

Das von Webb et al. vorgestellte Protokoll *RACS* [147] versucht, eine größere Menge von Cheats aus Anwendungs- und Protokollebene zu verhindern. Dafür führt es einen sogenannten *Referee*-Knoten ein, der die Autorität über den Spielstand besitzt. Teilnehmer senden alle ihre Ereignisnachrichten an diesen Referee. Dieser prüft alle eingehenden Nachrichten auf Cheating. Teilnehmer, deren Interessengebiete sich schneiden, können vom Referee angewiesen werden, ohne Umweg über den Referee direkt untereinander zu kommunizieren. Dabei sind sie jedoch weiterhin verpflichtet, alle Ereignisse zusätzlich an den Referee zu versenden. Gehen auf dem Kommunikationsweg zwischen den Teilnehmern bei direkter Kommunikation zu viele Nachrichten verloren oder versucht ein Teilnehmer, durch das Unterdrücken oder Verzögern von Nachrichten den anderen zu übervorteilen, kann dieser fordern, die Nachrichten wieder über den Referee zugestellt zu bekommen. Durch RACS wird eine dem Client/Server-Modell äquivalente Sicherheit gewährleistet. Allerdings muss der Referee jede Ereignisnachricht der Teilnehmer empfangen, überprüfen und gegebenenfalls an andere Teilnehmer weiterleiten. Hierdurch ist er nicht mehr wesentlich von einem gewöhnlichen zentralen Server zu unterscheiden. Durch die Möglichkeit, Teilnehmer direkt miteinander statt über den Server kommunizieren zu lassen, kann eine gewisse Bandbreiteneinsparung gegenüber einem klassischen Client/Server-Protokoll erreicht werden. Der sonstige Ressourcenverbrauch, also insbesondere Speicher und Rechenzeit, ändert sich jedoch nicht. RACS kann somit nicht die Skalierbarkeit erreichen, die in P2P-Systemen sonst möglich ist.

Goodman und Verbrugge schlagen *Peer Auditing* zur Verhinderung von Cheating vor [60]. In diesem Schema wird jedem Teilnehmer ein *Proxy* zugewiesen. Teilnehmer senden alle Ereignisnachrichten an ihren Proxy. Dieser berechnet die Auswirkungen des Ereignisses und sendet die Resultate an einen zentralen Server. Mit einer gewissen Wahrscheinlichkeit wird die Ereignisnachricht des Spielers an einen weiteren Teilnehmer des Spiels gesendet, dem sogenannten *Co-Auditor*. Dieser berechnet ebenfalls die Auswirkungen des Ereignisses. Der Server vergleicht daraufhin die Resultate von Proxy und Co-Auditor. Ergeben sich wesentliche Unterschiede in den Resultaten,

ist dies ein Hinweis darauf, dass entweder Proxy oder Co-Auditor cheaten. Der Server kann daraufhin selber die Resultate des Ereignisses berechnen und feststellen, welcher von beiden falsche Ergebnisse liefert. Alle Ereignisse, die nicht von einem Co-Auditor überprüft werden, werden vom Server einem Kurztest auf Plausibilität unterzogen, um offensichtliche Cheat-Versuche zu unterbinden. Als zusätzlicher Mechanismus wird allen Teilnehmern ein Vertrauenswert zugewiesen. Dieser wird vom Server berechnet und bei Empfang von korrekten Resultaten erhöht, bei implausiblen oder abweichenden Resultaten verringert. Mit *Peer Auditing* kann so ein hohes Maß an Cheat-Sicherheit erreicht werden. Allerdings wird für dieses Protokoll ein zentraler Server benötigt, der alle Ereignisnachrichten aller Teilnehmer empfängt. Ein gewisser Teil der Rechenlast kann im Vergleich zu einem klassischen Client/Server-System von den Proxies übernommen werden, Speicher und Bandbreitenbedarf des zentralen Servers unterscheiden sich jedoch nicht von klassischen Systemen.

Von Kabus et al. wird zur Verhinderung von Cheating *Mutual Checking* vorgeschlagen [74]. In diesem Ansatz wird die Spielwelt in Zonen unterteilt. Jeder Zone werden einige Teilnehmer als *Region Controller* zugewiesen. Die Teilnehmer innerhalb der Zone senden allen Region-Controllern ihrer Zone alle ihre Ereignisnachrichten zu. Diese überprüfen die Gültigkeit der Ereignisse. Weiterhin vergleichen sie, welcher Spielzustand aus diesen Ereignissen resultiert. Weichen die Zustände der einzelnen Controller voneinander ab, einigen sich diese auf einen gültigen Zustand. Setzt man voraus, dass die Mehrheit der Controller ehrlich ist, kann so eine dem Client/Server-Modell äquivalente Cheatsicherheit gewährleistet werden. Dies setzt aber voraus, dass die Zahl der Region-Controller pro Zone nicht zu klein ist. Ansonsten kann es vorkommen, dass aufgrund einer ungünstigen Zuweisung der Controller in einzelnen Zonen die Mehrheit der Controller unehrlich sind. Bei einer großen Zahl an Controllern pro Zone ergibt sich jedoch das Problem der Konsensfindung. Dieses ist nichttrivial: Hierzu ist das sogenannte *Problem der byzantinischen Generäle* zu lösen [87]. Da der Spielfluss erst weitergehen kann, wenn die Controller sich auf einen gültigen Spielstand geeinigt haben, kann durch das böswillige Erzeugen abweichender Zustände in den Controllern der Spielfluss für alle in der Zone befindlichen Teilnehmer erheblich verzögert werden, bis der Konsens zwischen den Controllern gefunden ist.

Ein alternativer Ansatz von Kabus et al. ist das *Log Auditing* [74]. Hier wird die Welt ebenfalls in Zonen aufgeteilt. Jeder Zone wird ein Teilnehmer als Region-Controller zugewiesen. Alle Teilnehmer senden ihre Ereignisnachrichten signiert an den Controller. Dieser berechnet die Auswirkungen der Ereignisse und versendet signiert die resultierenden Zustandsänderungen an die Spieler. Zusätzlich speichert er alle Nachrichten der Spieler. Zu festgelegten Zeiten sendet er den aktuellen Zustand und sämtliche Ereignisnachrichten der Spieler an eine zentrale, vertrauenswürdige Stelle. Diese prüft dann, ob der Controller alle Zustandsberechnungen korrekt durchgeführt hat. So soll ein Cheating durch den Region-Controller verhindert werden. Allerdings können so nicht alle Cheatmöglichkeiten des Controllers erkannt werden. Der Controller kann beispielsweise einzelne Ereignisse eines Spielers ignorieren und so die Gegenspieler dieses Spielers bevorzugen. Ein weiterer Nachteil dieses Ansatzes ist die Ressourcenbelastung des Controllers. Dieser agiert in dem vorgeschlagenen Protokoll de facto als Server für die gesamte Region. Er hat also denselben Rechen- und Bandbreitenaufwand, den ein Server in einem klassischen Client/Server-System hat. Der Speicherbedarf ist sogar größer als in einem Client/Server-System, da der

Controller sämtliche Nachrichten der Teilnehmer speichern muss. Es besteht so die Gefahr, dass der Controller überlastet wird; das System skaliert nicht.

Insgesamt können alle diese Cheatverhinderungsprotokolle eine hohe Sicherheit gegen Cheating erreichen. Allerdings wird dafür entweder ein zentraler Server benötigt, der alle Ereignisnachrichten der Spieler empfängt und bearbeitet, wodurch der Vorteil zu klassischen Client/Server-Systemen nur gering ist, oder es wird einem Teil der Teilnehmer Aufgaben übertragen, die in Lastsituationen von normalen Endsystemen nicht zu leisten sind. In beiden Fällen ist die Skalierbarkeit des Systems als gering einzuschätzen.

Eine Alternative zu der Verhinderung von Cheats durch das Protokoll ist es, Cheats nur zu detektieren und cheatende Spieler zu bestrafen. Dies ermöglicht die Vermeidung der Skalierungsprobleme der Cheatverhinderungsprotokolle. Es gibt einige Arbeiten, die die Detektion einzelner Cheatklassen ermöglichen. In der Regel wird versucht, Cheats auf der Anwendungsebene durch eine verhaltensbasierte Analyse zu finden. So kann die Steuerung einer Spielfigur durch ein externes Programm, einem sogenannten *Bot*, anhand der resultierenden Laufwege oder der Verteilung der erzeugten Nachrichten erkannt werden [32, 33, 135]. Ähnliche Techniken lassen sich verwenden, um sogenannte *Wall-Hacks* zu erkennen, bei denen Cheater Spielhindernisse undurchsichtig erscheinen lassen oder passierbar machen [88].

Diese Arbeiten beschäftigen sich jedoch nur mit der Erkennung des jeweiligen Cheats durch einen zentralen Server. Aus diesem Grund werden dort keine weiteren Maßnahmen vorgestellt, durch die diese Erkennungsstrategien in P2P-Systemen verwendet werden können. Die in diesem Kapitel vorgestellte Cheat-Erkennung für P2P-basierte virtuelle Welten kann diese Erkennungsstrategien als Module bei der verteilten Cheat-Erkennung einsetzen, um die jeweiligen Cheatklassen detektieren zu können.

7.3 Cheatermodell

Um die Effizienz von Maßnahmen gegen Cheating abschätzen zu können, muss zunächst ein Angreifermodell für Cheating festgelegt werden. Die in Abschnitt 7.1.2 erwähnte Cheatklassifizierung nach Neumann et al. legt durch ihre Einteilung von Cheats in Angriffe auf Vertraulichkeit, Integrität und Verfügbarkeit einer virtuellen Welt nahe, dass ein Cheater im Wesentlichen ein Sonderfall eines in Kommunikationsprotokollen betrachteten Angreifers ist.

Das in vielen Problemen der Netzsicherheit herangezogene Dolev-Yao-Angreifermodell [43] geht jedoch über die Fähigkeiten eines realistisch anzunehmenden Cheaters hinaus. Durch die definitionsgemäße Einschränkung der Ziele eines Cheaters auf Erlangung eines Vorteils in einer virtuellen Welt ergeben sich bei Cheatern Einschränkungen bei den anzunehmenden Angriffsmöglichkeiten und Angriffsmethoden.

In dieser Arbeit wird angenommen, dass Cheater in der Regel gewöhnliche Teilnehmer einer virtuellen Welt sind. Demnach wird davon ausgegangen, dass der Cheater keinen direkten Zugriff auf die Netzinfrastruktur hat. Topologisch gesehen befindet er sich also am Rande des Netzes. Er hat somit keinen Zugriff auf Nachrichten anderer Spieler, die nicht an ihn adressiert sind. Weiterhin hat er keinen Zugriff auf die Daten, die andere Spieler lokal verwalten.

Angriffsmöglichkeiten für den Cheater ergeben sich aus der lokalen Kontrolle über die Spielsoftware und dem Netzverkehr: Es wird davon ausgegangen, dass der Cheater volle *Kontrolle über die Spielsoftware* besitzt. Somit kann er alle lokal abgelegten Datenstrukturen ändern sowie das Ergebnis aller Berechnungen beeinflussen. Dies ermöglicht eine große Auswahl an Cheats auf Anwendungsebene. So kann ein Cheater die Steuerung des Spiels durch eine Computersteuerung automatisieren oder seine Eingaben algorithmisch optimieren. Diese Art des Cheatens wird *Botting* genannt. Mit sogenannten *Wall-Hacks* kann ein Cheater Hindernisse durchsichtig erscheinen lassen, um Gegner durch Hindernisse hindurch sehen zu können oder die Hindernisse passierbar machen. Der Cheater kann irreguläre Aktionen durchführen. So kann er zum Beispiel seine Laufgeschwindigkeit über die vom Spiel vorgegebene Grenze hinweg steigern. Durch die Veränderung von lokalen Datenstrukturen kann ein Cheater die Werte seiner Spielfigur ändern.

Weiterhin hat ein Cheater *Kontrolle über den vom Spiel erzeugten Datenverkehr*. Er kann Nachrichten modifizieren, duplizieren, willkürlich erzeugen, unterdrücken oder verzögern. Dadurch ergeben sich Cheatmöglichkeiten auf der Protokollebene. So kann ein Cheater beispielsweise durch die Manipulation oder Erzeugung gefälschter Nachrichten eine falsche Identität vortäuschen und somit Ereignisse im Namen anderer Spieler versenden. Durch das Versenden inkonsistenter Nachrichten kann er einem Teil der Spieler eine andere Folge von Ereignissen senden als den übrigen. Eine weitere Cheatmöglichkeit ist das Nichtdurchführen von Protokollaufgaben. So kann ein Cheater in QuON das Durchführen der Nachbarschaftsfindung unterdrücken.

Eine Kooperation zwischen Cheatern ist grundsätzlich möglich. Kooperierende Cheater können ihre Aktionen miteinander abstimmen, um so den individuellen Vorteil zu steigern oder zu versuchen, einer Entdeckung zu entgehen. Es wird jedoch davon ausgegangen, dass stets nur eine geringe Zahl von Cheatern auf diese Weise kooperiert. Insbesondere ist anzunehmen, dass nicht alle Nachbarn eines Cheaters ebenfalls Cheater sind.

Zusammengefasst handelt es sich bei dem in dieser Arbeit betrachteten Cheater um einen *lokalen* Angreifer, dessen Fähigkeiten sich auf die Manipulation des lokalen Systems beschränken. Auf diesem lokalen System hat er jedoch die volle Kontrolle und kann alle Berechnungen und Kommunikation auf dem lokalen System gezielt steuern.

7.4 Authentifizierung und Autorisierung

Um das Vortäuschen falscher Identitäten durch Cheater zu unterbinden und eine effektive Bestrafung der entdeckten Cheater zu ermöglichen, ist eine Authentifizierung und Autorisierung der Teilnehmer notwendig. Eine einfache Lösung ist die Einführung einer zentrale Autorität als Vertrauensanker. Die hier beschriebenen Verfahren lassen sich jedoch auch mit einer verteilten Authentifizierung und Autorisierung [145] kombinieren. Im Folgenden wird zunächst von einer zentralen Autorisierungsstelle ausgegangen.

Damit die Teilnehmer der virtuellen Welt gegenseitig ihre Autorisierung überprüfen können, stellt die Autorisierungsstelle nach erfolgreicher Authentifizierung ein signiertes, mit einer Gültigkeitsdauer versehenes Ticket aus. Das Ticket hat folgendes

Format: $H()$ ist eine sichere kryptographische Hashfunktion, $S_k()$ ist die kryptographische Signatur mit dem Schlüssel k, k^c der private Schlüssel der Autorisierungsstelle, t_0 der Ausstellungszeitpunkt und t_{end} das Ende des Gültigkeitszeitraumes, A die IP-Adresse des Teilnehmers und ID_A der Identifikator des Teilnehmers.

$$\text{Ticket}(A, ID_A, t_0, t_{end}, S_{k^c}(H(A, ID_A, t_0, t_{end})))$$

Dieses Ticket muss ein Teilnehmer vorzeigen, wenn er eine Verbindung mit einem anderen Teilnehmer aufbauen will oder – im Fall von zonenbasierten virtuellen Welten – einer neuen Zone beitreten will. Der Empfänger des Tickets kann dann anhand der im Ticket gespeicherten Daten feststellen, ob der Absender zu dem gegebenen Zeitpunkt mit gegebener Adresse und Identifikator am Netz teilnehmen darf. Entsprechen die im Ticket angegebenen Daten nicht den Daten des Absenders oder entspricht der signierte Hashwert nicht den dazugehörigen Daten, ist das Ticket ungültig. Jede Kommunikation mit dem Absender wird dann unterbunden. Dies stellt sicher, dass nur Teilnehmer mit gültigem Ticket an der virtuellen Welt teilnehmen können.

Nach Ablauf der Gültigkeitsdauer des Tickets muss der Spieler ein neues Ticket beantragen und seinen Kommunikationspartnern vorzeigen. Dies erlaubt einen einfachen Ausschluss von Cheatern aus der virtuellen Welt: Fallen Spieler durch Cheating auf, wird das Ausstellen eines neuen Tickets verweigert.

Hierdurch entsteht bei der Wahl des Gültigkeitszeitraumes für die Tickets jedoch ein Konflikt zwischen der Möglichkeit eines schnellen Ausschlusses von Cheatern und der Zahl an Ticketanträgen an die Autorisierungsstelle: Wird ein kurzer Gültigkeitszeitraum gewählt, können Cheater schnell aus dem System entfernt werden. Allerdings müssen alle, auch die nicht cheatenden Teilnehmer, relativ oft ihr Ticket erneuern. Ein langer Gültigkeitszeitraum senkt die Last der Autorisierungsstelle, ermöglicht aber Cheatern, nach ihrer Entdeckung eine relativ lange Zeit Teil der virtuellen Welt zu bleiben.

Dieses Problem lässt sich durch die Einführung eines Ticketrückrufs lösen: Fällt ein Teilnehmer durch Cheaten auf, wird von der Autorisierungsstelle ein signierter Ticketrückruf für das Ticket des Cheaters erstellt und an die Nachbarn des Cheaters oder den Koordinator der Zone des Cheaters versendet. Der Ticketrückruf hat folgendes Format:

$$\text{Ticketrückruf}(A, ID_A, t_0, S_{k^c}(H(A, ID_A, t_0)))$$

Der Empfänger des Rückrufs stellt daraufhin die Kommunikation mit dem Cheater ein und verteilt den Ticketrückruf gegebenenfalls an weitere Teilnehmer, die daraufhin ebenfalls die Kommunikation mit dem Cheater unterbinden. Somit wird der Cheater schnell aus dem Netz entfernt.

Um einen Wiedereintritt des Cheaters an anderer Stelle zu verhindern, verifizieren alle Teilnehmer stichprobenartig mit einer geringen Wahrscheinlichkeit die Gültigkeit eines Tickets online bei der Autorisierungsstelle. Ist das Ticket zurückgezogen worden, beantwortet die Autorisierungsstelle diese Anfrage mit dem Ticketrückruf für das entsprechende Ticket; der Cheater wird daraufhin durch den oben beschriebenen Mechanismus aus dem Netz entfernt.

7.5 Vertrauen

Das in diesem Kapitel vorgeschlagene verteilte Cheating-Erkennungssystem sieht vor, dass die Cheat-Erkennung von den Teilnehmern der virtuellen Welt durchgeführt wird. Dabei ist nicht auszuschließen, dass es zu falschen Anschuldigungen kommt. Diese können aus einer Fehlinterpretation des Verhaltens eines Teilnehmers resultieren, aber auch böswillige Falschanschuldigungen darstellen. Deswegen darf eine Cheat-Anschuldigung nicht unmittelbar zu einer Strafe für den Beschuldigten führen.

Aus diesem Grund wird ein Vertrauenssystem eingeführt. Wie die Authentifizierung wird das Vertrauenssystem der Einfachheit halber zunächst von einer zentralen Vertrauensstelle implementiert. Es ist jedoch auch eine Kombination mit verteilten Vertrauenssystemen denkbar, die einen globalen Vertrauenswert für Teilnehmer bereitstellen [1].

Die Vertrauensstelle berechnet für jeden Teilnehmer einen Vertrauenswert. Wird ein Teilnehmer des Cheatings verdächtigt, sinkt das Vertrauen in diesen Spieler. Dabei können verschiedene Faktoren berücksichtigt werden:

- *Schwere des Regelverstoßes:* Regelverstöße, die besonders schwerwiegend sind, beispielsweise weil sie die Stabilität des Netzes beeinträchtigen oder andere Teilnehmer in besonders starker Weise benachteiligen, führen zu einem stärkeren Vertrauensverlust als Regelverstöße, die nur geringe Auswirkungen haben. Der Gewichtungsfaktor für die Schwere des Regelverstoßes wird im Folgenden mit α_s bezeichnet.

- *Vertrauenswert des Anklägers*: Es wird angenommen, dass Anschuldigungen vertrauenswürdiger Ankläger vermutlich der Wahrheit entsprechen. Anschuldigungen seitens wenig vertrauenswürdiger Teilnehmer sind hingegen möglicherweise Falschanschuldigungen[3]. Deswegen werden Anschuldigungen durch Teilnehmer mit hohem Vertrauenswert stärker gewichtet als Anschuldigungen durch Teilnehmer mit niedrigem Vertrauenswert. Der vom Vertrauenswert der Anklägers abhängige Gewichtungsfaktor wird im Folgenden mit α_v bezeichnet werden. Dieser berechnet sich wie folgt aus dem Vertrauenswert v eines Teilnehmers und den Schwellenwerten v^+ und v^-. Teilnehmer mit einem Vertrauenswert oberhalb von v^+ werden als besonders vertrauenswürdig angesehen, Teilnehmer mit einem Vertrauenswert unterhalb von v^- als besonders unvertrauenswürdig. Dann ist

$$\alpha_v := \begin{cases} 1 & v > v^+ \\ 0 & v < v^- \\ \frac{v-v^-}{v^-+v^+} & sonst \end{cases}$$

- *Rolle des Beschuldigers*: Für die Überwachung der Teilnehmer wird in Abschnitt 7.6 die sogenannte Beobachterrolle eingeführt. Anschuldigungen, die von einem Teilnehmer in seiner Rolle als Beobachter eingereicht werden, werden stärker gewichtet als andere Anschuldigungen. Der rollenabhängige Gewichtungsfaktor wird im Folgenden als α_r bezeichnet.

[3]Dabei ist zu beachten, dass auch das Beschuldigen unschuldiger Teilnehmer als Cheating gewertet wird.

- *Indizien*: Kann ein Cheat-Verhalten durch Indizien oder Beweise belegt werden, wie beispielsweise durch die in Abschnitt 7.9 eingeführten signierten Nachrichten, wird die Cheat-Anschuldigung stärker gewichtet. Der Gewichtungsfaktor für die Indizienlage wird als α_i bezeichnet. Für nicht durch Indizien gestützte Anklagen beträgt der Faktor 1, für durch kryptographische Signaturen beweisbare Anschuldigungen α_i^{max} mit $\alpha_i^{max} \gg 1$.

Beim Eintreffen einer Cheatanschuldigung wird der Vertrauenswert des beschuldigten Teilnehmers um den Wert $\alpha_s \cdot \alpha_v \cdot \alpha_r \cdot \alpha_i$ verringert.

Um Cheat-Anschuldigungen ausgleichen zu können, wirken sich Zeitspannen ohne Anschuldigungen positiv auf den Vertrauenswert aus. Der Vertrauenswert wird logarithmisch bezüglich der Dauer dieser Zeitspannen erhöht. Durch die Wahl der Basis β des Logarithmus lässt sich die Geschwindungkeit der Vertrauenssteigerung steuern. Ist der Zeitpunkt der letzten Cheatanschuldigung gegen einen Teilnehmer $t^{'}$, so ist sein Vertrauenswert v_t zum Zeitpunkt t

$$v_t := v_{t'} + \log_\beta(t - t^{'})$$

Diese Anpassung ermöglicht eine relativ schnelle Erholung des Vertrauenswertes nach einer Falschanschuldigung. Das Erreichen von hohen Vertrauenswerten verlangt jedoch lange Phasen ohne Cheat-Anschuldigung. Da Fehlanschuldigungen in einem P2P-System nicht ausgeschlossen werden können, ist ein geringer Vertrauenswert noch kein Beweis für ein Cheating-Verhalten. Ein hoher Vertrauenswert ist jedoch ein gutes Indiz dafür, dass es sich um einen ehrlichen Teilnehmer handelt.

Der Vertrauenswert beeinflusst die Gültigkeitsdauer der Tickets der Teilnehmer. Teilnehmer mit einem niedrigen Vertrauenswert sind des Cheatings verdächtig und erhalten nur Tickets mit einer kurzen Gültigkeit. Teilnehmer mit hohem Vertrauen sind aller Wahrscheinlichkeit nach keine Cheater und erhalten dementsprechend länger gültige Tickets.

7.6 Gegenseitige Überwachung

Das Kernstück der Cheat-Erkennung ist die gegenseitige Überwachung der Teilnehmer. Jeder Teilnehmer, der Ereignisnachrichten oder Positionsmeldungen empfängt, überprüft diese auf mögliches Cheating. In zonenlosen Protokollen wird ein Teilnehmer also durch alle seine Nachbarn kontrolliert, in zonenbasierten Protokollen durch alle Teilnehmer der Zone und den Zonenkoordinator.

Um die Cheat-Erkennung möglichst umfassend und erweiterbar zu gestalten, wird die Überprüfung der Nachrichten auf eventuelles Cheaten von externen Modulen übernommen. Diese können verschiedene Erkennungsverfahren einsetzen und beliebig kombiniert werden. Mögliche Module sind dabei einfache Plausibilitätsprüfungen, die einfache Cheats wie das Ignorieren von Hindernissen oder das Überschreiten der maximalen Bewegungsgeschwindigkeit erkennen können, bis hin zu den in Abschnitt 7.2 behandelten verhaltensbasierten Erkennungsmechanismen, die auch komplexere Cheats wie das Einsetzten von *Bots* detektieren können.

Wird durch ein Modul der Cheat-Erkennung ein möglicher Regelbruch eines Teilnehmers erkannt, wird die Vertrauensstelle von diesem Ereignis informiert. Diese

verringert gemäß der in Abschnitt 7.5 beschriebenen Algorithmen den Vertrauenswert des angeklagten Teilnehmers. Dabei wird auch protokolliert, wie viele Teilnehmer das jeweilige Fehlverhalten melden. Trifft nur eine einzelne Anschuldigung ein, kann es sich dabei um eine versehentliche Fehlanschuldigung oder eine böswillige Falschanschuldigung handeln. Falls von einem Teilnehmer eine signifikante Zahl an Falschanschuldigungen eintrifft, wird dies als Cheatversuch des Anklägers betrachtet und dessen Vertrauenswert dementsprechend gesenkt.

Ein grundsätzliches Problem bei der gegenseitigen Überwachung in virtuellen Welten ist, dass Spieler, die in der Spielwelt nahe beieinander stehen, oft entweder gemeinsame oder entgegengesetzte Interessen haben. So bilden Spieler in virtuellen Welten oft Gruppen, die gemeinsam gewisse Ziele verfolgen. Diese werden möglicherweise Cheating seitens ihrer Mitspieler ignorieren und nicht an die Vertrauensstelle melden. Andererseits stehen Spieler in virtuellen Welten oft in einer Konkurrenz- oder Konfliktsituation zueinander. Sind Spieler Gegner in einem virtuellen Kampf, kann ein Spieler durch Falschanschuldigungen versuchen, seinen Konfliktgegner aus dem Spiel ausschließen zu lassen.

Um diese Interessenkonflikte zu verringern, wird jedem Spieler von der Vertrauensstelle ein *Beobachter* zugewiesen. Ein Spieler ist verpflichtet, seinem Beobachter alle Ereignisnachrichten und Positionsmeldungen zuzusenden. Das Nichtzusenden von Nachrichten wird als Cheatversuch gewertet.

Die Beobachter sollten von der Vertrauensstelle so gewählt werden, dass Interessenkonflikte unwahrscheinlich sind. Da der Beobachter als neutral eingeschätzt wird, haben Cheat-Anschuldigungen des Beobachters einen höheren Stellenwert als Cheat-Anschuldigungen anderer Teilnehmer. Bei der Zuweisung des Beobachters wird auch der Vertrauenswert der Teilnehmer berücksichtigt. Spieler mit geringem Vertrauenswert werden durch vertrauenswürdige Teilnehmer beobachtet. Damit keine dauerhafte Beziehung zwischen einem Teilnehmer und seinem Beobachter entstehen kann, wird in probabilistischen Zeitabständen jedem Teilnehmer ein neuer Beobachter zugewiesen. Wenn sich der Vertrauenswert eines Teilnehmers stark ändert, kann auf diese Weise ein neuer Beobachter zugewiesen werden.

Da Cheater wissen, dass Anschuldigungen von Beobachtern stärker bewertet werden als Anschuldigungen anderer Spieler, besteht die Gefahr, dass sie als Angriff auf die Cheat-Erkennung Beobachtern andere Ereignisnachrichten oder Positionsmeldungen senden als den übrigen Teilnehmern. Dadurch könnten sie Cheats durchführen, die die anderen Teilnehmer beeinflussen, aber vom Beobachter nicht bemerkt werden können. Aus diesem Grund überprüft der Beobachter Nachrichten stichprobenartig mit einer gewissen Wahrscheinlichkeit. Hierzu lässt er sich von einem willkürlich ausgesuchten Nachbarn oder Zonenmitglied des beobachteten Teilnehmers die entsprechende Nachricht zusenden. Sind die Nachrichten nicht identisch, wird dies als Cheatversuch gewertet.

Da ein Cheater dem Beobachter eine Folge von Nachrichten senden muss, die den Regeln der virtuellen Welt entsprechen, er mit dem Cheating gegenüber den anderen Teilnehmern jedoch eine Zustandsänderung zu seinen Gunsten hervorgerufen hat, ist es für den Cheater nicht einfach, den Zustand des Beobachters an den Zustand der anderen Teilnehmer anzugleichen, ohne seinen Vorteil wieder zu verlieren. Die Zustandsunterschiede zwischen Beobachter und anderen Teilnehmern müssen also aus

der Sicht des Cheaters eine gewisse Zeit aufrechterhalten werden, um ein erfolgreiches Cheating vertuschen zu können. Vergleicht der Beobachter in diesem Zeitraum eine von ihm empfangene Nachricht mit der von einem Nachbarn des Cheaters empfangenen Nachricht, kann er die Täuschung aufdecken.

Die Wahrscheinlichkeit einer Überprüfung der Nachrichten durch den Beobachter ist vom Vertrauenswert des beobachteten Teilnehmers abhängig. Je niedriger der Vertrauenswert, desto häufiger werden die an den Beobachter versendeten Nachrichten mit den an andere Teilnehmer versendeten Nachrichten abgeglichen. Damit haben Cheater nur wenig Zeit, die Täuschung des Beobachters aufzuheben.

7.7 Bestrafung von Cheatern

Um die virtuelle Welt möglichst frei von Cheatern zu halten und Teilnehmer der Welt vom Cheatern abzuschrecken, ist das Bestrafen von überführten Cheatern notwendig. Die wirksamste Bestrafung ist hier die Entfernung des Cheaters aus der Welt. Die Strafe kann dabei zeitlich begrenzt sein oder im Falle besonders böswilliger Cheats beziehungsweise bei wiederholtem Fehlverhalten dauerhaft gelten. In dem hier vorgestellten System lässt sich dies durch das Widerrufen des Tickets und der Verweigerung der Ticketausstellung an überführte Cheater erreichen.

Wichtig ist jedoch, die irrtümliche Bestrafung ehrlicher Teilnehmer zu minimieren. Deswegen sollte ein Teilnehmer nur bestraft werden, wenn er von einem besonders vertrauenswürdigen Teilnehmer unmittelbar beim Cheaten beobachtet wurde.

Die Vertrauensstelle initiiert einen Ausschluss eines Teilnehmers durch einen Ticketrückruf dann, wenn die folgenden zwei Bedingungen erfüllt sind:

- Der Vertrauenswert des Teilnehmers ist niedriger als ein festgelegter Schwellenwert v^-.
- Der Beobachter des Teilnehmers ist besonders vertrauenswürdig und stellt einen Regelverstoß des Teilnehmers fest.

Ein Beobachter gilt als besonders vertrauenswürdig, wenn sein Vertrauenswert v einen Schwellenwert v^+ überschreitet. Da der Vertrauenswert logarithmisch mit der Zeit ohne Cheatanschuldigung und somit sehr langsam wächst, hat sich ein Teilnehmer mit einem hohen Vertrauenswert über einen besonders langen Zeitraum als ehrlich erwiesen. Bei geeigneter Wahl von v^+ kann somit davon ausgegangen werden, dass es sich um einen ehrlichen Teilnehmer der virtuellen Welt handelt.

Cheat-Anschuldigungen von dritten Teilnehmern wirken sich somit nur auf den Vertrauenswert eines Teilnehmers aus. Sie können jedoch nicht direkte Ursache der Bestrafung eines Teilnehmers sein. So wird verhindert, dass eine Gruppe böswilliger Teilnehmer einen ehrlichen Spieler so lange mit Falschanschuldigungen bei der Vertrauensstelle anzeigt, bis dieser aus dem Spiel entfernt wird. Es bleibt also der in Abschnitt 7.5 aufgestellte Grundsatz gewahrt, nach dem ein niedriger Vertrauenswert kein Beweis für Cheating ist.

Da mutmaßlichen Cheatern stets ein besonders vertrauenswürdiger Beobachter zugewiesen wird, kann andererseits sichergestellt werden, dass cheatende Spieler schnell

aus dem Spiel entfernt werden: Bei Teilnehmern mit einem Vertrauenswert unterhalb des Schwellenwertes v^- reicht eine einzelne Cheat-Anschuldigung des Beobachters, um eine Bestrafung zur Folge zu haben.

In Verbindung mit kryptographischen Signaturen (siehe dazu Abschnitt 7.9) sind einige Cheats sicher nachzuweisen: So kann beispielsweise das Ignorieren eines Hindernisses mit zwei signierten aufeinanderfolgenden Positionsmeldungen bewiesen werden, wenn sich der Cheater in der ersten Nachricht auf der einen und in der zweiten Nachricht auf der anderen Seite des Hindernisses befindet, ein ordnungsgemäßes Passieren des Hindernisses in der Zeit zwischen den Nachrichten nach den Regeln des Spiels jedoch unmöglich ist. Kann ein Teilnehmer so das Cheating eines anderen Teilnehmers beweisen, kann von der Vertrauensstelle der Vertrauenswert des Cheaters unmittelbar auf den für einen Ausschluss nötigen Schwellenwert gesenkt werden. Weiterhin kann die Vertrauensstelle bei bewiesenen Cheats ohne die sonst nötige Beobachtung durch einen vertrauenswürdigen Beobachter eine Bestrafung eingeleiten.

7.8 Kontrolle der Spielfigur

Bislang nicht betrachtet wurde die Möglichkeit eines Cheaters, durch Manipulation lokal gespeicherter Daten seine Spielfigur regelwidrig zu verändern. Um dies verhindern zu können, muss der Zustand der Spielfigur sicher gespeichert werden und jede Änderung des Zustands überprüft werden.

Wenn ein Spieler nicht in die virtuelle Welt eingeloggt ist, ist es für eine sichere Speicherung ausreichend, den Zustand durch die Autorisierungsstelle zu signieren. Dadurch können Änderungen durch den Spieler einfach erkannt werden.

Wenn der Spieler in der virtuellen Welt aktiv ist, sind jedoch häufige Zustandsänderungen nicht ausgeschlossen. Diese Änderungen fortlaufend durch die Autorisierungsstelle signieren zu lassen, würde eine zu große Belastung dieser Stelle darstellen.

Stattdessen wird dem Beobachter eines Spielers bei der Zuweisung eine signierte Kopie des Zustandes zugestellt. Da der Beobachter alle Ereignisnachrichten des Spielers erhält, kann er alle Änderungen des Zustands nachvollziehen. Wird der Beobachter durch einen anderen Teilnehmer ersetzt, reicht er den aktuellen Zustand an den neuen Beobachter weiter. So können alle Zustandsänderungen nachvollzogen werden. Entdeckt ein Beobachter eine Abweichung des von ihm errechneten Zustands gegenüber dem vom Spieler postulierten Zustand, kann er dieses als Cheatversuch an die Vertrauensstelle weiterleiten. Um den Ausfall eines Beobachters kompensieren zu können, kann der aktuelle Zustand auch mit Nachbarn oder Zonenmitgliedern des beobachteten Teilnehmers abgeglichen werden.

Wenn sich ein Teilnehmer ausloggt, wird der aktuelle Zustand von der Autorisierungsstelle signiert und kann so bis zum nächsten Einloggen des Spielers von diesem nicht mehr geändert werden.

Die hier beschriebenen Mechanismen lassen sich nutzen, um den in Abschnitt 6.7.1 beschriebenen *Location Cache* zu implementieren: Die aktuelle Position ist Teil des Zustands der Spielfigur. Bei jeder Übergabe von einem Beobachter zum nächsten kann diese in einer zentralen oder verteilten Datenbank gesichert werden. Mit dieser Datenbank lässt sich beim Knotenbeitritt ein Teilnehmer ermitteln, dessen Position in der Nähe der gewünschten Beitrittsposition ist. Mit dieser Information lässt sich der Knotenbeitritt beschleunigen.

7.9 Kryptographische Absicherung

Das in Abschnitt 7.3 beschriebene Angreifermodell sieht vor, dass die hier betrachteten Cheater keinen Zugriff auf den Datenverkehr zwischen anderen Teilnehmern haben. Demnach ist es als Minimalsicherung ausreichend, die von der Autorisierungsstelle ausgestellten Tickets kryptographisch zu signieren und an die IP-Adresse des jeweiligen Teilnehmers zu binden. Durch die Verwendung von TCP als Transportschichtprotokoll[4] oder durch einen einfachen *Return Routability*-Test [72, 111], bei dem an den vorgeblichen Absender einer Nachricht eine Rückfrage zur Bestätigung der Nachricht gesendet wird, kann das Vortäuschen falscher Identitäten unterbunden werden.

In der Realität kann es jedoch vorkommen, dass Teilnehmer beispielsweise über ungesicherte Funknetzwerke auf die virtuelle Welt zugreifen. Befindet sich ein Angreifer in demselben Funknetzwerk, kann er durch Fälschen seiner Absenderadresse oder über *Man-in-the-Middle*-Angriffe die Identität dieses Teilnehmers annehmen. Die kryptographische Sicherung aller Nachrichten zwischen Teilnehmern, der Vertrauens- und der Autorisierungsstelle erlaubt eine Ausweitung des Angreifermodells auf derartige Angreifer.

