Universität Trier

Berücksichtigung der Infrastruktur- und Konsistenzaspekte zur Verbesserung der Skalierbarkeit und der Lastverteilung in verteilten virtuellen Umgebungen

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

M.Sc., Dipl.-Inform. (FH) Hermann Schloß

Trier, 10. März 2011

Amtierender Dekan: Prof. Dr. Ralf Münnich

Berichterstatter: Prof. Dr. Peter Sturm
Prof. Dr. Rainer Oechsle

Tag der Promotion: 10. März 2011

Vorwort

Das Hochschulprogramm des Landes Rheinland-Pfalz „Wissen schafft Zukunft“ befasst sich mit der Clusterbildung zwischen den Fachhochschulen (FH) und den Universitäten (Uni) an den Hochschulstandorten in Rheinland-Pfalz. Es eröffnet Master-Absolventen einer Fachhochschule die Möglichkeit, im Rahmen eines so genannten Kooperationskorridors FH - Uni zu promovieren. Im Rahmen dieses Programms begann ich im Oktober 2006 in der Forschungsgruppe „Systemsoftware und Verteilte Systeme“ von Prof. Dr. Peter Sturm an der Universität Trier mit meiner Promotion.

An erster Stelle möchte ich mich bei Prof. Dr. Peter Sturm bedanken. Ohne ihn wäre mein Promotionswunsch wahrscheinlich nicht in Erfüllung gegangen. Im Jahr 2006 standen die meisten Universitäten Master-Absolventen von Fachhochschulen noch sehr skeptisch gegenüber. Mit seiner liberalen Einstellung und der Aufnahme eines FH-Master-Absolventen in seine Arbeitsgruppe hat Prof. Dr. Peter Sturm als einer der ersten Universitätsprofessoren den Mut bewiesen, neue Wege in der Hochschulpolitik zu gehen. Ich bin Prof. Dr. Peter Sturm für die Chance, die er mir gegeben hat, und für sein Vertrauen, das er in mich gesetzt hat, genauso wie für seine fachgerechte, kompetente und sachliche Betreuung sehr dankbar.

Dem Gegenstand des Hochschulprogramms „Wissen schafft Zukunft“ und dem neuen hochschulpolitischen Zeitgeist wurde mit der Wahl der Gutachter Nachdruck verliehen, indem neben Prof. Dr. Peter Sturm auch Prof. Dr. Rainer Oechsle von der Fachhochschule Trier als Zweitgutachter (Berichterstatter und Prüfungsausschussmitglied) bestellt wurde.

Prof. Dr. Rainer Oechsle hat in meiner akademischen Laufbahn eine entscheidende Rolle gespielt. Unsere produktive Zusammenarbeit hat mit einem Projekt im Jahr 2003 begonnen. Daraus entstand im Jahr 2005 die erste gemeinsame Publikation, der in den Folgejahren noch einige weitere Publikationen folgten. Die hohen Ansprüche, die Prof. Dr. Rainer Oechsle an sich selbst und an seine Projektmitarbeiter stellt, und seine Fähigkeit, tiefgehende Probleme zu erfassen und sie konstruktiv anzugehen, waren für mich der Grund, ihn als Zweitgutachter vorzuschlagen. Ich danke ihm für fruchtbare Diskussionen, zahlreiche konstruktive Hinweise und für seine Bereitschaft, als Zweitgutachter zu fungieren.

Des Weiteren möchte ich mich ganz besonders bei Prof. Dr. Andreas Künkler bedanken, der mir bei meiner Entscheidung zu promovieren zur Seite gestanden

hat und alles daran gesetzt hat, dass mein Wunsch in Erfüllung geht. Exemplarisch seien an dieser Stelle die mehrmalige Verlängerung meiner befristeten Arbeitsverträge an der Fachhochschule Trier oder die Unterstützung bei organisatorischen Angelegenheiten im „Wissen schafft Zukunft“-Promotionsprogramm genannt.

Allen meinen Koautoren, AG-Mitgliedern an der Uni Trier und Studierenden der FH Trier, die mich mit ihren Projektarbeiten unterstützt haben, sei für ihre Hilfe gedankt. Mein Dank an dieser Stelle gilt besonders: Andreas Baumann, Angelo Beck, Jean Botev, Christian Bettinger, Markus Esch, Patrick Graz, Frank Hausen, Alex Höhfeld, Gennadiy Poryev, Bernd Klasen, Ingo Scholtes, Prof. Dr. Georg Schneider, Alex Vinzl und Benjamin Zech.

Prof. Dr. Andreas Lux, Kay Barzen, Angela Becker-Kob, Ebba Eeten, Gaby Elenz, Hermann Prüm, Markus Schwinn und Maximilian Witek danke ich dafür, dass sie meine Dissertation sowie meine Publikationen Korrektur gelesen haben. Prof. Dr. Andreas Lux danke ich außerdem für seine motivierenden, immer aufbauenden und auflockernden Gespräche, die wir während unserer gemeinsamen Fahrten zum FH-Arbeitsplatz führten.

Auch all denen, die an dieser Stelle nicht namentlich erwähnt wurden, mich aber durch ihren Glauben an mich und meine Arbeit unterstützt haben, sei ebenfalls gedankt.

Zu guter Letzt danke ich meinen Eltern und meiner Familie für ihre fortwährende Unterstützung. Die Tatsache, dass ich oftmals zu wenig Zeit für sie hatte, haben sie mit Verständnis aufgenommen und mir somit ein ideales Arbeitsumfeld ermöglicht.

Kurzfassung

Virtuelle Umgebungen erfreuen sich in den letzten Jahren einer großen Beliebtheit und können kontinuierlich wachsende Nutzerzahlen vorweisen. Deshalb wird erwartet, dass die ohnehin große Nutzergemeinde weiterhin wächst. Dieses Wachstum stellt jedoch die Entwickler und die Betreiber von virtuellen Umgebungen vor eine Herausforderung. Die meisten heutzutage erfolgreichen virtuellen Umgebungen basieren auf einer Client-Server-Infrastruktur, in der der Server für die Verwaltung der gesamten Umgebung zuständig ist. Bei einer gleichzeitigen Inanspruchnahme durch viele Nutzer werden die serverbasierten Umgebungen jedoch massiven Skalierbarkeits- und Lastverteilungsproblemen ausgesetzt. Diese Probleme führen beispielsweise dazu, dass Nutzer Wartezeiten beim Einloggen in Kauf nehmen müssen bzw. selbst im Falle eines erfolgreichen Logins die „überfüllten“ virtuellen Regionen nicht betreten dürfen. Deshalb wird in dieser Arbeit untersucht, ob eine Verbesserung der Skalierbarkeit und der Lastverteilung in virtuellen Umgebungen durch die Verwendung alternativer Infrastrukturansätze erreicht werden kann. Außerdem werden Maßnahmen zur weiteren Entlastung der Basis-Infrastruktur entwickelt.

Diese Verbesserung soll zum einen dadurch erzielt werden, dass anstelle eines Servers ein Server-Verbund (Peer-to-Peer-System) als eine Art Management-Schicht agieren soll, in der jeder Knoten nur einen bestimmten Teil der virtuellen Umgebung verwaltet. Dabei wird die gewünschte Lastverteilung durch die Aufteilung der Verantwortungsbereiche auf die Knoten der Management-Schicht erreicht. Gleichzeitig entsteht aber ein Mehraufwand für die Konstruktion bzw. Erhaltung einer solchen Management-Schicht. Deshalb werden in dieser Arbeit die Vor- und Nachteile eines Infrastrukturwechsels hin zu dezentralisierten Infrastruktur-Ansätzen untersucht.

Zum anderen wird eine spürbare Entlastung der Management-Schicht und somit eine Verbesserung der Skalierbarkeit und der Lastverteilung durch die Übertragung einiger Verwaltungsaufgaben an die Clients angestrebt. Zu diesen Verwaltungsaufgaben zählen die Gruppenkommunikation und das Konsistenz-Handling. Dazu wird in dieser Arbeit erforscht, inwiefern die Clients, die sich innerhalb derselben virtuellen Region befinden, selbst die Gruppenkommunikation über eine Multicast-Infrastruktur regeln können bzw. in die Wahrung der Konsistenz eines gemeinsamen Zustandes und der gemeinsam genutzten Objekte involviert werden können.

Bei dem clientbasierten Konsistenz-Handling sollte vor allem auf die besonderen Interaktivitäts- und Konsistenzanforderungen, die Nutzer an die virtuellen Umgebungen stellen, geachtet werden. Dazu wird ein neues Konsistenzmodell definiert, welches die Einhaltung der strengen Interaktivitätsanforderungen erzwingt und die Konsistenz mit einer bestimmten Wahrscheinlichkeit garantieren kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

Einleitung

Verteilte virtuelle Umgebungen *(Distributed Virtual Environments – DVEs)* und insbesondere Online-Spiele *(Massive Multiplayer Online Games – MMOGs)* erfreuen sich in den letzten Jahren einer großen Beliebtheit. Der rasche Anstieg der Zahl der Internetnutzer um 399.3% in den Jahren 2000 bis 2009[1] sowie die immer schneller werdenden Internetverbindungen gepaart mit der ständigen Weiter- bzw. Neuentwicklung von graphisch ansprechenden dreidimensionalen virtuellen Umgebungen bzw. Spielen mit packenden Handlungen sind nur einige Indizien dafür, dass sich die ohnehin große Nutzergemeinde in der nächsten Zeit weiterhin rapide vergrößern wird.