Hierzu wird das Ticket eines Teilnehmers von der Autorisierungsstelle nicht mehr an die IP-Adresse gebunden, sondern an den öffentlichen Schlüssel des Teilnehmers. Das derart modifizierte Ticket hat folgendes Format: $H()$ ist eine sichere kryptographische Hashfunktion, $S_k()$ ist die kryptographische Signatur mit dem Schlüssel k, k^c der private Schlüssel der Autorisierungsstelle, t_0 der Ausstellungszeitpunkt und t_{end} das Ende des Gültigkeitszeitraumes, k_A^p der öffentliche Schlüssel des Teilnehmers und ID_A der Identifikator des Teilnehmers.

$$\text{Ticket}(k_A^p, ID_A, t_0, t_{end}, S_{k^c}(H(k_A^p, ID_A, t_0, t_{end})))$$

Der Ticketinhaber signiert nun alle ausgehenden Nachrichten mit seinem privaten Schlüssel. Andere Teilnehmer können dann anhand des Tickets und des öffentlichen Schlüssels des Teilnehmers die Authentizität der Nachrichten prüfen.

Aus der Signierung aller Nachrichten ergibt sich ein weiterer Vorteil: Durch die Signatur sind die Nachrichten *unabstreitbar*. Damit können bei Cheat-Anschuldigungen die Nachrichten des mutmaßlichen Cheaters als Beweise verwendet werden. Gewisse Cheatklassen wie beispielsweise das Ignorieren von Hindernissen oder die Erhöhung der Bewegungsgeschwindigkeit sind durch die signierten Nachrichten des Cheaters leicht nachweisbar. Liegen der Vertrauensstelle unabstreitbare Beweise für das Fehlverhalten eines Teilnehmers vor, kann unmittelbar eine Bestrafung initiiert werden, ohne auf die sonst nötige direkte Beobachtung eines Cheats durch den Beobachter warten zu müssen.

Zusätzlich erschwert das Signieren von Nachrichten das Täuschen des Beobachters durch das Versenden unterschiedlicher Nachrichten an Beobachter und andere Teilnehmer. Sind die Nachrichten nicht signiert, muss der Beobachter seine Nachrichten

[4] Bei der Verwendung von TCP kann es aufgrund der benötigten Zeit für den 3-Wege-Handshake oder aufgrund von Sendewiederholungen zu erhöhten Latenzen kommen. Dennoch setzen einige virtuelle Welten, darunter „World of Warcraft“, TCP als Transportschichtprotokoll ein.

mit verschiedenen anderen Teilnehmern abgleichen, da für den Beobachter nicht zu unterscheiden ist, ob der von ihm beobachtete Teilnehmer oder der zum Vergleich der Nachrichten befragte Teilnehmer für eine Differenz zwischen den Nachrichten verantwortlich ist. Sind die Nachrichten hingegen signiert, reicht der Vergleich mit einer einzelnen Nachricht, um bei einer Differenz zwischen den Nachrichten den Verursacher der Differenz zweifelsfrei feststellen zu können. Ansonsten verläuft die Überprüfung der Nachrichten weiterhin wie in Abschnitt 7.6 beschrieben.

7.10 Integration in QuON

In den vorigen Abschnitten wurden die wesentlichen Mechanismen einer verteilten Cheat-Erkennung für virtuelle Welten vorgestellt. Die vorgestellten Mechanismen können Cheating auf der Anwendungsebene erkennen und bestrafen. Die Erkennung von Cheating auf der Protokollebene ist protokollspezifisch. Die für das im vorigen Kapitel vorgestellte Protokoll QuON benötigten Erweiterungen für eine Erkennung wesentlicher Cheats auf Protokollebene werden in den folgenden Abschnitten erläutert.

7.10.1 Überprüfung der Verbindungsnachbarn

Wie in Abschnitt 7.6 erläutert, ist es für eine effektive Cheat-Erkennung notwendig, dass der Beobachter Nachbarn des beobachteten Teilnehmers kennt, um gegebenenfalls die Konsistenz der ausgesendeten Nachrichten prüfen zu können. In zonenbasierten Protokollen kann dies über die Mitgliederliste der betreffenden Zone geschehen. Da in QuON keine solche Liste zur Verfügung steht, muss jeder Spieler seinem Beobachter zusätzlich zu seiner Position die Liste seiner Verbindungsnachbarn und temporären Nachbarn zusenden. Der in Abschnitt 6.5 beschriebene Algorithmus zur Versendung von Positionsmeldungen wird hierzu wie in 7.1 dargestellt erweitert. Somit werden dem Beobachter im Regelfall mindestens vier Nachbarn genannt, die für die Prüfung der vom Teilnehmer ausgesendeten Nachrichten verwendet werden können.

Es reicht allerdings nicht aus, sich auf die vom Teilnehmer versendeten Daten zu verlassen. Eine kooperierende Gruppe aus fünf Cheatern könnte ansonsten ihre Beobachter täuschen, indem sie sich gegenseitig als Verbindungsnachbarn ausgeben und sich bei Anfragen der Beobachter gegenseitig decken.

Diese Täuschung lässt sich aufdecken, indem sich die tatsächlichen Verbindungsnachbarn aktiv beim Beobachter melden. Hierzu fragen sie die Vertrauensstelle nach der Adresse des Beobachters eines ihrer Verbindungsnachbarn oder temporären Nachbarn. Dem so ermittelten Beobachter senden sie nun eine Nachricht des Verbindungsnachbarn oder temporären Nachbarn, der ihre Rolle beweist. Ein solcher Beweis kann durch eine Positionsmeldung des betreffenden Nachbarn erfolgen, die eine Liste von Verbindungsnachbarn und temporären Nachbarn enthält. Stellt der Beobachter fest, dass der beobachtete Teilnehmer den Absender der Nachricht nicht als Verbindungsnachbarn oder temporären Nachbarn gemeldet hat, ist dies als Cheatversuch zu werten.

Die aktive Benachrichtigung des Beobachters durch die Nachbarn wird ergänzend zu der in Abschnitt 7.6 beschriebenen vom Beobachter initiierten Überprüfung verwendet. Zwar ließe sich die Überprüfung auch ausschließlich über die aktive Benachrichtigung durch Nachbarn realisieren, aber um eine vom Vertrauenswert abhängige

Vorbedingung: $\mathcal{T}$: Menge der A bekannten Teilnehmer
Vorbedingung: O: Der Beobachter von Teilnehmer A
procedure SendePositionsmeldungen($\mathcal{T}$)
$\mathcal{N} \leftarrow$ BestimmeDirekteNachbarn($\mathcal{T}$)
$\mathcal{V} \leftarrow$ BestimmeVerbindungsnachbarn($\mathcal{T}$)
$\mathcal{S} \leftarrow$ BestimmeTemporäreNachbarn($\mathcal{T}, \mathcal{V}$)
/* Position an alle Nachbarn senden. */
for all $n \in \mathcal{N}$ **do**
send MOVE$(A, p, r^{AoI}) \mapsto n$
end for
/* An Beobachter, Verbindungs- und temporäre Nachbarn zusätzlich $\mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}$ senden, an Verbindungsnachbarn: Verbindungsnachbarstatus melden. */
for all $t \in \mathcal{V}$ **do**
send MOVE$(A, p, r^{AoI}, \mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}$,Verbindungsnachbaranfrage$) \mapsto t$
end for
for all $t \in \mathcal{S} \cup \{O\}$ **do**
send MOVE$(A, p, r^{AoI}, \mathcal{V}, \mathcal{P}_\mathcal{V}, \mathcal{S}, \mathcal{P}_\mathcal{S}) \mapsto t$
end for
$\mathcal{T} \leftarrow \mathcal{N} \cup \mathcal{V} \cup \mathcal{S}$
end procedure

Algorithmus 7.1 Versenden von Positionsmeldungen

Wahrscheinlichkeit der Überprüfung zu ermöglichen, müssten dann die Nachbarn den Vertrauenswert des beobachteten Teilnehmers kennen. Dies ist jedoch nicht wünschenswert, da sonst ein Cheater von einem kooperierenden Nachbarn stets über seinen aktuellen Vertrauenswert informiert werden könnte und sein Cheaten so steuern könnte, dass er den für die Bestrafung relevanten Schwellenwert v^- nicht unterschreitet.

7.10.2 Verweigerung der Nachbarschaftsfindung

Ein möglicher Cheat auf der Protokollebene ist das Verweigern der Ausführung von Protokollaufgaben. In QuON betrifft dies im Wesentlichen die Nachbarschaftsfindung. Da in QuON die Benachrichtigung über neue Nachbarn üblicherweise nicht nur von einem einzelnen Teilnehmer versendet wird, ist der Einfluss dieses Cheats relativ gering. Zudem lässt er sich einfach detektieren, da ein Teilnehmer, der von einigen seiner Nachbarn eine solche Benachrichtigung erhält anhand dieser Nachrichten überprüfen kann, ob einer seiner Nachbarn die Benachrichtigung unterdrückt hat.

Allerdings kann die Unterdrückung der Benachrichtigungen nur von den Nachbarn festgestellt werden. Der Beobachter hingegen kann dieses Verhalten nicht unmittelbar verifizieren. Somit kann gemäß der in Abschnitt 7.7 gegebenen Kriterien zunächst kein Ausschluss des Cheaters stattfinden.

Um eine Verifikation der Cheatanschuldigungen durch den Beobachter zu ermöglichen, kann die Vertrauensstelle den Beobachter anweisen, der virtuellen Welt in der Nähe des mutmaßlichen Cheaters beizutreten. Der Beobachter nimmt dort wie ein normaler Spieler am Protokollablauf teil. Er begibt sich in die Nachbarschaft

des mutmaßlichen Cheaters und kann so dessen Protokollkonformität unmittelbar überprüfen. Da dieses Verfahren für den Beobachter deutlich aufwändiger ist als die üblichen Prüfungsaufgaben, wird es von der Vertrauensstelle nur initiiert, wenn der Vertrauenswert des mutmaßlichen Cheaters aufgrund einer Vielzahl von Anschuldigungen unter einem Schwellenwert v^{--} liegt. Dieser Schwellenwert liegt unterhalb des normalerweise für einen Ausschluss ausreichenden Schwellenwert v^{-}. Das hier beschriebene Überprüfungsverfahren kann bei allen Cheats angewendet werden, die sich durch Nachbarn des Cheaters, jedoch nicht direkt durch den Beobachter, nachvollziehen lassen können.

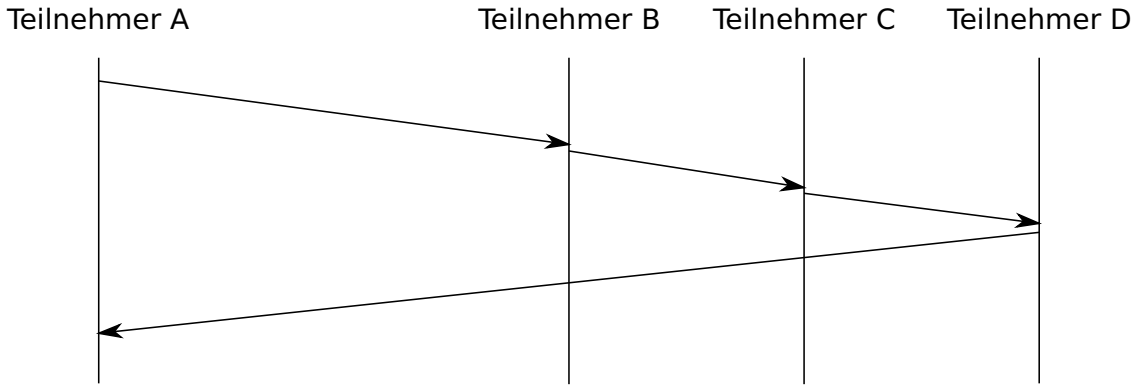
7.10.3 Sicheres Login

Eine weitere Cheatmöglichkeit auf Protokollebene betrifft den Beitrittsalgorithmus von QuON. Hier wird, wie in Abschnitt 6.7.1 beschrieben, die Beitrittsanfrage eines Teilnehmers rekursiv weitergeleitet, bis sie bei dem Teilnehmer angelangt ist, dessen Position der gewünschten Beitrittsposition am nächsten ist. Das Verfahren ist in Abbildung 7.1a visualisiert. Teilnehmer A will dem Netz beitreten. Seine Anfrage wird über Teilnehmer B und C weitergeleitet, bis sie bei Teilnehmer D, der sich am nächsten zu der gewünschten Beitrittsposition befindet, angelangt ist. Teilnehmer D sendet A daraufhin eine Bestätigung mit einer vorläufigen Nachbarsliste. Wenn sich ein Cheater auf dem Pfad der Nachricht befindet, kann er die Beitrittsanfrage verwerfen und so den Beitritt eines neuen Teilnehmers verhindern.

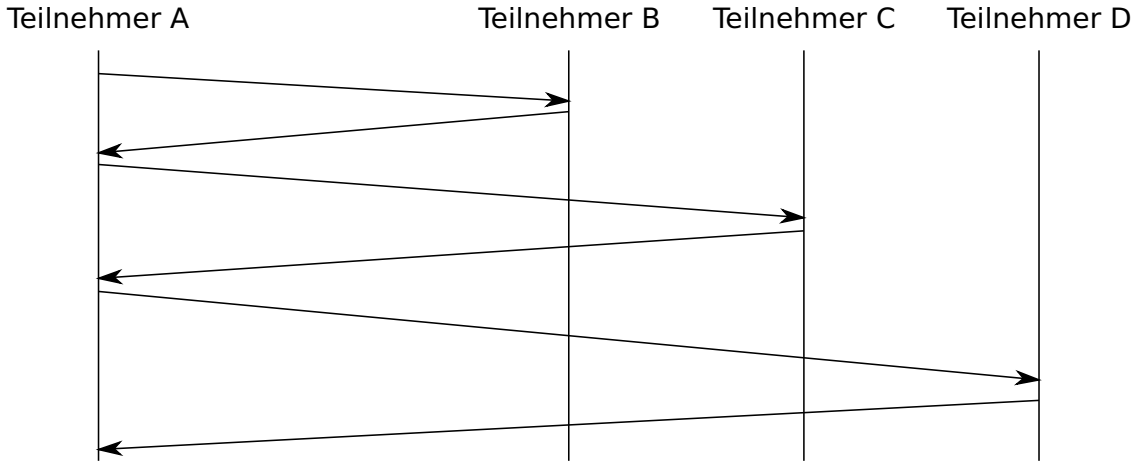
Um diesen Cheat erkennen zu können, kann der Beitrittsalgorithmus iterativ durchgeführt werden. Hierbei wird einem beitretenden Teilnehmer auf seine Anfrage mit der Adresse des nächsten Teilnehmers, an den die Anfrage zu richten ist, geantwortet. Der beitretende Teilnehmer kontaktiert diesen daraufhin und sendet wiederum die Beitrittsanfrage. Das Verfahren wird solange wiederholt, bis der beitretende Teilnehmer die Adresse des der Beitrittsposition nächstgelegenen Teilnehmers kennt. Von diesem erhält er dann, wie bei dem ursprünglichen Beitrittsalgorithmus, eine vorläufige Nachbarsliste, um dem Netz beitreten zu können. Diese Login-Variante ist in Abbildung 7.1b visualisiert.

Dieses Verfahren hat den Vorteil, dass ein Teilnehmer jeden Zwischenschritt im Netz kennt. Somit kann er nichtkooperative Teilnehmer, die eine Antwort auf seine Beitrittsanfrage verweigern, eindeutig identifizieren. Diese kann er dann der Vertrauensstelle als mutmaßliche Cheater melden. Allerdings kann dieser Cheat nicht unmittelbar durch einen Beobachter verifiziert werden und somit auch kein Netzausschluss des Cheaters erreicht werden. Zusätzlich wird der Cheat durch das Verfahren nicht verhindert; der neu hinzukommende Teilnehmer bleibt aus dem Netz ausgeschlossen.

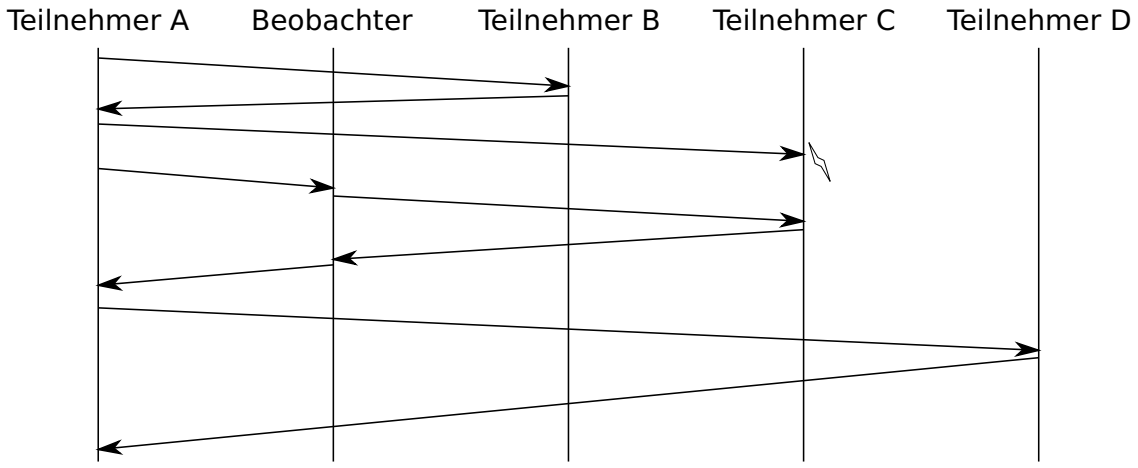
Deswegen wird der beitretende Spieler in einer solchen Situation von der Vertrauensstelle bei Beschwerdeeinreichung an den Beobachter des mutmaßlichen Cheaters verwiesen. Dieser fragt dann in Vertretung des beitretenden Spielers bei dem mutmaßlichen Cheater an. Verweigert dieser die Antwort, ist der Cheat für den Beobachter offensichtlich, und der Cheater kann aus dem Netz ausgeschlossen werden. Antwortet der mutmaßliche Cheater hingegen, leitet der Beobachter die Antwort an den beitretenden Spieler weiter. In diesem Fall wird der Cheat verhindert, da der mutmaßliche Cheater den Beitritt des neuen Teilnehmers nicht mehr behindern kann. Dieses Verfahren ist in Abbildung 7.1c dargestellt. Teilnehmer C ist in der Abbildung der mutmaßliche Cheater, der die initiale Anfrage von Teilnehmer A nicht beantwortet.



(a) Rekursives Login.



(b) Iteratives Login.



(c) Iteratives Login über Beobachter.

Abbildung 7.1 Verschiedene Login-Verfahren für QuON.

7.10.4 Ausschluss von Cheatern mit Ticketrückruf

In Abschnitt 7.4 wird beschrieben, wie Cheater durch einen Ticketrückruf der Vertrauensstelle aus der virtuellen Welt entfernt werden können. Bei zonenbasierten Protokollen geschieht dies einfach durch Publizierung des Ticketrückrufs über den Zonenkoordinator. In QuON gibt es diese Möglichkeit nicht. Für eine vollständige Entfernung des Cheaters aus dem Netz müssen alle Nachbarn die Verbindung zu dem Cheater abbrechen.

Hierzu leitet die Vertrauensstelle den Ticketrückruf an den Beobachter. Dieser wiederum leitet den Rückruf an alle ihm bekannte Verbindungsnachbarn und temporären Nachbarn des Cheaters weiter. Diese brechen die Verbindung mit dem Cheater ab und leiten den Ticketrückruf an alle ihnen bekannten Nachbarn weiter. Auch diese brechen die Verbindung zu dem Cheater ab und leiten den Rückruf an alle ihre Nachbarn weiter. Mit dieser Weiterleitung ist nun mit hoher Wahrscheinlichkeit das Interessengebiet des Cheaters vollständig abgedeckt. Die Empfänger müssen also nur noch die Verbindung zu dem Cheater abbrechen, ohne den Ticketrückruf weiterzuleiten. Der Cheater kennt nun keine Teilnehmer in der Umgebung mehr, die mit ihm zu kommunizieren bereit sind, und ist somit aus dem Netz ausgeschlossen.

7.11 Evaluierung

Um die Effektivität der vorgestellten Cheat-Erkennngsmaßnahmen zu evaluieren, wurden sie in OverSim simuliert. Dabei wurden die Cheater-Erkennungsrate und die Zeit zum Erkennen der Cheater gemessen. Zusätzlich wurde der Bandbreitenbedarf der Maßnahmen bestimmt.

7.11.1 Implementierung des Prototypes

In der Implementierung des Cheat-Erkennungssystems wurde der Fokus auf die Autorisierung (siehe Anschnitt 7.4), das Vertrauenssystem (siehe Abschnitt 7.5), die gegenseitige Überwachung (siehe Abschnitt 7.6) und die Bestrafung der erkannten Cheater (siehe Abschnitt 7.7) gelegt. Aus diesem Grund wurden komplexe verhaltensbasierte Erkennungsmechanismen, wie sie beispielsweise für die Erkennung von *Botting* nötig wären, nicht implementiert. Stattdessen wurden die Teilnehmer mit einer einfachen Plausibilitätsprüfung ausgestattet, die regelwidrige Bewegungen erkennen kann. Weiterhin ist der in Abschnitt 7.6 beschriebene Abgleich der vom Beobachter empfangenen Nachrichten mit den von den Nachbarn des beobachteten Teilnehmers empfangenen Nachrichten implementiert, um Angriffe auf die Cheat-Erkennung detektieren zu können. Die implementierten Erkennungsmodule sind ausreichend, um die Effektivität der Überwachungs- und Bestrafungsmechanismen zu testen.

Die in Abschnitt 7.9 beschriebene kryptographische Sicherung der Nachrichten durch Signaturen wurde ebenfalls nicht implementiert. Dadurch können in der Simulation keine Cheats von anderen Teilnehmern bewiesen werden; Strafen können ausschließlich auf Basis der Vertrauenswerte und der Überwachung durch vertrauenswürdige Beobachter ausgesprochen werden. Die in den Simulationen beobachteten Cheat-Erkennungsraten und Cheat-Erkennungzeiten könnten durch den Einsatz von Signaturen verbessert werden. In den untersuchten Szenarien könnte der Cheater gemäß des

in Abschnitt 7.7 beschriebenen Verfahrens mit einer einzelnen Cheatanschuldigung durch Beilegung der vom Cheater signierten Positionsmeldungen überführt und bestraft werden. Der Verzicht auf Signaturen in der Evaluierung hat den Vorteil, dass die Evaluierungsergebnisse auch auf komplexere Cheatverhalten übertragbar sind, die auch beim Einzatz von Signaturen nicht beweisbar sind.

Die Autorisierungsstelle und die Vertrauensstelle sind in der Implementierung als zentrale Dienste angelegt. Die Evaluierung des Bandbreitenbedarfs dieser Stellen zeigt, dass diese Dienste in den untersuchten Szenarien nur eine geringe Last zu tragen haben und somit die Skalierbarkeit von QuON nur wenig einschränken. Die Einführung einer verteilten Autorisierungs- und Vertrauensstelle ist dennoch eine sinnvolle mögliche Erweiterung der Cheat-Erkennung, da bei einem vollständig verteilten Dienst der Betrieb der Cheat-Erkennung nicht durch den Ausfall eines einzelnen zentralen Servers beeinträchtigt werden kann.

7.11.2 Metriken

Das wichtigste Kriterium für die Beurteilung eines Cheat-Erkennungssystems ist die *Cheater-Erkennungsrate*. Diese misst den Anteil erkannter Cheater an der Menge aller Cheater. Ein Cheat-Erkennungssystem kann nur dann als gut befunden werden, wenn der größte Teil der Cheater vom System erkannt werden.

Weiterhin ist die *Cheat-Dauer bis zur Entdeckung* relevant. Hierzu wird bei Cheatern die Zeit vom Beginn des Cheatens bis zu der Erkennung des Cheaters durch das Cheat-Erkennungssystem gemessen. Diese Zeit sollte möglichst gering sein, um Cheatern möglichst wenig Zeit zu geben, Schaden in der virtuellen Welt anzurichten. Eine lange Cheat-Dauer beeinflusst auch die Erkennungsrate: Verlässt der Cheater vor der Erkennung das Netz oder stellt er das Cheating vor der Erkennung ein, sinkt die Cheat-Erkennungsrate. Dies lässt sich nur durch eine schnelle Erkennung verhindern.

Die *Fehlerrate* misst den Anteil an fälschlicherweise wegen Cheating ausgeschlossenen ehrlichen Spielern. Die Bestrafung ehrlicher Spieler ist so gering wie möglich zu halten, da sonst ein Verlust von Spielern und somit finanzielle Einbußen für den Betreiber der Welt zu erwarten sind.

Zuletzt ist auch der *Bandbreitenbedarf* der Cheat-Erkennung relevant. Gemessen wurden hier der Bandbreitenbedarf der Cheat-Erkennung in Senderichtung[5] sowie der Anteil der Cheat-Erkennung am gesamten Bandbreitenbedarf. Der für die Cheat-Erkennung benötigte Datenverkehr sollte nur einen geringen Teil des insgesamt für die virtuelle Welt benötigten Bandbreitenbedarfs ausmachen.

7.11.3 Untersuchtes Cheat-Verhalten

Als erstes Cheat-Verhalten wurde exemplarisch für ein *dauerhaftes Cheating*, bei dem sich der Cheater durchgängig regelwidrig verhält, die *Erhöhung der Laufgeschwindigkeit* untersucht. Cheater, die in den Simulationen dieses Cheat-Verhalten zeigten, überschritten die maximale regelgerechte Geschwindigkeit um 10%.

[5] Wie in Abschnitt 5.2.1 erläutert, ist dies aufgrund der meist asymmetrischen Endkundenanbindungen an das Internet der im Vergleich zur Empfangsrichtung relevantere Bandbreitenbedarf.

Als Beispiel für ein *sporadisches Cheat-Verhalten* wurde das *Ignorieren von Hindernissen* gewählt. Hier bewegt sich der Cheater die meiste Zeit regelkonform. Von Zeit zu Zeit ignoriert er jedoch Hindernisse, statt diese ordnungsgemäß zu umlaufen.

Als Beispiel für einen *Angriff auf die Cheaterkennung* wurde die *Täuschung des Beobachters* simuliert. Dabei sendet der Cheater an den Beobachter falsche Positionsmeldungen. Wird dieser Angriff auf die Cheaterkennung nicht erkannt, kann damit Cheating vertuscht werden. Um einen Vorteil für den Cheater zu erreichen, muss dieser Angriff mit einem weiteren Cheat-Verhalten kombiniert werden. In der Simulation sollte jedoch die Erkennung dieser Art der Täuschung unabhängig von weiterem Cheating untersucht werden, sodass sich der Cheater in der Simulation ansonsten regelgerecht in der virtuellen Welt bewegt. Da durch das Fehlen weitergehender Cheats eine Erkennung des Cheaters erschwert wird, handelt es sich hierbei um eine worst-case-Betrachtung dieser Angriffsform.

Bei der Evaluierung wurde die in Abschnitt 7.9 beschriebene kryptographische Signatur der Positionsmeldungen nicht vorgenommen. Cheating ließ sich also in den Simulationen nicht zweifelsfrei beweisen. Somit muss ausschließlich auf die in Abschnitt 7.7 beschriebene Erkennung zurückgegriffen werden, die auf unmittelbaren Beobachtungen vertrauenswürdiger Beobachter basiert. Durch die Verwendung von Signaturen würden die Erkennungsraten steigen und die Erkennungszeiten sinken.

7.11.4 Einflussgrößen

Eine wichtige Einflussgröße der Cheat-Erkennung in Verbindung mit QuON ist die durchschnittliche Zahl an direkten Nachbarn eines Teilnehmers der virtuellen Welt. Je mehr Nachbarn ein Teilnehmer besitzt, um so mehr Teilnehmer können das Verhalten des Teilnehmers beurteilen und gegebenenfalls als Cheating identifizieren. Eine höhere Anzahl an Nachbarn sollte also die Cheat-Erkennung beschleunigen und die Cheat-Erkennungsrate steigern. In den Simulationen wurde die Zahl der Nachbarn wie auch bei der Evaluierung bisheriger Protokolle in Abschnitt 5.2.2 durch die Bildung von Gruppen gesteuert. Je größer die Gruppe ist, in der sich ein Teilnehmer bewegt, desto größer ist die durchschnittliche Zahl seiner Nachbarn. Die in den Simulationen verwendeten maximalen Gruppengrößen waren 1 (keine Gruppenbildung), 20 und 40.

Die zweite untersuchte Einflussgröße ist der durchschnittliche Anteil von Cheatern an der Teilnehmerzahl der virtuellen Welt. Da Cheater nicht aktiv an der Cheat-Erkennung mitarbeiten und diese gegebenenfalls zu stören versuchen, ist zu erwarten, dass ein höherer Cheater-Anteil die Cheat-Erkennungszeiten erhöhen und eventuell die Cheat-Erkennungsrate senken kann. Realistische Cheater-Anteile sind stark vom jeweiligen Spiel und der Zielgruppe des Spiels abhängig. Die öffentlichen Statistiken eines Anti-Cheat-Programms der Online-Spieleplattform „Steam“ weisen einen Cheateranteil von knapp unter 9% aus [138]. In Umfragen der „European Sports League“, nach eigener Angabe die größte und älteste Online-Liga Europas, werden Cheater-Anteile von 2 bis 4,5% ermittelt [113]. In den hier betrachteten Szenarien werden deshalb zwei verschiedene Cheater-Anteile von 5% und 10% untersucht.

7.11.5 Szenarien

Die evaluierten Szenarien sind an die in Abschnitt 5.2.3 untersuchten Szenarien angelehnt. Für die Simulation der virtuellen Welt wurde wiederum der *SimpleGameClient*

verwendet. In der virtuellen Welt bewegten sich 500 Teilnehmer. Diese traten zu Beginn der Simulation vor dem Beginn von Messungen dem Netz bei. Anschließend blieben die Teilnehmer für die gesamte Simulationszeit Teil des Netzes, sofern sie nicht wegen Cheatings ausgeschlossen wurden. Um die Evaluierungen auf die Cheat-Erkennung zu konzentrieren, wurde Teilnehmerfluktuation nicht betrachtet.

Nach Abschluss der Initialisierung begannen einige Teilnehmer zu cheaten. Abhängig von der Wahl des Cheater-Anteils an der Teilnehmerzahl von 5% oder 10% zeigten im Simulationsverlauf 25 oder 50 Teilnehmer regelwidriges Verhalten. Die Cheater wurden zufällig aus der Menge der Teilnehmer ausgewählt. Dabei wurden die in Abschnitt 7.11.3 erläuterten Cheat-Varianten *Überschreitung der Maximalgeschwindigkeit*, *Ignorieren von Hindernissen* oder *Täuschen des Beobachters* eingesetzt. In jedem Simulationslauf wurde jeweils nur eines der Verhalten eingesetzt. Nachdem der letzte Teilnehmer mit dem Cheating begonnen hatte, wurde noch ein Zeitraum von 500 Sekunden simuliert, in der dieser Cheater von der Cheat-Erkennung überführt werden konnte. Nach Ablauf dieser Zeit wurde die Simulation beendet.

Das Interessengebiet der Teilnehmer war in diesen Simulationen ein Kreis mit einem Radius von 50 m um die Position des Teilnehmers. Die reguläre Bewegungsgeschwindigkeit betrug 5 m/s. Cheater, die das Cheat-Verhalten „Überschreitung der Maximalgeschwindigkeit“ zeigten, bewegten sich mit einer um 10% erhöhten Bewegungsgeschwindigkeit, also mit 5,5 m/s. Es wurden von allen Teilnehmern 6 Positionsmeldungen pro Sekunde generiert und an die Nachbarn sowie den Beobachter versendet.

Die Spielfeldgröße der virtuellen Welt betrug 1.000 m mal 1.000 m. Die Teilnehmerdichte wurde durch die Verwendung des *GroupRoaming*-Bewegungsmodells variiert. Dabei wurden Gruppengrößen von 1, also individuelle Bewegung der Teilnehmer ohne Gruppenbildung, 20 und 40 untersucht.

Für die Modellierung des unterliegenden Netzwerkes wurde wie bei den anderen Evaluierungen in dieser Arbeit das *SimpleUnderlay*-Modell gewählt. Alle Simulationen wurden mit 50 verschiedenen Seeds wiederholt. Die abgebildeten Ergebnisse sind die Mittelwerte dieser Durchläufe zusammen mit dem 99%-Konfidenzintervall. Die Simulationsparameter sind in Tabelle 7.1 zusammengefasst.

7.11.6 Ergebnisse

In den folgenden Abschnitten werden die Evaluierungsergebnisse dargestellt. Dabei werden die Metriken Cheat-Erkennungsrate, Cheat-Erkennungszeit, Fehlerrate und Bandbreitenbedarf betrachtet.

7.11.6.1 Cheat-Erkennungsrate

Abbildung 7.2 zeigt die Cheat-Erkennungsrate der verschiedenen Cheat-Verhalten in Abhängigkeit zu der Gruppengröße. Sowohl beim Cheat „Überschreitung der Maximalgeschwindigkeit“ (siehe Abbildung 7.2a) als auf beim Cheat „Ignorieren von Hindernissen“ (siehe Abbildung 7.2b) liegt die Erkennungsrate konstant bei 100%. Es werden also zuverlässig alle Cheater erkannt, die eines dieser Cheat-Verhalten zeigen. Cheater, die das Verhalten „Täuschen des Beobachters“ (siehe Abbildung 7.2c) zeigen, werden größtenteils erkannt. Bei einer Gruppengröße von 1 werden zwischen 98% und 99%, bei einer Gruppengröße von 40 werden alle Cheater erkannt. Der

Parameter	Wert
Simulationszeit	500 Sekunden nach Start Cheating letzer Cheater
Anzahl Seeds	50
Teilnehmerzahl	500
Spielfeldgröße	1 km*1 km
Bewegungsmodell	GroupRoaming
max. Gruppengröße	1, 20, 40
Cheatingverhalten	Überschreitung der Maximalgeschwindigkeit Ignorieren von Hindernissen Täuschen des Beobachters
Cheaterwahrscheinlichkeit	5%, 10%
Sitzungsmodell	NoChurn
Interessengebietsradius	50 m
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Vektornorm	euklidische Norm
Netzmodell	SimpleUnderlay

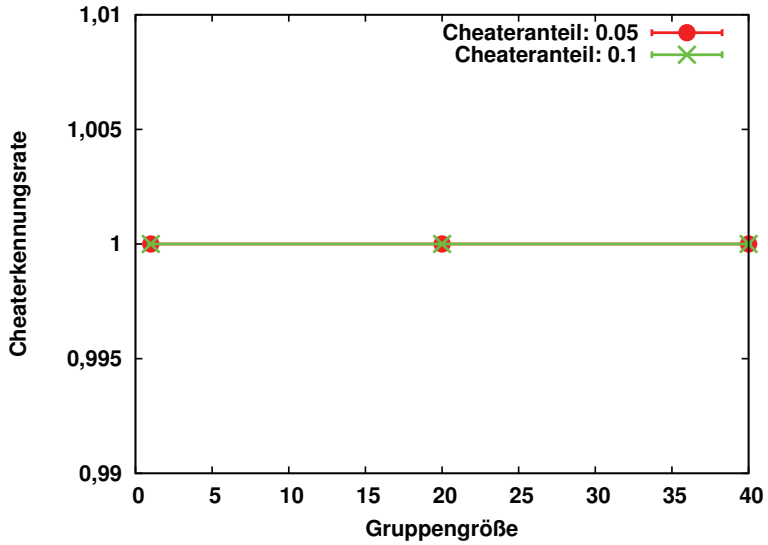
Tabelle 7.1 Simulationsparameter.

Cheater-Anteil hat dabei keinen statistisch signifikanten Einfluss auf die Ergebnisse. Die Ergebnisse zeigen, dass die verteilte Cheat-Erkennung effizient Cheater identifizieren kann.

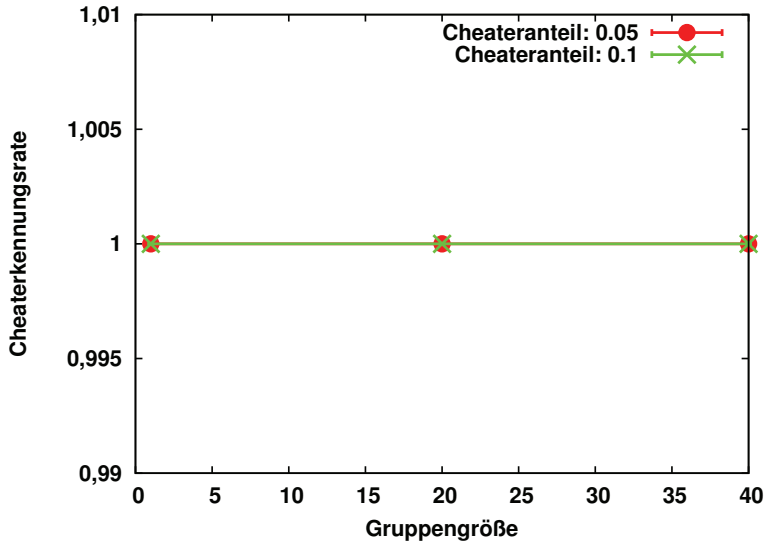
7.11.6.2 Cheat-Erkennungszeit

In Abbildung 7.3 sind die Erkennungszeiten der verschiedenen Cheat-Verhalten in Abhängigkeit zu der Gruppengröße gezeigt. Das Cheat-Verhalten „Überschreitung der Maximalgeschwindigkeit“ wird am schnellsten erkannt. Die zu diesem Verhalten gehörenden Simulationsergebnisse sind in Abbildung 7.3a gezeigt. Bei einer Gruppengröße von 1 beträgt die Erkennungszeit ungefähr 1,5 Sekunden. Die Erkennungszeit sinkt mit der Gruppengröße auf 0,75 Sekunden bei einer Gruppengröße von 40. Permanente Cheats können also mit der verteilten Cheat-Erkennung in sehr kurzer Zeit erkannt werden. Bei größerer Gruppengröße gibt es mehr Teilnehmer, die den Cheat beobachten und melden können, wodurch die Erkennungszeit bei größeren Gruppengrößen sinkt. Bei einem Cheater-Anteil von 10% ist die Erkennungszeit leicht über der Erkennungszeit bei einem Cheater-Anteil von 5%, die Unterschiede sind jedoch statistisch nicht signifikant.