Die Popularität dieser Umgebungen treibt deren Weiterentwicklung voran, zeigt aber auch gleichzeitig die Grenzen der momentan verwendeten Technologien auf. Bei einer gleichzeitigen Inanspruchnahme durch viele Nutzer sind die verwendeten zentralisierten Client-Server-Infrastrukturen, trotz ihrer hohen Rechen- und Speicherkapazitäten, oftmals überlastet. Dadurch werden die zentralisierten serverbasierten Umgebungen massiven Skalierbarkeits- und Lastverteilungsproblemen ausgesetzt. Diese Tatsache kann dazu führen, dass die Nutzer einer virtuellen Umgebung bei der Anmeldung mit Wartezeiten rechnen oder gar auf andere Server ausweichen müssen. Diese Umstände führen zu einer Unzufriedenheit der Nutzer, die nach Möglichkeit minimiert oder gar vermieden werden soll. Die Wachstumsrate der Zahl der Internetnutzer lässt vermuten, dass die klassische Client-Server-Netzwerk-Technologie den Skalierbarkeitsanforderungen nicht gerecht werden kann. Um die erforderliche Skalierbarkeit in den virtuellen Umgebungen sicherstellen zu können, müssen neue Wege beschritten werden. Deshalb werden in dieser Arbeit unterschiedliche Alternativen zu der herkömmlichen Client-Server-Architektur betrachtet und Ansätze untersucht, die zur Verbesserung der Skalierbarkeit und Lastverteilung in den virtuellen Umgebungen beitragen können.

Die meisten heute erfolgreichen virtuellen Umgebungen basieren auf einer Client-Server-Infrastruktur, in der ein Server die Verantwortung und Verwaltung für die gesamte Umgebung übernimmt (siehe Abbildung 1.1). Bei hohen Nutzerzahlen erfahren solche Infrastrukturen inhärent Skalierbarkeits- und Lastverteilungsprobleme.

[1] `http://www.internetworldstats.com`

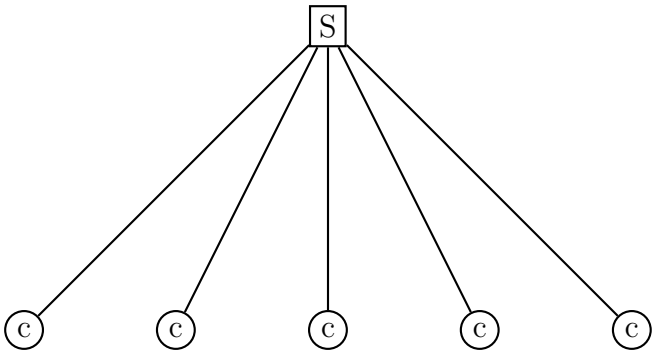


Abbildung 1.1. Client-Server-Infrastruktur

Der erste Verbesserungsansatz wäre die Verwendung eines dezentralisierten Management-Overlays anstelle eines einzelnen Servers (siehe Abbildung 1.2). Beim

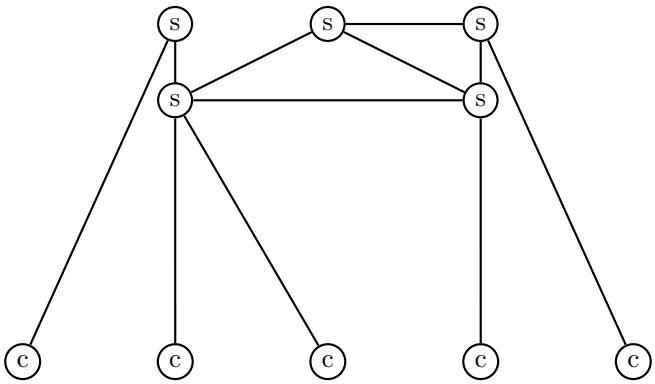


Abbildung 1.2. Client-Server-Overlay-Infrastruktur

Übergang von einer zentralisierten Client-Server-Infrastruktur hin zu einer dezentralisierten Peer-to-Peer-Infrastruktur wird das Management der gesamten Umgebung von einem dezentralisierten Peer-to-Peer-Overlay übernommen. Bei diesem Ansatz übernimmt jeder Knoten des Overlays die Verantwortung für eine ihm zugeteilte virtuelle Region. Somit wird durch die Aufteilung der Verantwortungsbereiche auf die einzelnen Knoten des Overlays eine Lastverteilung erreicht. Gleichzeitig entsteht aber ein Mehraufwand für die Konstruktion bzw. Erhaltung eines solchen Overlays. Auch das clientseitige Erfragen von Informationen, die außerhalb der eigenen virtuellen Region liegen, kann nicht so effizient bewerkstelligt werden, wie es bei einem Server der Fall ist. In dieser Arbeit werden die entsprechenden Vor- und Nachteile eines solchen Infrastrukturwechsels abgewogen. Außerdem wird gezeigt, dass der Einsatz von Overlays, die nach einem Zufallsprinzip aufgebaut sind, die gewünschte Lastverteilung bei geringbleibenden Konstruktionskosten mit sich bringt.

Der zweite Verbesserungsansatz liegt in der Übertragung von einigen Verwaltungsaufgaben, die bis jetzt beim Server lagen, an die Clients. So könnten

die Clients beispielsweise direkt in einer Peer-to-Peer-Manier miteinander kommunizieren, ohne den Server in Anspruch zu nehmen (siehe Abbildung 1.3). Bei diesem Ansatz müssen die Clients sich selbst um eine effiziente Gruppenkommunikation durch den Aufbau einer skalierbaren und effizienten Multicast-Infrastruktur kümmern. Eine weitere Verwaltungsaufgabe, die von den Clients teilweise übernommen werden kann, betrifft das Konsistenz-Handling. Das clientbasierte Konsistenz-Handling bedeutet, dass die Clients selbst für die Wahrung der Konsistenz eines gemeinsamen Zustandes bzw. der gemeinsam genutzten Objekte innerhalb einer virtuellen Region zuständig sein sollten.

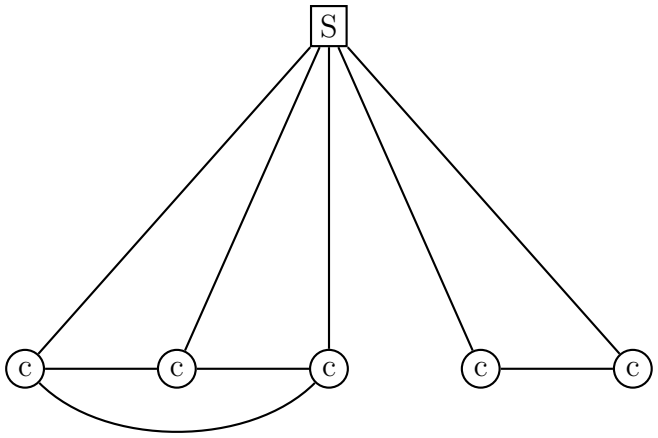


Abbildung 1.3. Client-Overlay-Server-Infrastruktur

Bekanntermaßen sind Konsistenz und Interaktivität zwei gegensätzliche Kriterien, die häufig nicht im gleichen Maß erfüllt werden können. Die Tatsache, dass Nutzer vermehrt Inhalte selbst generieren und in der entsprechenden Umgebung für andere Nutzer sichtbar platzieren können, steigert die Attraktivität der virtuellen Umgebungen weiterhin, erhöht aber gleichzeitig die Ansprüche an die Basis-Infrastruktur und das Konsistenz-Handling. Somit müssen sich die Entwickler von solchen Umgebungen bzw. die Forscher auf diesem Gebiet nicht nur mit der Frage nach einer skalierbaren Infrastruktur auseinandersetzen, sondern sich auch mit der Bestimmung eines für das jeweilige Anwendungsszenario optimalen Trade-offs zwischen Konsistenz und Interaktivität befassen. Aus diesem Grund wird in dieser Arbeit ein neues Konsistenzmodell definiert, welches die Einhaltung der strengen Interaktivitätsanforderungen erzwingt und die Konsistenz mit einer bestimmten Wahrscheinlichkeit garantieren kann.

In dieser Dissertation wird die Verbesserung der Skalierbarkeit und der Lastverteilung in virtuellen Umgebungen sowohl durch die Verwendung eines strukturierten Management-Overlays anstelle eines Servers als auch durch die clientgeregelte Gruppenkommunikation und das clientbasierte Konsistenz-Handling angestrebt. Abbildung 1.4 stellt vereinfacht den damit verbundenen Infrastrukturwandel dar.