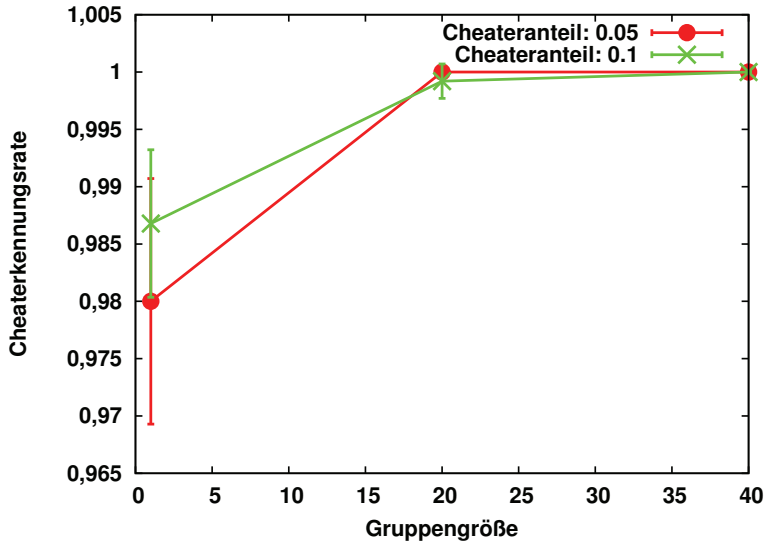
Abbildung 7.3b zeigt die Simulationsergebnisse für das Cheat-Verhalten „Ignorieren von Hindernissen“. Da dies ein sporadisches Cheat-Verhalten darstellt, wird für die Erkennung eine längere Zeit benötigt als für Verhalten „Überschreitung der Maximalgeschwindigkeit“. Bei einer Gruppengröße von 1 benötigt die Erkennung im Durchschnitt zwischen 7,5 und 8 Sekunden. Bei größeren Gruppen sinkt die Erkennungszeit deutlich, bei einer Gruppengröße von 40 beträgt sie ungefähr 2 Sekunden. Bei Gruppen dieser Größe reicht ein Regelbruch des Cheaters, um den Vertrauenswert so weit sinken zu lassen, dass die Observierung durch einen besonders vertrauenswürdigen Beobachter veranlasst wird. Bei einem weiteren Regelbruch wird dann die Bestrafung eingeleitet. Auch hier ist die Erkennungszeit bei einem Cheater-Anteil von 10% etwas größer als die Erkennungszeit bei einem Cheater-Anteil von 5%.



(a) Cheat-Erkennungsrate „Überschreitung der Maximalgeschwindigkeit“.

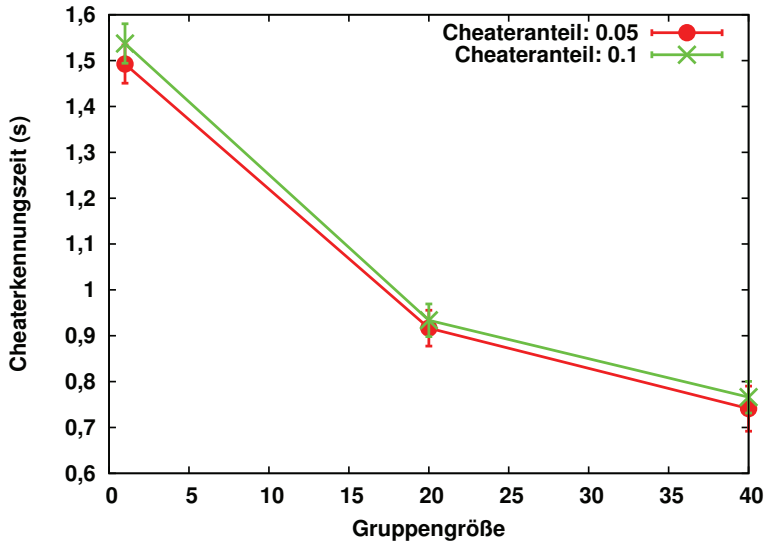


(b) Cheat-Erkennungsrate „Ignorieren von Hindernissen“.

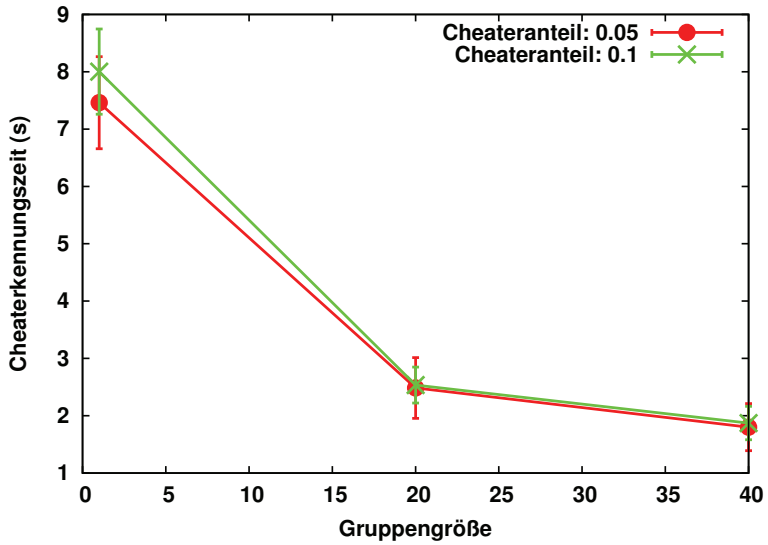


(c) Cheat-Erkennungsrate „Täuschen des Beobachters“.

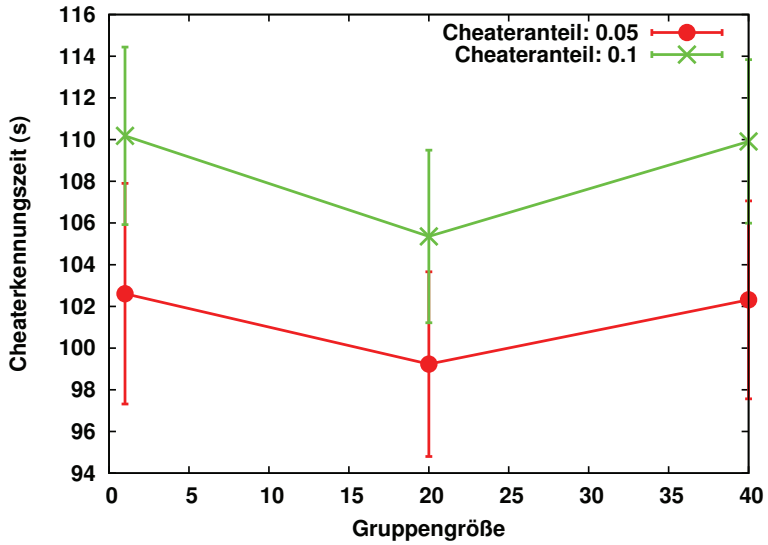
Abbildung 7.2 Evaluierungsergebnisse: Cheat-Erkennungsrate in Abhängigkeit der Gruppengröße.



(a) Cheat-Erkennungszeit „Überschreitung der Maximalgeschwindigkeit“.



(b) Cheat-Erkennungszeit „Ignorieren von Hindernissen“.



(c) Cheat-Erkennungszeit „Täuschen des Beobachters“.

Abbildung 7.3 Evaluierungsergebnisse: Cheat-Erkennungszeit in Abhängigkeit der Gruppengröße.

In Abbildung 7.3c ist die durchschnittliche Erkennungszeit für das Cheat-Verhalten „Täuschen des Beobachters“ gezeigt. Die Zeit ist deutlich länger als bei den anderen Cheat-Verhalten: Bei einer Gruppengröße von 1 beträgt sie bei einem Cheater-Anteil von 5% 102 Sekunden, bei einem Cheater-Anteil von 10% 110 Sekunden. Bei einer Gruppengröße von 40 wird für die Erkennung dieselbe Zeit benötigt. Bei einer Gruppengröße von 20 wird etwas weniger Zeit benötigt: Bei einem Cheater-Anteil von 5% 100 Sekunden, bei einem Cheater-Anteil von 10% 106 Sekunden. Die Abweichung gegenüber den anderen Gruppengrößen ist allerdings statistisch nicht signifikant.

Die Erkennungszeiten sind hier nicht signifikant mit der Gruppengröße korreliert, da die Erkennung nur vom Beobachter unter Mithilfe der Verbindungsnachbarn des beobachteten Teilnehmers erfolgen kann. Es ist anzumerken, dass dieses Cheat-Verhalten für den Cheater nur dann einen Vorteil bringt, wenn er es mit einem anderen Cheat kombiniert. Durch die daraus resultierenden Anschuldigungen anderer Teilnehmer würde der Vertrauenswert des Cheaters sinken und damit auch die Erkennungszeit verkürzt werden.

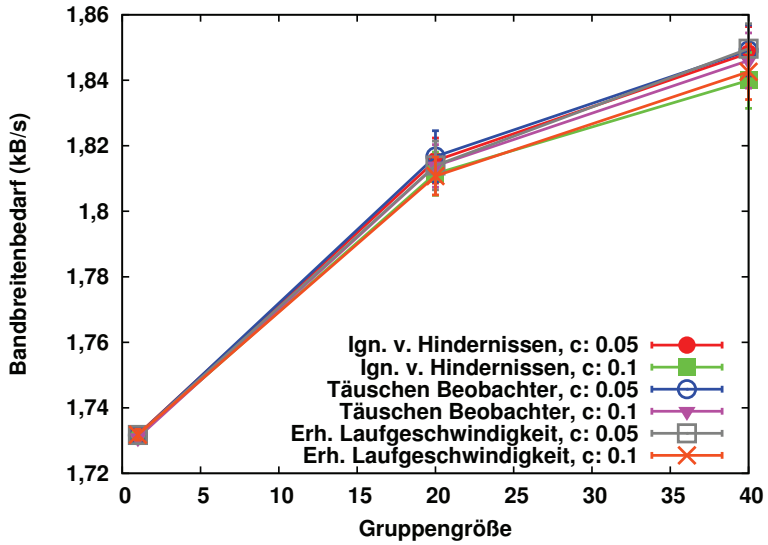
7.11.6.3 Fehlerrate

Die Fehlerrate war über alle Simulationsläufe hinweg sehr gering. Da nur in wenigen Simulationsläufen ehrliche Spieler fälschlicherweise ausgeschlossen wurden, wurde die Fehlerrate über alle Simulationläufe hinweg aggregiert. Nimmt man alle Simulationsläufe zusammen, wurden 450.000 verschiedene Teilnehmer simuliert. Davon waren 416.250 Teilnehmer ehrlich. Insgesamt wurden 44 ehrliche Teilnerhmer des Spiels verweisen, was eine Fehllerrate von ungefähr 0,01% ergibt.

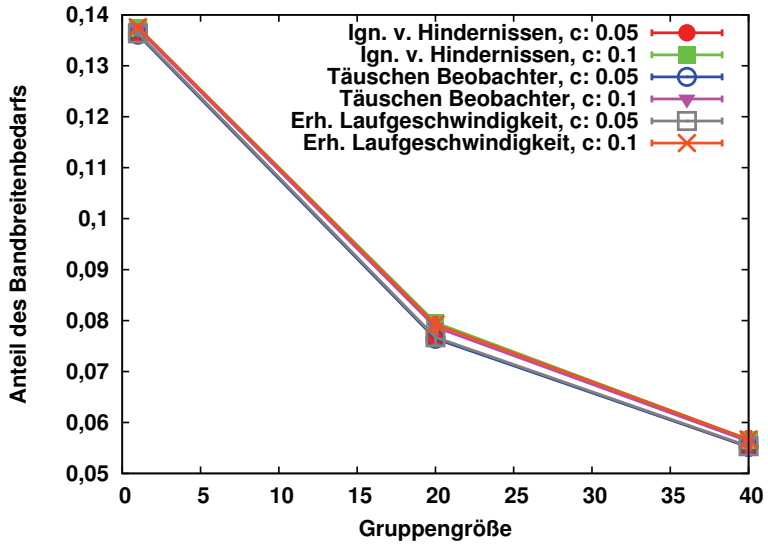
7.11.6.4 Bandbreitenbedarf

Abbildung 7.4 zeigt den Bandbreitenbedarf der Cheat-Erkennung bei den verschiedenen Cheat-Verhalten und Cheater-Anteilen (in der Abbildung mit „c“ bezeichnet) in Abhängigkeit der Gruppengröße. Abbildung 7.4a zeigt dabei den absoluten Bandbreitenbedarf, der durchschnittlich von jedem Teilnehmer geleistet werden muss. Bei einer Gruppengröße von 1 beträgt dieser bei allen untersuchten Szenarien etwa 1,73 kByte/s. Bei steigender Gruppengröße steigt der Bandbreitenbedarf für die Cheat-Erkennung leicht, da mehr Teilnehmer einen Cheater beobachten und melden. Bei einer Gruppengröße von 20 liegt er zwischen 1,81 kByte/s und 1,82 kByte/s, bei einer Gruppengröße von 40 zwischen 1,84 kByte/s und 1,86 kByte/s. Da Cheater nach ihrem Ausschluss aus dem Netz keinerlei Datenverkehr mehr erzeugen, ist der Bandbreitenbedarf bei höheren Cheater-Anteilen etwas größer als bei geringeren Cheater-Anteilen.

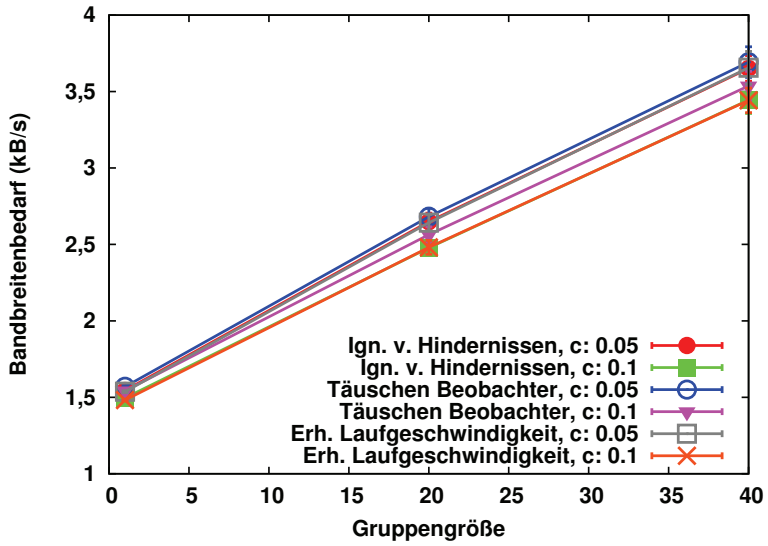
Abbildung 7.4b zeigt den Anteil der Cheat-Erkennung am Gesamtbandbreitenbedarf der Teilnehmer. Bei einer Gruppengröße von 1 beträgt der Anteil ungefähr 13,8%. Da bei einer Gruppengröße von 1 die Teilnehmer relativ wenige Nachbarn haben, ist der durch das Versenden aller Nachrichten an den Beobachter entstehende Mehrbedarf spürbar. Bei einer Gruppengröße von 20 sinkt der Anteil der Cheat-Erkennung auf ungefähr 7,8% des Gesamtbandbreitenbedarfs, bei einer Gruppengröße von 40 auf ungefähr 5,6%. In diesem Szenario wird also der weitaus größte Teil des Bandbreitenbedarfs durch QuON erzeugt. Dies zeigt, dass die Cheat-Erkennung keinen negativen Einfluss auf die Skalierungseigenschaften von QuON bei steigender Nachbarszahl hat.



(a) Durchschnittlicher Bandbreitenbedarf Cheat-Erkennung.



(b) Durchschnittlicher Anteil Cheat-Erkennung am Gesamtbandbreitenbedarf.



(c) Durchschnittlicher Bandbreitenbedarf Autorisierungsstelle und Vertrauensstelle.

Abbildung 7.4 Evaluierungsergebnisse: Bandbreitenbedarf in Abhängigkeit der Gruppengröße.

In den Simulationen wurden Autorisierungsstelle und Vertrauensstelle als zentrale Server implementiert. Der Bandbreitenbedarf dieser Komponenten ist in Abbildung 7.4c dargestellt. Bei einer Gruppengröße von 1 beträgt der Bandbreitenbedarf der beiden Komponenten zusammengenommen 1,5 kByte/s. Bei größeren Gruppen steigt der Bedarf auf 2,5 kByte/s bei einer Gruppengröße von 20 und ungefähr 3,5 kByte/s bei einer Gruppengröße von 40. Der Anstieg ist dem höheren Aufwand für die Verifizierung der Tickets der Teilnehmer geschuldet. Da Teilnehmer bei höheren Gruppengrößen mehr Nachbarn haben, wird entsprechend öfter die Online-Verifizierung der Tickets durchgeführt (Siehe hierzu auch Abbildung A.4 in Anhang A.2). Bei höheren Cheater-Anteilen liegt der Bandbreitenbedarf aufgrund der größeren Anzahl beobachteter Cheats etwas höher als bei geringeren Cheater-Anteilen. Bezogen auf die 500 Teilnehmer in der Simulation beträgt der Bandbreitenbedarf zwischen 3 B/s und 7 B/s pro Teilnehmer.

7.11.7 Bewertung

Die Evaluierungsergebnisse zeigen, dass die vorgestellte verteilte Cheat-Erkennung effizient Fehlverhalten böswilliger Teilnehmer virtueller Welten erkennen und bestrafen kann. So wurden in der verschiedenen Szenarien fast alle Cheater erkannt. Die Erkennung der Cheater geschieht größtenteils in wenigen Sekunden. Es bleibt einem Cheater somit nur wenig Zeit, mit seinem Fehlverhalten einen Schaden zu erzeugen. Nur das Cheat-Verhalten „Täuschen des Beobachters“ wird erst nach einer etwas längeren Zeit erkannt, da es nur vom Beobachter und nicht von den anderen Nachbarn erkannt werden kann. Um durch dieses Verhalten einen Vorteil zu erlangen, muss es jedoch mit einem anderen Cheat-Verhalten kombiniert werden. Dieses kann jedoch auch von den anderen Teilnehmern beobachtet werden und senkt somit die Erkennungszeit deutlich. Die Fehlerrate in den Simulationen betrug 0,01%. Es werden also nur sehr wenige ehrliche Spieler durch die Cheat-Erkennung fälschlicherweise des Cheatens bezichtigt.

Der Bandbreitenbedarf der Cheat-Erkennung bleibt insgesamt gering. Es wurde in allen Szenarien ein durchschnittlicher Bandbreitenbedarf von weniger als 1,9 kByte/s gemessen. Zudem steigt der Bandbreitenbedarf nur wenig mit der Gruppengröße und somit der Zahl der Nachbarn. Folglich ist in Szenarien mit hoher Nachbarszahl und damit hoher Last der Anteil der Cheat-Erkennung am gesamten Bandbreitenbedarf gering. Bei einer Gruppengröße von 40 liegt er beispielsweise bei lediglich 5,6%.

Auch bei der Verwendung einer zentralen Authentisierungs- und Vertrauensstelle bleibt die Cheat-Erkennung skalierbar: Der Bandbreitenbedarf der genannten Stellen liegt je nach Teilnehmerdichte bei wenigen Bytes pro Sekunde. Bei einer einfachen 100 Mbit/s-Netzanbindung für die Vertrauens- und Autorisierungsstelle ließen sich bereits mehrere Millionen Teilnehmer unterstützen. Die verteilte Cheat-Erkennung lässt sich um verteilte Vertrauens- und Authentisierungsmechanismen erweitern und somit ganz ohne zentrale Komponenten betreiben.

Insgesamt ist die hier vorgestellte verteilte Cheat-Erkennung somit eine effiziente Möglichkeit, Cheater in auf P2P-Protokollen basierenden virtuellen Welten zu erkennen und auszuschließen.

7.12 Zusammenfassung

In diesem Kapitel wurde eine verteilte Cheat-Erkennung für P2P-basierte virtuelle Welten vorgestellt. Diese bietet eine skalierbare und effiziente Möglichkeit, Cheater aus einer virtuellen Welt fernzuhalten, ohne die bei bisherigen Protokollen zur Cheat-Verhinderung üblichen Latenzeinbußen hinnehmen zu müssen.

Die Cheat-Erkennung erfordert eine Authentifizierung und Autorisierung der Teilnehmer. Diese Dienste können durch einen einfachen zentralen Server bereitgestellt werden. Es ist aber auch eine Bereitstellung durch eine verteilte Autorisierungsstelle vorstellbar. Von der Autorisierungsstelle wird den Teilnehmern ein Ticket ausgestellt, mit dem sie ihre Teilnahmeberechtigung gegenüber anderen Teilnehmern beweisen können.

Allen Teilnehmern wird ein Vertrauenswert zugeordnet. Dieser Vertrauenswert wird durch Cheat-Anschuldigungen verringert und durch längere Perioden ohne Cheat-Anschuldigungen erhöht. Der Vertrauenswert beeinflusst die Gültigkeitsdauer der Tickets und wird bei der Bewertung von Cheat-Anschuldigungen berücksichtigt. Die Vertrauensstelle kann wie die Autorisierung durch einen zentralen Server verwaltet werden. Auch hier ist alternativ die Verwendung eines verteilten Vertrauensdienstes möglich.

Das Kernstück der Cheat-Erkennung ist die gegenseitige Überwachung der Teilnehmer. Jeder Teilnehmer prüft die ihm von Nachbarn zugesendeten Nachrichten auf Plausibilität und mögliches Fehlverhalten. Ein Cheat-Verdacht wird der Vertrauensstelle gemeldet. Um mögliche Interessenkonflikte zu vermeiden, werden Teilnehmer nicht nur von ihren Nachbarn, sondern auch von unabhängigen Beobachtern überwacht. Diese werden von der Vertrauensstelle zugewiesen. Von Beobachtern festgestelltes Fehlverhalten wirkt sich stärker auf den Vertrauenswert eines Teilnehmers aus als von Nachbarn beobachtetes Fehlverhalten. Teilnehmer mit geringem Vertrauen bekommen Beobachter zugewiesen, die ein besonders hohes Vertrauen genießen.

Die Beobachter spielen auch eine wichtige Rolle bei der Bestrafung von Cheatern: Ein Teilnehmer wird bestraft, wenn sein Vertrauenswert einen Schwellenwert unterschreitet und sein Beobachter, der gemäß der Zuordnung von Beobachtern einen hohen Vertrauenswert besitzt, einen Cheat direkt beobachtet. Zusätzlich können Strafen auch ohne Einwirkung des Beobachters eingeleitet werden, wenn die Cheats mit kryptographisch gesicherten Nachrichten bewiesen werden können.

Eine wirkungsvolle Strafe ist der Ausschluss aus der virtuellen Welt. Er lässt sich durch einen Rückruf des Tickets des Cheaters durchführen. Hierzu wird von der Vertrauensstelle ein signierter Ticketrückruf generiert und von den Nachbarn des Cheaters in der virtuellen Welt verbreitet. Die Nachbarn des Cheaters unterbinden daraufhin jegliche Kommunikation mit dem Cheater.

Die Evaluierung der Cheat-Erkennung zeigt, dass mit den hier vorgeschlagenen Mechanismen Cheater effizient und schnell erkannt werden können. Dabei ist die Fehlerrate der Erkennung gering. Der Bandbreitenbedarf der Cheat-Erkennung ist nur ein geringer Teil des für die Teilnahme in der virtuellen Welt nötigen Gesamtbandbreitenbedarfs.

8. Evaluierung

Um die Leistungsfähigkeit des in dieser Arbeit entwickelten Protokolls QuON zu testen, wurden Evaluierungen durchgeführt, die in diesem Kapitel vorgestellt werden. Zunächst wird eine simulative Evaluierung durchgeführt. Im Anschluss wird QuON im Forschungsnetz G-Lab evaluiert.

8.1 Evaluierung im Simulator

Zunächst wird QuON mit dem in Kapitel 4 vorgestellten Simulationswerkzeug OverSim evaluiert. Als erstes werden dabei Varianten des Protokolls verglichen, um eine gute Parametrisierung zu finden. Anschließend wird QuON bei unterschiedlichem Sitzungsverhalten der Teilnehmer und bei verschiedenen Teilnehmerdichten untersucht. Zuletzt wird QuON mit bisherigen Protokollen für virtuelle Welten verglichen.

8.1.1 Metriken

Bei der Evaluierung von QuON werden dieselben Metriken verwendet, die auch bei der Evaluierung bisheriger Protokollvorschläge für virtuelle Welten in Abschnitt 5.2.1 beschrieben wurden. Diese Metriken umfassen die *Nachrichtenverzögerung* der Positionsmeldungen, den *Bandbreitenbedarf* und die *Nachbarschaftskenntnis*.

Für eine störungsfreie Interaktion zwischen den Teilnehmern der virtuellen Welt ist es wichtig, dass Positionsmeldungen und Ereignisnachrichten der Teilnehmer mit einer geringen Verzögerung zugestellt werden. Dabei sollte die Zustellzeit 200 ms nicht überschreiten [126]. In den Simulationen wurde die Zeit gemessen, die durchschnittlich zwischen dem Absenden einer Positionsmeldung und dem Empfang dieser Meldung durch andere Teilnehmer vergeht.

Da in P2P-Protokollen Aufgaben von den Teilnehmern übernommen werden, die in klassischen Client/Server-Systemen von den Servern geleistet werden, ist der Bandbreitenbedarf von P2P-Systemen auf Seiten der Teilnehmer üblicherweise höher als bei Client/Server-Systemen. Vor allem in Senderichtung ist die verfügbare

Bandbreite der Teilnehmer jedoch relativ gering; nur wenige übliche Endkundenanschlüsse überschreiten eine Kapazität von 1 MBit/s, einige Anschlüsse sind sogar auf 384 kBit/s beschränkt [50]. Der Bandbreitenbedarf eines Protokolls sollte demnach diese Größe nicht überschreiten. Gemessen wurde in den Simulationen der durchschnittliche Bandbreitenbedarf der Teilnehmer in Senderichtung in tausend Bytes pro Sekunde (kByte/s).

Eine weitere wichtige Metrik ist die Nachbarschaftskenntnis. Diese gibt an, welcher Anteil an Teilnehmern alle Teilnehmer in seinem Interessengebiet kennt. Dieser Anteil sollte möglichst nahe an 100% liegen. Ist dies nicht der Fall, werden mögliche Interaktionen verhindert.

Bei der Evaluierung des Knotenbeitritts wird die Zeit gemessen, die vom Versenden einer Beitrittsanfrage eines Teilnehmers bis zu dessen erfolgreichem Netzbeitritt vergeht.

8.1.2 Einflussgrößen

Analog zur Evaluierung bisheriger Protokollvorschläge für virtuelle Welten wird wie in Abschnitt 5.2.2 die *Teilnehmerdichte* als wesentliche Einflussgröße in den Simulationen betrachtet. Dabei wird zwischen *durchschnittlicher Teilnehmerdichte* und *lokaler Teilnehmerdichte* unterschieden.

Die durchschnittliche Teilnehmerdichte ergibt sich aus der Anzahl der Teilnehmer bezogen auf die Größe der Spielwelt. Sie wurde in den Simulationen durch die Wahl verschiedener Spielfeldgrößen variiert. Bei einer hohen Teilnehmerdichte haben Teilnehmer eine höhere Zahl anderer Teilnehmer in ihrem Interessengebiet. Dadurch steigt in Szenarien mit hoher Teilnehmerdichte in der Regel auch der Bandbreitenbedarf. Bei zonenbasierten Protokollen müssen bei hoher Teilnehmerdichte die Zonengrößen entsprechend verkleinert werden, um die Zahl der Teilnehmer pro Zone nicht zu groß werden zu lassen. Dies kann abhängig vom jeweiligen Protokoll Nachrichtenverzögerung und Nachbarschaftskenntnis beeinflussen.

Bei der lokalen Teilnehmerdichte wird hingegen die Ungleichverteilung der Teilnehmer in der virtuellen Welt betrachtet. Da virtuelle Welten in der Regel das Zusammenspiel der Teilnehmer fördern, ist die Bildung von Spielergruppen üblich. Diese bestehen aus Teilnehmern, die ein gemeinsames Ziel in der virtuellen Welt verfolgen und räumlich zusammenbleiben. Durch eine derartige Gruppenbildung ist die Teilnehmerdichte in verschiedenen Bereichen des Spielfelds unterschiedlich groß: Innerhalb einer Gruppe ist die Teilnehmerdichte hoch, in Spielfeldbereichen, in denen sich keine Gruppen aufhalten, hingegen sehr niedrig. In den Simulationen wurde die lokale Teilnehmerdichte beeinflusst, indem sich die Teilnehmer zu Gruppen verschiedener Größe zusammenschlossen.

Eine andere wichtige Einflussgröße ist das Sitzungsverhalten der Nutzer. Relevant sind hier insbesondere die durchschnittlichen *Sitzungslängen* und das Verhältnis zwischen Sitzungsende durch *geordnetes Abmelden* und Sitzungsende durch *Knotenausfall*. Beide Größen wirken sich auf die Nachbarschaftskenntnis aus. Bei kürzeren Sitzungszeiten steigt die Fluktuationsrate der Teilnehmer. Da sowohl der Beitritt von Teilnehmern zum Netz als auch der Austritt von Teilnehmern aus dem Netz potentielle Risiken für die Nachbarschaftskenntnis bergen, führen kürzere Sitzungszeiten in der Regel zu einer Verschlechterung der Nachbarschaftskenntnis.

Wird die Sitzung eines Teilnehmers durch einen Ausfall statt durch ein geordnetes Abmelden beendet, steigen die Risiken für die Nachbarschaftskenntnis. So können bei einem geordneten Abmelden eines Teilnehmers in QuON mittels des in Abschnitt 6.7.2 beschriebenen Abmeldeverfahrens die Aufgaben des sich abmeldenden Teilnehmers proaktiv auf andere Teilnehmer übertragen werden. Bei einem Ausfall kann nicht auf dieses Verfahren zurückgegriffen werden. Der Ausfall muss stattdessen durch das in Abschnitt 6.7.4 beschriebene Backup-Verfahren kompensiert werden.

Der Kommunikationsaufwand von QuON ist in der Regel nicht von der Gesamtzahl der Teilnehmer, sondern nur von der Zahl der Nachbarn eines Teilnehmers abhängig. Eine Ausnahme ist hier das in Abschnitt 6.7.1 beschriebene Beitrittsverfahren. Die rekursive Weiterleitung der Beitrittsanfrage hat einen von der Gesamtzahl der Teilnehmer abhängigen Aufwand. Ohne Verwendung eines Login-Caches beträgt er bei n Teilnehmern $O(n)$ bei ungünstiger Verteilung der Teilnehmer. Bei Gleichverteilung der Teilnehmer beträgt der Aufwand $O(\sqrt{n})$. Bei der Evaluierung des Knotenbeitritts wird daher die Teilnehmerzahl als Einflussgröße verwendet.

8.1.3 Szenarien

Zunächst werden in der Evaluierung verschiedene QuON-Varianten miteinander verglichen. Zuerst wird in Abschnitt 8.1.4 die von QuON verwendete Vektornorm betrachtet. Die möglichen Normen – die Maximumsnorm und die euklidische Norm – sind in Abschnitt 6.2 beschrieben. QuON wird mit den Normen bei verschiedenen Teilnehmerdichten simuliert.

Im Anschluss werden in Abschnitt 8.1.5 die verschiedenen Methoden zur Bandbreiteneinsparung in QuON, die in Abschnitt 6.6.3 beschriebene alternative Nachbarsfindung und die in Abschnitt 6.9 beschriebenen dynamischen Interessengebietsgrößen, untersucht. Es wird das Bandbreiteneinsparungspotential verglichen und gegen mögliche Einschränkungen in der Nachbarschaftskenntnis abgewogen. Dabei wird als Einflussgröße die Teilnehmerdichte betrachtet.

Danach wird in Abschnitt 8.1.6 der Teilnehmerbeitritt in QuON evaluiert. Hier wird bei verschiedener Teilnehmerzahl gemessen, wie viel Zeit durchschnittlich von der Versendung einer Beitrittsanfrage durch einen Teilnehmer bis zu dessen erfolgreichem Netzbeitritt vergeht.

Anschließend wird in Abschnitt 8.1.7 der Einfluss des Sitzungsverhaltens der Teilnehmer auf QuON untersucht. Dabei werden Simulationen mit verschiedenen durchschnittlichen Sitzungslängen durchgeführt. Weiterhin wurde der Anteil an geordneten Abmeldungen variiert.

Zuletzt wird in Abschnitt 8.1.8 QuON mit den bereits in Abschnitt 5.2 evaluierten bisherigen Protokollen verglichen. Hierbei wird als Einflussgröße die Teilnehmerdichte betrachtet.

Bei der Evaluierung werden dabei dieselben Szenarien betrachtet, die bereits in der Evaluierung der bisherigen Protokolle in Abschnitt 5.2.3 beschrieben wurden. Die virtuelle Welt wird von der Applikation *SimpleGameClient* simuliert. In der virtuellen Welt bewegen sich durchschnittlich 500 Teilnehmer. Das Sitzungsverhalten der Teilnehmer wird von dem *ParetoChurn*-Sitzungsmodell gesteuert. Die mittlere

Parameter	Wert
Simulationszeit	2 Stunden nach Initialisierungsphase
Anzahl Seeds	30
Teilnehmerzahl	500
Spielfeldgröße	1 km*1 km, 5 km*5 km, 10 km*10 km
Bewegungsmodell	GroupRoaming
max. Gruppengröße	1, 5, 10, 20, 40, 60
Sitzungsmodell	ParetoChurn
Sitzungszeit	100 Minuten
Graceful-leave	50%
Interessengebietsradius	50 m
Bewegungsgeschwindigkeit	5 m/s
Frequenz Positionsmeldungen	6/s
Backupnachbarn	1/Quadrant
Interessengebietspuffer	5 m
Vektornorm	euklidische Norm
Netzmodell	SimpleUnderlay

Tabelle 8.1 Simulationsparameter.

Sitzungszeit beträgt dabei, sofern nicht explizit angegeben, 100 Minuten. Die Sitzungen werden zufällig entweder mit einem geordneten Abmelden oder einem *fail-stop*-Ausfall des Teilnehmers beendet.

Das Interessengebiet der Teilnehmer ist ein Kreis um die aktuelle Position des Teilnehmers mit dem Radius 50 m. Die Bewegungsgeschwindigkeit der Teilnehmer in der virtuellen Welt beträgt 5 m/s. Dabei werden von den Teilnehmern 6 Positionsmeldungen pro Sekunde generiert.

Die in den Simulationen betrachteten Spielfeldgrößen sind, sofern nicht explizit angegeben, 10.000 m mal 10.000 m, 5.000 m mal 5.000 m und 1.000 m mal 1.000 m. Die Bewegung der Teilnehmer wird von dem *GroupRoaming*-Modell gesteuert. Die maximale Gruppengrößen sind dabei 1 (also keine Gruppenbildung), 5, 10, 20, 40 und 60. Die Parameter sind so gewählt, dass sie den Werten heute existierender virtueller Welten möglichst nahe kommen[1].

Bei QuON wird der in Abschnitt 6.7.4 beschriebene Backup-Mechanismus genutzt. Dabei wird ein Backup-Nachbar pro Quadrant verwendet. Ebenso findet der in Abschnitt 6.8 beschriebene Interessengebietspuffer Anwendung. Er hat einen Radius von 5 Metern.

Das unterliegende Netz wird von dem *SimpleUnderlay*-Modell simuliert. Alle Simulationen wurden mit 30 verschiedenen Seeds wiederholt. Gezeigt sind die Durchschnittswerte dieser Wiederholungen und das 99% Konfidenzintervall. Die Simulationsparabeter sind in Tabelle 8.1 zusammengefasst.

8.1.4 Evaluierung der Vektornormen

Für QuON wurden in Abschnitt 6.2 zwei verschiedene Vektornormen vorgeschlagen: Die Maximumsnorm und die euklidische Norm. Mit der Maximumsnorm lassen sich

[1] Siehe dazu auch Abschnitt 5.2.3.

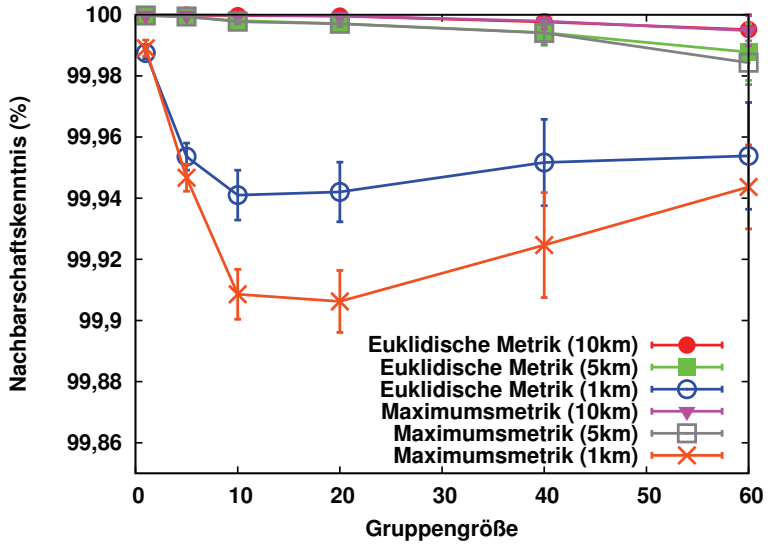
einige Eigenschaften von QuON in idealisierten Szenarien beweisen. Mit der euklidischen Norm haben die Interessengebiete eine kleine Fläche, wodurch Bandbreite eingespart werden kann.

In diesem Abschnitt werden die Vektornormen bei verschiedenen Teilnehmerdichten untersucht. Die Simulationsparameter entsprechen – mit Ausnahme der Wahl der Vektornorm – denen in Tabelle 8.1. Die Nachrichtenverzögerung ist in QuON von der gewählten Norm und der Teilnehmerdichte unabhängig. Alle Nachrichten werden in einem Hop auf Overlay-Ebene zugestellt, die resultierende Nachrichtenverzögerung entspricht also der Durchschnittsverzögerung zwischen den Knoten im unterliegenden Netzwerk. Sie betrug in den hier durchgeführten Simulationen mit dem SimpleUnderlay-Modell zwischen 85 ms und 88 ms.

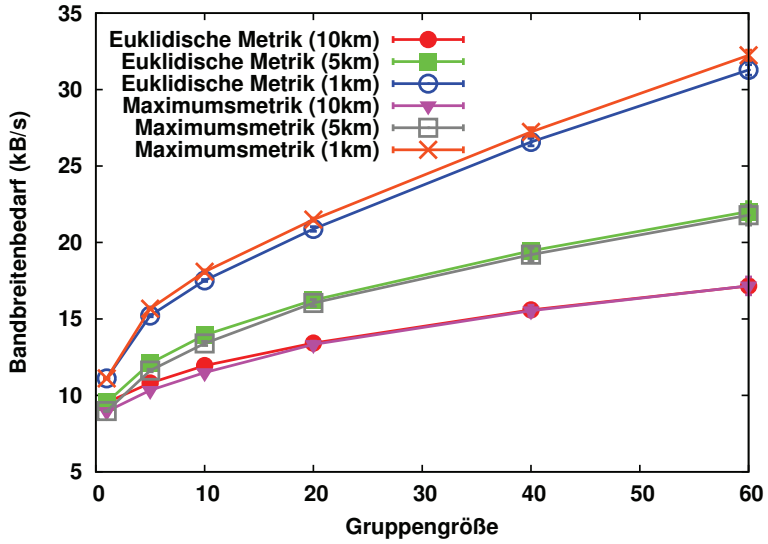
In Abbildung 8.1 werden die Evaluierungsergebnisse von QuON bei Verwendung der verschiedenen Vektornormen gezeigt. Abbildung 8.1a zeigt die Nachbarschaftskenntnis. Bei einer Spielfeldgröße von 10.000 m mal 10.000 m hat die Wahl der Norm keinen statistisch signifikanten Einfluss auf die Nachbarschaftskenntnis. Sowohl bei der euklidischen Norm als auch bei der Maximumsnorm wird bei einer Gruppengröße von 1 eine Nachbarschaftskenntnis von über 99,999% erreicht. Die Nachbarschaftskenntnis sinkt bei steigender Gruppengröße langsam, bei einer Gruppengröße von 60 beträgt sie 99,995%. Auch bei einer Spielfeldgröße von 5.000 m mal 5.000 m ergeben sich keine signifikanten Unterschiede zwischen den beiden Normen. Bei einer Gruppengröße von 1 wird wiederum eine Nachbarschaftskenntnis von mehr als 99,999% erreicht. Bei einer Gruppengröße von 60 sinken die Durchschnittswerte auf 99,987% für die euklidische Norm und 99,984% für die Maximumsnorm. Die mit der Gruppengröße fallende Nachbarschaftskenntnis erklärt sich, wie in Abschnitt 6.10 erläutert, dadurch, dass bei Gruppenbildung der Ausfall eines Verbindungsnachbarn mehrere Teilnehmer betreffen kann und so die Nachbarsfindung gestört werden kann. Insgesamt sind die beobachteten Werte jedoch auch bei großen Gruppen noch sehr gut.

Auch bei einer Spielfeldgröße von 1.000 m mal 1.000 m sind die Unterschiede bei der Verwendung verschiedener Vektornormen gering, jedoch etwas deutlicher als bei den größeren Spielfeldgrößen. Bei Verwendung der Euklidischen Norm wird bei einer Gruppengröße von 1 eine Nachbarschaftskenntnis von 99,99% erreicht. Bei einer Gruppengröße von 10 sinkt sie auf ein Minimum von 99,94%, um dann bis zu einer Gruppengröße von 60 auf 99,95% zu steigen. Bei einer Gruppengröße von 1 erreicht QuON bei Verwendung der Maximumsnorm ebenfalls eine Nachbarschaftskenntnis von 99,99%. Diese sinkt jedoch bei einer Gruppengröße von 20 bis auf 99,905%, um von dort bis auf 99,94% bei einer Gruppengröße von 60 zu steigen. Die zunächst fallende Nachbarschaftskenntnis lässt sich wie bei den größeren Spielfeldern erklären. Ab einer gewissen Gruppengröße ist die lokale Teilnehmerdichte so hoch, dass, wie in Abschnitt 6.10 erläutert, durch die hohe Zahl an direkten Nachbarn der Ausfall von Verbindungsnachbarn wieder kompensiert werden kann und so die Nachbarschaftskenntnis wieder steigt.

Es überrascht zunächst, dass die euklidische Norm eine bessere Nachbarschaftskenntnis aufweisen kann, da unter idealisierten Bedingungen für die Maximumsnorm eine perfekte Nachbarschaftskenntnis garantiert werden kann (vergleiche hierzu Abschnitt 6.2 und Anhang B.2), bei der euklidischen Norm jedoch nicht. Diese Überlegungen gelten jedoch nur in einer idealisierten Umgebung ohne Nachrichtenverzögerun-



(a) Nachbarschaftskenntnis von QuON bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.



(b) Bandbreitenbedarf von QuON bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.

Abbildung 8.1 Evaluierungsergebnisse: Einfluss der Wahl der Vektornorm in QuON.

gen, Paketverluste und Knotenausfälle. Die Simulationsergebnisse zeigen, dass die in QuON implementierten Mechanismen, um Knotenausfälle und Paketverzögerungen zu kompensieren, in Zusammenarbeit mit der euklidischen Norm besser arbeiten als in Zusammenhang mit der Maximumsnorm.

Abbildung 8.1b zeigt den Bandbreitenbedarf von QuON bei Verwendung der verschiedenen Metriken bei verschiedenen Spielfeldgrößen in Abhängigkeit von der Gruppengröße. Wie auch bei der Nachbarschaftskenntnis sind die Unterschiede zwischen den Normen bei den Spielfeldgrößen 10.000 m mal 10.000 m und 5.000 m mal 5.000 m nur gering. Bei einer Spielfeldgröße von 10.000 m mal 10.000 m benötigt QuON bei einer Gruppengröße von 1 bei Verwendung der Maximumsnorm 9 kByte/s und bei Ver-

wendung der euklidischen Norm 9,5 kByte/s. Bei einer Gruppengröße von 60 steigt der Bandbreitenbedarf auf 17,1 kByte/s unabhängig von der verwendeten Norm.

Wählt man die Spielfeldgröße von 5.000 m mal 5.000 m, entspricht der Bandbreitenbedarf bei einer Gruppengröße von 1 den Ergebnissen des größeren Spielfeldes mit 9 kByte/s bei Verwendung der Maximumsnorm und 9,5 kByte/s bei Verwendung der euklidischen Norm. Der Bandbreitenbedarf steigt jedoch mit steigender Gruppengröße stärker an und erreicht bei einer Gruppengröße von 60 21,7 kByte/s bei Verwendung der Maximumsnorm und 22 kByte/s bei Verwendung der euklidischen Norm.

Aufgrund der größeren Fläche des Interessengebiets bei Verwendung der Maximumsnorm ist es zunächst überraschend, dass QuON bei der Verwendung der euklidischen Norm mehr Bandbreite benötigt als bei Verwendung der Maximumsnorm. Bei großen Spielfeldgrößen sind zufällige Begegnungen zwischen Gruppen selten. Innerhalb der Gruppen sind die Teilnehmer in der Regel so dicht beinander, dass sie sich stets im Interessengebiet aller anderen Gruppenmitglieder befinden. Somit wird die Zahl der direkten Nachbarn in diesen Szenarien durch die Gruppengröße dominiert. Die Zahl an direkten Nachbarn ist also bei großen Spielfeldern von der verwendeten Vektornorm unabhängig. Siehe hierzu auch die Messungen der Zahl der Nachbarn in Abbildung A.5a in Anhang A.3. Bei der Zahl an temporären Nachbarn gibt es jedoch Unterschiede zwischen den Normen: Wie Abbildung A.5b in Anhang A.3 zeigt, ergibt sich bei der euklidischen Norm eine leicht höhere Zahl an temporären Nachbarn. Somit ergibt sich insgesamt bei Verwendung der euklidischen Norm eine geringfügig größere Zahl an Nachbarn. Bei großen Gruppengrößen ist dieser Unterschied jedoch zu klein, um sich relevant auf den Gesamtbandbreitenbedarf auszuwirken.

Bei der Spielfeldgröße 1.000 m mal 1.000 m sind die Unterschiede zwischen den Normen deutlicher. Im Gegensatz zu den größeren Spielfeldgrößen benötigt QuON mit der Maximumsnorm mehr Bandbreite als mit der euklidischen Norm. Bei einer Gruppengröße von 1 liegt der Bandbreitenbedarf mit 11 kByte/s für beide Normen noch auf gleicher Höhe. Bei größeren Gruppen wird bei Verwendung der Maximumsnorm mehr Bandbreite benötigt als bei Verwendung der euklidischen Norm. Bei einer Gruppengröße von 60 benötigt QuON mit der Maximumsnorm 32,2 kByte/s, mit der euklidischen Norm 31,3 kByte/s und damit etwa 3% weniger.

Bei dieser Spielfeldgröße zeigt sich nun der erwartete Mehrbedarf der Maximumsnorm aufgrund der größeren Fläche des Interessengebiets. Bei kleinen Spielfeldern sind zufällige Begegnungen zwischen Teilnehmern oder Teilnehmergruppen üblich. Bei einer größeren Fläche des Interessengebiets erhöht sich somit die Zahl der direkten Nachbarn. Dies zeigt auch die Evaluierung der Anzahl der Nachbarn in Abbildung A.5a in Anhang A.3. Wie bei den größeren Spielfeldern ergibt sich eine geringfügig kleinere Zahl an temporären Nachbarn bei der Maximumsnorm (siehe dazu Abbildung A.5b in Anhang A.3). Der Unterschied in der Anzahl der direkten Nachbarn ist jedoch deutlich größer und dominiert somit den Bandbreitenbedarf.

Insgesamt zeigen die Simulationen, dass die Unterschiede zwischen den untersuchten Abstandsnormen eher gering bleiben. Insgesamt sind die mit der euklidischen Norm erzielten Ergebnisse jedoch besser als mit der Maximumsnorm. Während bei größeren Spielfeldern und den damit verbundenen geringen Teilnehmerdichten die Wahl der Vektornormen nur einen vernachlässigbar kleinen Einfluss auf die Ergebnisse

hat, bietet die euklidische Norm bei kleinem Spielfeld und hoher Teilnehmerdichte eine bessere Nachbarschaftskenntnis bei geringerem Bandbreitenbedarf. Aus diesem Grund wird in allen weiteren Evaluierungen von QuON die euklidische Norm verwendet.

8.1.5 Möglichkeiten der Bandbreiteneinsparung

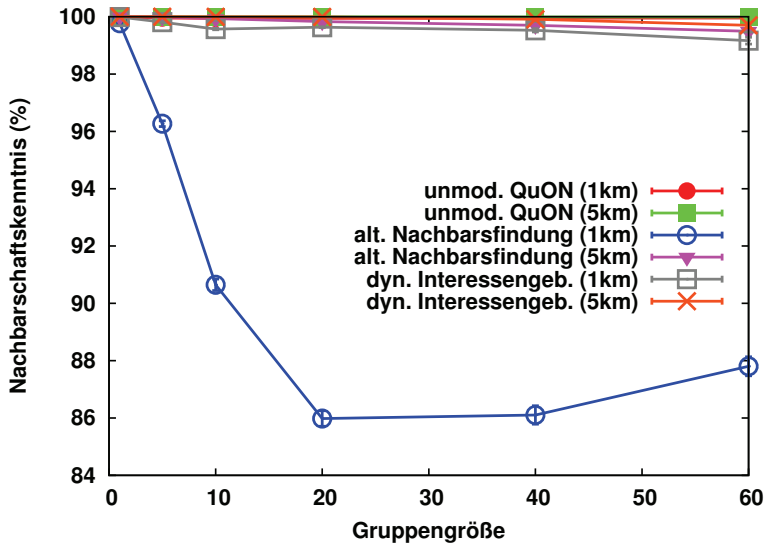
In diesem Abschnitt werden Bandbreiteneinsparungsmöglichkeiten in QuON evaluiert. Dabei werden die in Abschnitt 6.6.3 vorgestellte alternative Nachbarsfindung und die in Abschnitt 6.9 vorgestellte dynamische Interessengebietsgröße betrachtet und mit dem unmodifizierten QuON verglichen. Als Einflussgröße wird auch hier die Teilnehmerdichte verwendet. Die Simulationsparameter entsprechen denen in Tabelle 8.1. Wie bei der Evaluierung der Vektornormen ändern sich die Nachrichtenverzögerungen über alle Simulationen nur wenig. Sie liegen zwischen 85 ms und 89 ms.

In Abbildung 8.2 werden die Evaluierungsergebnisse der Bandbreiteneinsparungsmechanismen bei verschiedenen Spielfeldgrößen gezeigt. Die Spielfeldgröße 10.000 m mal 10.000 m wurde in der Darstellung der Übersichtlichkeit halber weggelassen, da die dort auftretenden Bandbreitenanforderungen ohnehin nur gering sind. Die dort beobachteten Tendenzen entsprechen denen bei der Spielfeldgröße 5.000 m mal 5.000 m. Die Ergebnisse für die Spielfeldgröße 10.000 m mal 10.000 m finden sich in Abbildung A.6 in Anhang A.3.

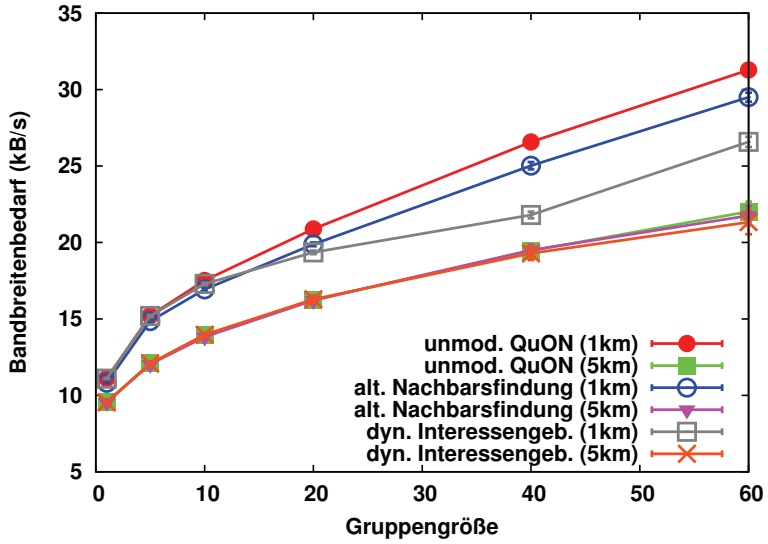
Abbildung 8.2a zeigt die mit den verschiedenen Bandbreiteneinsparungsmechanismen erreichte Nachbarschaftskenntnis in Abhängigkeit von der Gruppengröße. Das unmodifizierte QuON erreicht erwartungsgemäß die beste Nachbarschaftskenntnis. Bei einer Spielfeldgröße von 5.000 m mal 5.000 m beträgt diese bei einer Gruppengröße von 1 über 99,999%. Bis zu einer Gruppengröße von 60 sinkt die Nachbarschaftskenntnis auf 99,987%. Bei der Verwendung der alternativen Nachbarsfindung wird bei einer Gruppengröße von 1 ebenfalls eine Nachbarschaftskenntnis von über 99,999% erreicht. Sie fällt jedoch deutlich stärker als beim unmodifizierten QuON auf 99,5% bei einer Gruppengröße von 60. Bei einer Gruppengröße von 1 kann QuON mit dynamischer Interessengebietsgröße mit mehr als 99,999% die gleiche Nachbarschaftskenntnis wie die anderen Varianten erreichen. Bei größeren Gruppen fällt der Wert auf 99,7% bei einer Gruppengröße von 60 und damit stärker als beim unmodifizierten QuON, jedoch weniger stark als bei der alternativen Nachbarsfindung.

Bei der Spielfeldgröße von 1.000 m mal 1.000 m ist der Unterschied in der Nachbarschaftskenntnis des unmodifizierten QuONs zu den anderen Varianten, insbesondere zu der alternativen Nachbarschaftsfindung, deutlich größer. Bei einer Gruppengröße von 1 liegt der Wert für unmodifiziertes QuON bei 99,99%. Bei einer Gruppengröße von 10 erreicht die Nachbarschaftskenntnis mit 99,94% ein Minimum. Von dort steigt die Nachbarschaftskenntnis bis zu einer Gruppengröße von 60 auf 99,95%. Die Werte sind äquivalent zu den in der Evaluierung der Vektornormen in Abschnitt 8.1.4 beschriebenen Werten für QuON mit euklidischer Metrik.

Bei der alternativen Nachbarschaftsfindung wird bereits bei der Gruppengröße 1 nur noch eine Nachbarschaftskenntnis von 99,78% erreicht. Diese fällt stark ab auf 86% bei einer Gruppengröße von 20. Von diesem Minimum verbessert sie sich leicht auf 88% bei der Gruppengröße 60. Da bei der alternativen Nachbarsfindung die direkten



(a) Nachbarschaftskenntnis von QuON bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.



(b) Bandbreitenbedarf von QuON bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.

Abbildung 8.2 Evaluierungsergebnisse: Einfluss von Bandbreiteneinsparungsmechanismen in QuON.

Nachbarn nicht bei der Nachbarsfindung helfen können, ist die Verschlechterung der Nachbarschaftskenntnis bei größeren Gruppen deutlicher ausgeprägt als bei unmodifiziertem QuON. Bei Gruppen größer als 20 kommt es dennoch zu einer Verbesserung der Nachbarschaftskenntnis, da bei hoher Teilnehmerdichte der Backup-Mechanismus ausgefallene Verbindungsnachbarn schneller ersetzen kann.

Die Nachbarschaftskenntnis von QuON mit dynamischen Interessengebieten liegt wiederum zwischen den Werten für unmodifiziertes QuON und der alternativen Nachbarschaftsfindung. Bei der Gruppengröße 1 werden dort 99,99% erreicht. Bei einer Gruppengröße 60 fällt der Wert auf 99,16% ab. Bei der dynamischen Adaption der Interessengebiete setzt die Stabilisierung der Nachbarschaftskenntnis bei großen Gruppen nicht ein, da es bei hoher Teilnehmerdichte zu einer häufigeren Adaption

der Interessengebietsgröße und somit zu größeren Risiken für die Nachbarschaftskenntnis kommt[2].

Der Bandbreitenbedarf der QuON-Varianten ist in Abbildung 8.2b gezeigt. Bei der Spielfeldgröße 5.000 m mal 5.000 m liegen die Messwerte der QuON-Varianten dicht beieinander. Bei einer Gruppengröße von 1 benötigen alle Varianten 9,5 kByte/s. Der Messwert steigt mit steigender Gruppengröße an. Bei einer Gruppengröße von 60 erreicht er 22 kByte/s für das unmodifizierte QuON, 21,7 kByte/s für die alternative Nachbarsfindung und 21,3 kByte/s für die dynamischen Interessengebiete. Der Bandbreitenbedarf von QuON ist in diesen Szenarien insgesamt relativ niedrig, die Einsparmöglichkeiten sind entsprechend gering.

Bei einer Spielfeldgröße von 1.000 m mal 1.000 m sind die Unterschiede zwischen den Protokollvarianten deutlicher. Bei einer Gruppengröße von 1 liegt der Bandbreitenbedarf für unmodifiziertes QuON bei 11,1 kByte/s. Bei einer Gruppengröße von 60 steigen die Werte auf 31,3 kByte/s. Da es bei höherer Teilnehmerdichte häufiger zu Begegnungen zwischen Teilnehmern kommt und somit die Nachbarsfindung einen größeren Anteil am Gesamtbandbreitenbedarf hat, weist die alternative Nachbarsfindung nun einen sichtbaren Bandbreitenvorteil auf. Bei der Gruppengröße 1 beträgt der Bandbreitenbedarf 10,8 kByte/s. Bei einer Gruppengröße von 60 steigt der Wert auf 29,5 kByte/s und liegt damit knapp 6% unter dem entsprechenden Wert für unmodifiziertes QuON.

Bis zu einer Gruppengröße von 10 bleibt der Bandbreitenbedarf von QuON mit dynamischen Interessengebietsgrößen annähernd gleich zum unmodifizierten QuON. Bei einer Gruppengröße von 1 beträgt er 11,1 kByte/s. Bei größeren Gruppen ist jedoch eine deutliche Einsparung sichtbar, da hier die dynamische Anpassung der Interessengebietsgröße häufiger eingesetzt wird. Bei einer Gruppengröße von 60 beträgt der Bandbreitenbedarf 26,5 kByte/s. Bei dem durch die höhere Teilnehmerdichte bedingten höheren Bandbreitenbedarf zeigen sich somit etwas größere Einsparpotentiale. Die dynamische Adaption der Interessengebiete kann bei einer Gruppengröße von 60 15% der Bandbreite einsparen.

Betrachtet man die nur geringe Bandbreitenverringerung der alternativen Nachbarsfindung im Vergleich zu unmodifiziertem QuON und die mit der Modifikation verbundenen deutlichen Einbußen der Nachbarschaftskenntnis, ist der Einsatz der alternativen Nachbarsfindung in den hier betrachteten Szenarien nicht empfehlenswert. Bei unmodifiziertem QuON benötigt die Nachbarsfindung nur einen geringen Anteil am Gesamtbandbreitenbedarf, sodass das Einsparpotential nur gering ist. Da die alternativen Nachbarsfindung völlig ohne Redundanz auskommt, ist eine Verschlechterung der Nachbarschaftskenntnis bei hohen Teilnehmerdichten aufgrund von Latenzen, Paketverlusten und Teilnehmerausfällen nicht zu vermeiden.

Dynamische Interessengebiete können den Bandbreitenbedarf signifikant verringern. Dabei muss aber eine gewisse Verschlechterung der Nachbarschaftskenntnis in Kauf genommen werden. Sofern keine unmittelbare Gefahr der Überlastung der Teilnehmer besteht, ist die Gefahr, Teilnehmer in ihren Interaktionen einzuschränken, schwerer zu gewichten als eine Bandbreiteneinsparung von 15%. In der weiteren Evaluierung wird daher das unmodifizierte QuON verwendet.

[2]Vergleiche dazu auch Abschnitt 6.9.3.

Besteht hingegen eine konkrete Gefahr der Überlastung, ist, wie in Abschnitt 6.9.3 beschrieben, die Verwendung dynamischer Interessengebiete eine geeignete Technik um Überlastsituationen zu vermeiden.

8.1.6 Teilnehmerbeitritt

Der Beitritt eines Teilnehmers zu einem QuON-Netz wird durch das Versenden einer Beitrittsanfrage des Teilnehmers an ein zufälliges Mitglied des Netzes initiiert. Von dort wird die Anfrage rekursiv im Netz weitergeleitet, bis sie den Teilnehmer erreicht, der der gewünschten Beitrittsposition am nächsten ist. Der Aufwand dieses Verfahrens ist als einziger Teil von QuON von der Gesamtzahl der Teilnehmer im Netz abhängig. Bei n Teilnehmern benötigt die Nachricht im schlechtesten Fall $O(n)$ Weiterleitungsschritte und bei Gleichverteilung der Teilnehmer $O(\sqrt{n})$. Um den Beitritt zu beschleunigen, kann, wie in Abschnitt 6.7.1 beschrieben, ein *Location-Cache* eingesetzt werden.

In diesem Abschnitt wird das Teilnehmerbeitrittsverfahren mit und ohne Location-Cache evaluiert. Hierzu wurde in OverSim ein QuON-Netz mit 500, 1.000, 3.000, 6.000 und 10.000 Teilnehmern simuliert. Nach Abschluss der Initialisierungsphase traten diesem Netz durchschnittlich 50 weitere Teilnehmer bei. Bei diesen Teilnehmern wurde die Zeit zwischen Absendung der Beitrittsanfrage der Teilnehmer bis zu ihrem erfolgreichen Beitritt in das Netz gemessen. Die Spielfeldgröße wurde in den Simulationen mit 100.000 m mal 100.000 m so groß gewählt, dass die Teilnehmer durchschnittlich keine direkten Nachbarn hatten. In dem Szenario konnten so bei der Weiterleitung der Beitrittsanfrage keine Teilnehmer übersprungen werden. Bei der gemessenen Zeit handelt es sich also um eine worst-case-Zeit für den Knotenbeitritt.

Der Location-Cache wurde in der Simulation der Einfachheit halber von einer zentralen Instanz bereitgestellt. Dabei wurde eine triviale Implementierung eines Caches verwendet, die alte Einträge nicht verwirft, sondern nur beim Eintreffen aktualisierter Daten überschreibt. Die Evaluierungsergebnisse zeigen, dass auch eine solche einfache zentrale Lösung praxistauglich ist. Bei einem derartigen Cache wird für jeden Teilnehmer der virtuellen Welt Speicher für einen Identifikator und die Position benötigt. Diese Daten lassen sich pro Teilnehmer in 12 Bytes speichern. Damit lassen sich auch auf einfacher Hardware die Positionen mehrerer Millionen Teilnehmer speichern.

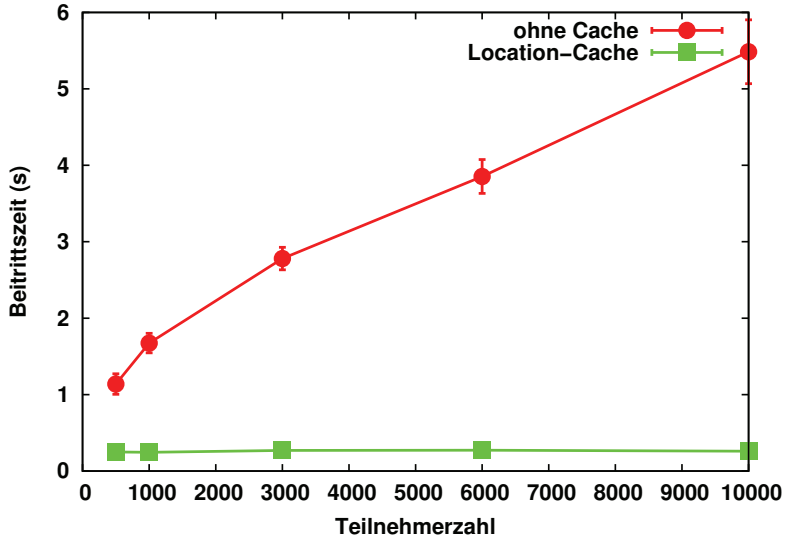
Um beitretende Teilnehmer nicht aufgrund veralteter Daten an zu weit von der gewünschten Beitrittsposition entfernte Teilnehmer zu verweisen, wurden die Einträge jeweils mit einem Malus versehen, der sich aus dem Alter des Eintrags und der maximalen Bewegungsgeschwindigkeit der Teilnehmer ergab.

Alle Teilnehmer des Netzes versendeten ein mal pro Minute ihre aktuelle Position an den Location-Cache. Dabei wurde der Bandbreitenbedarf des Caches in Empfangsrichtung gemessen. Eine höhere Aktualisierungsfrequenz würde die Genauigkeit des Caches erhöhen und somit die Beitrittsgeschwindigkeit senken. Dabei würde aber der Bandbreitenbedarf steigen. Die Simulationsergebnisse zeigen, dass sich mit der gewählten Frequenz sehr gute Ergebnisse erzielen lassen.

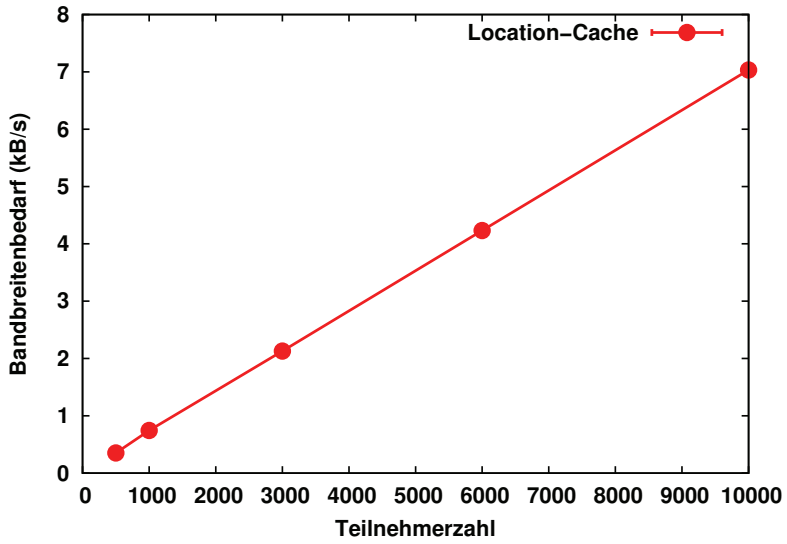
Die Abweichungen der Simulationsparameter von den in Tabelle 8.1 angegebenen beziehungsweise zusätzliche Parameter sind in Tabelle 8.2 gezeigt.

Parameter	Wert
Simulationszeit	500 s nach Initialisierungsphase
Teilnehmerzahl	500, 1.000, 3.000, 6.000, 10.000
Spielfeldgröße	100 km*100 km
Bewegungsmodell	RandomRoaming
Sitzungsmodell	NoChurn
Aktualisierungsfrequenz Position	1/Minute

Tabelle 8.2 Simulationsparameter Teilnehmerbeitritt.



(a) Beitrittszeit in QuON in Abhängigkeit der Teilnehmerzahl.



(b) Bandbreitenbedarf des Login-Caches QuON in Abhängigkeit der Teilnehmerzahl.

Abbildung 8.3 Evaluierungsergebnisse: Beitritt zum QuON-Netz.

Abbildung 8.3 zeigt die Ergebnisse der Evaluierung. In Abbildung 8.3a ist die für den Teilnehmerbeitritt benötigte Zeit dargestellt. Ohne Verwendung des Login-Caches vergehen bei 500 Teilnehmern durchschnittlich 1,1 Sekunden, bis ein neuer Teilnehmer dem Netz beigetreten ist. Die für den Login benötigte Zeit steigt proportional zur Wurzel der Teilnehmerzahl und erreicht bei 10.000 Teilnehmern 5,5 Sekunden.

Die Simulationsergebnisse bestätigen somit die Aufwandsabschätzungen der für den Teilnehmerbeitritt benötigten Zeit. Die bei 10.000 Teilnehmern benötigten 5,5 Sekunden liegen noch in einem annehmbaren Bereich. In großen virtuellen Welten ist jedoch zu erwarten, dass die für den Beitritt benötigte Zeit zu lang wird.

Bei Verwendung eines Login-Caches ist die für den Beitritt benötigte Zeit unabhängig von der Gesamtteilnehmerzahl und bleibt zwischen 245 ms und 270 ms. Die Beitrittsanfrage eines Teilnehmers muss also nur selten weitergeleitet werden, um zu dem für die Beitrittsposition verantwortlichen Teilnehmer zu gelangen. Damit kann auch in großen virtuellen Welten ein schneller Teilnehmerbeitritt gewährleistet werden.

Der Bandbreitenbedarf des Login-Caches ist in Abbildung 8.3b abgebildet. Da jeder Teilnehmer in festen Intervallen seine Position an den Login-Cache sendet, steigt dessen Bandbreitenbedarf mit der Teilnehmerzahl. Bei 1.000 Teilnehmern liegt der Bandbreitenbedarf bei 350 B/s, bei 10.000 Teilnehmern bei 7 kByte/s.

Der Bandbreitenbedarf des Login-Caches ist bei Verwendung eines zentralen Caches somit erwartungsgemäß linear von der Teilnehmerzahl abhängig. Der absolute Betrag ist dabei jedoch relativ gering. Es wird in der Simulation nur eine Bandbreite von 0,7 B/s pro Teilnehmer benötigt. Damit ließen sich bereits mit einer DSL-Internetanbindung mehrere Millionen Teilnehmer unterstützen. Durch eine Verringerung der Positionsaktualisierungsfrequenz der Teilnehmer ließe sich der Bandbreitenbedarf weiter senken, dies könnte jedoch die Teilnehmerbeitrittszeit steigern.

8.1.7 Einfluss des Sitzungsverhaltens der Teilnehmer

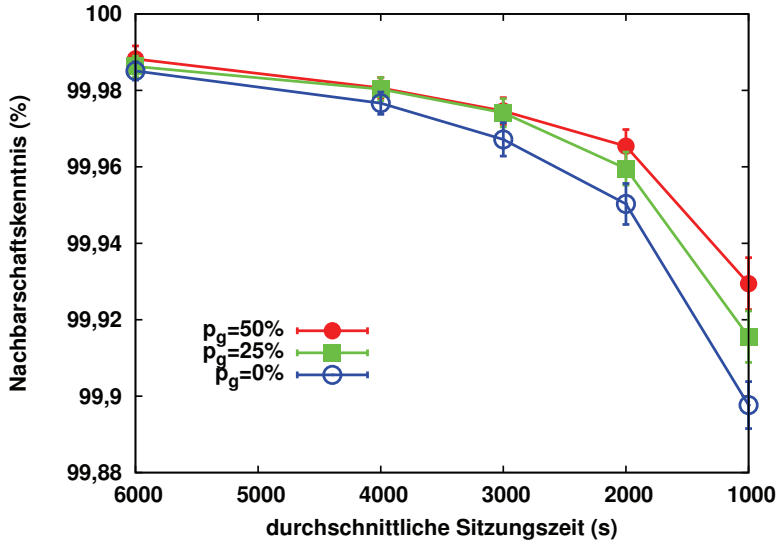
Um den Einfluss des Sitzungsverhaltens auf QuON zu evaluieren, wurden Simulationen mit verschiedenen Sitzungslängen und verschiedenen Wahrscheinlichkeiten von Teilnehmerausfällen durchgeführt. Dabei wurde die Sitzungslänge von der sonst verwendeten durchschnittlichen Sitzungszeit von 100 Minuten, also 6.000 Sekunden, auf 4.000, 3.000, 2.000 und 1.000 Sekunden verringert. Dabei wurde eine Ausfallwahrscheinlichkeit von 50%, 75% und 100% – entsprechend einer Wahrscheinlichkeit für ein geordnetes Abmelden der Teilnehmer nach Sitzungsende p_g von 50%, 25% und 0% – untersucht. Die Simulationsergebnisse dieser Szenarien sind in Abbildung 8.4 gezeigt. Die Nachrichtenverzögerung ist wie in den anderen Simulationen vom Sitzungsverhalten unabhängig und liegt in allen Simulationen zwischen 84 ms und 88 ms.

Knotenfluktuation übt einen Einfluss auf die Gruppenbildung aus: Bei hoher Fluktuation werden Teilnehmer öfter aus Gruppen entfernt, neu hinzukommende Teilnehmer befinden sich jedoch möglicherweise an einer anderen Stelle des Spielfelds und benötigen einige Zeit, bis sie ihre Gruppe erreichen. Somit ist bei hoher Fluktuation die beobachtete effektive Gruppengröße deutlich kleiner als bei geringer Fluktuation. Um eine hierdurch verursachte Verfälschung der Simulationsergebnisse zu vermeiden, wurde die Gruppenbildung für diese Simulationen deaktiviert und mit einem *Random Waypoint*-Modell als Bewegungsmodell simuliert. Die Spielfeldgröße betrug dabei 1.000 m mal 1.000 m.