Wie in der Abbildung 1.5 angedeutet, werden im Laufe dieser Arbeit der Wechsel der Management-Infrastruktur von einem Server zu einem Peer-to-Peer-System

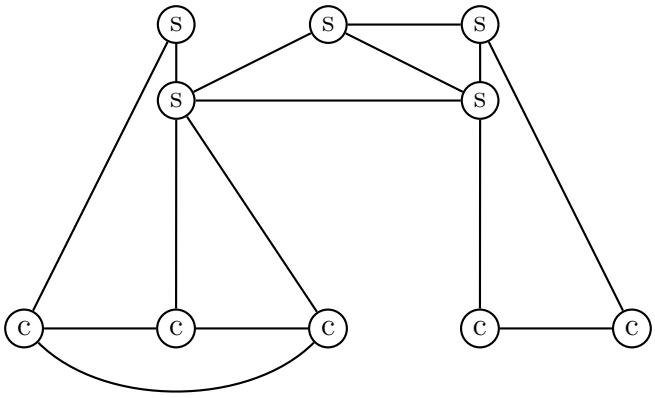


Abbildung 1.4. Client-Overlay-Server-Overlay-Infrastruktur

und die Übertragung von Verwaltungsaufgaben an die Clients unabhängig voneinander untersucht. Aber auch die clientbasierte Multicast-Kommunikation und das

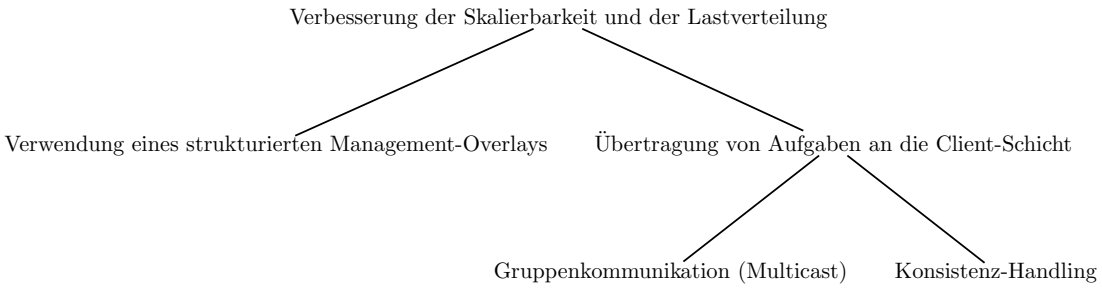


Abbildung 1.5. Verbesserung der Skalierbarkeit und der Lastverteilung

clientbasierte Konsistenz-Handling werden im weiteren Verlauf weitestgehend unabhängig voneinander betrachtet[2].

In Kapitel 2 wird zuerst die Terminologie, die in den virtuellen Umgebungen zum Einsatz kommt, eingeführt. Dann werden die bekanntesten Vertreter der strukturierten Overlay-Netzwerke vorgestellt, die für den Einsatz als Management-Schicht in einer virtuellen Umgebung in Frage kommen könnten. Danach werden die Grundlagen der Multicast-Kommunikation besprochen. Da der Konsistenz in virtuellen Umgebungen in dieser Arbeit eine zentrale Bedeutung beigemessen wird, werden in Kapitel 2 auch unterschiedliche Konsistenzaspekte und -verfahren ausführlich behandelt.

Um die Machbarkeit der in dieser Arbeit beschriebenen Ansätze und Konzepte im praktischen Einsatz evaluieren zu können, wurden zu Demonstrations- und Untersuchungszwecken ein DVE-Framework und basierend darauf ein Multiplayer-Online-Spiel entwickelt. Diese werden in Kapitel 3 beschrieben.

[2] In weiteren Arbeiten könnte geklärt werden, ob und inwiefern die Wahl einer (Multicast-)Infrastruktur auf das Konsistenz-Handling Einfluss nehmen kann.

Im weiteren Verlauf der Arbeit werden in Kapitel 4 zwei Overlay-Netzwerke bezüglich der für DVEs relevanten Eigenschaften miteinander verglichen und die Abbildung von virtuellen Regionen auf die verantwortlichen Konten im entsprechenden Overlay-Netzwerk diskutiert.

In Kapitel 5 werden zwei Multicast-Ansätze vorgestellt, die im Laufe dieser Arbeit, unter Berücksichtigung der besonderen Anforderungen von virtuellen Umgebungen, entwickelt wurden. Darüber hinaus werden die beiden Ansätze den existierenden Multicast-Infrastrukturen gegenübergestellt.

In Kapitel 6 wird ein neues Konsistenzmodell definiert, welches die Einhaltung der strengen Interaktivitätsanforderungen erzwingt und die Konsistenz in einer virtuellen Umgebung mit einer bestimmten Wahrscheinlichkeit garantieren kann.

Kapitel 7 fasst die wesentlichen Inhalte dieser Arbeit noch einmal zusammen und stellt den wissenschaftlichen Beitrag heraus. Ein Ausblick auf weitere wichtige Aspekte, die in dieser Arbeit nicht ausreichend untersucht wurden, jedoch einer tiefergehenden Betrachtung bedürfen, wird in Kapitel 8 gegeben.

Kapitel A (im Anhang) beschreibt die Fuzzy-Logik-basierte Berechnung eines Trade-off-Parameters, welcher zur Anpassung von Konsistenzalgorithmen verwendet wird.

2

Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die für diese Arbeit benötigten Grundlagen vermittelt sowie eine Übersicht über verwandte Arbeiten gegeben. Begonnen wird mit einer Einführung in die Terminologie einer virtuellen Umgebung und einer Vorstellung bekannter und forschungsrelevanter Umgebungen (Abschnitt 2.1).

Die in dieser Arbeit betrachtete Verbesserung der Skalierbarkeit und der Lastverteilung soll unter anderem durch die Verwendung eines strukturierten Peer-to-Peer-Overlays anstelle eines Servers für Management-Aufgaben erreicht werden. Die dafür erforderlichen Grundlagen und Begrifflichkeiten sowie eine Übersicht der existierenden Overlay-Ansätze und der entsprechenden Simulationsumgebungen werden in Abschnitt 2.2 vorgestellt.

Ein anderer Verbesserungsansatz liegt in der Übertragung einiger Verwaltungsaufgaben an die Clients. Bei diesem Ansatz müssen die Clients sich selbst unter anderem um eine effiziente Gruppenkommunikation durch den Aufbau einer skalierbaren und effizienten Multicast-Infrastruktur kümmern. Die Grundlagen der Multicast-Kommunikation werden in Abschnitt 2.3 vermittelt.

Der Schwerpunkt dieser Arbeit liegt auf der Zusicherung der Konsistenz in virtuellen Umgebungen. Deshalb müssen die entsprechenden Grundlagen ebenfalls erarbeitet werden. In Abschnitt 2.4 wird der Konsistenzbegriff zuerst allgemein definiert und dann auf die virtuellen Umgebungen übertragen. Im Laufe dieses Abschnittes werden unterschiedliche Konsistenz-Facetten und Verfahren besprochen.

2.1 Verteilte virtuelle Umgebungen

Virtuelle Umgebungen vermitteln ihren Nutzern das Gefühl, zusammen mit anderen Nutzern in einem Raum zu sein, und kombinieren somit die wichtigen Eigenschaften von physischer und sozialer Präsenz. Die heutigen virtuellen Umgebungen und Spiele mit ihren graphisch ansprechenden, meist dreidimensionalen Zugängen bieten eine Vielzahl an packenden Handlungen und Szenarien und ermöglichen es Tausenden von Spielern zu interagieren und zu kommunizieren, zusammen bestimmte Aufgaben zu bewältigen oder gegen andere Spieler bei Wettbewerben anzutreten. Die Mimik oder die Gestik bzw. die Bewegungsabläufe der einzelnen

Avatare sind dabei von großer Bedeutung. Die Konsistenz der entsprechenden Avatar-Zustände (Position, etc.) spielt deshalb eine wichtige Rolle. Da die Nutzer einer solchen Umgebung ihre Entscheidungen oftmals in Echtzeit treffen müssen, stellen solche Umgebungen ebenfalls sehr hohe Anforderungen an die Interaktivität.

Grundsätzlich wird bei virtuellen Umgebungen zwischen Online-Spielen und Online-Welten unterschieden. Online-Spiele – im weiteren Verlauf einfach nur Spiele oder MMOGs genannt – verfügen über eine vordefinierte Handlung und ein definiertes Ziel, welches von Spielern nach Möglichkeit erreicht werden soll. Dabei können die Spieler ihre Eigenschaften durch das Lösen von Aufgaben verbessern und dadurch bestimmte Vorteile im Spiel erreichen.

Die (Mit-)Gestaltungsmöglichkeit zählt zu den größten Unterscheidungsmerkmalen einer virtuellen Welt im Vergleich zu einem Spiel. Außerdem ist es nicht zwingend erforderlich Aufgaben zu lösen, um bestimmte Tätigkeiten verrichten zu können oder in bestimmte virtuelle Regionen eintreten zu dürfen.

Da verteilte virtuelle Welten und Spiele auf einer ähnlichen Infrastruktur aufbauen und vergleichbare Benutzerzahlen aufweisen, wird im weiteren Verlauf dieser Arbeit auf die explizite Differenzierung zwischen virtuellen Welten und Spielen verzichtet. Diese werden unter dem Oberbegriff virtuelle Umgebungen bzw. DVEs zusammengefasst.

Bevor im weiteren Verlauf des Abschnitts einige virtuelle Umgebungen vorgestellt werden, wird zunächst stichwortartig die Terminologie eingeführt, die im Laufe der gesamten Arbeit verwendet wird:

- Eine *virtuelle Umgebung* kann vereinfacht als eine Ansammlung zentralisiert (Server) oder verteilt (Overlay) verwalteter *Objekte* und der partizipierenden *Avatare* angesehen werden.
- *Objekte* sind stationäre oder bewegliche Gegenstände (Häuser, Bäume, Fahrzeuge, Power-Ups, etc.) in einer virtuellen Umgebung. Es wird zwischen *statischen* und *dynamischen* Objekten unterschieden. Die statischen Objekte sind nicht veränderbar und können direkt bei Erwerb des Spiels oder der Umgebung auf einem Medium komplett ausgeliefert werden. Die Eigenschaften dynamischer Objekte können sich ändern oder von den Nutzern geändert werden. Deshalb müssen die aktuellen Informationen über diese Objekte immer bei Bedarf nachgeladen werden. Die Objektänderungen müssen an alle Teilnehmer in der entsprechenden virtuellen Region propagiert werden.
- Ein *Non-Player Character (NPC)* ist ein Computer-gesteuerter Objekt, dessen Bewegungen nach einem bestimmten Algorithmus ablaufen und dessen Verhaltensmuster programmatisch beschrieben und deshalb deterministisch sind. Aus diesem Grund kann das Verhalten eines NPC meistens vorhergesagt werden. Die Zustandsaktualisierungen eines NPC müssen ebenfalls an alle betroffenen Teilnehmer propagiert werden.
- Ein *Avatar* ist ein graphischer Stellvertreter einer realen Person in einer virtuellen Umgebung, mit dem sich diese Person durch die virtuelle Umgebung bewegen oder mit anderen Nutzern interagieren kann. Das menschliche Verhalten

ist nicht deterministisch, deshalb können die Bewegungen eines Avatars nicht immer genau vorhergesagt werden. Ein Avatar kann über bestimmte Eigenschaften wie zum Beispiel seine Position verfügen, die für die anderen Avatare sichtbar bzw. zugreifbar sind und deshalb konsistent gehalten werden müssen.

- Ein *Client* steht im Kontext dieser Arbeit stellvertretend sowohl für einen Nutzer, der mit seinem Avatar in einer virtuellen Umgebung mit anderen Nutzern interagiert, als auch für den Rechner eines Nutzers, der als Knoten in dem entsprechenden Netzwerk angesehen werden kann.
- Der *Zustand* eines virtuellen Objektes (Avatars, Gegenstandes) ist durch seine Attributwerte (Farbe, Position, etc.) zu einem bestimmten Zeitpunkt definiert [Snowdon und Munro, 2001]. Informationen, die zur Veränderung des Zustandes dieses Objektes führen, sowie die Veränderungen selbst sollten für alle betroffenen Teilnehmer sichtbar und nachvollziehbar sein.
- Eine *Interaktion* ist laut [Margery et al., 1999] eine konkurrierende Manipulation von Objekt- bzw. Avatar-Zuständen durch mindestens zwei Nutzer.
- Aus Plausibilitätsgründen wird im Laufe dieser Arbeit von der Aufteilung einer virtuellen Umgebung in mehrere disjunkte Partitionen *(Areas of Interest – AoIs)*, die separat verwaltet werden können, ausgegangen[1].
- Unter *AoI-Management* versteht man die gezielte Verteilung von Aktualisierungen bzw. relevanten Informationen nur an die Nutzer, die an diesen Informationen interessiert sind und sich in einer bestimmten virtuellen Region – AoI – befinden.
- Clients organisieren sich interessenbasiert in *AoI-Gruppen*, um eine gezielte Verteilung von Aktualisierungen bzw. relevanten Informationen an die Gruppenmitglieder zu ermöglichen.
- Um die Verfügbarkeit von Gegenständen in der virtuellen Umgebung zu steigern, erhält jeder Nutzer nach der Anmeldung *Kopien (Replikate)* der in seiner AoI befindlichen Objekte. Clients können auf diese Kopien unter Berücksichtigung der Konsistenzaspekte zugreifen.
- Beim *Konsistenz-Handling* geht es darum, einen konsistenten Zustand aller Objekte einer AoI zu wahren.

Die ersten Computer-Spiele – Vorreiter der heutigen virtuellen Umgebungen –, die über ein Netzwerk mit mehreren Personen gespielt werden konnten, entstanden Ende der 70er Jahre[2]. Anfangs wurden diese Spiele nur in geschlossenen Netzwerken mit wenigen Spielern gespielt. Mit der Verbreitung des Internets fanden diese Spiele immer mehr Anhänger und schafften es, über die Grenzen geschlossener Netzwerke herauszukommen. Die bekanntesten Vertreter der aktuellen virtuellen Umgebungen werden im Folgenden kurz vorgestellt.

SecondLife[3] (Abbildung 2.1 (Quelle: `http://second-life.softonic.de`)) ist eine beliebte dreidimensionale virtuelle Umgebung, welche den Nutzern die Möglichkeit

[1] Die Aufteilung von virtuellen Umgebungen in nicht-disjunkte AoIs wird u.a. in [Esch et al., 2009a] behandelt.

[2] `http://www.livinginternet.com/d/di_major.htm`

[3] `http://www.secondlife.com`

einräumt ihre Umgebung mit zu gestallten. Wie in solchen Umgebungen üblich, interagieren und kommunizieren die Nutzer untereinander mithilfe ihrer Avatare. Diese serverbasierte virtuelle Umgebung weist bereits bei vergleichbar geringen Nutzerzahlen (ca. 50) Skalierbarkeitsprobleme auf.

Abbildung 2.1. Screenshot: SecondLife

In Analogie zu SecondLife stellt *Active Worlds*[4] eine dreidimensionale virtuelle Umgebung dar. Auch hier kommen Avatare zur Nutzerinteraktion zum Einsatz. Anderes als in SecondLife, können Nutzer in Active Words eigene Welten erschaffen, die nur für ihre Nutzergruppe zugängig sind.

Diese Möglichkeit bietet auch das heute bekannteste Online-Rollenspiel *World of Warcraft*[5]. Diese abgeschlossenen Welten in World of Warcraft werden jedoch im Gegensatz zu Active Worlds nicht von den Nutzern generiert, sondern von den Providern zur Verfügung gestellt. World of Warcraft ist momentan die virtuelle Umgebung mit den höchsten Benutzerzahlen. Nach Informationen vom Dezember 2008 zählt diese Umgebung bereits mehr als 11,5 Millionen Abonnenten[6]. Da World of Warcraft jedoch auf einer zentralisierten Infrastruktur basiert, ist diese Umgebung auch massiven Skalierbarkeitsproblemen ausgesetzt, die mit bestimmten Einschränkungen für die Benutzer verbunden sind (z.B. lange Wartezeiten oder verzögerte Systemrückmeldungen). Nach inoffiziellen Angaben können gleichzeitig ca. 2300 Nutzer auf einem Server spielen. Die Grenze für die Anzahl der Nutzer, die sich gleichzeitig an einem virtuellen Ort befinden können, liegt bei ca. 100.

In *The Lord of the Rings online*[7] (*LotRo*) (Abbildung 2.2 (Quelle: `http://www.lord-of-the-rings-online.de`)) basieren die Handlungen, die Inhalte und die Charaktere

[4] `http://www.activeworlds.com/`
[5] `http://www.worldofwarcraft.com`
[6] `http://eu.blizzard.com/de/press/081223.html`
[7] `http://www.lotro.com/`

auf den Büchern von *John Ronald Reuel Tolkien*. LotRo baut auf einer ähnlichen zentralisierten Infrastruktur wie World of Warcraft auf. Das Spiel zeigt aufgrund sehr ansprechender Inhalte und Handlungen ein hohes Wachstumspotenzial.

Abbildung 2.2. Screenshot: LotRo

Neben den Problemen der Identifikation einer skalierbaren Basis-Infrastruktur bzw. der Bestimmung eines akzeptablen Trade-offs zwischen Konsistenz und Interaktivität werfen virtuelle Umgebungen viele weitere interdisziplinäre Fragestellungen auf, die bis heute noch gar nicht oder nicht befriedigend gelöst sind. Viele Forscher in der ganzen Welt befassen sich mit diesen Problemen und Fragestellungen. Deshalb werden nachfolgend auch einige forschungsrelevante virtuelle Umgebungen vorgestellt.

Croquet [Smith et al., 2003] ist eine weitere dreidimensionale virtuelle Umgebung, deren Nutzer miteinander über ihre Avatare interagieren. Croquet basiert auf einer Peer-to-Peer-Infrastruktur und weist deshalb eine bessere Skalierbarkeit als die serverbasierten Umgebungen auf.

HYMS [Kim et al., 2004] ist eine hybride Architektur für MMOGs, in der bestimmte Server-Funktionalitäten an ausgewählte Clients übertragen werden, um die Serverlast zu minimieren. Diese ausgewählten Clients sind in einem Peer-to-Peer-Netzwerk organisiert.

SimMud [Knutsson et al., 2004] ist ein MMOG, das auf dem Peer-to-Peer-Overlay *Pastry* [Rowstron und Druschel, 2001a] aufbaut. Der Aspekt der Selbstorganisation findet hier bei der Bildung von Spielergruppen seine Anwendung. Diese Gruppen sind dann für die Verteilung des Spielzustandes sowie für die Konsistenzhaltung von gemeinsam genutzten Objekten zuständig. Der Spielzustand wird mittels der Anwendungsschicht-Multicast-Infrastruktur *Scribe* [Castro et al., 2002] unter allen Clients verbreitet.