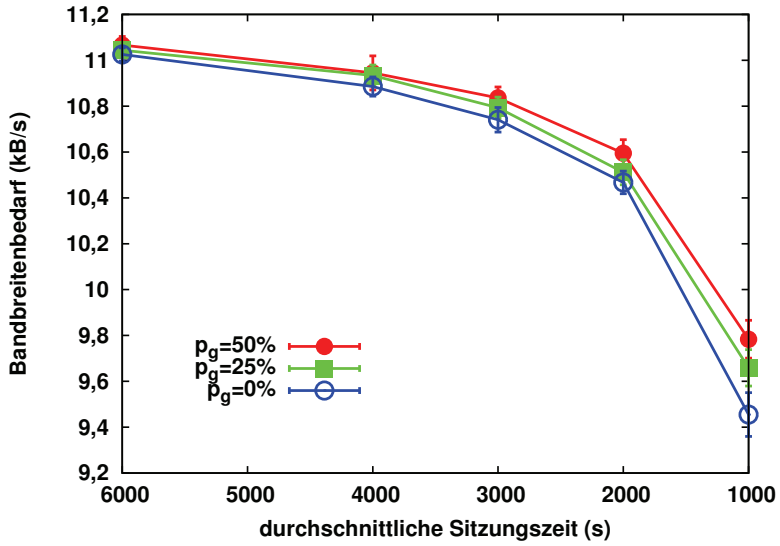
Die Abweichungen der Simulationsparameter von den in Tabelle 8.1 engegebenen beziehungsweise zusätzliche Parameter sind in Tabelle 8.3 gezeigt.

Parameter	Wert
Spielfeldgröße	1 km*1 km
Sitzungszeit	6.000 s, 4.000 s, 3.000 s, 2.000 s, 1.000 s
Graceful-leave	50%, 25%, 0%
max. Gruppengröße	1

Tabelle 8.3 Simulationsparameter Einfluss Sitzungszeit.



(a) Nachbarschaftskenntnis von QuON bei verschiedenen graceful-leave in Abhängigkeit der Sitzungslänge.



(b) Bandbreitenbedarf von QuON bei verschiedenen graceful-leave in Abhängigkeit der Sitzungslänge.

Abbildung 8.4 Evaluierungsergebnisse: Einfluss des Sitzungsverhaltens.

Abbildung 8.4a zeigt die Nachbarschaftskenntnis von QuON in Abhängigkeit von der Sitzungslänge bei verschiedenen Wahrscheinlichkeiten p_g für ein geordnetes Abmelden der Teilnehmer nach Sitzungsende. Erwartungsgemäß sinkt bei kurzen Sitzungszeiten die Nachbarschaftskenntnis. Bei geringen Wahrscheinlichkeiten für geordnetes Abmelden ist sie zudem schlechter als bei hohen Wahrscheinlichkeiten. Bei einer Sitzungslänge von 6.000 Sekunden erreicht QuON bei $p_g = 50\%$ eine Nach-

barschaftskenntnis von 99,988%, bei $p_g = 25\%$ eine Nachbarschaftskenntnis 99,985% und bei $p_g = 0\%$ eine Nachbarschaftskenntnis 99,984%. Bis zu einer Sitzungslänge von 2.000 Sekunden sinken die Werte leicht auf 99,966% für $p_g = 50\%$, 99,96% für $p_g = 25\%$ und 99,95% für $p_g = 0\%$. Bei noch kürzeren Sitzungslängen ist die Verschlechterung der Nachbarschaftskenntnis deutlicher auf 99,93% für $p_g = 50\%$, 99,92% für $p_g = 25\%$ und 99,9% für $p_g = 0\%$ bei einer Sitzungslänge von 1000 Sekunden. Insgesamt bleibt die Nachbarschaftskenntnis damit aber im guten Bereich. Auch bei einer extrem kurzen durchschnittlichen Sitzungszeit von 1000 Sekunden und einer extrem geringen Abmeldewahrscheinlichkeit p_g von 0% sinkt die Nachbarschaftskenntnis nicht unter 99,9%. Bei vergleichbaren Protokollen wie Vast liegt in dem untersuchten Szenario bei einer Sitzungslänge von 6000 Sekunden und einer Abmeldewahrscheinlichkeit p_g von 50% die Nachbarschaftskenntnis bereits unter 97%[3].

In Abbildung 8.4b ist der Bandbreitenbedarf von QuON in Abhängigkeit von der Sitzungslänge bei verschiedenen Wahrscheinlichkeiten für ein geordnetes Abmelden der Teilnehmer nach Sitzungsende dargestellt. Dabei sinkt der Bandbreitenbedarf mit der durchschnittlichen Sitzungslänge und steigt mit einer höheren Wahrscheinlichkeit für geordnetes Abmelden. Bei einer Sitzungslänge von 6.000 Sekunden wird eine Bandbreite von 11,06 kByte/s bei $p_g = 50\%$, 11,04 kByte/s bei $p_g = 25\%$ und 11,02 kByte/s bei $p_g = 0\%$ benötigt. Bei einer durchschnittlichen Sitzungslänge von 1.000 Sekunden beträgt der Bandbreitenbedarf 9,78 kByte/s bei $p_g = 50\%$, 9,66 kByte/s bei $p_g = 25\%$ und 9,46 kByte/s bei $p_g = 0\%$.

Die Ursache für den geringeren Bandbreitenbedarf bei kürzeren durchschnittlichen Sitzungszeiten und geringeren Wahrscheinlichkeiten für eine geordnetes Abmelden ist, dass beide Faktoren dazu führen, dass ein Teilnehmer häufiger seine Verbindungsnachbarn verliert. Die durchschnittliche Zahl der Verbindungsnachbarn ist auch in Abbildung A.7 in Anhang A.3 dargestellt. Bei einer geringeren Zahl von Verbindungsnachbarn muss ein Teilnehmer seine Positionsmeldungen an weniger Nachbarn senden und spart so Bandbreite. Allerdings kann es bei einem fehlenden Verbindungsnachbarn zu Unzuverlässigkeiten in der Nachbarsfindung kommen. Dies führt, wie oben dargelegt, zu Einbußen der Nachbarschaftskenntnis.

Dabei bleibt die Nachbarschaftskenntnis insgesamt gut. Auch bei einer unrealistisch kurzen durchschnittlichen Sitzungszeit von 1.000 Sekunden (16 1/3 Minuten) und einer unrealistisch geringen Wahrscheinlichkeit für ein geordnetes Abmelden von 0% unterschreitet die Nachbarschaftskenntnis 99,9% nicht. In Studien werden bei Nutzern virtueller Welten deutlich höhere durchschnittliche Sitzungszeiten ermittelt [30, 110, 133, 134]. Die geringste in diesen Studien gemessene durchschnittliche Sitzungszeit lag bei 54 Minuten [133].

8.1.8 Vergleich mit bisherigen Protokollen

Zum Abschluss der simulativen Evaluierung wurde QuON mit den in Kapitel 5 untersuchten bisherigen Protokollen verglichen. Hierbei wurde, analog zu der Evaluierung in Abschnitt 5.2, der Einfluss der Teilnehmerdichte auf die Protokolle untersucht. Da SimMUD und N-Tree aufgrund ihrer schlechten Nachbarschaftskenntnis nicht für einen Einsatz in virtuellen Welten geeignet sind[4], wurden hier für den Vergleich nur

[3] Ein ausführlicherer Vergleich von QuON mit bisherigen Protokollen findet sich in Abschnitt 8.1.8.

[4] Siehe hierzu Abschnitt 5.2.4.3.

Vast und PubSubMMOG herangezogen. Die Simulationsparameter entsprechen den in Tabelle 8.1 angegebenen Werten. Die Simulationsergebnisse sind in Abbildung 8.5 dargestellt. Da die Untersuchung der Spielfeldgröße 10.000 m mal 10.000 m im Vergleich zur Spielfeldgröße 5.000 m mal 5.000 m keine qualitativ neuen Ergebnisse zeigt, werden die Ergebnisse der Spielfeldgröße 10.000 m mal 10.000 m zur Verbesserung der Übersichtlichkeit hier in den Abbildungen nicht gezeigt. Sie finden sich in Abbildung A.8 in Anhang A.3.

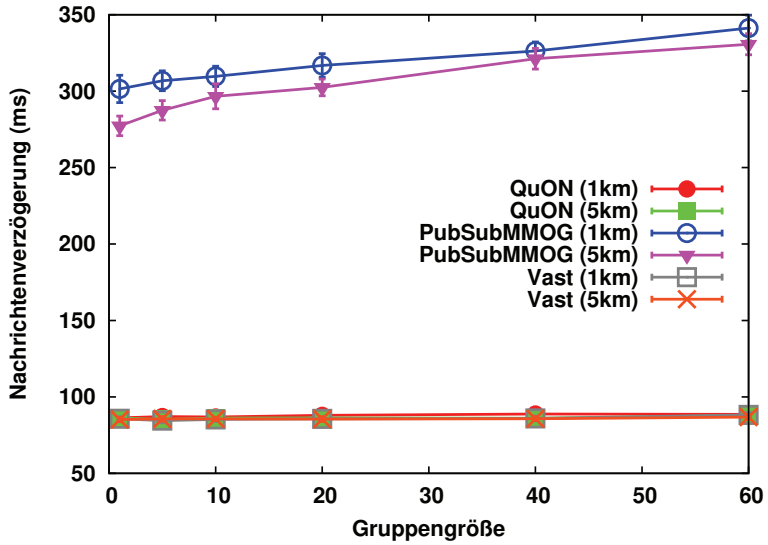
Abbildung 8.5a zeigt die Nachrichtenverzögerung der Protokolle in Abhängigkeit von der Spielfeldgröße. In QuON und Vast werden die Positionsmeldungen direkt vom Absender an die direkten Nachbarn versendet. Die dabei beobachteten Nachrichtenverzögerungen hängen im Wesentlichen vom Transport der Nachrichten durch das unterliegende Netzwerk ab. Sie sind also nicht wesentlich durch die Teilnehmerdichte beeinflusst. Sowohl bei QuON als auch bei Vast beträgt die beobachtete Nachrichtenverzögerungen bei allen betrachteten Spielfeldgrößen weniger als 90 ms.

In PubSubMMOG sind die Nachrichtenverzögerungen deutlich größer. So benötigen Positionsmeldungen bei Spielfeldgröße 5.000 m mal 5.000 m bei der Gruppengröße 1 277 ms. Die Nachrichtenverzögerung steigt mit steigender Gruppengröße und erreicht bei einer Gruppengröße von 60 330 ms. Die Nachrichtenverzögerungen bei der Spielfeldgröße 1.000 m mal 1.000 m liegen in PubSubMMOG bei der Gruppengröße 1 bei 301 ms, und bei der Gruppengröße 60 bei 340 ms. Der Anstieg der Latenzen bei höherer Teilnehmerdichte wird, wie in Abschnitt 5.1.2.3 erläutert, durch die in PubSubMMOG eingesetzten Lastverteilbäume verursacht. Abhängig von der Teilnehmerdichte benötigt PubSubMMOG die drei- bis vierfache Zeit der anderen Protokolle, um Positionsmeldungen zuzustellen. Die von PubSubMMOG erzeugten Nachrichtenverzögerungen liegen damit oberhalb der für virtuelle Welten empfohlenen maximalen Verzögerung von 200 ms [126].

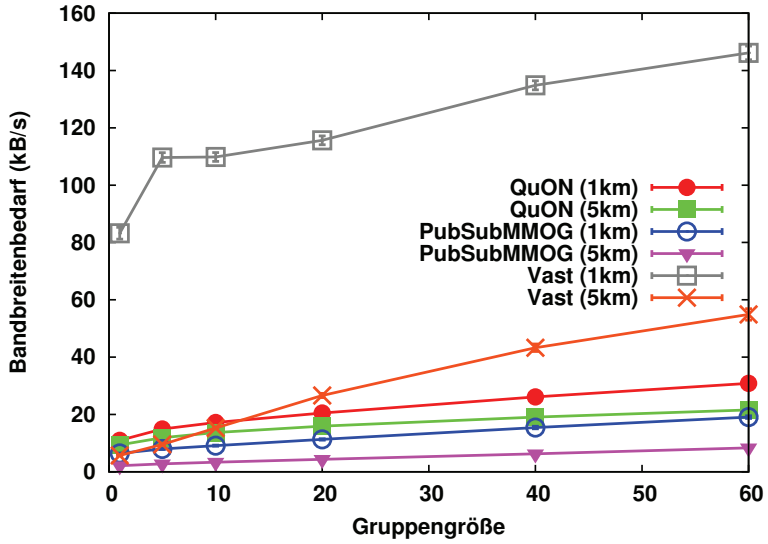
In Abbildung 8.5b ist der Bandbreitenbedarf der Protokolle in Abhängigkeit von der Spielfeldgröße gezeigt. QuON benötigt bei der Spielfeldgröße 5.000 m mal 5.000 m bei der Gruppengröße 1 9 kByte/s. Bei steigender Gruppengröße steigt der Bandbreitenbedarf auf 22 kByte/s bei einer Gruppengröße von 60. Bei der Spielfeldgröße 1.000 m mal 1.000 m ist der Bandbreitenbedarf aufgrund der höheren Teilnehmerdichte und damit einhergehenden größeren Zahl an direkten Nachbarn höher. Bei der Gruppengröße 1 beträgt er ungefähr 11 kByte/s. Bei der Gruppengröße von 60 erreicht er 31 kByte/s.

Der Bandbreitenbedarf von Vast ist bei fast allen Gruppengrößen deutlich höher als der Bandbreitenbedarf von QuON. Bei einer Spielfeldgröße von 5.000 m mal 5.000 m liegt er mit 5,7 kByte/s noch unter dem Wert von QuON. Bei Gruppengrößen von mehr als 10 ist der Bandbreitenbedarf von Vast jedoch größer als der von QuON. Bei einer Gruppengröße von 60 steigt der Bedarf von Vast auf 55 kByte/s und übertrifft dabei den entsprechenden Bedarf von QuON um über 150%. Bei der Spielfeldgröße von 1.000 m mal 1.000 m liegt der Bandbreitenbedarf von Vast deutlich höher als der der anderen Protokolle. Hier wird bei der Gruppengröße von 1 83 kByte/s benötigt, bei der Gruppengröße 60 146 kByte/s und damit über 370% mehr als QuON. Eine genauere Analyse des Bandbreitenbedarfs von Vast findet sich in Abschnitt 5.1.4.3.

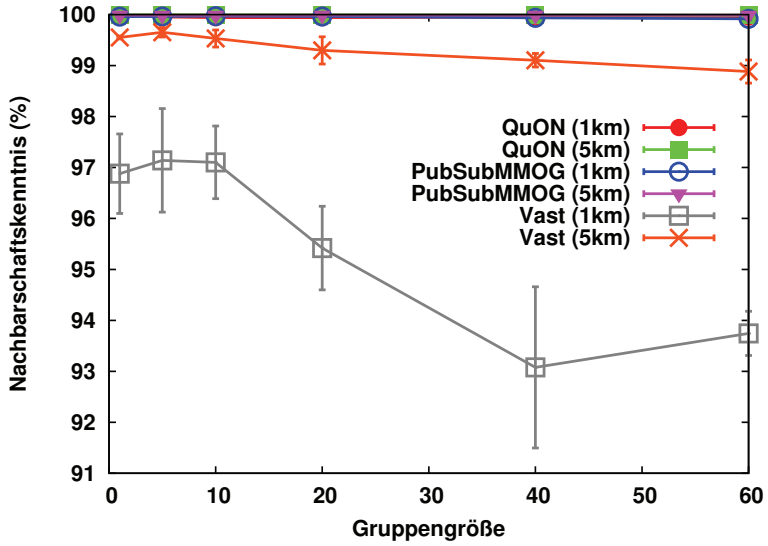
PubSubMMOG benötigt von den hier betrachteten Protokollen die geringste Bandbreite. Bei einer Spielfeldgröße von 5.000 m mal 5.000 m werden bei Gruppengröße



(a) Nachrichtenverzögerung bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.



(b) Bandbreitenbedarf bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.



(c) Nachbarschaftskenntnis bei verschiedenen Spielfeldgrößen in Abhängigkeit der Gruppengröße.

Abbildung 8.5 Vergleich von QuON mit bisherigen Protokollen.

1 2,1 kByte/s benötigt, bei einer Gruppengröße von 60 8,3 kByte/s, etwa 40% des Bandbreitenbedarfs von QuON. Bei der Spielfeldgröße 1.000 m mal 1.000 m benötigt PubSubMMOG bei einer Gruppengröße von 1 6,4 kByte/s und bei einer Gruppengröße von 60 19 kByte/s, was 60% des Bandbreitenbedarfs von QuON entspricht.

Damit benötigt PubSubMMOG signifikant weniger Bandbreite als QuON. Der relative Vorteil von PubSubMMOG sinkt aber mit steigender Teilnehmerdichte. Gängige ADSL-Anschlüsse bieten genug Ressourcen für den Bandbreitenbedarf beider Protokolle. Der Bandbreitenbedarf von Vast überfordert hingegen übliche Breitbandanschlüsse.

Abbildung 8.5c zeigt die Nachbarschaftskenntnis der Protokolle in Abhängigkeit von der Spielfeldgröße. Die von QuON und PubSubMMOG erreichte Nachbarschaftskenntnis bleibt in allen betrachteten Szenarien über 99,9%. Bei der Spielfeldgröße 5.000 m mal 5.000 m liegt die Nachbarschaftskenntnis von QuON und PubSubMMOG bei allen Gruppengrößen über 99,98%, bei der Spielfeldgröße von 1.000 m mal 1.000 m ist der schlechteste Wert von QuON 99,94% bei einer Gruppengröße von 10, der schlechteste Wert von PubSubMMOG 99,91% bei einer Gruppengröße von 60.

Die Nachbarschaftskenntnis von Vast bleibt stets unter diesen Werten. Bei der Spielfeldgröße 5.000 m mal 5.000 m erreicht Vast bei einer Gruppengröße von 1 eine Nachbarschaftskenntnis von 99,5%. Diese sinkt mit größerer Gruppengröße bis auf 98,8% bei einer Gruppengröße von 60. Bei der Spielfeldgröße 1.000 m mal 1.000 m erreicht Vast bei Gruppen kleiner als 10 eine Nachbarschaftskenntnis von ungefähr 97%. Bei größeren Gruppen sinkt dieser Wert auf 93% bis 94%. Dabei streuen die Ergebnisse stärker als bei den anderen Protokollen.

Die Nachbarschaftskenntnis von PubSubMMOG und QuON ist in allen betrachteten Szenarien gut. Bei Vast ist die Nachbarschaftskenntnis insbesondere bei hohen Teilnehmerdichten deutlich schlechter. Einige Teilnehmer müssten bei Vast mit einer spürbaren Einschränkung ihrer Interaktionsmöglichkeiten rechnen.

Zusammengefasst bietet QuON gute Nachrichtenverzögerung und Nachbarschaftskenntnis bei mittlerem Bandbreitenbedarf. PubSubMMOG bietet zwar eine gute Nachbarschaftskenntnis und einen guten Bandbreitenbedarf, führt allerdings zu hohen Nachrichtenverzögerungen. Vast kann gute Nachrichtenverzögerungen bieten, weist jedoch eine nur mittlere Nachbarschaftskenntnis auf und benötigt deutlich zu viel Bandbreite. Da Nachrichtenverzögerung und Nachbarschaftskenntnis in der Regel wichtigere Kriterien für virtuelle Welten sind als der Bandbreitenbedarf (siehe hierzu Abschnitt 2.1.3), ist QuON somit den anderen hier betrachteten Protokollen überlegen.

8.1.9 Schlussfolgerungen

Die Ergebnisse der simulativen Evaluierung bestätigen die Aufwandsabschätzung in Abschnitt 6.10. In allen untersuchten Szenarien kann QuON eine sehr gute Nachrichtenverzögerung von weniger als 90 ms bieten. Diese Latenz ist unabhängig von Teilnehmerdichte und Sitzungsverhalten der Teilnehmer. Dabei werden gute Werte für die Nachbarschaftskenntnis erreicht. In allen untersuchten Szenarien bleibt die Nachbarschaftskenntnis von QuON oberhalb von 99,9%. Der Bandbreitenbedarf ist dabei proportional zur Zahl der Nachbarn und somit der Teilnehmerdichte. Er bleibt

aber auch bei Szenarien mit hoher Teilnehmerdichte unterhalb von 35 kByte/s und damit für gängige Internetanschlüsse beherrschbar.

Auch bei hoher Teilnehmerfluktuation bleibt die von QuON gebildete virtuelle Welt stabil. Teilnehmerausfälle können von den Backup-Nachbarn kompensiert werden.

Im Vergleich zu bisherigen Protokollen zeigt sich QuON den anderen untersuchten Protokollen deutlich überlegen. Es kann eine sehr gute Nachrichtenverzögerung mit einer sehr guten Nachbarschaftskenntnis verbinden und erreicht in beiden Metriken die Werte der jeweils besten untersuchten bisherigen Protokolle.

8.2 Evaluierung im Forschungsnetz

Um zu zeigen, dass sich die Simulationsergebnisse von QuON auch auf physische Netze übertragen lassen, wurde QuON im Forschungsnetz G-Lab [123, 136] evaluiert.

8.2.1 Auswahl des Forschungsnetzes

Das Forschungsnetz G-Lab besteht aus einem Netz von etwa 160 Knoten, welche an sechs deutschen Universitäten lokalisiert sind. Die Standorte sind mit Gigabit-Ethernet an den DFN-Backbone angeschlossen. Hierdurch ergeben sich sehr geringe Nachrichtenverzögerungen. Eine Evaluierung in G-Lab entspricht deshalb eher einer Proof-of-concept-Implementierung.

Ein anderes Testnetz, welches nicht wie G-Lab unrealistisch gute Bedingungen bietet, ist PlanetLab [18, 109]. PlanetLab besteht aus über 1000 Knoten, die weltweit im Internet an über 500 Standorten verteilt sind. Die Netzknoten in PlanetLab sind jedoch in der Regel überlastet. Aufgrund der Überlastung kommt es teilweise zu Zeiträumen von mehreren hundert bis wenigen tausend Millisekunden, in denen einem Prozess, der auf einem PlanetLab-Knoten läuft, keine Rechenzeit zugewiesen wird. Die hierdurch resultierenden Verzögerungen sind so hoch, dass für latenzsensitive Anwendungen wie virtuelle Welten keine aussagekräftigen Messwerte zu ermitteln sind. Aus diesem Grund wurde die in den folgenden Abschnitten vorgestellte Evaluierung nur im Forschungsnetz G-Lab durchgeführt.

8.2.2 Szenario

Zum Zeitpunkt der Evaluierung waren im Forschungsnetz 137 Knoten für die Evaluierung verfügbar. Auf jedem dieser Knoten wurde eine Instanz von OverSim gestartet, in der jeweils ein QuON-Teilnehmer erzeugt wurde. Diese Teilnehmer wurden über die OverSim-Echtnetzanbindung mit dem *SingleHost-Underlay*[5] miteinander verbunden.

Im Zusammenspiel mit dem SimpleGameClient bildeten die 137 Teilnehmer eine gemeinsame virtuelle Welt. Da aufgrund des Fehlens einer zentralen Koordinationsinstanz für die Bildung von Teilnehmergruppen das *GroupRoaming*-Modell nicht für die Steuerung der Teilnehmer eingesetzt werden konnte, wurde stattdessen das *HotspotRoaming*-Modell verwendet.

Mit diesem Modell wurden auf dem 300 m mal 300 m Meter großen Spielfeld vier Hotspots definiert. Nach einer Verweilzeit von 5 Sekunden in einem Hotspot oder

[5] Siehe hierzu auch Abschnitt 4.9.

beim Erreichen eines Ziels außerhalb des Hotspots wählte sich ein Teilnehmer ein neues Ziel. Dieses Ziel befand sich mit einer Wahrscheinlichkeit von 60% in einem zufällig gewählten Hotspot und mit einer Wahrscheinlichkeit von 40% außerhalb eines Hotspots. Durch die Wahl des Hotspots ergaben sich Stellen mit höherer Teilnehmerdichte auf dem Spielfeld. Dabei wurde nicht wie bei der in Abschnitt 6.9.3 durchgeführten Evaluierung eine Überlastung der Teilnehmer in den Hotspots herbeigeführt; die mit dem Szenario erreichte Teilnehmerdichte ist vergleichbar mit der höchsten der in den Simulationen untersuchten Teilnehmerdichte, die bei einer Spielfeldgröße von 1.000 m mal 1.000 m bei einer maximalen Gruppengröße von 60 erreicht wird.

Die weitere Parametrisierung des SimpleGameClients und von QuON entspricht den in der Simulation betrachteten Szenarien. Das Interessengebiet der Teilnehmer war ein Kreis um die aktuelle Position des Teilnehmers mit dem Radius 50 m. Die Bewegungsgeschwindigkeit der Teilnehmer betrug 5 m/s. Jeder Teilnehmer generierte pro Sekunde 6 Positionsmeldungen.

8.2.3 Ergebnisse

In der Evaluierung wurde von den einzelnen Teilnehmern die Verzögerung der Positionsmeldungen, der Bandbreitenbedarf und die Nachbarschaftskenntnis gemessen. Um ohne zentrale Instanz die Nachbarschaftskenntnis bestimmen zu können, legten die Teilnehmer zu festgelegten Zeiten ihre Position sowie die Adressen aller ihrer Nachbarn in einer Datei ab. Anhand dieser Dateien konnte dann eine Topologie der virtuellen Welt erstellt und damit die Nachbarschaftsbeziehungen der Teilnehmer bestimmt werden. Allerdings ist diese Messmethode davon abhängig, dass die Erstellung der Dateien von allen Teilnehmern möglichst exakt zur selben Zeit durchgeführt wird. Hierzu wurden die Netzknoten in G-Lab mittels NTP [97] synchronisiert.

Während der Evaluierung fiel ein Netzknoten von G-Lab aus, sodass nur die Ergebnisse der verbleibenden 136 Knoten verwendet werden konnten.

In Abbildung 8.6 ist der Verlauf der Nachbarschaftskenntnis in der virtuellen Welt über den Verlauf von 22 ⅓ Stunden aufgezeichnet. Dabei lag der Durchschnitt der Nachbarschaftskenntnis bei 99,97% und damit nahe an den bei der simulativen Evaluierung ermittelten Werten. Nur an vier von 268 Messpunkten war die Nachbarschaftskenntnis kleiner als 100%. Zum Zeitpunkt 290 fehlte bei fünf Teilnehmern Wissen über je einen Nachbarn. Dies lässt sich durch den oben erwähnten Knotenausfall erklären. Bei den anderen drei Messpunkten, an denen die Nachbarschaftskenntnis unter 100% lag, fehlte je einmal einem, einmal zwei und einmal drei Teilnehmern Wissen über je einen Nachbarn.

Die durchschnittliche Nachrichtenverzögerung über die gesamte Zeit der Evaluierung und über alle Teilnehmer lag bei 7,2 ms, der durchschnittliche Bandbreitenbedarf bei 57,4 kByte/s.

Der Speicherbedarf für OverSim lag bei ungefähr 12 MB. Die CPU-Auslastung der Knoten lag bei weniger als 1%.

8.2.4 Bewertung der Ergebnisse

Bei der Bewertung der Evaluierungsergebnisse ist zu beachten, dass sich die Netzknoten in G-Lab nur in wenigen Standorten in Deutschland befinden und über einen

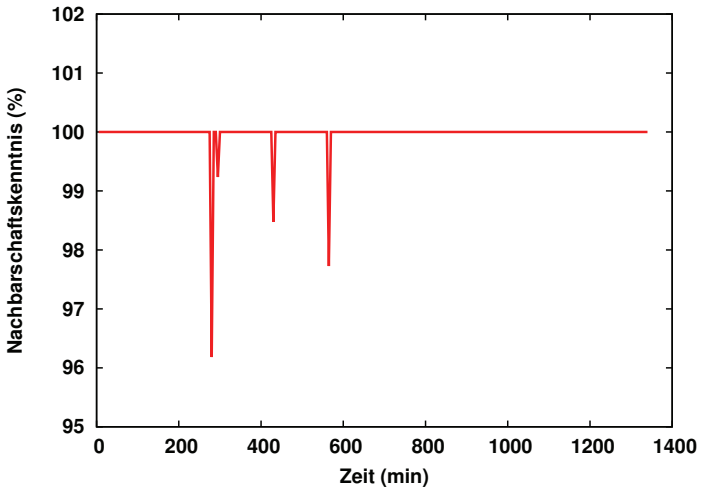


Abbildung 8.6 Verlauf der Nachbarschaftskenntnis.

überdurchschnittlich guten Netzzugang verfügen. Aufgrund dieser Topologie sind die beobachteten Nachrichtenverzögerungen deutlich geringer als bei einer globalen Verteilung der Teilnehmer. Weiterhin sind Teilnehmerausfälle in G-Lab verhältnismäßig selten. Im Verlauf der Evaluierung fiel genau ein Netzknoten in G-Lab aus.

Dennoch zeigen die Ergebnisse, dass QuON in physischen Netzen bei einem geringen Bandbreitenbedarf eine sehr gute Nachbarschaftskenntnis erreichen kann. Die guten Messwerte in G-Lab belegen, dass sich die in der Simulation gewonnenen Ergebnisse auch in physischen Netzen erreichen lassen. Der geringe Speicherbedarf und die geringe CPU-Last zeigt, dass auch ressourcenschwache Endgeräte wie beispielsweise Smartphones in der Lage sind, an einer QuON-basierten virtuellen Welt teilzunehmen.

8.3 Zusammenfassung

In diesem Kapitel wurde QuON in verschiedenen Szenarien simulativ und im Forschungsnetz G-Lab evaluiert. In den Evaluierungen wurde gezeigt, dass QuON eine sehr gute Nachrichtenverzögerung und eine sehr gute Nachbarschaftskenntnis bieten kann. Dabei ist die Nachrichtenverzögerung von Sitzungsverhalten und Teilnehmerdichte unabhängig. Die Nachbarschaftskenntnis sinkt bei hoher Teilnehmerdichte, hoher Teilnehmerfluktuation und hohen Ausfallraten, bleibt aber in allen untersuchten Szenarien über 99,9%. Der Bandbreitenbedarf bleibt auch bei hohen Teilnehmerdichten gering genug, um übliche Netzanschlüsse nicht zu überlasten.

Im Vergleich zu bisherigen P2P-Protokollen für virtuelle Welten ist QuON den anderen untersuchten Protokollen überlegen, da kein anderes untersuchtes Protokoll eine gute Nachrichtenverzögerung mit einer guten Nachbarschaftskenntnis und mäßigem Bandbreitenbedarf verbinden kann.

Auch in physischen Netzen können, wie die Untersuchung im Forschungsnetz G-Lab zeigt, die guten Werte der Simulation erreicht werden. QuON kann somit die in Abschnitt 2.1.3 beschriebenen Anforderungen voll erfüllen.

9. Zusammenfassung und Ausblick

Virtuelle Welten sind grafische virtuelle Umgebungen, in denen eine große Zahl an Teilnehmern miteinander interagieren können. Die Teilnehmer werden in der virtuellen Welt durch einen sogenannten Avatar repräsentiert. Die Interaktionsmöglichkeiten der Teilnehmer basieren in der Regel auf der Position dieses Avatars in der virtuellen Welt.

Bislang verwenden virtuelle Welten üblicherweise Client/Server-basierte Protokolle. Bei diesen Protokollen wird die maximale Zahl unterstützter Teilnehmer durch die verfügbare Serverkapazität begrenzt. Kommerzielle virtuelle Welten nutzen verschiedene Techniken wie Sharding, Instanziierung und Zonen, um eine größere Zahl von Teilnehmern unterstützen zu können. Teilweise werden auch komplexe Proxy-Architekturen verwendet, die die verfügbaren Ressourcen der Server besser auf die Teilnehmer aufteilen können.

Wie in dieser Arbeit dargelegt wurde, kann jedoch keine dieser Techniken eine zufriedenstellende Skalierbarkeit bieten. Zur Unterstützung einer wachsenden Teilnehmerzahl muss stets die verfügbare Serverkapazität gesteigert werden. Peer-to-Peer-(P2P-)Protokolle sind eine Möglichkeit, diese Skalierungsprobleme zu überwinden. In P2P-Protokollen werden Aufgaben nicht von zentralen Servern, sondern von den Teilnehmern selber übernommen. Die verfügbaren Ressourcen steigen also mit der Zahl der Teilnehmer. Ist der Ressourcenbedarf pro Teilnehmer wie bei QuON geringer als der Ressourcenzuwachs, sind die Voraussetzungen für skalierbare virtuelle Welten gegeben.

Virtuelle Welten stellen neben der Skalierbarkeit noch weitere wichtige Anforderungen an das verwendete Protokoll. Die im Rahmen der Arbeit betrachteten Anforderungen sind eine geringe Nachrichtenverzögerung, eine hohe Nachbarschaftskenntnis, ein geringer Bandbreitenbedarf und eine Cheating-Erkennung. Bisherige P2P-Protokolle können nicht alle dieser Anforderungen zugleich erfüllen.

9.1 Ergebnisse der Arbeit

Um bisherige P2P-Protokolle für virtuelle Welten evaluieren zu können und um die Entwicklung neuer Protokolle zu vereinfachen, war die Entwicklung eines Simulationswerkzeugs nötig. In Zusammenarbeit mit Kollegen entstand deshalb das Simulationsrahmenwerk *OverSim*. OverSim erfüllt wichtige Anforderungen wie Skalierbarkeit, Flexibilität, realistische Netzmodellierung und eine realistische Modellierung von Nutzer- und Anwendungsverhalten besser als bisherige Simulationswerkzeuge. Weitere wichtige von OverSim gebotene Eigenschaften, die von bisherigen Simulationswerkzeugen oft vernachlässigt werden, sind Echtnetzanbindung, die Erstellung aussagekräftiger Statistiken, Visualisierung, eine ausführliche Dokumentation und eine einfache Erweiterbarkeit.

Besonderen Wert bei der Entwicklung von OverSim wurde auf ein modulares Design gelegt. Dadurch lassen sich einzelne Komponenten einer Simulation, wie beispielsweise die Netzabstraktion, transparent durch andere Module ersetzen. Dies ermöglicht eine flexible Anpassung von OverSim an die jeweiligen Anforderungen verschiedener Simulationen.

OverSim wurde als Open-Source-Software auf der Webseite www.oversim.org veröffentlicht und wird inzwischen international von Forschungseinrichtungen für die Entwicklung und Evaluierung von Overlayprotokollen eingesetzt.

Auf der Basis von OverSim wurden in dieser Arbeit bisherige Protokolle für virtuelle Welten evaluiert und miteinander verglichen. Bislang fehlte in der Literatur eine solche vergleichende Untersuchung.

Bisherige P2P-Protokolle für virtuelle Welten lassen sich in die Kategorien vollvermaschte Protokolle, zonenbasierte Protokolle und zonenlose Protokolle unterteilen. Die zonenbasierten Protokolle lassen sich weiter in ALM-basierte Protokolle, Supernode-basierte Protokolle und Protokolle mit dynamischen Zonen unterteilen. Während vollvermaschte Protokolle nicht skalieren und somit für einen Einsatz in virtuellen Welten nicht geeignet sind, unterstützen die anderen Kategorien prinzipiell beliebig viele Teilnehmer, da die Teilnehmerlast hier nur von der Zahl der Teilnehmer in der Umgebung der virtuellen Welt und nicht von der Gesamtzahl der Teilnehmer abhängt.

Aus jeder der Kategorien wurde nun ein bisheriges Protokoll – im Einzelnen SimMUD, PubSubMMOG, N-Tree und Vast – ausgewählt und mit OverSim evaluiert. Als Ergebnis der Evaluierung wurde festgestellt, dass Protokolle mit statischen Zonen die hohen Anforderungen virtueller Welten an die Nachrichtenverzögerung nicht uneingeschränkt erfüllen können. SimMUD als ALM-basiertes Protokoll benötigte für die Zustellung von Positionsmeldungen ungefähr 200 ms, das Supernode-basierte Protokoll PubSubMMOG sogar bis zu 350 ms. Verzögerungen von 200 ms können von Spielern als Einschränkung empfunden werden [126].

Protokolle mit dynamischen Zonen können die erforderliche Nachbarschaftskenntnis nicht gewährleisten. In einigen Szenarien wurde bei dem Protokoll N-Tree eine Nachbarschaftskenntnis von unter 60% beobachtet.

Zonenlose Protokolle könnten die Anforderungen virtueller Welten erfüllen, bisherige Voronoi-basierte Protokolle zeigen jedoch einen zu hohen Bandbreitenbedarf. Bei

hoher Teilnehmerdichte wurde bei Vast ein durchschnittlicher Bandbreitenbedarf von mehr als 140 kByte/s gemessen.

Da keines der untersuchten Protokolle die Anforderungen virtueller Welten voll erfüllen konnte, wurde das neue Protokoll *QuON* entwickelt. Dieses verbindet die gute Nachrichtenverzögerung zonenloser Protokolle mit einer sehr guten Nachbarschaftskenntnis und einem im Vergleich zu bisherigen zonenlosen Protokollen deutlich geringeren Bandbreitenbedarf.

QuON ist ein zonenloses sogenanntes „Mutual-Notification"-Protokoll. Bei diesen Protokollen haben alle Teilnehmer eine direkte Verbindung zu allen Teilnehmern, mit denen sie interagieren können. Das Finden neuer Nachbarn geschieht über Benachrichtigungen dieser direkten Nachbarn oder zusätzlicher Verbindungsnachbarn, die zusätzlich den Netz-Zusammenhalt garantieren.

Durch die Verbindungen zu den direkten Nachbarn können Ereignisnachrichten und Positionsnachrichten in QuON mit sehr geringer Nachrichtenverzögerung zugestellt werden. In den Simulationen entsprach die durchschnittliche Nachrichtenverzögerung von QuON mit knapp 90 ms stets der durchschnittlichen Ende-zu-Ende-Verzögerung zwischen den Teilnehmern im verwendeten SimpleUnderlay-Modell.

Die Wahl der Verbindungsnachbarn gewährleistet eine sehr gute Nachbarschaftskenntnis. Mittels spezieller Backup-Nachbarn kann auch bei Knotenfluktuation die Nachbarschaftskenntnis gewahrt bleiben. Auch bei sehr hoher Teilnehmerfluktuation mit durchschnittlichen Sitzungszeiten von 1.000 Sekunden, wobei keiner der Teilnehmer nach Ablauf der Sitzung eine geordnete Abmeldung durchführte sondern durch einen simulierten *fail-stop*-Ausfall aus dem Netz entfernt wurde, blieb die Nachbarschaftskenntnis bei 99,9%. Von den untersuchten bisherigen Protokollen konnte nur PubSubMMOG eine vergleichbare Nachbarschaftskenntnis aufweisen.

Der Protokoll-Overhead für die Sicherung des Netzzusammenhaltes und die Nachbarsfindung in QuON bleiben verhältnismäßig gering. Der Bandbreitenbedarf von QuON bleibt so deutlich unter den Werten bisheriger zonenloser Protokolle. So benötigt das Protokoll Vast bei hoher Teilnehmerdichte mehr als 140 kByte/s. In demselben Szenario werden von QuON nur 31 kByte/s benötigt.

Die Evaluierungsergebnisse sind in Tabelle 9.1 zusammengefasst. Von den untersuchten Protokollen kann nur QuON zugleich die Anforderunge der geringen Nachrichtenverzögerung, der hohen Nachbarschaftskenntnis und des geringen Bandbreitenbedarfs erfüllen.

In einer Evaluierung im Testnetz G-Lab konnten die guten Ergebnisse der Simulation bestätigt werden: In G-Lab lag die Nachbarschaftskenntnis bei 99,97%, die Nachrichtenverzögerung aufgrund der guten Vernetzung der in G-Lab eingesetzten Knoten bei 7,2 ms. Der Bandbreitenbedarf lag bei höherer Teilnehmerdichte als in den Simulationen bei 57,4 kByte/s. Die geringe Speichernutzung von 12 MB und die geringe CPU-Last der G-Lab-Knoten von etwa 1% zeigen, dass die Anforderungen von QuON auch von ressourcenschwachen Geräten erfüllt werden können.

Ein mögliches Problem bei P2P-Protokollen für virtuelle Welten sind durch Hotspots, also Bereiche der virtuellen Welt mit besonders hoher Teilnehmerdichte, hervorgerufene Überlastsituationen. Die Überlast kann in QuON vermieden werden, indem die Teilnehmer bei drohender Überlast dynamisch ihre Interessengebiete verklei-

Protokoll	Art	Nachrichtenverzögerung	Nachbarschaftskenntnis	Bandbreitenbedarf
QuON	zonenlos	⊕	⊕	◯
SimMUD	Zonen, ALM	⊖	⊖	⊕
PubSubMMOG	Zonen, Supernode	⊖	⊕	⊕
N-Tree	dynamische Zonen	⊕	⊖	◯
Vast	zonenlos	⊕	◯	⊖

Tabelle 9.1 Vergleich von QuON mit bisherigen P2P-Protokollen für virtuelle Welten.

nern. Abhängig von der Wahl des Adaptionsalgorithmus kann so eine hohe Bandbreiteneinsparung in Hotspots erzielt werden, ohne die Nachbarschaftskenntnis deutlich zu beeinträchtigen. In der Simulation konnte der durchschnittliche Bandbreitenbedarf in Hotspots um 75% gesenkt werden. Dabei war die Nachbarschaftskenntnis nur 0,8 Prozentpunkte schlechter als ohne dynamische Adaption der Interessengebiete.

Ein weiteres, von bisherigen P2P-Protokolle oft vernachlässigtes, Problem ist der Schutz der virtuellen Welt vor Cheatern – unehrlichen Teilnehmern, die sich durch Regelbruch Vorteile gegenüber anderen verschaffen wollen. Mit einer verteilten Cheating-Erkennung können Cheater in einer virtuellen Welt gefunden und aus der Welt entfernt werden. Dabei müssen keine Einbußen in der Skalierbarkeit oder der Nachrichtenverzögerung hingenommen werden.

Das Kernstück der Cheat-Erkennung ist die gegenseitige Überwachung der Teilnehmer. Um den Einfluss von Interessenkonflikten zu mindern, werden die Aktionen der Teilnehmer nicht nur von ihren Nachbarn, sondern auch von unabhängigen Beobachtern überprüft. Anschuldigungen des Beobachters wirken sich stärker aus als Anschuldigungen von Nachbarn. Wird ein Teilnehmer, dessen Vertrauenswert unterhalb eines Schwellenwertes liegt, von einem vertrauenswürdigen Beobachter des Cheatings bezichtigt, wird der mutmaßliche Cheater von der virtuellen Welt ausgeschlossen.

In Simulationen konnte gezeigt werden, dass mit der hier beschriebenen Cheat-Erkennung Cheater schnell und zuverlässig erkannt und bestraft werden können. Die Cheating-Verhalten „Überschreitung der Maximalgeschwindigkeit“ und „Ignorieren von Hindernissen“ wurden in allen Fällen erkannt, das Cheating-Verhalten „Täuschen des Beobachters“ mit einer Wahrscheinlichkeit von mehr als 98%. Die Erkennungszeiten lagen bei wenigen Sekunden bei den Cheating-Verhalten „Überschreitung der Maximalgeschwindigkeit“ und „Ignorieren von Hindernissen“, beim Cheating-Verhalten „Täuschen des Beobachters“ bei weniger als zwei Minuten. Dabei war die Fehlerrate mit 0,01% gering. Auch der Bandbreitenbedarf bleib mit maximal 1,86 kByte/s niedrig.

9.2 Zukünftige Arbeiten

Mobilität wird im heutigen Internet insbesondere durch die Verbreitung netzfähiger Smartphones immer wichtiger. In dieser Arbeit wurden Mobilitätsaspekte nicht betrachtet. Ein einfacher Ansatz ist es, Mobilitätsaspekte auf der Vermittlungsschicht

beispielsweise durch Mobile IP [108] zu behandeln. Dies kann jedoch zu erhöhten Nachrichtenverzögerungen führen. Für eine optimale Behandlung von Mobilität müssen Erweiterungen zu QuON entwickelt werden, die die besonderen Anforderungen virtueller Welten berücksichtigen. Dabei muss insbesondere die schnelle Mitteilung einer geänderten Vermittlungsschichtadresse an die Nachbarn eines mobilen Teilnehmers gewährleistet werden. Eine direkte Information der Nachbarn ist möglich, sofern nicht diese gleichzeitig ebenfalls die Adresse wechseln. In diesem Fall könnten beispielsweise die Verbindungsnachbarn oder andere gemeinsame Nachbarn die neue Adresse übermitteln. Eine mögliche Erweiterung wäre auch das Auswerten von Cross-Layer-Informationen, um einen bevorstehenden Handover vorab kommunizieren zu können.

Network Address Translation (NAT) ist im heutigen Internet weit verbreitet. Die Verwendung von NAT erschwert den Einsatz von P2P-Technologien in der Regel. Die im Zusammenhang mit NAT entstehenden Probleme können durch den Einsatz von Techniken wie STUN [120] und *UDP hole punching* [52] abgemildert werden. QuON lässt sich um eine Unterstützung dieser Techniken erweitern, um auch über NAT angebundene Teilnehmer unterstützen zu können.

Die in dieser Arbeit vorgestellte Cheat-Erkennung kann auf zentralen oder dezentralen Autorisierungs- und Vertrauensstellen basieren. Dabei wurden jedoch vorrangig zentrale Lösungen untersucht. Das Verhalten dezentraler Autorisierungs- und Vertrauenssysteme kann von dem Verhalten zentraler Systeme abweichen. Beispielweise kann es eine gewisse Zeit dauern, bis der Vertrauenswert in einem dezentralen Vertrauenssystem aktualisiert wird, oder es können in dem System inkonsistente Vertrauenswerte auftreten. Dies kann die Effizienz der Cheat-Erkennung beeinträchtigen. Es sind somit weiteren Untersuchungen notwendig, die die Einflüsse der Verwendung dezentraler Autorisierungs- und Vertrauenssysteme genauer betrachten.

In dieser Arbeit wurde hauptsächlich die Verteilung von Positionsmeldungen und Ereignisnachrichten an Teilnehmer einer virtuellen Welt betrachtet. Oft sind für den Betrieb einer virtuellen Welt auch Zustandsdaten notwendig, die gespeichert werden müssen. Um einen möglichst skalierbaren Betrieb der virtuellen Welt zu ermöglichen, sollte auch die Speicherung der Zustandsdaten mittels P2P-Protokollen von den Teilnehmern übernommen werden. Dabei bietet sich die Speicherung in einer verteilten Hashtabelle an. Diese muss verschiedene Anforderungen erfüllen: Zunächst muss der Zustand zuverlässig auch ohne regelmäßige Auffrischung gehalten werden können. Weiterhin muss der Zugriff auf den Zustand von verschiedenen Teilnehmern stets konsistente Daten liefern. Dabei sollten die Antwortzeiten gering bleiben. Zusätzlich muss die Speicherung sicher sein, um Cheatern keine Möglichkeit zu geben, unbemerkt den Zustand zu ihren Gunsten zu verändern.

A. Zusätzliche Messergebnisse

A.1 Evaluierung bisheriger Protokolle

In Kapitel 5 wurden bisherige P2P-Protokolle für virtuelle Welten vorgestellt und untersucht. In den folgenden Abschnitten finden sich ergänzende Messungen zu den in Abschnitt 5.2 vorgestellten Ergebnissen.

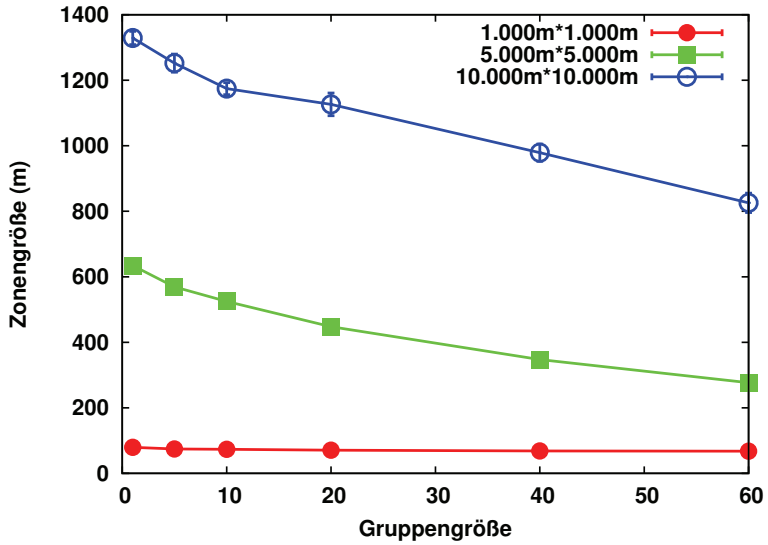
A.1.1 N-Tree

In den Simulationen des Protokolls N-Tree in Abschnitt 5.2 wurden die Zonengrößen und die Zahl der Zonenmitglieder gemessen. In Abbildung A.1a ist die durchschnittlich von Teilnehmern beobachtete Zonengröße abhängig von der Gruppengröße bei verschiedenen Spielfeldgrößen dargestellt. Um diese Zonengröße zu messen, beobachteten alle Teilnehmer regelmäßig die Größe ihrer aktuellen Zone. Über diese Messungen wurde dann der Durchschnitt gebildet. Zonen mit vielen Teilnehmern gehen somit stärker in den Messwert ein als Zonen mit wenigen Teilnehmern. Die Maßzahl erlaubt eine Abschätzung der Frequenz der Zonenübertritte von Spielern.

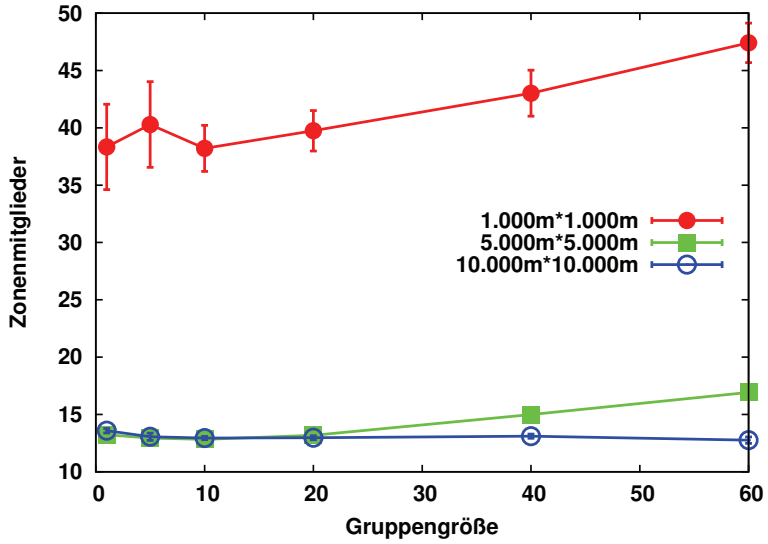
Die beobachtete Zonengröße sinkt bei steigender Teilnehmerdichte. Dies gilt sowohl für die über die Spielfeldgröße gesteuerte globale Teilnehmerdichte also auch für die über die Gruppengröße gesteuerte lokale Teilnehmerdichte. Bei sehr hoher Teilnehmerdichte, wie sie bei einer Spielfeldgröße von 1.000 m mal 1.000 m gegeben ist, erreicht die beobachtete Zonengröße die konfigurierte Minimalgröße (siehe hierzu auch Abschnitt 5.1.3.2).

Abbildung A.1b zeigt die Zahl der von Teilnehmern beobachteten Zonenmitglieder abhängig von der Gruppengröße bei verschiedenen Spielfeldgrößen. Hier wurde regelmäßig von jedem Spieler die Zahl der Teilnehmer gemessen, die sich in derselben Zone befanden. Die Zahl gibt die durchschnittliche Zahl an Teilnehmern an, an die ein Spieler seine Positionsmeldungen senden muss.

Diese Zahl bleibt bei geringer Teilnehmerdichte unterhalb von 15 durchschnittlich beobachteten Nachbarn. Bei der Spielfeldgröße 5.000 m mal 5.000 m steigt sie bei



(a) Durchschnittliche von Teilnehmern beobachtete Zonengröße in N-Tree bei verschiedenen Spielfeldgrößen.



(b) Durchschnittliche Zahl an von Teilnehmern beobachteten Zonenmitglieder in N-Tree bei verschiedenen Spielfeldgrößen.

Abbildung A.1 Zusätzliche Messergebnisse für N-Tree.

großen Gruppen auf 17. Da bei der Spielfeldgröße 1.000 m mal 1.000 m wie oben beschrieben die Zonen in der Regel ihre Minimalgröße nicht überschreiten, ist hier die Zahl der beobachteten Nachbarn deutlich größer. Sie steigt von 37 bei einer Gruppengröße von 1 auf 48 bei einer Gruppengröße von 60. Aufgrund der geringen Nachbarschaftskenntnis bei der Spielfeldgröße 1.000 m mal 1.000 m[1] schwanken die Messwerte deutlich.

A.1.2 SimMUD

In den Simulationen des Protokolls SimMUD in Abschnitt 5.2 wurde die durchschnittliche Zahl der von Teilnehmern durchgeführten Zonenwechsel pro Minute

[1]Siehe Abschnitt 5.2.4.3.

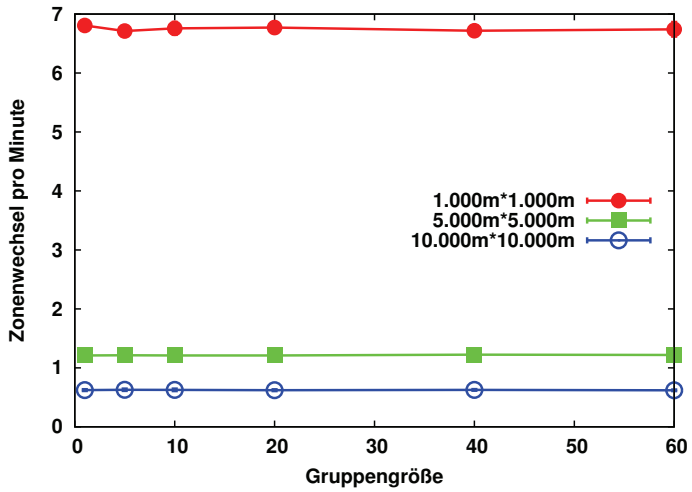


Abbildung A.2 Durchschnittliche Zahl der Zonenwechsel pro Minute in SimMUD bei verschiedenen Spielfeldgrößen.

abhängig von der Gruppengröße bei verschiedenen Spielfeldgrößen gemessen. Die Messergebnisse sind in Abbildung A.2 dargestellt.

Dabei ist die Zahl der Zonenwechsel nicht von der Gruppengröße abhängig. Stattdessen steigt sie bei sinkender Spielfeldgröße, da bei kleineren Spielfeldern bei SimMUD auch kleinere Zonen verwendet werden. Bei einer Spielfeldgröße von 10.000 m mal 10.000 m liegt die Zahl der Zonenwechsel pro Minute bei 0,8, bei einer Spielfeldgröße von 5.000 m mal 5.000 m bei 1,2 und bei einer Spielfeldgröße von 1.000 m mal 1.000 m bei 6,8.

A.1.3 Vast

In den Simulationen des Protokolls Vast in Abschnitt 5.2 wurde der durchschnittliche Bandbreitenaufwand für Nachbarsfindung und den Backup-Mechanismus abhängig von der Gruppengröße bei verschiedenen Spielfeldgrößen gemessen. Die Messergebnisse sind in Abbildung A.3 dargestellt.

Der Bandbreitenbedarf für Nachbarsfindung und den Backup-Mechanismus steigt mit der lokalen und globalen Teilnehmerdichte. Bei einer Spielfeldgröße von 10.000 m mal 10.000 m liegt er bei einer Gruppengröße von 1 bei 2 kByte/s, bei einer Gruppengröße von 60 bei 18 kByte/s. Bei einer Spielfeldgröße von 5.000 m mal 5.000 m werden bei einer Gruppengröße von 1 2 kByte/s benötigt, bei einer Gruppengröße von 60 bereits 40 kByte/s. Bei einer Spielfeldgröße von 1.000 m mal 1.000 m ist der Bandbreitenbedarf deutlich höher. Bei einer Gruppengröße von 1 liegt er bei 61 kByte/s, bei einer Gruppengröße von 60 erreicht er 112 kByte/s. Die verhältnismäßig geringere Steigung ab einer Gruppengröße 5 bei der Spielfeldgröße von 1.000 m mal 1.000 m erklärt sich dadurch, dass durch den geringen Abständen zwischen Gruppen in der virtuellen Welt die Zahl der Backup-Nachbarn und Grenznachbarn nicht mehr so stark mit der Gruppengröße ansteigen und dadurch die Vergrößerung des Aufwands für Backup-Mechanismus und Nachbarsfindung geringer ausfällt[2].

[2]Vergleiche auch Abschnitt 5.2.4.2.

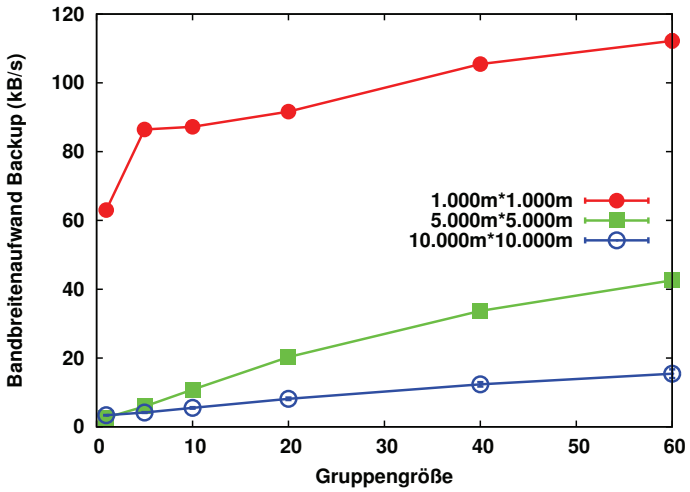


Abbildung A.3 Bandbreitenaufwand für Nachbarsfindung und Backup-Mechanismus in Vast bei verschiedenen Spielfeldgrößen.

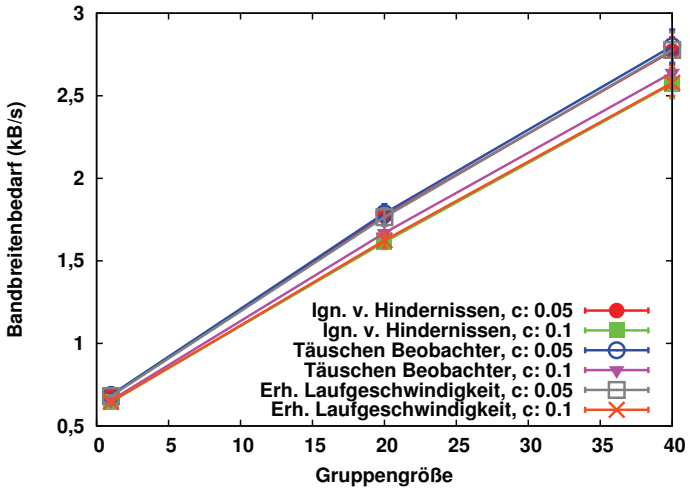


Abbildung A.4 Bandbreitenaufwand für Ticketvalidierung der Autorisierungsstelle in Abhängigkeit der Gruppengröße.

A.2 Evaluierung Cheat-Erkennung

Bei der Evaluierung der Cheat-Erkennung in Abschnitt 7.11 wurde der durchschnittliche Bandbreitenaufwand der Autorisierungsstelle für die Online-Verifizierung der Tickets in den verschiedenen Szenarien abhängig von der Gruppengröße gemessen. Die Messergebnisse sind in Abbildung A.4 dargestellt.

Der Bandbreitenbedarf steigt erwartungsgemäß linear mit der Gruppengröße. Die gemessenen Werte liegen bei einer Gruppengröße von 1 bei etwa 0,7 kByte/s und steigen bei einer Gruppengröße von 60 auf Werte zwischen 2,6 kByte/s und 2,8 kByte/s. Das beobachtete Cheatingverhalten hat keinen wesentlichen Einfluss auf den Bandbreitenbedarf. Bei hohem Cheater-Anteil liegt der Bandbreitenbedarf etwas niedriger als bei geringem Cheater-Anteil.

A.3 Evaluierung QuON

Bei der Evaluierung der Abstandsnormen für QuON in Abschnitt 8.1.4 wurde die Zahl der direkten und temporären Nachbarn der Teilnehmer gemessen. Die Ergebnisse sind in Abbildung A.5 gezeigt. Abbildung A.5a zeigt die Zahl der direkten Nachbarn in Abhängigkeit von der Gruppengröße bei verschiedenen Spielfeldgrößen. Erwartungsgemäß steigt die durchschnittliche Zahl an direkten Nachbarn linear mit der Gruppengröße. Bei großen Spielfeldern – also einer Spielfeldgröße von 10.000 m mal 10.000 m und 5.000 m mal 5.000 m – ist die Zahl der direkten Nachbarn unabhängig von der verwendeten Abstandsnorm. Bei der Spielfeldgröße 1.000 m mal 1.000 m ist die Zahl der direkten Nachbarn bei Verwendung der Maximumsnorm etwas größer als bei Verwendung der euklidischen Norm.

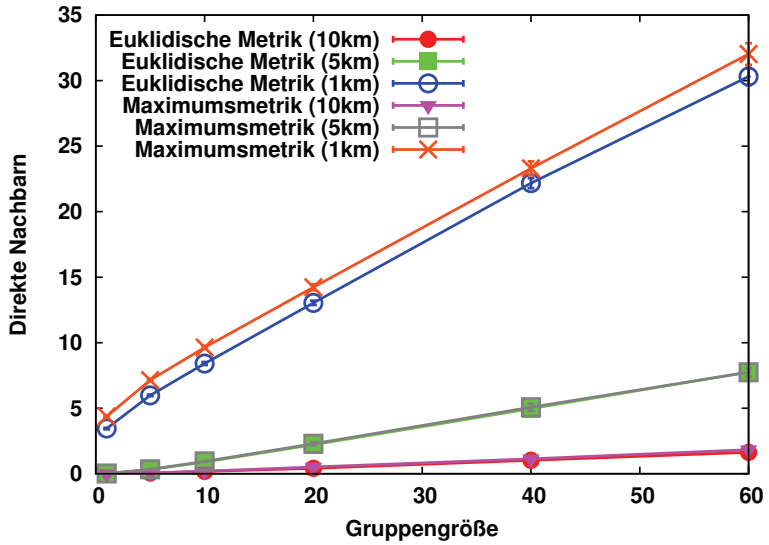
In Abbildung A.5b ist die Zahl der temporären Nachbarn in Abhängigkeit von der Gruppengröße bei verschiedenen Spielfeldgrößen gezeigt. In allen Szenarien steigt dabei die Zahl der temporären Nachbarn mit der Gruppengröße. Bei größeren Gruppen flacht die Steigung deutlich ab, teilweise sinkt die Zahl bei einer Gruppengröße von 60 wieder. Bei den Spielfeldgröße 10.000 m mal 10.000 m und 5.000 m mal 5.000 m bei Gruppengrößen zwischen 5 und 40 und bei der Spielfeldgröße von 1.000 m mal 1.000 m bei Gruppengrößen zwischen 1 und 20 ist zu sehen, dass bei Verwendung der euklidischen Norm die temporäre Nachbarnschaftsbeziehung häufiger auftritt als bei Verwendung der Maximumsnorm. Die Unterschiede betragen maximal 10% und bleiben somit gering.

Bei der Evaluierung der Techniken zur Bandbreiteneinsparung in QuON in Abschnitt 8.1.5 wurde aus Gründen der Übersichtlichkeit die Simulationsergebnisse für die Spielfeldgröße 10.000 m mal 10.000 m weggelassen. Abbildung A.6 zeigt diese Ergebnisse. In Abbildung A.6a ist die Nachbarschaftskenntnis in Abhängigkeit von der Gruppengröße dargestellt. Bei kleinen Gruppen ist die Nachbarschaftskenntnis aller Varianten nahe bei 100%. Bei größeren Gruppen sinkt die Nachbarschaftskenntnis. Bei der Gruppengröße 60 liegt sie bei 99,995% für unmodifizierten QuON, 99,97% für dynamische Interessengebiete und 99,93% für die alternative Nachbarsfindung.

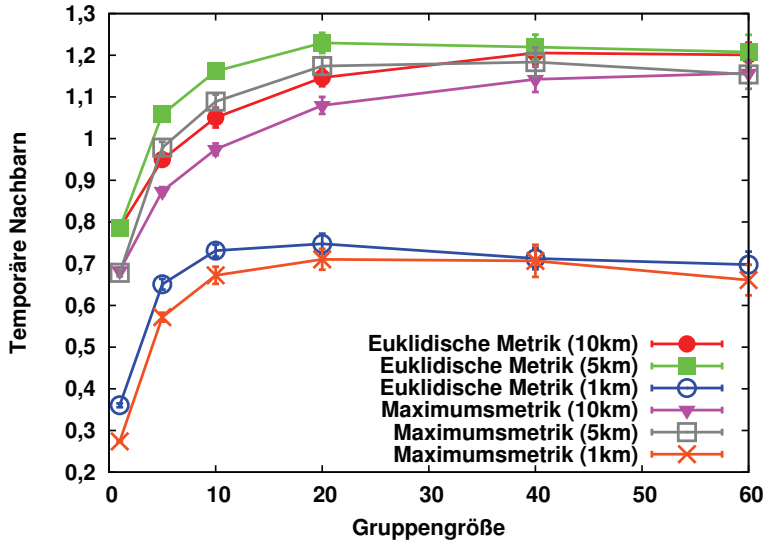
In Abbildung A.6b ist der Bandbreitenbedarf der Varianten dargestellt. Die Unterschiede zwischen den Varianten sind dabei vernachlässigbar gering. Bei der Gruppengröße 1 beträgt der Bandbreitenbedarf 9,5 kByte/s. Er steigt bis auf 17 kByte/s bei der Gruppengröße 60.

Bei der Evaluierung des Einflusses des Sitzungsverhaltens der Teilnehmer auf QuON in Abschnitt 8.1.7 wurde die durchschnittliche Zahl an Verbindungsnachbarn in Abhängigkeit von der Sitzungslänge bei verschiedenen graceful-leave-Wahrscheinlichkeiten p_g gemessen. Die Messergebnisse sind in Abbildung A.7 gezeigt. Dabei sinkt die Zahl der Verbindungsnachbarn mit der Sitzungslänge. Bei einer durchschnittlichen Sitzungslänge von 6000 Sekunden besitzt jeder Teilnehmer durchschnittlich 3.9 Verbindungsnachbarn, bei einer durchschnittlichen Sitzungszeit von 1000 Sekunden abhängig von p_g zwischen 3,4 und 3,55 Verbindungsnachbarn. Bei hohen Werten für p_g wird auch eine höhere Zahl an Verbindungsnachbarn gemessen.

Beim Vergleich von QuON mit bisherigen Protokollen in Abschnitt 8.1.8 wurden aus Gründen der Übersichtlichkeit die Simulationsergebnisse für die Spielfeldgröße 10.000 m mal 10.000 m weggelassen. Abbildung A.8 zeigt diese Ergebnisse. In Abbildung A.8a ist die Nachrichtenverzögerung in Abhängigkeit von der Gruppengröße

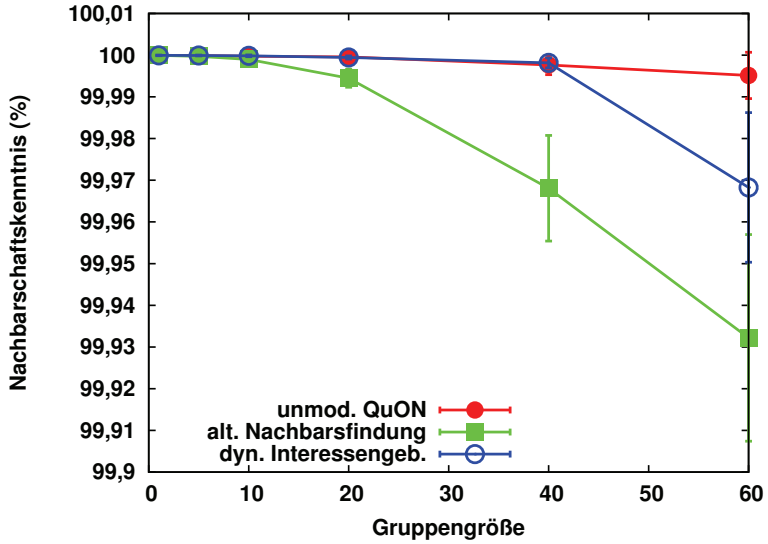


(a) Durchschnittliche Zahl an direkten Nachbarn in Abhängigkeit der Gruppengröße

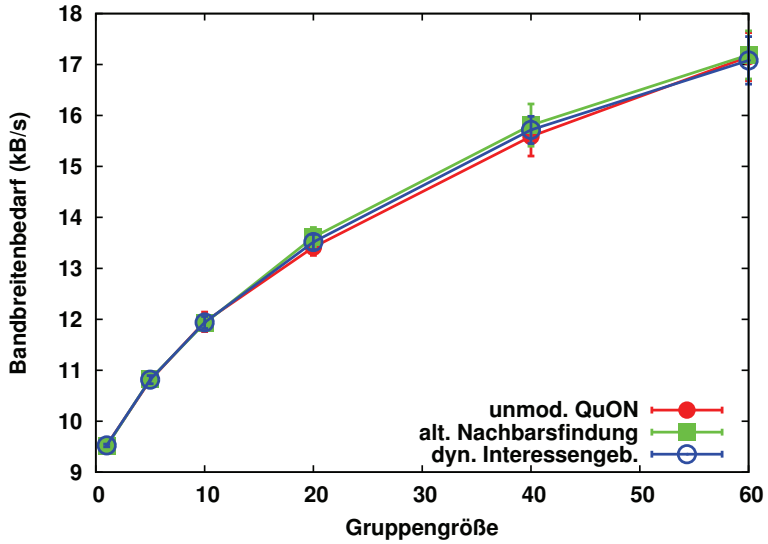


(b) Durchschnittliche Zahl an temporären Nachbarn in Abhängigkeit der Gruppengröße

Abbildung A.5 Nachbarszahlen bei der Evaluierung der in QuON verwendeten Abstandsnormen bei verschiedenen Spielfeldgrößen.



(a) Nachbarschaftskenntnis in Abhängigkeit der Gruppengröße.



(b) Bandbreitenbedarf in Abhängigkeit der Gruppengröße.

Abbildung A.6 Evaluierungsergebnisse: Einfluss von Bandbreiteneinsparungsmechanismen in QuON bei Spielfeldgröße 10.000 m mal 10.000 m.

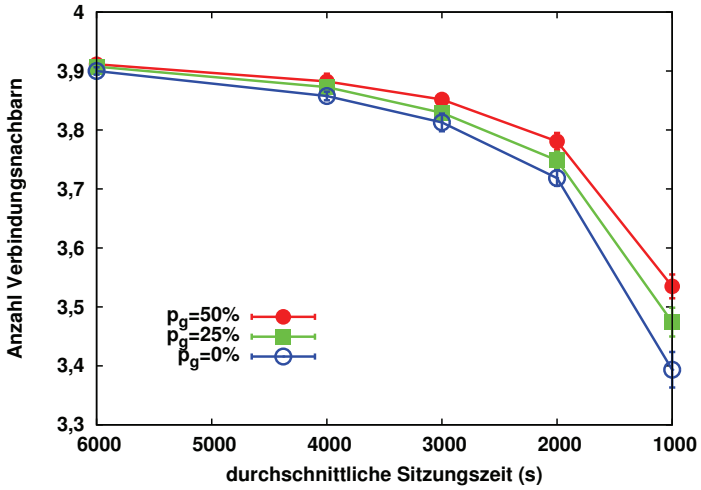
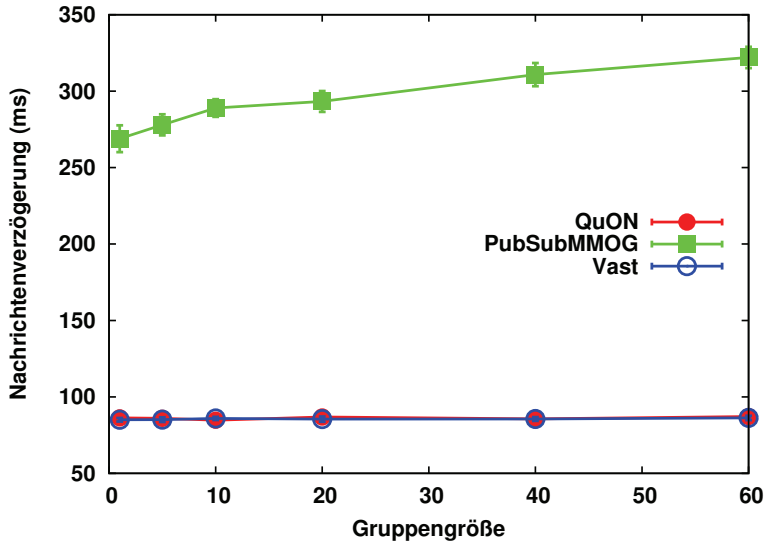


Abbildung A.7 Durchschnittliche Zahl an Verbindungsnachbarn in Abhängigkeit der Sitzungslänge.

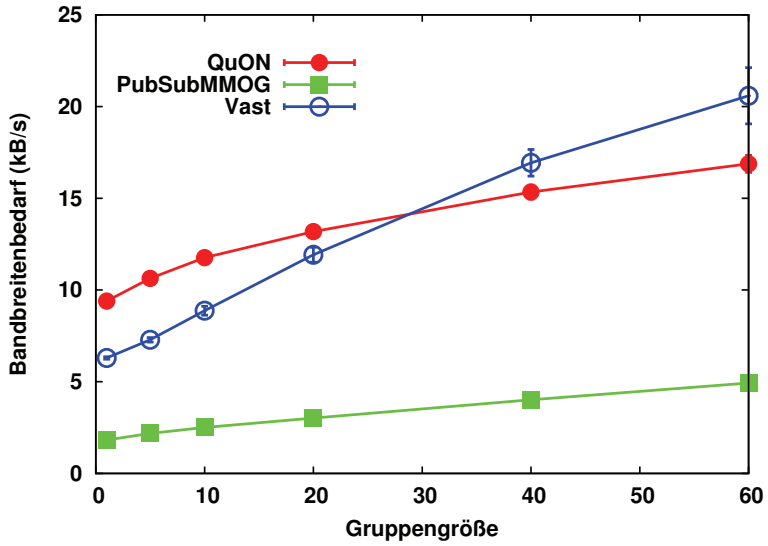
dargestellt. Bei QuON und Vast beträgt die Nachrichtenverzögerung unabhängig von der Gruppengröße ungefähr 90 Sekunden, dies entspricht der durchschnittlichen Ende-zu-Ende-Verzögerung zwischen zwei Netzknoten im für die Simulation verwendeten SimpleUnderlay-Modell. Die Nachrichtenverzögerung von PubSubMMOG beträgt bei der Gruppengröße 1 270 ms und steigt bis auf 320 ms bei der Gruppengröße 60.