Mediator [Fan et al., 2007] ist ein Design-Framework für die Entwicklung von Peer-to-Peer-basierten MMOGs und DVEs. Das Framework folgt einem hybriden Ansatz und unterteilt Netzwerkkomponenten in die so genannten *Super-* und *Common-Peers*. Um das Skalierbarkeitsproblem zu umgehen, werden in Mediator Server-Funktionalitäten an die Peers übertragen. Neben anderen Aspekten widmen sich die Mediator-Entwickler hauptsächlich dem Problem der effizienten Suche in Peer-to-Peer-Architekturen. Dabei stützen sie sich auf *FreePastry*[8] – eine Open-Source-Implementierung des Pastry-Overlay-Netzwerkes.

In [Yu und Vuong, 2005] ist eine zweischichtige Peer-to-Peer-Infrastruktur, die als Basis für dezentralisierte virtuelle Umgebungen dienen soll, beschrieben. Die erste Schicht wird vom strukturierten Pastry-Overlay gebildet und übernimmt Verwaltungsaufgaben. Die Benutzerinteraktion findet in einer Peer-to-Peer-Manier in der zweiten Schicht statt. Basierend auf dieser zweischichtigen Infrastruktur haben die Entwickler eine prototypische Spielanwendung implementiert.

Das MMOG *Time Prisoners* [Rhalibi und Merabti, 2005] baut auf dem JXTA-Peer-to-Peer-Overlay[9] auf. Das Besondere bei diesem Ansatz ist, dass es zu Beginn auf einer reinen Client-Server-Infrastruktur basiert. Mit einer steigenden Benutzeranzahl vollzieht diese Infrastruktur einen Wandel zu der obengenannten Peer-to-Peer-Architektur.

Weitere DVEs, die im Rahmen des Promotionsvorhabens untersucht wurden, sind *Myriad* [Schaeffer et al., 2005], *CyberWalk* [Chim et al., 2003] und *Solipsis* [Keller und Simon, 2003].

Im weiteren Verlauf des Kapitels werden die benötigten Infrastruktur- und Konsistenzgrundlagen vermittelt.

2.2 Overlay-Netzwerke und DVEs

Um den großen Nutzerzahlen gerecht zu werden, stellen Spielentwickler meistens mehrere autonome Spiel-Server zur Verfügung. Da bei diesem serverbasierten Ansatz jeder Server eine eigene Welt- oder Spielinstanz verwaltet, ist die Kommunikation bzw. Interaktion von Spielern, die auf unterschiedlichen Servern eingeloggt sind, stark eingeschränkt[10]. Aufgrund einer ausbalancierten Last- und Zuständigkeitsverteilung kann ein Peer-to-Peer-Overlay eine einzige virtuelle Umgebung aufspannen. Da sich nun alle Nutzer in der gleichen Umgebung befinden, entstehen keine Kommunikations- bzw. Interaktionseinschränkungen mehr. Diese gegensätzlichen Eigenschaften sind in der Abbildung 2.3 dargestellt.

In einem Peer-to-Peer-Overlay übernimmt jeder Knoten die Verantwortung für eine ihm zugeteilte virtuelle Region (AoI). Eine Aufteilung der Welt in verschiedene AoIs, die dezentral verwaltet werden, macht es möglich, dass die Welt in kleinere Partitionen geteilt wird, bei denen eine Kontrolleinheit für die Verwaltung einer oder mehrerer AoIs zuständig ist. Alle AoIs zusammengefasst bilden somit wieder

[8] `http://freepastry.org/FreePastry/`
[9] `https://jxta.dev.java.net/`
[10] Es gibt Ansätze, die Chat-Anwendung über die Servergrenzen hinweg anbieten.

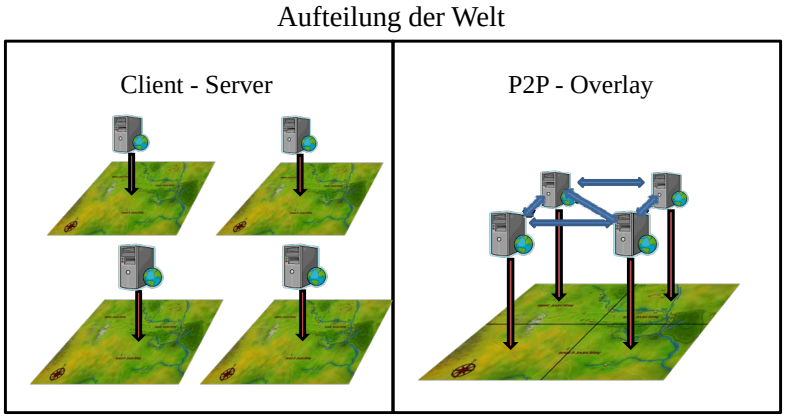


Abbildung 2.3. Aufteilung der Zuständigkeiten

eine komplette Welt. Der Vorteil dieses Vorgehens besteht darin, dass die Teilbereiche in der Größe variabel sind und bei Bedarf die Welt auf beliebig viele Kontrolleinheiten unterteilt werden kann. Je höher der Grad der Aufteilung ist, desto geringer fällt die Größe der einzelnen AoIs aus und desto weniger Verwaltungsaufwand muss eine Kontrolleinheit innerhalb einer AoI betreiben. Damit wird die gewünschte Lastverteilung durch die Aufteilung der Verantwortungsbereiche auf die Knoten des Overlays erreicht.

Da bereits viele ausgereifte Overlay-Ansätze existieren, die in unterschiedlichen Bereichen erfolgreich eingesetzt werden, wurde in dieser Arbeit der Fokus auf Identifikation eines für DVEs am besten geeigneten Ansatzes gelegt. Die bekanntesten Vertreter der strukturierten Overlays werden in Abschnitt 2.2.1 vorgestellt. Da die Identifikation eines geeigneten Overlay-Netzwerkes durch Simulation seines Verhaltens erfolgen sollte, werden einige Simulationsumgebungen, die im Laufe dieser Arbeit untersucht wurden, in Abschnitt 2.2.2 betrachtet.

2.2.1 Overlay-Netzwerke

Peer-to-Peer-Netzwerke unterscheiden sich grundsätzlich im Grad der Dezentralisierung. In diesem Zusammenhang spricht man von zentralisierten, dezentralisierten und hierarchischen Ansätzen. Bei einem zentralisierten Ansatz (z.B. Napster[11]) ist eine Kontrollinstanz für die Bewältigung der Verwaltungsaufgaben zuständig. Lediglich die Kommunikation unter den Peers findet in der Peer-to-Peer-Manier statt. Im Gegensatz dazu sind bei einem dezentralisierten Ansatz (z.B. Gnutella [Jovanovic et al., 2001]) die Peers selbst für die Netzwerkverwaltung zuständig. Bei einem hierarchischen Ansatz (z.B. JXTA) sind die Peers in mehrere Hierarchiestufen aufgeteilt. Je nach Hierarchiestufe ist ein Peer in die Netzwerkverwaltung eingebunden oder nicht.

Der Einsatz von Peer-to-Peer-Netzwerken findet in den letzten Jahren verstärkt Anwendung. Wurden diese Systeme anfangs als Basis für File-Sharing- bzw.

[11] `www.napster.com`

Gruppen-Chat-Anwendungen verwendet, so hat sich die Spannweite der Anwendungen in den letzten Jahren deutlich vergrößert. Einige Forschungsgruppen an renommierten Instituten befassen sich mit dem Entwurf und der Entwicklung von Overlay-Netzwerken, die eine zusätzliche Publish/Lookup-Funktionalität bereitstellen, die von unterschiedlichen Peer-to-Peer-Anwendungen wie zum Beispiel DVEs genutzt werden können. Zu solchen Overlay-Netzwerken zählen zum Beispiel *Chord* [Stoica et al., 2001], *Pastry* [Rowstron und Druschel, 2001a], *CAN* [Ratnasamy et al., 2001a], *Kademlia* [Maymounkov und Mazieres, 2002], *Tapestry* [Zhao et al., 2004] und *P-Grid* [Aberer, 2001]. Sie finden in vielen unterschiedlichen Bereichen – von verteilten Speichersystemen (*OceanStore* [Kubiatowicz et al., 2000]) bis hin zu IP-Fernsehen (Joost[12]) – ihren Einsatz. In dieser Arbeit werden Overlay-Netzwerke gemäß [Zech, 2007] definiert:

Definition 2.1 (Overlay-Netzwerk). *Netzwerke, die ein physikalisches Netzwerk, zum Beispiel das Internet, als zugrunde liegendes Kommunikationsmedium verwenden und darauf basierend auf der Anwendungsebene eine eigene logische Verbindungsstruktur aufbauen, nennt man* ***Overlay-Netzwerke****. Die darunterliegende Netzwerkschicht wird in diesem Zusammenhang als* ***Underlay*** *bezeichnet.*

Grundsätzlich wird zwischen unstrukturierten und strukturierten Overlays unterschieden. Unstrukturierte Peer-To-Peer-Overlays bestehen aus einer Menge von Netzwerkknoten und einer Menge von Verbindungen, die diese Knoten zufällig – das heißt ohne die Anwendung eines bestimmten Algorithmus – verbinden. In das Beitreten und das Verlassen eines Knotens sind lediglich seine direkten Nachbarn involviert. Somit haben diese Operationen nur lokale Auswirkungen und betreffen das gesamte Netz nicht. Diese zufällige Struktur skaliert sehr gut bezüglich der Teilnehmeranzahl, d.h. beliebig viele Knoten können ohne großen Kommunikations- und Rechenaufwand aufgenommen werden. Die Suche in solchen Netzwerken gestaltet sich jedoch äußerst ineffizient, da hier nicht aus der Objekt-ID[13] auf den Knoten geschlossen werden kann, der das Objekt besitzt. Es gibt somit keine Möglichkeit, das Objekt „schnell“ zu finden. Suchanfragen werden entweder mittels *Flooding*- oder *Random-Walk*-Verfahren umgesetzt. Beim Flooding wird eine Suchanfrage an die bekannten Nachbarn solange weitergeleitet, bis die maximale Hop-Anzahl erreicht ist. Leider kann es bei einer zu kleinen Hop-Grenze dazu kommen, dass ein Objekt nicht gefunden wird, obwohl es im Netzwerk vorhanden ist. Beim Random-Walk-Verfahren werden die Anfragen nur an einen zufälligen Nachbarn weitergeleitet. Damit wird zwar die Netzlast reduziert, die Zeitdauer der Suche erhöht sich jedoch enorm im Vergleich zu Flooding.