In Abbildung A.8b ist der Bandbreitenbedarf der Protokolle in Abhängigkeit von der Gruppengröße dargestellt. Der Bandbreitenbedarf steigt in allen Protokollen mit steigender Gruppengröße: In PubSubMMOG steigt er von 2 kByte/s bei der Gruppengröße 1 auf 5 kByte/s bei der Gruppengröße 60, in QuON von 9 2 kByte/s bei der Gruppengröße 1 auf 17 kByte/s bei der Gruppengröße 60 und bei Vast von 6 kByte/s bei der Gruppengröße 1 auf 21 kByte/s bei der Gruppengröße 60. Vast zeigt dabei die größte Steigerung des Bandbreitenbedarfs der in diesem Szenario untersuchten Protokolle.

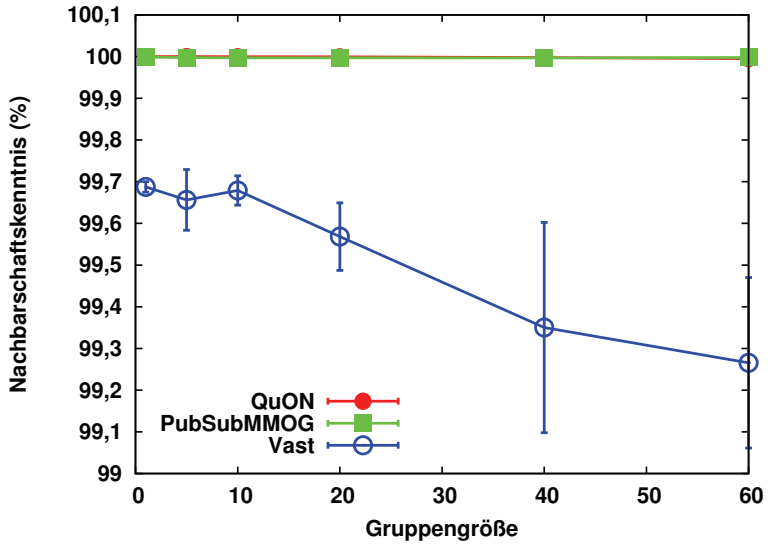
Abbildung A.8c zeigt die Nachbarschaftskenntnis der Protokolle in Abhängigkeit von der Gruppengröße. Bei PubSubMMOG und QuON blieben die Messwerte dabei unabhängig von der Gruppengröße nahe bei 100%. Bei Vast liegen sie niedriger und fallen mit steigender Gruppengröße. Bei der Gruppengröße 1 erreicht Vast eine Nachbarschaftskenntnis von 99,7% und bei einer Gruppengröße von 60 eine Nachbarschaftskenntnis von 99,3%.



(a) Nachrichtenverzögerung in Abhängigkeit der Gruppengröße.



(b) Bandbreitenbedarf in Abhängigkeit der Gruppengröße.



(c) Nachbarschaftskenntnis in Abhängigkeit der Gruppengröße.

Abbildung A.8 Vergleich von QuON mit bisherigen Protokollen bei Spielfeldgröße 10.000 m mal 10.000 m.

B. Beweise

B.1 Korrektheit der Verbindungsnachbarbestimmung

Um die Korrektheit der Verbindungsnachbarbestimmung zu beweisen, muss gezeigt werden, dass bei einer Bewegung eines Teilnehmers, die in einer Änderung der Nachbarsbeziehungen resultiert, die beteiligten Nachbarn mit der von Verbindungsnachbarn und temporären Nachbarn versendeten Informationen die neuen Nachbarsbeziehungen korrekt bestimmen können. Ausgehend von einem Graphen mit korrekten Nachbarsbeziehungen bleibt in diesem Fall nach jeder Bewegung der Graph korrekt. Hierbei wird von einem idealisierten Netz ohne Nachrichtenverluste und ohne Nachrichtenverzögerungen ausgegangen. O.B.d.A. kann so davon ausgegangen werden, dass jede Bewegung so aufgeteilt werden kann, dass in jeder Teilbewegung nur zwischen genau einem Nachbarspaar ein Quadrantenübertritt erfolgt.[1] Alle im Folgenden betrachteten Bewegungsschritte sind derartige Teilbewegungen. Ist t^0 der Zeitpunkt vor und t^1 der Zeitpunkt nach einer solchen Bewegung eines Teilnehmers $\vec{A}$, gilt somit für zwei beliebige Teilnehmer B und C

$$\begin{aligned} &t^0 : \vec{A} \in Q_B^i \wedge \vec{A} \in Q_C^j \\ &t^1 : (\vec{A} \in Q_B^{i'} \wedge i \neq i') \Rightarrow \vec{A} \in Q_C^j \end{aligned} \tag{B.1.1}$$

Die Bewegungsschritte werden unmittelbar an alle Nachbarn kommuniziert. Teilnehmerausfälle werden nicht betrachtet. Weiterhin wird hier von der Verwendung der Maximumsnorm $d(A, B)$ ausgegangen. Der Betrag der Differenz der x-Koordinate zweier Knoten A und B wird mit $d_x(A, B)$, der Betrag der Differenz der y-Koordinate mit $d_y(A, B)$ angegeben. Für die Aussage „B ist Verbindungsnachbar von A“ wird

[1] Dies impliziert, dass es Teilnehmern nicht möglich ist, die exakte x- oder y-Koordinate eines in der Nähe befindlichen anderen Teilnehmers einzunehmen. Da die Genauigkeit der im Protokoll verwendeten Koordinaten deutlich über der Anzeigegenauigkeit in gängigen virtuellen Welten liegt, lässt sich dieses technisch leicht und ohne sichtbare Einschränkung für die Teilnehmer durchsetzen.

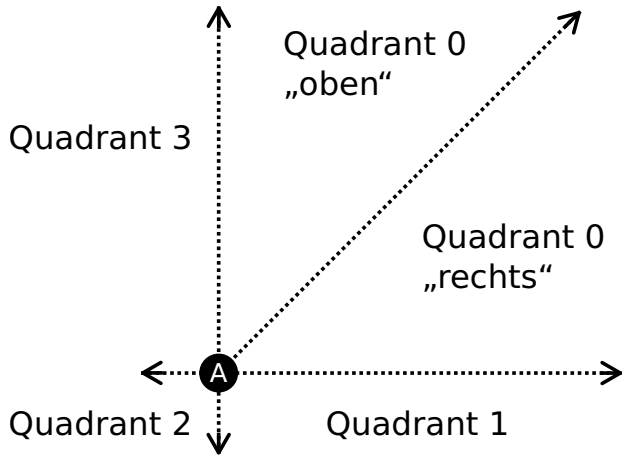


Abbildung B.1 Quadranten von Knoten A mit aufgeteiltem Quadrant 0.

die Notation $A \rhd B$ verwendet. Gemäß Abschnitt 6.4.3 gilt dann: A ist Verbindungsnachbar oder temporärer Nachbar von B. Die Notation kann bei Bedarf präzisiert werden mit $A \rhd^i B$ für die Aussage „B ist Verbindungsnachbar von A in Quadrant i“. Es gilt gemäß der Definition der Verbindungsnachbarbeziehung in Abschnitt 6.4.2

$$A \rhd^i B \Rightarrow \forall X \in Q_A^i : d(A,B) \leq d(A,X) \tag{B.1.2}$$

Die Relation $A \bowtie B$ wird definiert als $A \bowtie B :\Leftrightarrow A \rhd B \vee B \rhd A$. Als Hilfskonstrukt wird eine Unterteilung eines Quadranten entlang der Winkelhalbierenden der Quadrantengrenzen eingeführt. Hierdurch entstehen zwei Hälften, die im Falle von Quadrant 0 mit „oben“ und „rechts“ beziehungsweise Q^{0o} und Q^{0r} bezeichnet werden. Dies veranschaulicht Abbildung B.1.

Da QuON bezüglich der Quadranten symmetrisch ist, wird hier nur die Verbindungsnachbarbeziehung in Quadrant 0 eines Teilnehmers betrachtet. Für die anderen Quadranten lässt sich dieser Beweis analog führen.

Betrachtet man einen Knoten A mit einem Verbindungsnachbarn B zum Zeitpunkt t^0 und einem Knoten C, der zum Zeitpunkt t^1 neuer Verbindungsnachbar von A werden soll, kann sich diese Veränderung der Nachbarschaftsbeziehungen durch eine Bewegung von A, B oder C ergeben. Bei der Bewegung können alle Knoten in demselben Quadranten von A bleiben, oder es kann ein Quadrantenwechsel stattfinden. Es ergeben sich somit sechs Fälle, durch die eine Änderung der Verbindungsnachbarbeziehung $A \rhd^0 B$ hervorgerufen werden kann:

1. $t^0 : A \rhd^0 B, \vec{C} \in Q_A^0$
   $t^1 : d(A,B) > d(A,\vec{C})$
   Knoten C bewegt sich innerhalb des Quadranten und ist nun näher als der alte Verbindungsnachbar B.

2. $t^0 : A \rhd^0 B, \vec{C} \notin Q_A^0$
   $t^1 : \vec{C} \in Q_A^0 \wedge d(A,B) > d(A,\vec{C})$
   Knoten C betritt den Quadranten und ist A dort näher als der alte Verbindungsnachbar B.

3. $t^0 : A \rhd^0 \vec{B}, C \in Q_A^0$
   $t^1 : d(A, \vec{B}) > d(A, C)$
   Der Verbindungsnachbar B bewegt sich innerhalb des Quadranten und ist nun weiter entfernt als Knoten C.

4. $t^0 : A \rhd^0 \vec{B}, C \in Q_A^0$
   $t^1 : \vec{B} \notin Q_A^0, C \in Q_A^0$
   Der Verbindungsnachbar B verlässt den Quadranten.

5. $t^0 : \vec{A} \rhd^0 B, C \in Q_A^0$
   $t^1 : d(\vec{A}, B) > d(\vec{A}, C)$
   Knoten A bewegt sich, sodass Knoten C nun näher ist als der alte Verbindungsnachbar B.

6. $t^0 : \vec{A} \rhd^0 B, C \in Q_A^0$
   $t^1 : B \notin Q_{\vec{A}}^0, C \in Q_{\vec{A}}^0$
   Knoten A bewegt sich, sodass sein Verbindungsnachbar B nicht mehr im selben Quadranten liegt.

Es muss bewiesen werden, dass für jeden dieser Fälle zum Zeitpunkt t^0 gilt: $t^0 : A \bowtie C \vee \exists X : A \bowtie X \bowtie C$. Dies ermöglicht Knoten A zum Zeitpunkt t^1, die Verbindungsnachbarbeziehung $A \rhd C$ aufzubauen.

Fall 1: Knoten C bewegt sich innerhalb des Quadranten und ist nun näher als der alte Verbindungsnachbar B.

$$t^0 : A \rhd^0 B, \vec{C} \in Q_A^0$$
$$t^1 : d(A, B) > d(A, \vec{C})$$

Zunächst wird hier der Quadrant 0 in die oben beschriebenen Hälften aufgeteilt. Dann lassen sich bis auf Symmetrie zwei Fälle unterscheiden: Der neue Verbindungsnachbar nähert sich aus der Hälfte, in der sich der alte Verbindungsnachbar befindet, oder er nähert sich aus der anderen Hälfte. O.B.d.A wird hier der Fall betrachtet, dass sich der Verbindungsnachbar in der „rechten“ Hälfte befindet.

Fall 1a: Der neue Nachbar ist in derselben Hälfte wie der alte Verbindungsnachbar.

$$\vec{C} \in Q_A^{0r}, B \in Q_A^{0r}$$

Die Annäherung eines Knotens $\vec{C}$ an einen Knoten A aus der Hälfte des bisherigen Verbindungsnachbarn zum Zeitpunkt t^0 ist in Abbildung B.2 gezeigt. Da Knoten B Verbindungsnachbar von Knoten A in Quadrant 0 ist, kann sich nach (B.1.2) in dem schraffiert gezeichneten Bereich kein weiterer Knoten befinden.

$\vec{C}$ ist entweder oberhalb oder unterhalb von B. Es gilt: $t^0 : \vec{C} \in Q_B^0 \wedge B \in Q_{\vec{C}}^2$ wenn $\vec{C}$ oberhalb von B ist oder $t^0 : \vec{C} \in Q_B^1 \wedge B \in Q_{\vec{C}}^3$ wenn $\vec{C}$ unterhalb von B liegt.

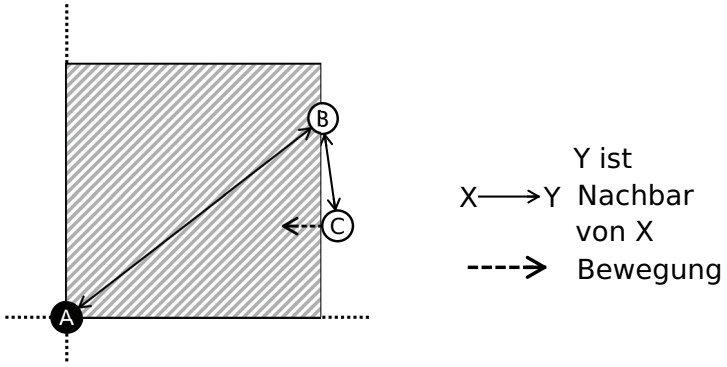


Abbildung B.2 Knoten C nähert sich Knoten A aus der Hälfte des bisherigen Verbindungsnachbarn.

Aus (B.1.1) folgt, dass sich zwischen Knoten B und $\vec{C}$ kein weiterer Knoten X befinden kann. Befände sich ein solcher Knoten X zwischen B und $\vec{C}$ gälte für $\vec{C}$ oberhalb B:

$$\begin{aligned} & t^0 : \vec{C} \in Q_B^0, B \in Q_{\vec{C}}^2, X \in Q_B^0, X \in Q_{\vec{C}}^2, \vec{C} \in Q_X^0 \\ & t^1 : \vec{C} \in Q_B^3, B \in Q_{\vec{C}}^1 \text{ (sonst } d(A, \vec{C}) > d(A, B)) \\ \Rightarrow\ & t^1 : X \in Q_{\vec{C}}^1 \text{ (sonst } X \notin Q_B^0) \\ \Rightarrow\ & t^1 : \vec{C} \in Q_X^3 \text{ (Widerspruch zu (B.1.1))} \end{aligned} \tag{B.1.3}$$

Analog gälte für B oberhalb $\vec{C}$:

$$\begin{aligned} & t^0 : \vec{C} \in Q_B^1, B \in Q_{\vec{C}}^3, X \in Q_B^1, X \in Q_{\vec{C}}^3, \vec{C} \in Q_X^1 \\ & t^1 : \vec{C} \in Q_B^2, B \in Q_{\vec{C}}^0 \text{ (sonst } d(A, \vec{C}) > d(A, B)) \\ \Rightarrow\ & t^1 : X \in Q_{\vec{C}}^0 \text{ (sonst } X \notin Q_B^1) \\ \Rightarrow\ & t^1 : \vec{C} \in Q_X^2 \text{ (Widerspruch zu (B.1.1))} \end{aligned} \tag{B.1.4}$$

Ist $\vec{C}$ oberhalb B folgt aus (B.1.2) und (B.1.3):

$$t^0 : \nexists X \in Q_{\vec{C}}^2 : d(\vec{C}, B) \geq d(\vec{C}, X) \Rightarrow \vec{C} \rhd^2 B \tag{B.1.5}$$

Ist B oberhalb $\vec{C}$ folgt aus (B.1.2) und (B.1.4):

$$t^0 : \nexists X \in Q_B^1 : d(B, \vec{C}) \geq d(B, X) \Rightarrow B \rhd^1 \vec{C} \tag{B.1.6}$$

$\vec{C}$ ist also zum Zeitpunkt t^0 Verbindungsnachbar (B.1.6) oder temporärer Nachbar (B.1.5) von B. Somit gilt $t^0 : A \bowtie B \bowtie \vec{C}$

Fall 1b: Der neue Nachbar ist in einer anderen Hälfte als der alte Verbindungsnachbar.

$$\vec{C} \in Q_A^{0o}, B \in Q_A^{0r}$$

Die Annäherung eines Knotens aus der anderen Hälfte ist in Abbildung B.3 gezeigt. B ist wiederum Verbindungsnachbar von Knoten A in Quadrant 0. Gemäß (B.1.2) kann sich in der grau schraffierten Fläche kein weiterer Knoten befinden.

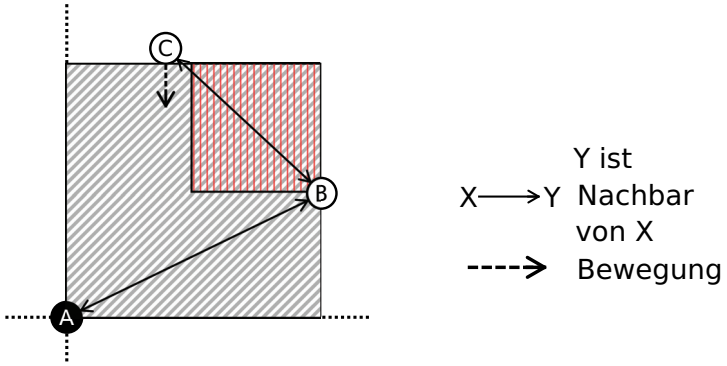


Abbildung B.3 Knoten C nähert sich Knoten A aus der bezüglich des alten Verbindungsnachbarn anderen Hälfte.

Ist $d_x(B,\vec{C}) \geq d_y(B,\vec{C})$, bleibt Knoten $\vec{C}$ also wie im Bild also außerhalb der rot schraffierten Fläche, muss B nach (B.1.2) Verbindungsnachbar von $\vec{C}$ sein:

$$t^0 : \nexists X \in Q^1_{\vec{C}} : d(B,\vec{C}) \geq d(X,\vec{C}) \Rightarrow \vec{C} \rhd^1 B \tag{B.1.7}$$

Ist hingegen $d_x(B,\vec{C}) \leq d_y(B,\vec{C})$, tritt $\vec{C}$ also in die in Abbildung B.3 rot schraffierte Fläche ein, muss $\vec{C}$ Verbindungsnachbar von B sein:

$$t^0 : \nexists X \in Q^3_B : d(B,\vec{C}) \geq d(B,X) \Rightarrow B \rhd^3 \vec{C} \tag{B.1.8}$$

Wie in Fall 1a ist $\vec{C}$ also zum Zeitpunkt t^0 Verbindungsnachbar (B.1.8) oder temporärer Nachbar (B.1.7) von B. Es gilt also $t^0 : A \bowtie B \bowtie \vec{C}$.

Fall 2: Knoten C betritt den Quadranten, und ist A dort näher als der alte Verbindungsnachbar B.

$$t^0 : A \rhd^0 B, \vec{C} \notin Q^0_A$$
$$t^1 : \vec{C} \in Q^0_A \wedge d(A,B) > d(A,\vec{C})$$

Wie in Fall 1 kann hier der Quadrant wieder in einen „rechten“ und einen „oberen“ Teil geteilt werden. Unterscheiden lassen sich dann die Fälle, dass der sich bewegende Knoten den Quadranten im selben Teil betritt wie der Verbindungsnachbar oder in den anderen Teil eintritt. O.B.d.A. wird wiederum der Fall betrachtet, dass sich der alte Verbindungsnachbar in der „rechten“ Hälfte befindet.

Fall 2a: Der neue Nachbar tritt in den Teil ein, in dem sich der alte Verbindungsnachbar befindet.

$$t^1 : \vec{C} \in Q^{0r}_A, B \in Q^{0r}_A$$

Tritt der neue Verbindungsnachbar in den Teil in den Quadranten ein, in dem sich der alte Verbindungsnachbar befindet, ist die Situation, wie in Abbildung B.4

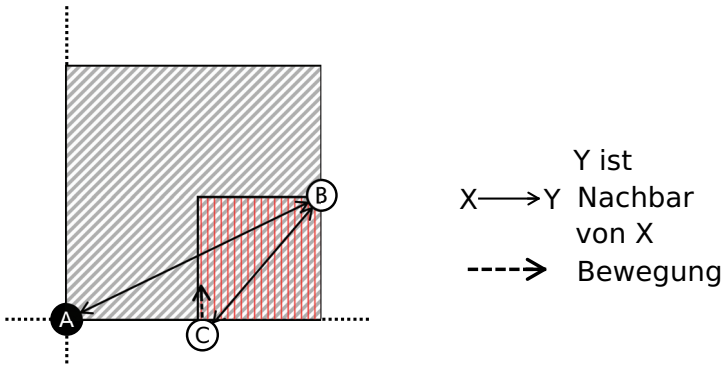


Abbildung B.4 Knoten C nähert sich Knoten A aus Quadrant 1.

verdeutlicht, äquivalent zu Fall 1b. Ist $d_x(B,\vec{C}) \geq d_y(B,\vec{C})$, bleibt $\vec{B}$ also außerhalb der in der Abbildung rot schraffierten Fläche, gilt $t^0 : C \rhd^0 \vec{B}$. Ist hingegen $d_x(B,\vec{C}) \leq d_y(B,\vec{C})$, tritt $\vec{B}$ also in die in rot schraffierte Fläche ein, gilt $t^0 : \vec{B} \rhd^2 C$.

$\vec{C}$ ist also zum Zeitpunkt t^0 Verbindungsnachbar oder temporärer Nachbar von B. Es gilt $t^0 : A \bowtie B \bowtie C$.

Fall 2b: Der neue Nachbar tritt in den Teil ein, in dem sich der alte Verbindungsnachbar nicht befindet.

$$t^1 : \vec{C} \in Q_A^{0o}, B \in Q_A^{0r}$$

Tritt der neue Nachbar, wie in Abbildung B.5 gezeigt, auf der gegenüberliegenden Seite in den Quadranten ein, muss Knoten A Verbindungsnachbar von Knoten C sein:

Es kann sich nach (B.1.1) zu t^0 kein Knoten zwischen A und $\vec{C}$ befinden. Gäbe es einen solchen Knoten X würde gelten:

$$\begin{aligned} & t^0 : \vec{C} \in Q_A^3, A \in Q_{\vec{C}}^1, X \in Q_A^3, X \in Q_{\vec{C}}^1, \vec{C} \in Q_X^3 \\ & t^1 : \vec{C} \in Q_A^0, A \in Q_{\vec{C}}^2 \text{ (nach Voraussetzung Fall 2)} \\ \Rightarrow\ & t^1 : X \in Q_{\vec{C}}^2 \text{ (sonst } X \notin Q_A^3) \\ \Rightarrow\ & t^1 : \vec{C} \in Q_X^0 \text{ (Widerspruch zu (B.1.1))} \end{aligned} \tag{B.1.9}$$

Aus (B.1.2) und (B.1.9) folgt nun:

$$t^0 : \nexists X \in Q_{\vec{C}}^1 : d(\vec{C},A) \geq d(\vec{C},X) \Rightarrow \vec{C} \rhd^1 A$$

$\vec{C}$ muss also zum Zeitpunkt t^0 Verbindungsnachbar oder temporärer Nachbar von A sein: $t^0 : A \bowtie C$.

Fall 3: Der Verbindungsnachbar B bewegt sich innerhalb des Quadranten und ist nun weiter entfernt als Knoten C.

$$\begin{aligned} & t^0 : A \rhd^0 \vec{B}, C \in Q_A^0 \\ & t^1 : d(A,\vec{B}) > d(A,C) \end{aligned}$$

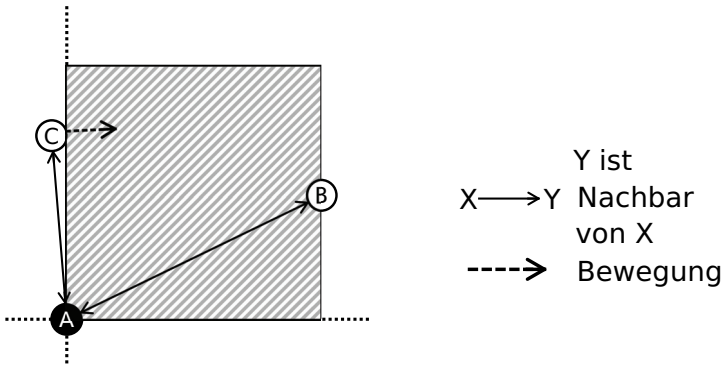


Abbildung B.5 Knoten C nähert sich Knoten A aus Quadrant 3.

Dieser Fall ist äquivalent zu Fall 1: Eine Bewegung des alten Verbindungsnachbarn nach oben ist äquivalent zu einer Bewegung des neuen Verbindungsnachbarn nach unten, und eine Bewegung des alten Verbindungsnachbarn nach rechts ist äquivalent zu einer Bewegung des neuen Verbindungsnachbarn nach links. Demnach gilt wie in Fall 1: $t^0 : A \bowtie B \bowtie C$.

Fall 4: Der Verbindungsnachbar B verlässt den Quadranten.

$$t^0 : A \rhd^0 \vec{B}, C \in Q_A^0$$
$$t^1 : \vec{B} \notin Q_A^0, C \in Q_A^0$$

Auch hier wird der Quadrant in einen „rechten“ und einen „oberen“ Teil geteilt. Es ergeben sich zwei Fälle: Entweder befindet sich der neu zu findende Verbindungsnachbar in dem Teil, aus dem der alte Verbindungsnachbar den Quadranten verlässt, oder er befindet sich in dem anderen Teil. O.B.d.A. wird hier der Fall betrachtet, dass sich der alte Verbindungsnachbar im „oberen“ Teil befindet.

Fall 4a: Der neue Nachbar befindet sich in dem Teil, in dem sich der alte Verbindungsnachbar nicht befindet.

$$t^0 : C \in Q_A^{0r}, \vec{B} \in Q_A^{0o}$$

Der neue Verbindungsnachbar befindet sich in dem jeweils anderen Teil des Quadranten. Dies ist in Abbildung B.6 gezeigt. Hier bewegt sich der alte Verbindungsnachbar B aus den Quadranten 0 eines Knotens A. Der neu zu findende Verbindungsnachbar ist C.

Aus (B.1.2) folgt:

$$\nexists X \in Q_A^0 : d(A, C) \geq d(C, X) \tag{B.1.10}$$

Somit gilt:

$$\begin{aligned} & C \rhd^2 A \vee \exists N \in Q_C^2 \setminus Q_A^0 : C \rhd^2 N \\ \Rightarrow\ & C \rhd^2 A \vee \exists N \in Q_A^1 : C \rhd^2 N \text{ (denn: } \forall X \in Q_A^2 \cup Q_A^3 : d(C, A) < d(C, X)) \end{aligned} \tag{B.1.11}$$

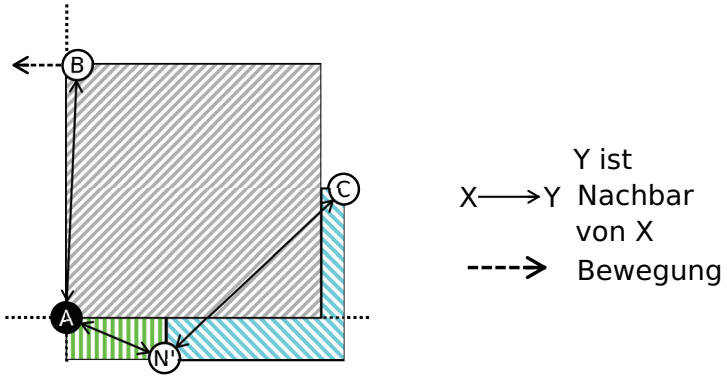


Abbildung B.6 Knoten B verlässt den Quadranten, der neue Verbindungsnachbar befindet sich im anderen Teil.

Ist $C \rhd^2 A$, ist C temporärer Nachbar von A. A kann somit trivial zu t^1 den Status der Beziehung ändern. Ansonsten wird $\mathcal{N}$ als die Menge aller Knoten in Q_A^1, die eine Verbindung zu C haben, und $N^{'}$ als das nächste Element in $\mathcal{N}$ zu A definiert:

$$\begin{aligned} &\mathcal{N} := \{X \in Q_A^1 : C \rhd X \vee X \rhd C\} \\ &N^{'} := N^{'} \in \mathcal{N} \wedge (\forall N \in \mathcal{N} : d(A, N^{'}) \leq d(A, N)) \end{aligned} \tag{B.1.12}$$

Es kann nun keinen Knoten geben, der sich zwischen C und $N^{'}$, also in der in Abbildung B.6 türkis schraffiert gezeichneten Fläche, befindet. Gäbe es einen solchen Knoten X würde gelten:

$$\begin{aligned} &d(C, X) < d(C, N^{'}) \wedge d(N^{'}, X) < d(C, N^{'}) \\ \Rightarrow &C \not\rhd N^{'} \wedge N^{'} \not\rhd C \text{ (Widerspruch zu (B.1.12))} \end{aligned} \tag{B.1.13}$$

Ebenso kann es keinen Knoten geben, der sich zwischen A und $N^{'}$, also in der in Abbildung B.6 grün schraffiert gezeichneten Fläche, befindet. Gäbe es einen solchen Knoten X würde gelten:

$$\begin{aligned} &d(A, X) < d(A, N^{'}) \\ &X \rhd^0 C \text{ (nach (B.1.2) und (B.1.13)) (Widerspruch zu (B.1.12))} \end{aligned} \tag{B.1.14}$$

Somit gilt

$$d_x(A, N^{'}) \geq d_y(A, N^{'}) \Rightarrow N^{'} \rhd^3 A \tag{B.1.15}$$

$$d_x(A, N^{'}) \leq d_y(A, N^{'}) \Rightarrow A \rhd^1 N^{'} \tag{B.1.16}$$

Aus (B.1.11), (B.1.12), (B.1.15) und (B.1.16) folgt nun:

$$t^0 : A \bowtie C \vee A \bowtie N^{'} \bowtie C$$

Fall 4b: Der neue Nachbar befindet sich in dem Teil, in dem sich der alte Verbindungsnachbar befindet.

$$t^0 : C \in Q_A^{0o}, \vec{B} \in Q_A^{0o}$$

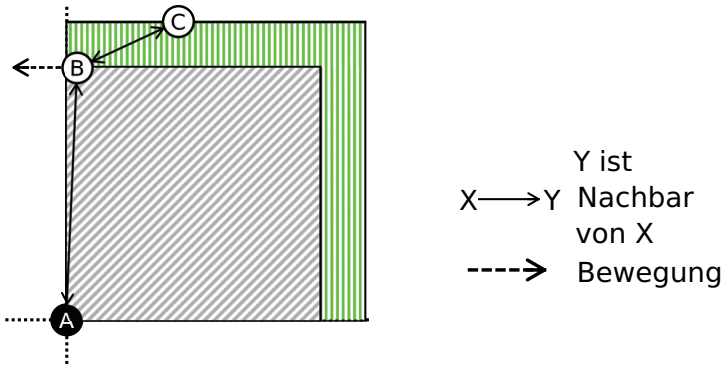


Abbildung B.7 Knoten B verlässt den Quadranten, der neue Verbindungsnachbar befindet sich im selben Teil.

Der Fall, dass sich der neue Verbindungsnachbar C im selben Teil des Quadranten befindet wie der alte Verbindungsnachbar B, ist in B.7 gezeigt. Da C in t^1 Verbindungsnachbar wird, kann es nach (B.1.1) und (B.1.2) keinen Knoten außer $\vec{B}$ geben, der in Quadrant 0 von A näher an A ist als C (im Bild grün schraffiert):

$$t^0 : \nexists X \in Q_A^0 \setminus \{\vec{B}\} : d(X, A) < d(C, A)$$

Somit gilt:

$$t^0 : d_X(\vec{B}, C) \geq d_Y(\vec{B}, C) \Rightarrow C \rhd^2 \vec{B}$$
$$t^0 : d_X(\vec{B}, C) \leq d_Y(\vec{B}, C) \Rightarrow \vec{B} \rhd^0 C$$

C ist also zu t^0 Verbindungsnachbar oder temporärer Nachbar von $\vec{B}$. Es gilt somit $t^0 : A \bowtie \vec{B} \bowtie C$.

Fall 5: Knoten A bewegt sich, sodass Knoten C nun näher ist als der alte Verbindungsnachbar B.

$$t^0 : \vec{A} \rhd^0 B, C \in Q_A^0$$
$$t^1 : d(\vec{A}, B) > d(\vec{A}, C)$$

Dies kann nur den Fall sein, wenn der alte und der neue Verbindungsnachbar sich in unterschiedlichen Teilen von Quadrant 0 von A befinden. Damit ist dieser Fall äquivalent zu Fall 1b. Eine Bewegung von A nach oben entspricht einer Bewegung von C nach unten. Eine Bewegung von A nach rechts entspricht einer Bewegung von C nach links. Demnach gilt analog zu Fall 1b: $t^0 : \vec{A} \bowtie B \bowtie C$.

Fall 6: Knoten A bewegt sich, sodass sein Verbindungsnachbar B nicht mehr im selben Quadranten liegt.

$$t^0 : \vec{A} \rhd^0 B, C \in Q^0_A$$
$$t^1 : B \notin Q^0_{\vec{A}}, C \in Q^0_{\vec{A}}$$

Dieser Fall ist äquivalent zu Fall 4. Eine Bewegung des Knotens nach oben ist äquivalent zu einer Bewegung des Verbindungsnachbarn nach unten, und eine Bewegung des Knotens nach rechts ist äquivalent zu einer Bewegung des Verbindungsnachbarn nach links. Analog zu Fall 4 gilt dann:

$$t^0 : \vec{A} \bowtie C \vee \vec{A} \bowtie B \bowtie C \vee \exists N' : \vec{A} \bowtie N' \bowtie C$$

Somit ist für alle möglichen Bewegungsschritte, die zu einer Änderung der Verbindungsnachbarbeziehung führen können, gezeigt, dass gilt:

$$t^0 : A \bowtie C \vee \exists X : A \bowtie X \bowtie C$$

□

B.2 Korrektheit der Nachbarsfindung

Die Korrektheit der Nachbarsfindung folgt unmittelbar aus der in Abschnitt B.1 bewiesenen Korrektheit der Verbindungsnachbarbestimmung: Ist in einem Quadranten der Verbindungsnachbar B eines Knoten A außerhalb des Interessengebiets von A, muss jeder Knoten C, der in diesem Quadranten in das Interessengebiet eintreten will, dem Knoten näher kommen als Knoten B. Somit wird Knoten C vor Eintritt in das Interessengebiet Verbindungsnachbar von Knoten A. A ist somit zum Zeitpunkt des Eintritts in sein Interessengebiet über die Position von C informiert.

Befindet sich der Verbindungsnachbar B innerhalb des Interessengebiets von Knoten A, überdeckt das Interessengebiet von Knoten B das gesamte Interessengebiet von Knoten A im jeweiligen Quadranten. Somit ist jeder Knoten, der in diesem Quadranten in das Interessengebiet von A eintritt, ein direkter Nachbar von B. B kann somit A den Eintritt des neuen Nachbarn melden.