Viele Forschungsgruppen beschäftigen sich gegenwärtig hauptsächlich nur mit der Entwicklung strukturierter Overlay-Netzwerke. Die Motivation dahinter liegt in der Bestrebung, die Suche in einem Overlay-Netzwerk effizient zu gestalten. Im Gegensatz zu den unstrukturierten Overlays werden die strukturierten Over-

[12] http://www.joost.com/

[13] Bei den meisten Overlays wird die Objekt-ID durch die Anwendung einer Hash-Funktion auf den Objektnamen ermittelt. P-Grid verwendet hierfür ein alternatives Mapping-Verfahren, welches im Gegensatz zu Hash-Funktionen ähnliche Namen auf ähnliche Kennungen abbildet [Aberer, 2001].

lays anhand eines bestimmten Algorithmus aufgebaut. Die meisten strukturierten Overlay-Netzwerke basieren auf dem Prinzip der *Distributed Hash Tables* (DHT). Nach dem Beitritt erhält jeder Knoten im Overlay eine eindeutige ID, die seine virtuelle Position im Overlay festlegt. Objekte bzw. Daten, die in einem Peer-to-Peer-Netzwerk öffentlich zugänglich gemacht werden sollen, müssen zuerst im Overlay *veröffentlicht* (Publish-Operation) werden. Dabei wird das Objekt anhand seiner ID dem zuständigen Knoten im Overlay zugeordnet. Erst nach dem Veröffentlichen können Objekte bzw. Daten effizient wieder *aufgefunden* (Lookup-Operation) werden, indem die Suchanfrage anhand der Objekt-ID in Richtung des zuständigen Knotens geroutet wird. DHTs garantieren für die angebotenen Publish- bzw. Lookup-Operationen eine obere Komplexitätsschranke von $O(log(n))$. Somit steht der schlechten Skalierbarkeit der strukturierten Overlays eine gute Suchperformance gegenüber.

Die Untersuchung von Overlay-Netzwerken fand in dieser Arbeit zum Zwecke der Identifikation einer Infrastruktur statt, die für den Einsatz als Management-Schicht, zu deren Aufgaben u.a. die Verwaltung bzw. Verteilung von Weltdaten zählt, am besten geeignet ist. Die Kriterien dafür sind zum Beispiel die Konstruktionskosten oder die Suchperformance. Im Folgenden werden die bekanntesten strukturierten Overlay-Netzwerke vorgestellt.

Chord

Chord [Stoica et al., 2001] ist ein skalierbares Overlay-Netzwerk für verteilte Internet-Anwendungen. Chord verwendet DHTs für die Abbildung von Objekten auf Overlay-Knoten. Diese Knoten werden anhand ihrer IDs in einer Ringstruktur angeordnet. Ein Objekt wird auf den Knoten abgebildet, dessen ID größer oder gleich der Objekt-ID – das heißt dem Hashwert des Objekts – ist, und der sich am nächsten (numerisch) an der Objekt-ID befindet. Jeder Knoten besitzt somit einen eindeutigen Zuständigkeitsbereich.

Neben den Verbindungen zu seinem direkten Vorgänger und Nachfolger im Ring verwaltet ein Chord-Knoten eine Liste von Nachfolgern mit einer beliebigen, aber festen Größe. Die Nachfolger-Verbindungen werden für die einfache, d.h. nicht effiziente Suche, verwendet, wobei eine Suchanfrage einfach entlang des Rings von Knoten zu Knoten über diese Verbindungen weitergereicht wird. Die Nachfolger-Liste dient hauptsächlich der Fehlertoleranz, wird aber auch bei der „normalen“ Suche eingesetzt. Wenn der direkte Nachfolger eines Knotens auf die Anfragen nicht reagiert, wird an seiner Stelle der nächste in die Nachfolger-Liste eingetragene Knoten als direkter Nachfolger verwendet.

Die Routing-Tabellen, die in Chord für die effiziente Suche verwendet werden, bezeichnet man als *Finger Tables*. Durch die Verwendung der entsprechenden Tabellen-Einträge wird eine logarithmische Nachrichtenkomplexität garantiert. Schlägt die effiziente Suche fehl, so kann das Objekt immer noch mit den Mitteln der einfachen Suche aufgefunden werden.

Ein Knoten wird anhand seiner ID im Ring identifiziert. Beim Eintritt eines neuen Knotens in das Overlay wird dieser entsprechend seiner ID in den Ring ein-

gefügt. Die benachbarten Knoten passen dabei ihre Vorgänger- und Nachfolger-Verbindungen entsprechend der neuen Situation an. Der neue Knoten bestimmt die Einträge seiner Finger Table mithilfe eines bekannten Knotens aus dem Netzwerk. Die entsprechenden Zuständigkeitsbereiche verändern sich mit Eintritt bzw. Austritt von Knoten.

In der ursprünglichen Version berücksichtigt Chord bei der Konstruktion des Overlays die Topologie der darunterliegenden Netzwerkschicht nicht [Zech, 2007]. Es gibt jedoch Ansätze [Dabek et al., 2004], die Chord dahingehend erweitern.

Chord ist nicht zuletzt aufgrund seiner vergleichsweise einfachen, aber sehr effizienten Netzwerkstruktur eines der bekanntesten Peer-To-Peer-Netzwerke wissenschaftlicher Herkunft [Mahlmann und Schindelhauer, 2007].

Tapestry

Wie die meisten anderen strukturierten Overlay-Netzwerke basiert *Tapestry* [Zhao et al., 2004] auf dem Prinzip der DHTs. Dabei werden Netzwerkknoten genauso wie Objekte auf eindeutige Schlüssel aus demselben Schlüsselraum abgebildet. Im Gegensatz zu Chord wird in Tapestry die Underlay-Topologie bei der Konstruktion des Overlays zum Zwecke der Latenz-Minimierung berücksichtigt.

Die Suche in Tapestry basiert auf dem in [Plaxton et al., 1999] vorgestellten Routing-Verfahren. Die Weiterleitung einer Suchanfrage (Lookup) geschieht anhand der Einträge aus der Routing-Tabelle, die bei Tapestry *Neighbor-Map* genannt wird. Die Routing-Tabelle eines Tapestry-Knotens (n_a) enthält mehrere Stufen (engl. Level). Dabei repräsentieren die Spalten der Neighbor-Map die Stufen und die Zeilen die Ziffern der Kennungen in der Position der jeweiligen Stufe + 1. Das heißt, in einer Stufe l speichert n_a Referenzen zu den Knoten, deren ID das *längste gemeinsame Präfix* der Länge l mit der ID von n_a hat. Hat n_a zum Beispiel die $ID = 1234$ und n_b die $ID = 1211$, dann speichert n_a die Referenz zu n_b an der 1. Position der 2. Stufe. Die Position ergibt sich aus der $l + 1$-ten Ziffer der betrachteten $ID = 1211$. Die Knotenanzahl, die ein Feld der Neighbor-Map fassen kann, ist durch einen Konfigurationsparameter begrenzt. Die Tabelle 2.1 zeigt beispielhaft einen Ausschnitt einer Neighbor-Map des Knotens n_a.

Level / Pos	0	1	2	$\cdots$
0	0123	1023	1201	$\cdots$
1	n_a	1123	1211	$\cdots$
2	2345	n_a	1223	$\cdots$
3	3345	1345	n_a	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Tabelle 2.1. Neighbor-Map des Knotens $n_a = 1234$

Es ist möglich, dass sich mehrere Knoten ein gemeinsames Präfix β mit einem Knoten n_a teilen. Derjenige unter ihnen mit der geringsten physikalischen

Entfernung wird als *Primärnachbar* (engl. Primary Neighbor) bezeichnet. Alle anderen sind dann die *Sekundärnachbarn* (engl. Secondary Neighbors). Publish- und Lookup-Anfragen werden im Tapestry-Overlay vorzugsweise über die Primärnachbarn Ziffer für Ziffer weitergeleitet.

In der Abbildung 2.4 ist ein Beispielszenario dargestellt, in dem der Knoten mit der $ID = 1234$ das Objekt mit der $ID = 3477$ veröffentlicht (gestrichelte Linie). Der Knoten (1234), der das zu veröffentlichende Objekt (3477) besitzt, wird als *Server* bezeichnet. Die Veröffentlichung wird Knoten für Knoten (Ziffer für Ziffer) weitergeleitet: $3{*}{*}{*} \Rightarrow 34{*}{*} \Rightarrow 347{*} \Rightarrow 3477$ (mit $* \in \{x \in \mathbb{N} \mid x \leq 9\}$), bis sie bei einem Knoten ankommt, dessen ID der ID des Objektes numerisch am nächsten ist (*Surrogate Routing*). Der Knoten mit der „nächsten“ $ID = 3478$ verwaltet fortan das Objekt und wird als *Object Root* bezeichnet.

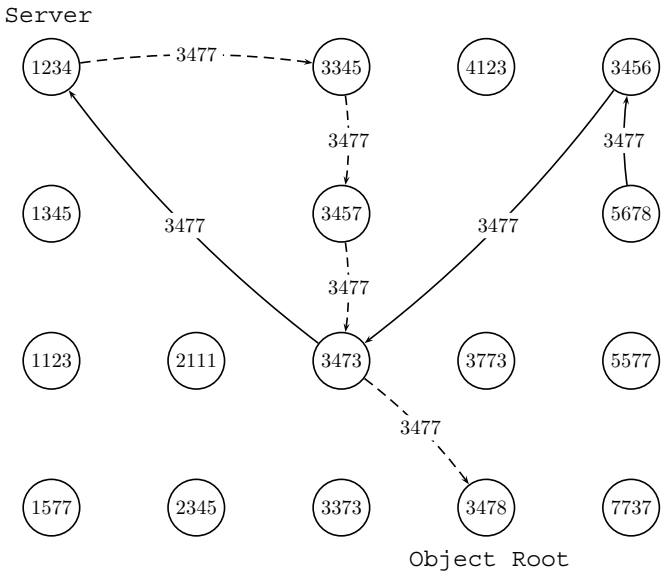


Abbildung 2.4. Routing in Tapestry

Das Tapestry-Overlay unterscheidet sich von anderen Ansätzen darin, dass es eine Art alternatives Routing – *Surrogate Routing* genannt – unterstützt. Diese Art von Routing ist notwendig, falls auf dem Weg, den eine Anfrage nimmt und der durch die Objekt-ID vorgegeben ist, „Löcher“ in der Neighbor-Map vorkommen. Ein solches „Loch“ kommt dann zu Stande, wenn in dem Feld auf der Stufe l und an der Position p kein Eintrag vorhanden ist. In diesem Fall wird die entsprechende Ziffer der Objekt-ID um eins erhöht und es wird versucht, erneut ab der neuen Position zu routen. Dies wird solange wiederholt, bis kein Knoten mehr auf der aktuellen Stufe oder darüberliegenden Stufen vorhanden ist.

Eine weitere Besonderheit bei Tapestry stellen die so genannten *Backpointers* dar. Dabei speichern alle Knoten, die auf dem Pfad der Veröffentlichung liegen, die Referenz zu dem Server, bevor sie die Veröffentlichung (Publish) weiterleiten. Dadurch wird ein Objektzeiger oftmals gefunden, bevor die Anfrage den Objekt-Root-Knoten erreicht. Diese Eigenschaft ist in der Abbildung 2.4 ebenfalls dar-

gestellt. Die Suchanfrage (durchgezogene Linie) des Knotens 5678 findet einen Objektzeiger zum Server bei dem Knoten 3473.

Wie bereits erwähnt, dient Tapestry als Basis für das verteilte Speichersystem OceanStore [Kubiatowicz et al., 2000].

Pastry

Der von [Rowstron und Druschel, 2001b] entwickelte DHT-basierte Such- und Routing-Dienst *Pastry* ist für den Einsatz in sehr großen Netzwerken geeignet. Ähnlich wie bei Tapestry wird in Pastry die Underlay-Topologie bei der Konstruktion des Overlays zum Zwecke der Latenz-Optimierung berücksichtigt.

Für das Routing verwendet Pastry drei Informationsquellen. Die *Routing-Table* in Pastry entspricht der Neighbor-Map des Tapestry-Overlays und basiert auf exakt demselben Verfahren. Eine weitere Menge – *Leaf Set* – ist die Menge der numerisch nächsten Knoten zur eigenen Knoten-ID. Daneben existiert ein so genanntes *Neighborhood-Set*, das die Verbindungen zu den Knoten mit einer geringen topologischen Entfernung verwaltet.

Beim Absetzen bzw. Weiterleiten einer Suchanfrage prüft ein Knoten zuerst, ob die Ziel-ID im Bereich des eigenen *Leaf Set* enthalten ist. Ist dies der Fall, so wird die Anfrage an den Knoten weitergeleitet, dessen ID der Ziel-ID am nächsten (numerisch) ist. Wenn kein passender Knoten im *Leaf Set* existiert, dann wird die Anfrage anhand der Einträge in der Routing-Tabelle weitergeleitet. Das Neighborhood-Set wird nicht für die effiziente Suche verwendet, sondern als eine Fallback-Lösung betrachtet.

Pastry wurde bereits als Grundlage einiger verteilter Anwendungen wie *Past* (verteiltes Speichersystem) [Rowstron und Druschel, 2001c], *Scribe* (verteilte Multicast-Infrastruktur) [Castro et al., 2002] oder *Squirrel* (verteilter Webcache) [Iyer et al., 2002] eingesetzt.

CAN

Ein ebenfalls DHT-basierter Ansatz namens *CAN* (Content Addressable Network) wurde von [Ratnasamy et al., 2001b] vorgestellt. Grundsätzlich kann beim Aufbau von CAN die darunterliegende Topologie berücksichtigt werden. Die Knoten-Lokalität wurde jedoch von den Autoren nicht näher betrachtet.

Das *CAN*-Overlay spannt einen d-dimensionalen kartesischen Koordinatenraum auf, der unter den Overlay-Knoten aufgeteilt wird. Dabei sind die Knoten in einer Art Gitterstruktur auf bestimmte Zonen verteilt. Je nach der Dimension kann eine Zone als ein Punkt auf einer Linie (Dimension = 1), ein Rechteck (Dimension = 2), ein Hexaeder (Dimension = 3), etc. angesehen werden. Ein Knoten speichert dabei die Verbindungen zu den verantwortlichen Knoten in den angrenzenden Zonen. Rand-Knoten im Gitter verwalten Verbindungen zu den Knoten am anderen Rand, so als ob sie direkt benachbart wären.

Die Suchanfragen werden in CAN entlang der orthogonalen Zeiger weitergeleitet. Hierbei wird derjenige Zeiger ausgewählt, dessen euklidischer Abstand zum

Zielpunkt am kleinsten ist. Dieses Verfahren wird so lange wiederholt, bis der Eigentümer der Zone, in der der Zielpunkt liegt, gefunden wird.

Um dem CAN-Netzwerk beizutreten, kontaktiert ein neuer Knoten einen beliebigen Knoten im Overlay. Daraufhin wird der entsprechende Zuständigkeitsbereich auf die beiden Knoten aufgeteilt und die Nachbarschaftsverbindungen werden angepasst.

Eine verteilte File-Sharing-Anwendung [Elmas und Özkasap, 2004] wurde auf Basis von CAN implementiert.

P-Grid-Overlay

Im Gegensatz zu den meisten anderen DHT-basierten strukturierten Peer-to-Peer-Overlays baut *P-Grid* [Aberer, 2001] auf einem Präfixbaum-Konzept (Trie) [Freedman und Vingralek, 2002] auf. Dieses besagt, dass die Position eines Knotens innerhalb des Baumes durch seinen binären Pfad P eindeutig bestimmt ist. Dabei entspricht die Position eines Knotens idealerweise einem Blatt in dem Präfixbaum. Der Zuständigkeitsbereich eines Knotens ist ebenfalls durch den besagten Pfad vorgegeben. Ein P-Grid-Knoten mit dem Pfad 01 hält zum Beispiel alle Referenzen zu den veröffentlichten Objekten, deren Bezeichnung auf einem binären String mit dem Präfix 01 abgebildet wird (Abbildung 2.5). Das heißt, dass jeder P-Grid-Knoten mit einem Pfad P für alle Objekte, deren binäre Repräsentation mit P beginnt, zuständig ist.

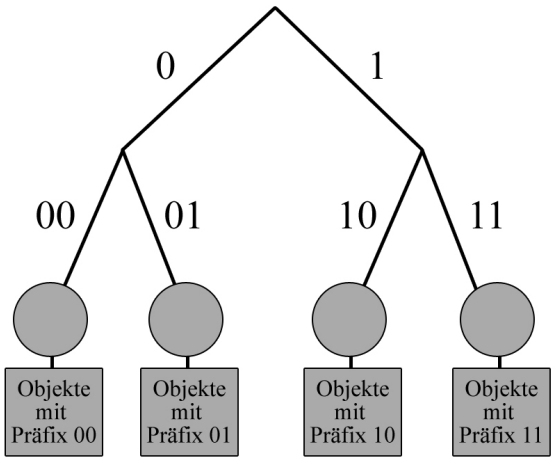


Abbildung 2.5. P-Grid-Baum (Quelle: [Bettinger, 2008])

Um Suchanfragen weiterleiten zu können, verwendet jeder P-Grid-Knoten Routing-Tabellen variabler Größe. Für jeden Substring seines binären Pfades speichert der Knoten Referenzen zu den Knoten, deren binäre Pfade sich in der letzten Stelle des entsprechenden Substrings unterscheiden. 1 ist zum Beispiel der erste Substring des Pfades 101. Folglich speichert der Knoten an der entsprechenden Stelle in der Routing-Tabelle die Referenzen zu den (ihm bekannten) Knoten, deren binärer Pfad mit einer 0 beginnt. Der zweite Substring ist 10, deshalb speichert

der Knoten an der entsprechenden Stelle in der Routing-Tabelle Referenzen zu den Knoten mit Pfad-Präfix 11, usw. (siehe Tabelle 2.2 bzw. Abbildung 2.6). Auf diese Art und Weise speichert ein Knoten Referenzen zu mindestens einem anderen Knoten in den Teilbäumen, die in seinem eigenen Pfad nicht enthalten sind.