Demnach kann in jedem Fall ein Knoten von seinem Verbindungsnachbarn über neu in sein Interessengebiet eintretenden Knoten informiert werden. □

Literaturverzeichnis

[1] Aberer, Karl ; Despotovic, Zoran: Managing trust in a peer-2-peer information system. In: *Proceedings of the tenth international conference on Information and knowledge management (CIKM '01), Atlanta, GA, USA*, ACM, 2001, S. 310–317. – ISBN 1-58113-436-3

[2] Activeworlds Inc.: *Active Worlds*. http://www.activeworlds.com/. – (Überprüft am: 16.12.2010)

[3] Ahmed, Dewan T. ; Shirmohammadi, Shervin: A microcell oriented load balancing model for collaborative virtual environments. In: *IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2008 (VECIMS 2008), Tarent, Italien*, Juli 2008, S. 86 –91

[4] Ahmed, Dewan T. ; Shirmohammadi, Shervin ; Oliveira, Jauvane: Improving gaming experience in zonal MMOGs. In: *Proceedings of the 15th international conference on Multimedia (MULTIMEDIA '07), Augsburg*, ACM, 2007, S. 581–584. – ISBN 978-1-59593-702-5

[5] Alexander, Leigh: *Blizzard Bans 320,000 Warcraft, Diablo Accounts*. http://www.gamasutra.com/view/news/28183/Blizzard_Bans_320000_Warcraft_Diablo_Accounts.php. April 2010. – (Überprüft am: 18.11.2010)

[6] Armbrust, Michael ; Fox, Armando ; Griffith, Rean ; Joseph, Anthony D. ; Katz, Randy ; Konwinski, Andy ; Lee, Gunho ; Patterson, David ; Rabkin, Ariel ; Stoica, Ion ; Zaharia, Matei: A view of cloud computing. In: *Communications of the ACM* 53 (2010), April, S. 50–58. – ISSN 0001-0782

[7] Backhaus, Helge ; Krause, Stephan: Voronoi-based adaptive scalable transfer revisited: gain and loss of a Voronoi-based peer-to-peer approach for MMOG. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07), Melbourne, Australien*, ACM, September 2007, S. 49–54

[8] Backhaus, Helge ; Krause, Stephan: QuON: a quad-tree-based overlay protocol for distributed virtual worlds. In: *International Journal of Advanced Media and Communication* 4 (2010), März, Nr. 2, S. 126 – 139

[9] Banerjee, Suman ; Bhattacharjee, Bobby ; Kommareddy, Christopher: Scalable Application Layer Multicast. In: *Proceedings of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02), Pittsburgh, PA, USA* Bd. 32, Oktober 2002, S. 205–217

[10] Barab, S. ; Dodge, T. ; Tuzun, H. ; Job-Sluder, K. ; Jackson, C. ; Arici, A. ; Job-Sluder, L. ; Carteaux, R. ; Jr. ; Gilbertson, J. ; Heiselt, C.: *The Quest Atlantis Project: A socially-responsive play space for learning.* Kap. 7, S. 159–186. In: Shelton, B. E. (Hrsg.) ; Wiley, D. (Hrsg.): *The Educational Design and Use of Simulation Computer Games.* Rotterdam, The Netherlands : Sense Publishers, 2007. – ISBN 978-90-8790-156-1

[11] Bartle, Richard: *Early MUD History.* http://www.mud.co.uk/richard/mudhist.htm. November 1990. – (Überprüft am: 10.01.2011)

[12] Baset, Salman A. ; Schulzrinne, Henning G.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In: *25th IEEE International Conference on Computer Communications (INFOCOM 2006), Barcelona, Spanien*, April 2006. – ISSN 0743-166X

[13] Bauer, Daniel ; Rooney, Sean ; Scotton, Paolo: Network infrastructure for massively distributed games. In: *Proceedings of the 1st workshop on Network and system support for games (NetGames '02), Braunschweig*, ACM, 2002, S. 36–43. – ISBN 1-58113-493-2

[14] Baughman, Nathaniel E. ; Liberatore, Marc ; Levine, Brian N.: Cheat-proof playout for centralized and peer-to-peer gaming. In: *IEEE/ACM Trans. Netw.* 15 (2007), Nr. 1, S. 1–13. – ISSN 1063-6692

[15] Baumgart, Ingmar ; Heep, Bernhard ; Krause, Stephan: OverSim: A Flexible Overlay Network Simulation Framework. In: *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, Mai 2007, S. 79–84

[16] Baumgart, Ingmar ; Heep, Bernhard ; Krause, Stephan: OverSim: A scalable and flexible overlay framework for simulation and real network applications. In: *Ninth International Conference on Peer-to-Peer Computing (IEEE P2P), Seattle, WA, USA*, September 2009, S. 87–88

[17] Baumgart, Ingmar ; Heep, Bernhard ; Krause, Stephan: OverSim: Ein skalierbares und flexibles Overlay-Framework für Simulation und reale Anwendungen. In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 32, Issue 3 (2009), Oktober, S. 179–182

[18] Bavier, Andy ; Bowman, Mic ; Chun, Brent ; Culler, David ; Karlin, Scott ; Muir, Steve ; Peterson, Larry ; Roscoe, Timothy ; Spalink, Tammo ; Wawrzoniak, Mike: Operating system support for planetary-scale network services. In: *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, CA, USA*, USENIX Association, 2004, S. 19–19

[19] Bharambe, Ashwin R. ; Rao, Sanjay ; Seshan, Srinivasan: Mercury: a scalable publish-subscribe system for internet games. In: *Proceedings of the 1st workshop on Network and system support for games (NetGames '02), Braunschweig*, ACM, 2002, S. 3–9. – ISBN 1-58113-493-2

[20] BigWorld Pty: *BigWorld Server.* http://www.bigworldtech.com/technology/server.php. – (Überprüft am: 07.01.2011)

[21] Blizzard Entertainment: *Diablo II and Warcraft III Battle.net Bans*. http://forums.battle.net/thread.html?topicId=24401612571&sid=3000. April 2010. – (Überprüft am: 18.11.2010)

[22] Blizzard Entertainment Inc.: *Anzahl der Abonnenten von World Of Warcraft® Steigt weltweit auf 12 Millionen*. http://eu.blizzard.com/de-de/company/press/pressreleases.html?101007. – (Überprüft am: 19.12.2010)

[23] Blizzard Entertainment Inc.: *World of Warcraft*. http://eu.battle.net/wow/de/. – (Überprüft am: 16.12.2010)

[24] Burns, Brendan: *Darkstar: the java game server*. O'Reilly Media, Inc., 2007. – ISBN 0596514840

[25] Camp, Tracy ; Boleng, Jeff ; Davies, Vanessa: A survey of mobility models for ad hoc network research. In: *Wireless Communications and Mobile Computing* 2 (2002), August, Nr. 5, S. 483 – 502

[26] Castro, Miguel ; Druschel, Peter ; Kermarrec, Anne-Marie ; Rowstron, Antony I. T.: Scribe: a large-scale and decentralized application-level multicast infrastructure. In: *IEEE Journal on Selected Areas in Communications* 20 (2002), Oktober, Nr. 8, S. 1489–1499. – ISSN 0733-8716

[27] CCP Explorer: *Log the alts in (New PCU) New Record: 60,453*. http://www.eveonline.com/ingameboard.asp?a=topic&threadID=1331799&page=3#66. 2010. – (Überprüft am: 16.12.2010)

[28] CCP hf: *EVE Online*. http://www.eveonline.com/. – (Überprüft am: 16.12.2010)

[29] Chawathe, Yatin ; Ratnasamy, Sylvia ; Breslau, Lee ; Lanham, Nick ; Shenker, Scott: Making gnutella-like P2P systems scalable. In: *SIGCOMM '03, Karlsruhe*, ACM Press, 2003, S. 407–418. – ISBN 1-58113-735-4

[30] Chen, Kuan-Ta ; Huang, Polly ; Lei, Chin-Laung: Game traffic analysis: an MMORPG perspective. In: *Computer Networks* 50 (2006), Nr. 16, S. 3002–3023. – ISSN 1389-1286

[31] Chen, Kuan-Ta ; Huang, Polly ; Lei, Chin-Laung: How sensitive are online gamers to network quality? In: *Communications of the ACM* 49 (2006), Nr. 11, S. 34–38. – ISSN 0001-0782

[32] Chen, Kuan-Ta ; Jiang, Jhih-Wei ; Huang, Polly ; Chu, Hao-Hua ; Lei, Chin-Laung ; Chen, Wen-Chin: Identifying MMORPG bots: a traffic analysis approach. In: *EURASIP Journal of Advanced Signal Processing* 2009 (2009), Januar, S. 3:1–3:22. – ISSN 1110-8657

[33] Chen, Kuan-Ta ; Pao, Hsing-Kuo K. ; Chang, Hong-Chung: Game bot identification based on manifold learning. In: *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '08), Worcester, MA, USA*, ACM, 2008, S. 21–26. – ISBN 978-1-60558-132-3

[34] Chu, Yang hua ; Rao, Sanjay G. ; Seshan, Srinivasan ; Zhang, Hui: A case for end system multicast. In: *IEEE Journal on Selected Areas in Communications* 20 (2002), Oktober, Nr. 8, S. 1456 – 1471. – ISSN 0733-8716

[35] Claypool, Mark ; Claypool, Kajal: Latency and player actions in online games. In: *Communications of the ACM* 49 (2006), Nr. 11, S. 40–45. – ISSN 0001-0782

[36] Cohen, Bram: Incentives Build Robustness in BitTorrent. In: *1st Workshop on Economics of Peer-to-Peer Systems, Berkley, CA, USA*, Juni 2003

[37] Conrad, Michael ; Hof, Hans-Joachim: A Generic, Self-Organizing, and Distributed Bootstrap Service for Peer-to-Peer Networks. In: *Proceedings of New Trends in Network Architectures and Services: 2nd International Workshop on Self-Organizing Systems (IWSOS 2007), The Lake District, UK*, September 2007

[38] Consalvo, Mia: *Cheating: gaining advantage in videogames.* MIT Press, 2007. – ISBN 9780262033657

[39] Cronin, Eric ; Filstrup, Burton ; Jamin, Sugih: Cheat-proofing dead reckoned multiplayer games. In: *International Conference on Application and Development of Computer Games, Hong Kong, China*, 2003

[40] Dabek, Frank ; Zhao, Ben ; Druschel, Peter ; Kubiatowicz, John ; Stoica, Ion: Towards a Common API for Structured Peer-to-Peer Overlays. In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, 2003 (Lecture Notes in Computer Science 2735), S. 33–44. – ISBN 978-3-540-40724-9

[41] Diffie, Whitfield ; Hellman, Martin: New directions in cryptography. In: *IEEE Transactions on information Theory* 22 (1976), Nr. 6, S. 644–654. – ISSN 0018-9448

[42] Dinger, Jochen ; Waldhorst, Oliver: Decentralized Bootstrapping of P2P Systems: A Practical View. In: *Procedings of the 8th International Conference on Networking (IFIP TC6) , Aachen*, Mai 2009, S. 703–715

[43] Dolev, Danny ; Yao, Andrew: On the security of public key protocols. In: *IEEE Transactions on Information Theory* 29 (1983), März, Nr. 2, S. 198 – 208. – ISSN 0018-9448

[44] Doru Constantinescu, Adrian P.: Implementation of Application Layer Multicast in OverSim. In: *4th Euro-FGI Workshop on New Trends in Modelling, Quantitative Methods and Measurements, Ghent, Belgien*, 2007

[45] Eastlake 3rd, Donald ; Hansen, Tony: *US Secure Hash Algorithms (SHA and HMAC-SHA).* RFC 4634 (Informational). Juli 2006. – URL http://www.ietf.org/rfc/rfc4634.txt

[46] Entropia Universe: *Mindark PE AB.* http://www.entropiauniverse.com/. – (Überprüft am: 16.12.2010)

[47] Esch, Markus ; Scholtes, Ingo: A scale-free and self-organized P2P overlay for massive multiuser virtual environments. In: *5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2009 (CollaborateCom 2009), Washington D.C., USA*, November 2009

[48] Fan, Lu ; Taylor, Hamish ; Trinder, Phil: Mediator: a design framework for P2P MMOGs. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07), Melbourne, Australien*, ACM, 2007, S. 43–48. – ISBN 978-0-9804460-0-5

[49] Fannar, Halldor: *The Server Technology of EVE Online: How to Cope With 300,000 Players on One Server*. http://www.gdcvault.com/play/109/The-Server-Technology-of-EVE. – Lecture Talk on GDC Austin 2008, (Überprüft am: 19.12.2010)

[50] Federal Communications Commission: *SIXTH BROADBAND DEPLOYMENT REPORT*. http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-10-129A1.pdf. Juli 2010. – FCC-Nr. FCC-10-129A1

[51] Fiedler, Stefan ; Wallner, Michael ; Weber, Michael: A communication architecture for massive multiplayer games. In: *Proceedings of the 1st workshop on Network and system support for games (NetGames '02), Braunschweig*, ACM, 2002, S. 14–22. – ISBN 1-58113-493-2

[52] Ford, Bryan ; Srisuresh, Pyda ; Kegel, Dan: Peer-to-peer communication across network address translators. In: *Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC '05), Anaheim, CA, USA*, USENIX Association, 2005, S. 13–13

[53] Frey, Davide ; Royan, Jérôme ; Piegay, Romain ; Kermarrec, Anne-Marie ; Fessant, Fabrice L. ; Anceaume, Emmanuelle: Solipsis: A decentralized architecture for virtual environments. In: *Proceedings of 1st International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality 2008 (MMVE 2008), Reno, NV, USA*, März 2008

[54] Funcom: *Age of Conan*. http://www.ageofconan.com/. – (Überprüft am: 07.01.2011)

[55] Gai, Anh-Tuan ; Viennot, Laurent: Broose: a practical distributed hashtable based on the de-Bruijn topology. In: *Fourth International Conference on Peer-to-Peer Computing (P2P 2004), Zürich, Schweiz*, August 2004, S. 167–174

[56] Gamer, Thomas ; Scharf, Michael: Realistic Simulation Environments for IP-based Networks. In: *Digital Proceedings of 1st International Workshop on OMNeT++ (Hosted by SIMUTools '08: 1st International Conference on Simulation Tools and Techniques), Marseille, Frankreich*, ICST, März 2008. – ISBN 978-963-9799-20-2

[57] García, Pedro ; Pairot, Carles ; Mondéjar, Rubén ; Pujol, Jordi ; Tejedor, Helio ; Rallo, Robert: PlanetSim: A New Overlay Network Simulation Framework. In: *Fifth International Workshop on Software Engineering and Middleware (SEM 2005), Lissabon, Portugal* Bd. Volume 3437/2005, 2005, S. 123–136. – ISBN 978-3-540-25328-0

[58] GauthierDickey, Chris ; Lo, Virginia ; Zappala, Daniel: Using n-trees for scalable event ordering in peer-to-peer games. In: *Proceedings of the international workshop on Network and operating systems support for digital audio and video (NOSSDAV '05), Stevenson, WA, USA*, ACM, 2005, S. 87–92. – ISBN 1-58113-987-X

[59] GauthierDickey, Chris ; Zappala, Daniel ; Lo, Virginia ; Marr, James: Low latency and cheat-proof event ordering for peer-to-peer games. In: *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '04), Cork, Irland*, ACM, 2004, S. 134–139. – ISBN 1-58113-801-6

[60] Goodman, Josh ; Verbrugge, Clark: A peer auditing scheme for cheat elimination in MMOGs. In: *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '08), Worcester, MA, USA*, ACM, 2008, S. 9–14. – ISBN 978-1-60558-132-3

[61] Griwodz, Carsten: State replication for multiplayer games. In: *Proceedings of the 1st workshop on Network and system support for games (NetGames '02), Braunschweig*, ACM, 2002, S. 29–35. – ISBN 1-58113-493-2

[62] Guinness World Records: *Most expensive virtual property.* http://www.guinnessworldrecords.com/Search/Details/Mostexpensivevirtual-property/63530.htm. – (Überprüft am: 11.02.2011)

[63] Gummadi, Krishna P. ; Saroiu, Stefan ; Gribble, Steven D.: King: estimating latency between arbitrary internet end hosts. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW '02), Marseille, Frankreich*, ACM, 2002, S. 5–18. – ISBN 1-58113-603-X

[64] Hosseini, Mojtaba ; Ahmed, Dewan T. ; Shirmohammadi, Shervin ; Georganas, Nicolas D.: A Survey of Application-Layer Multicast Protocols. In: *IEEE Communications Surveys Tutorials* 9 (2007), Nr. 3, S. 58 –74. – ISSN 1553-877X

[65] Hsiao, Tsun-Yu ; Yuan, Shyan-Ming: Practical middleware for massively multiplayer online games. In: *IEEE Internet Computing* 9 (2005), September, Nr. 5, S. 47 – 54. – ISSN 1089-7801

[66] Hu, Shun-Yun ; Chen, Jui-Fa ; Chen, Tsu-Han: VON: a scalable peer-to-peer network for virtual environments. In: *IEEE Network* 20 (2006), Juli, Nr. 4, S. 22 –31. – ISSN 0890-8044

[67] Hu, Shun-Yun ; Liao, Guan-Ming: Scalable peer-to-peer networked virtual environment. In: *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames '04), Portland, OR, USA*, ACM, 2004, S. 129–133. – ISBN 1-58113-942-X

[68] Iimura, Takuji ; Hazeyama, Hiroaki ; Kadobayashi, Youki: Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In: *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames '04), Portland, OR, USA*, ACM, 2004, S. 116–120. – ISBN 1-58113-942-X

[69] Jagex Ltd.: *RuneScape*. http://www.runescape.com/. – (Überprüft am: 16.12.2010)

[70] Jelasity, Márk ; Montresor, Alberto ; Jesi, Gian P. ; Voulgaris, Spyros: *The Peersim Simulator*. http://peersim.sf.net. – (Überprüft am: 18.11.2010)

[71] Jiang, Jehn-Ruey ; Chiou, Jiun-Shiang ; Hu, Shun-Yun: Enhancing Neighborship Consistency for Peer-to-Peer Distributed Virtual Environments. In: *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops (ICDCSW '07), Toronto, Kanada*, IEEE Computer Society, 2007, S. 71. – ISBN 0-7695-2838-4

[72] Johnson, David ; Perkins, Charles ; Arkko, Jari: *Mobility Support in IPv6*. RFC 3775 (Proposed Standard). Juni 2004. – URL http://www.ietf.org/rfc/rfc3775.txt

[73] Kaashoek, M. Frans ; Karger, David R.: Koorde: A Simple Degree-Optimal Distributed Hash Table. In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, 2003 (Lecture Notes in Computer Science 2735/2003), S. 98–107. – ISBN 978-3-540-40724-9

[74] Kabus, Patric ; Terpstra, Wesley W. ; Cilia, Mariano ; Buchmann, Alejandro P.: Addressing cheating in distributed MMOGs. In: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames '05), Hawthorne, NY, USA*, ACM, 2005. – ISBN 1-59593-156-2

[75] Katsaros, Konstantinos ; Kemerlis, Vasilios P. ; Stais, Charilaos ; Xylomenos, George: A BitTorrent module for the OMNeT++ simulator. In: *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems 2009 (MASCOTS '09), London, UK*, September 2009. – ISSN 1526-7539

[76] Kawahara, Yoshihiro ; Aoyama, Tomonori ; Morikawa, Hiroyuki: A Peer-to-Peer Message Exchange Scheme for Large-Scale Networked Virtual Environments. In: *Telecommunication Systems* 25 (2004), S. 353–370. – ISSN 1018-4864

[77] Keller, Joaquin ; Simon, Gwendal: Toward a peer-to-peer shared virtual reality. In: *22nd International Conference on Distributed Computing Systems, Workshops, (ICDCSW '02), Wien, Östereich*, Juli 2002, S. 695 – 700

[78] Kim, Beob ; You, Kang: A Dynamic Hierarchical Map Partitioning for MMOG. In: Min, Geyong (Hrsg.) ; Di Martino, Beniamino (Hrsg.) ; Yang, Laurence (Hrsg.) ; Guo, Minyi (Hrsg.) ; Ruenger, Gudula (Hrsg.): *Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops, Sorrento, Italien*. Springer Berlin / Heidelberg, 2006 (Lecture Notes in Computer Science 4331), S. 813–822

[79] Knutsson, Björn ; Lu, Honghui ; Xu, Wei ; Hopkins, Bryan: Peer-to-peer support for massively multiplayer games. In: *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), Hong Kong, China* Bd. 1, März 2004. – ISSN 0743-166X

[80] Kolberg, Mario ; Buford, John: An XCAST multicast implementation for the OverSim simulator. In: *Proceedings of the 7th IEEE conference on Consumer communications and networking conference (CCNC'10), Las Vegas, NV, USA*, IEEE Press, 2010, S. 764–768. – ISBN 978-1-4244-5175-3

[81] Kovacevic, Aleksandra ; Kaune, Sebastian ; Mukherjee, Patrick ; Liebau, Nicolas ; Steinmetz, Ralf: Benchmarking Platform for Peer-to-Peer Systems. In: *it - Information Technology (Methods and Applications of Informatics and Information Technology)* 49 (2007), September, Nr. 5, S. 312–319

[82] Krasnyansky, Maxim ; Thiel, Florian: *Universal TUN/TAP device driver*. http://www.kernel.org/doc/Documentation/networking/tuntap.txt. – (Überprüft am: 28.10.2010)

[83] Krause, Stephan: A Case for Mutual Notification: A Survey of P2P Protocols for Massively Multiplayer Online Games, Worcester, MA, USA. In: *Proceedings of NetGames 2008 Network and Systems Support for Games*, Oktober 2008, S. 28–33

[84] Krause, Stephan: Coping with Hotspots: AOI Adaption Strategies for P2P Networked Virtual Environments. In: *International Conference on Ultra Modern Telecommunications, St. Petersburg, Russland*, Oktober 2009

[85] Krause, Stephan ; Hübsch, Christian: Scalable application-layer multicast simulations with oversim. In: *Proceedings of the 7th IEEE conference on Consumer communications and networking conference (CCNC'10), Las Vegas, NV, USA*, IEEE Press, 2010, S. 314–318. – ISBN 978-1-4244-5175-3

[86] Kunzmann, Gerald ; Nagel, Robert ; Hossfeld, Tobias ; Binzenhofer, Andreas ; Eger, Kolja: Efficient Simulation of Large-Scale P2P Networks: Modeling Network Transmission Times. In: *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP '07), Neapel, Italien*, IEEE Computer Society, 2007, S. 475–481. – ISBN 0-7695-2784-1

[87] Lamport, Leslie ; Shostak, Robert ; Pease, Marshall: The Byzantine Generals Problem. In: *ACM Transactions on Programming Language Syststems* 4 (1982), Juli, S. 382–401. – ISSN 0164-0925

[88] Laurens, Peter ; Paige, Richard F. ; Brooke, Philip J. ; Chivers, Howard: A Novel Approach to the Detection of Cheating in Multiplayer Online Games. In: *12th IEEE International Conference on Engineering Complex Computer Systems, Auckland, Neuseeland*, Juli 2007, S. 97 –106

[89] Lee, Hsiu-Hui ; Sun, Chin-Hua: Load-balancing for peer-to-peer networked virtual environment. In: *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames '06), Singapur*, ACM, 2006. – ISBN 1-59593-589-4

[90] Li, Jinyang ; Stribling, Jeremy ; Morris, Robert ; Kaashoek, M. Frans ; Gil, Thomer M.: A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In: *24th Annual Joint Conference of the IEEE*

Computer and Communications Societies (INFOCOM 2005), Miami, FL, USA Bd. 1, März 2005, S. 225–236

[91] Lim, Jungyoul ; Chung, Jaeyong ; Kim, Jinryong ; Shim, Kwanghyun: A Dynamic Load Balancing for Massive Multiplayer Online Game Server. In: Harper, Richard (Hrsg.) ; Rauterberg, Matthias (Hrsg.) ; Combetto, Marco (Hrsg.): *Entertainment Computing - ICEC 2006*. Springer Berlin / Heidelberg, 2006 (Lecture Notes in Computer Science 4161), S. 239–249

[92] Linden Research, Inc.: *Second Life*. http://secondlife.com/. – (Überprüft am: 16.12.2010)

[93] Linden Research, Inc.: *Second Life User Statistics*. http://secondlife.com/xmlhttp/secondlife.php. – (Überprüft am: 19.12.2010)

[94] Lombardi, Julian: *Open Cobalt*. http://www.opencobalt.org. – (Überprüft am: 16.12.2010)

[95] Ma, Alex: *CAIDA : tools : measurement : skitter*. http://www.caida.org/tools/measurement/skitter/. Juni 2008. – (Überprüft am: 08.06.2008)

[96] Maymounkov, Petar ; Mazières, David: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In: *Peer-to-Peer Systems: First International Workshop (IPTPS 2002), Cambridge, MA, USA*, März 2002 (Lecture Notes in Computer Science 2429), S. 53–65

[97] Mills, David ; Martin, Jim ; Burbank, Jack ; Kasch, William: *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905 (Proposed Standard). Juni 2010. – URL http://www.ietf.org/rfc/rfc5905.txt

[98] Morillo, Pedro ; Moncho, Walter ; Orduña, Juan M. ; Duato, Jose: Providing Full Awareness to Distributed Virtual Environments Based on Peer-to-Peer Architectures. In: *Advances in Computer Graphics*. Springer, 2006 (Lecture Notes in Computer Science 4035), S. 336–347. – ISBN 978-3-540-35638-7

[99] Morningstar, Chip ; Farmer, F. Randall: *The lessons of Lucasfilm's Habitat*. Kap. 10, S. 273–302. In: Benedikt, Michael L. (Hrsg.): *Cyberspace: first steps*. Cambridge, MA, USA : MIT Press, 1991. – ISBN 0-262-02327-X

[100] Naicken, Stephen ; Basu, Anirban ; Livingston, Barnaby ; Rodhetbhai, Sethalat: A Survey of Peer-to-Peer Network Simulators. In: *Proceedings of The Seventh Annual Postgraduate Symposium (PGNET), Liverpool, UK*, 2006

[101] NCsoft: *Guild Wars*. http://www.guildwars.com/. – (Überprüft am: 07.01.2011)

[102] Neumann, Christoph ; Prigent, Nicolas ; Varvello, Matteo ; Suh, Kyoungwon: Challenges in peer-to-peer gaming. In: *ACM SIGCOMM - Computer Communication Review* 37 (2007), Nr. 1, S. 79–82. – ISSN 0146-4833

[103] Ng, T.S.E. ; Zhang, Hui: Predicting Internet network distance with coordinates-based approaches. In: *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, USA* 1 (2002), S. 170–179. – ISSN 0743-166X

[104] OMG Labs: *Eve-Online Status Monitor*. http://eve-offline.net/?server=tranquility. – (Überprüft am: 16.12.2010)

[105] Open Wonderland Foundation: *Open Wonderland*. http://openwonderland.org/. – (Überprüft am: 16.12.2010)

[106] OpenSimulator.org: *OpenSimulator*. http://opensimulator.org/. – (Überprüft am: 11.02.2011)

[107] Pellegrino, Joseph D. ; Dovrolis, Constantinos: Bandwidth requirement and state consistency in three multiplayer game architectures. In: *Proceedings of the 2nd workshop on Network and system support for games (NetGames '03), Redwood City, CA, USA*, ACM, 2003, S. 52–59. – ISBN 1-58113-734-6

[108] Perkins, Charles: *IP Mobility Support for IPv4, Revised*. RFC 5944 (Proposed Standard). November 2010. – URL http://www.ietf.org/rfc/rfc5944.txt

[109] Peterson, Larry ; Anderson, Tom ; Culler, David ; Roscoe, Timothy: A blueprint for introducing disruptive technology into the Internet. In: *ACM SIGCOMM - Computer Communication Review* 33 (2003), Nr. 1, S. 59–64. – ISSN 0146-4833

[110] Pittman, Daniel ; GauthierDickey, Chris: A measurement study of virtual populations in massively multiplayer online games. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07), Melbourne, Australien*, ACM, 2007, S. 25–30. – ISBN 978-0-9804460-0-5

[111] Qiu, Ying ; Zhou, Jianying ; Deng, Robert: Security Analysis and Improvement of Return Routability Protocol. In: Burmester, Mike (Hrsg.) ; Yasinsac, Alec (Hrsg.): *Secure Mobile Ad-hoc Networks and Sensors*. Springer Berlin / Heidelberg, 2006 (Lecture Notes in Computer Science 4074), S. 174–181

[112] Quax, Peter ; Dierckx, Jeroen ; Cornelissen, Bart ; Lamotte, Wim: ALVIC versus the internet: redesigning a networked virtual environment architecture. In: *International Journal on Computer Games Technology* (2008), Januar, S. 4:1–4:9. – ISSN 1687-7047

[113] Reining, Jan-Philipp: *Cheater, der Versuch einer Analyse (Teil 2)*. http://www.esl.eu/de/esl_blog/news/123103/. Mai 2010. – (Überprüft am: 23.11.2010)

[114] Reynolds, Craig W.: Flocks, herds and schools: A distributed behavioral model. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87), Anaheim, CA, USA*, ACM Press, 1987, S. 25–34. – ISBN 0-89791-227-6

[115] Rhea, Sean ; Geels, Dennis ; Roscoe, Timothy ; Kubiatowicz, John: Handling churn in a DHT. In: *Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC '04), Boston, MA, USA*, USENIX Association, 2004, S. 10–10

[116] Ricci, Laura ; Salvadori, Andrea: Nomad: virtual environments on P2P voronoi overlays. In: *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems (OTM'07), Vilamoura, Portugal*, Springer-Verlag, 2007, S. 911–920. – ISBN 3-540-76889-0, 978-3-540-76889-0

[117] Rieche, Simon ; Wehrle, Klaus ; Fouquet, Marc ; Niedermayer, Heiko ; Petrak, Leo ; Carle, Georg: Peer-to-Peer-Based Infrastructure Support for Massively Multiplayer Online Games. In: *4th IEEE Consumer Communications and Networking Conference, 2007 (CCNC 2007), Las Vegas, NV, USA* IEEE (Veranst.), 2007, S. 763–767. – ISBN 1424406676

[118] Rivest, Ronald L. ; Shamir, Adi ; Adleman, Leonard: A method for obtaining digital signatures and public-key cryptosystems. In: *Communications of the ACM* 21 (1978), Februar, S. 120–126. – ISSN 0001-0782

[119] Robles, Rosslin J. ; Yeo, Sang-Soo ; Moon, Young-Deuk ; Park, Gilcheol ; Kim, Seoksoo: Online Games and Security Issues. In: *Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking (FGCN '08), Sanya, China*. Washington, DC, USA : IEEE Computer Society, Dezember 2008, S. 145–148. – ISBN 978-0-7695-3431-2

[120] Rosenberg, Jonathan ; Mahy, Rohan ; Matthews, Philip ; Wing, Dan: *Session Traversal Utilities for NAT (STUN)*. RFC 5389 (Proposed Standard). Oktober 2008. – URL http://www.ietf.org/rfc/rfc5389.txt

[121] Rowstron, Antony ; Druschel, Peter: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: *Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg*, Springer, November 2001 (Lecture Notes in Computer Science 2218). – ISSN 0302-9743

[122] Schmieg, Arne ; Stieler, Michael ; Jeckel, Sebastian ; Kabus, Patric ; Kemme, Bettina ; Buchmann, Alejandro: pSense - Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games. In: *IEEE International Conference on Peer-to-Peer Computing (P2P2008), Los Alamitos, CA, USA*, IEEE Computer Society, 2008, S. 247–256. – ISBN 978-0-7695-3318-6

[123] Schwerdel, Dennis ; Günther, Daniel ; Henjes, Robert ; Reuther, Bernd ; Müller, Paul: German-Lab Experimental Facility. In: *Future Internet Symposium (FIS) 2010, Berlin*, September 2010

[124] Sebayang, Andreas: *Details zu Charakter-Transfers in World of Warcraft*. http://www.golem.de/0606/46092.html. – (Überprüft am: 07.01.2011)

[125] Shudo, Kazuyuki ; Tanaka, Yoshio ; Sekiguchi, Satoshi: Overlay Weaver: An overlay construction toolkit. In: *Computer Communications* 31 (2008), Nr. 2, S. 402 – 412. – Special Issue: Foundation of Peer-to-Peer Computing. – ISSN 0140-3664

[126] Smed, Jouni ; Kaukoranta, Timo ; Hakonen, Harri: Aspects of Networking in Multiplayer Computer Games. In: Sing, Loo W. (Hrsg.) ; Man, Wan H. (Hrsg.) ; Wai, Wong (Hrsg.): *Proceedings of International Conference on Application and Development of Computer Games in the 21st Century, Hong Kong, China*, November 2001, S. 74–81

[127] Smeet Communications GmbH: *Smeet*. http://smeet.com. – (Überprüft am: 16.12.2010)

[128] Sony Online Entertainment LLC.: *PlanetSide*. http://planetside.station.sony.com/. – (Überprüft am: 16.12.2010)

[129] Steiner, Moritz ; En-Najjary, Taoufik ; Biersack, Ernst W.: A global view of kad. In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07), San Diego, CA, USA*, ACM, 2007, S. 117–122. – ISBN 978-1-59593-908-1

[130] Stoica, Ion ; Morris, Robert ; Karger, David ; Kaashoek, M. Frans ; Balakrishnan, Hari: Chord: A scalable peer-to-peer lookup service for internet applications. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01), San Diego, CA, USA*, ACM, 2001, S. 149–160. – ISBN 1-58113-411-8

[131] Sueselbeck, Richard ; Schiele, Gregor ; Seitz, Sebastian ; Becker, Christian: Adaptive Update Propagation for Low-Latency Massively Multi-User Virtual Environments. In: *Proceedings of 18th Internatonal Conference on Computer Communications and Networks (ICCCN 2009), San Francisco, CA, USA*, IEEE Computer Society, 2009

[132] Süselbeck, Richard ; Schiele, Gregor ; Becker, Christian: Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud. In: *Proceedings of the 8th Annual Workshop on Network and Systems Support for Games (NetGames '09), Paris, Frankreich*, IEEE Press, 2009, S. 14:1–14:2. – ISBN 978-1-4244-5605-5

[133] Suznjevic, Mirko ; Dobrijevic, Ognjen ; Matijasevic, Maja: MMORPG Player actions: Network performance, session patterns and latency requirements analysis. In: *Multimedia Tools and Applications* 45 (2009), Nr. 1-3, S. 191–214. – ISSN 1380-7501

[134] Tarng, Pin-Yun ; Chen, Kuan-Ta ; Huang, Polly: An analysis of WoW players' game hours. In: *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '08), Worcester, MA, USA*, ACM, 2008, S. 47–52. – ISBN 978-1-60558-132-3

[135] Thawonmas, Ruck ; Kashifuji, Yoshitaka ; Chen, Kuan-Ta: Detection of MMORPG bots based on behavior analysis. In: *Proceedings of the 2008*

International Conference on Advances in Computer Entertainment Technology (ACE '08), Yokohama, Japan, ACM, 2008, S. 91–94. – ISBN 978-1-60558-393-8

[136] Tran-Gia, Phuoc ; Feldmann, Anja ; Steinmetz, Ralf ; Eberspächer, Jörg ; Zitterbart, Martina ; Müller, Paul ; Schotten, Hans: *G-Lab White Paper Phase 1 - Studien und Experimentalplattform für das Internet der Zukunft*. https://www.german-lab.de/fileadmin/Press/G-Lab_White_Paper_Phase1.pdf. Januar 2009. – (Überprüft am: 18.11.2010)

[137] Triebel, Tonio ; Guthier, Benjamin ; Effelsberg, Wolfgang: Skype4Games. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07), Melbourne, Australien*, ACM, 2007, S. 13–18. – ISBN 978-0-9804460-0-5

[138] vacbanned.com: *VACBanned: Check if a SteamID is VAC banned or not* . http://www.vacbanned.com/view/statistics. November 2010. – (Überprüft am: 23.11.2010)

[139] Varga, András: *The INET Framework*. http://inet.omnetpp.org. – (Überprüft am: 18.11.2010)

[140] Varga, András: *OMNeT++*. http://www.omnetpp.org. – (Überprüft am: 18.11.2010)

[141] Varga, András: The OMNeT++ Discrete Event Simulation System. In: *Proceedings of the European Simulation Multiconference (ESM2001), Prag, Tschechien*, Juni 2001

[142] Varga, András ; Hornig, Rudolf: An overview of the OMNeT++ simulation environment. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops (Simutools '08), Marseille, Frankreich*, ICST, 2008. – ISBN 978-963-9799-20-2

[143] Varvello, Matteo ; Biersack, Ernst ; Diot, Christophe: Dynamic clustering in delaunay-based P2P networked virtual environments. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07), Melbourne, Australien*, ACM, 2007, S. 105–110. – ISBN 978-0-9804460-0-5

[144] Vik, Knut-Helge: Game state and event distribution using proxy technology and application layer multicast. In: *Proceedings of the 13th annual ACM international conference on Multimedia (MULTIMEDIA '05), Singapur*, ACM, 2005, S. 1041–1042. – ISBN 1-59593-044-2

[145] Wacker, Arno ; Schiele, Gregor ; Schuster, Sebastian ; Weis, Torben: Towards an authentication service for Peer-to-Peer based Massively Multiuser Virtual Environments. In: *International Journal of Advanced Media and Communication* 2 (2008), Dezember, S. 364–379. – ISSN 1462-4613

[146] Webb, Steven D. ; Soh, Sieteng: Cheating in networked computer games: a review. In: *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts (DIMEA '07), Perth, Australien*, ACM, 2007, S. 105–112. – ISBN 978-1-59593-708-7

[147] Webb, Steven D. ; Soh, Sieteng ; Lau, William: RACS: a referee anti-cheat scheme for P2P gaming. In: *17th International workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV'07), Urbana-Champaign, IL, USA*, Juni 2007, S. 37–42

[148] Wehrle, Klaus ; Reber, Jochen ; Kahmann, Verena: A Simulation Suite for Internet Nodes with the Ability to Integrate Arbitrary Quality of Service Behavior. In: *Proceedings of Communication Networks And Distributed Systems Modeling And Simulation Conference (CNDS 2001), Phoenix, AZ, USA*, 2001, S. 115–122

[149] Yamamoto, Shinya ; Murata, Yoshihiro ; Yasumoto, Keiichi ; Ito, Minoru: A distributed event delivery method with load balancing for MMORPG. In: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames '05), Hawthorne, NY, USA*, ACM, 2005. – ISBN 1-59593-156-2

[150] Yan, Jeff ; Randell, Brian: A systematic classification of cheating in online games. In: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames '05), Hawthorne, NY, USA*, ACM, 2005. – ISBN 1-59593-156-2

[151] Yao, Zhongmei ; Leonard, Derek ; Wang, Xiaoming ; Loguinov, Dmitri: Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks. In: *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols (ICNP '06), Santa Barbara, CA, USA*, IEEE Computer Society, 2006, S. 32–41. – ISBN 1-4244-0593-9

[152] Ye, Meng ; Cheng, Lang: System-performance modeling for massively multiplayer online role-playing games. In: *IBM Systems Journal* 45 (2006), Nr. 1, S. 45 –48. – ISSN 0018-8670

[153] Yeo, Chai K. ; Lee, Bu-Sung ; Er, Meng H.: A survey of application level multicast techniques. In: *Computer Communications* 27 (2004), Nr. 15, S. 1547 – 1568. – ISSN 0140-3664

[154] Yu, Anthony ; Vuong, Son T.: MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In: *Proceedings of the international workshop on Network and operating systems support for digital audio and video (NOSSDAV '05), Stevenson, WA, USA*, ACM, 2005, S. 99–104. – ISBN 1-58113-987-X