Teilbaum	Referenz
0	**0**01
11	**11**1
100	**100**

Tabelle 2.2. Routing-Tabelle von $p = 101$

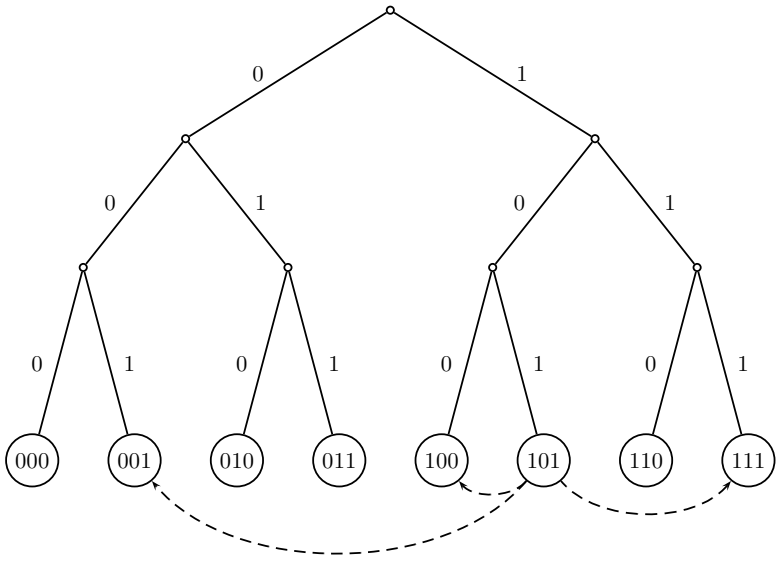


Abbildung 2.6. Zeiger von $p = 101$

Der Aufbau eines P-Grid-Overlays ist ein zufallsgesteuerter Prozess. Das heißt, um dem Netzwerk beizutreten, muss ein neuer Knoten lediglich ein paar Informationen (z.B. Routing-Tabellen-Einträge) mit seinem Kontaktknoten austauschen. Der vollständige Aufbau von Routing-Tabellen etc. geschieht dann zufallsgesteuert im Laufe des Lebenszyklus des entsprechenden Knotens. Während des Aufbaus einer P-Grid-Infrastruktur kann es vorkommen, dass mehrere Knoten dieselbe Position in dem Präfixbaum haben und deshalb denselben Pfad besitzen. Diese Knoten sind dann für den gleichen Objektbestand zuständig. Zur Auflösung von Knoten mit demselben Pfad wird der probabilistische Algorithmus aus [Datta et al., 2003] verwendet.

Einige Peer-to-Peer-Anwendungen, darunter der verteilte Speicher *UniStore* [Karnstedt et al., 2007] und die *PIX-Grid*-Plattform zum Austausch von Fotos [Aberer et al., 2003], wurden bereits auf der Basis des P-Grid-Overlays implementiert.

In der ursprünglichen Version berücksichtigt P-Grid die Knoten-Lokalität nicht.

2.2.2 Simulationsumgebungen

Im vorhergehenden Abschnitt wurden einige bekannte Overlays sowie verteilte Anwendungen, die auf diesen Overlays aufbauen, vorgestellt. Im Laufe dieser Arbeit war es erforderlich zu untersuchen, welches Overlay-Netzwerk sich am besten für den Dienst als Management-Schicht in einer DVE eignet. Um eine objektive Antwort auf diese Frage geben zu können, musste das Verhalten von Overlay-Netzwerken unter Berücksichtigung der Anwendungsanforderungen simuliert und evaluiert werden. Dafür sollte entweder eine geeignete existierende Simulationsumgebung identifiziert werden oder eine neue maßgeschneiderte Simulationsumgebung entwickelt werden.

In diesem Abschnitt werden einige existierende Umgebungen vorgestellt, die zur Simulation von Overlay-Netzwerken eingesetzt werden. Die meisten Simulationsumgebungen stellen auch einige Overlay-Implementierungen zu Demonstrationszwecken bereit und besitzen Schnittstellen zur Implementierung von weiteren Overlays. Einige dieser Umgebungen verfügen auch über eine Visualisierungskomponente, die das Verhalten eines Overlays unter bestimmten Bedingungen graphisch darstellt.

VisPastry[14] ist ein Visualisierungsframework für das Pastry-Overlay-Netzwerk und die darauf basierende Scribe-Multicast-Infrastruktur [Castro et al., 2002]. Mithilfe von VisPastry erhält ein Nutzer einen Einblick in die Konstruktion des Pastry-Overlays sowie in den Aufbau des Multicast-Baums von Scirbe. Da VisPastry speziell für die Visualisierung des Pastry-Overlays entwickelt wurde, werden keine weiteren Overlay-Implementierungen unterstützt.

Chord-Overlay verfügt ebenfalls über eine Simulations- und Visualisierungsumgebung namens *Visualizer*[15]. Visualizer stellt die Informationen von ausgewählten Knoten im Chord-Ring graphisch dar. Auf diese Art und Weise ist es möglich nachzuvollziehen, in welchem Zustand sich der Chord-Ring gegenwärtig befindet und wie sich das Beitreten und das Ausscheiden von Knoten auf den Chord-Ring auswirkt. Visualizer unterstützt keine weiteren Implementierungen und kann nur zur Darstellung des Chord-Overlays verwendet werden.

OverSim [Baumgart et al., 2007] ist ein Simulator für Overlay-Netzwerke, der auf *OMNeT++*[16] basiert. Ein Vorteil von OverSim gegenüber den obengenannten Visualisierungsumgebungen liegt darin, dass OverSim für die Simulation von unterschiedlichen Overlay-Implementierungen eingesetzt werden kann. OverSim stellt die Implementierungen der Overlay-Netzwerke Chord, Kademlia [Maymounkov und Mazieres, 2002] und Gia [Chawathe et al., 2003] bereit und verfügt über eine API, die dem Nutzer die Implementierung eigener Overlays ermöglicht. Zusätzlich kann OverSim mit unterschiedlichen Modellen der Underlay-

[14] `http://research.microsoft.com/~antr/Pastry`
[15] `http://pdos.csail.mit.edu/chord/howto.html`
[16] `http://www.omnetpp.org/`

Netzwerk-Topologie kombiniert werden. Netzwerke mit bis zu 10^6 Knoten können mit OverSim simuliert werden.

P2PSim [Li et al., 2005] ist ein weiterer Overlay-Simulator. P2PSim unterstützt die Implementierungen von Chord, Tapestry, Kelips [Gupta et al., 2003], Kademlia und OneHop [Gupta et al., 2004]). Der Fokus bei P2PSim liegt auf der Simulation von Lookup-Anfragen unter Berücksichtigung des Knotenausfalls. Ähnlich wie OverSim kann P2PSim mit unterschiedlichen Underlay-Modellen kombiniert werden. Es können insgesamt bis zu 3000 Knoten simuliert werden.

Die Simulationsumgebung *OverlayWeaver*[17] unterstützt die Simulation der Overlay-Netzwerke Koorde [Kaashoek und Karger, 2003], Chord, Kademlia, Pastry und Tapestry. Außerdem bietet OverlayWeaver eine Benutzerschnittstelle für die Entwicklung und Simulation eigener Overlays. Darüber hinaus ist OverlayWeaver für die Simulation von Multicast-Ansätzen geeignet. Netzwerke mit bis zu 4000 Knoten können mit diesem Werkzeug simuliert werden. Es werden jedoch keine Underlay-Modelle unterstützt.

Die Umgebung *Narses* [Maniatis et al., 2003] simuliert Anwendungsszenarien mit einer hohen Netzbelastung und untersucht die Netzwerk-Skalierbarkeit. Narses unterstützt Underlay-Modellierung und kann Netzwerke mit bis zu 600 Knoten simulieren.

PlanetSim-Simulator [Garcia et al., 2005] unterstützt die Simulation der Overlays Chord- und *Symphony* [Manku et al., 2003] und stellt eine API zur Entwicklung von nutzerspezifischen Overlay-Ansätzen bereit. Das Netzwerk wird dabei nach und nach aufgebaut. Das heißt, man fängt mit einem Knoten an und fügt sukzessive die anderen Knoten hinzu. Auf diese Art und Weise können Netzwerke mit bis zu 10^6 Knoten entstehen. Generierte Netzwerke können abgespeichert und später wieder geladen werden. Auch eine einfache Underlay-Abstraktion ist verfügbar.

PeerSim[18] ist ein skalierbarer Overlay-Simulator, der in der Lage ist, bis zu 10^7 Knoten zu simulieren. Die Simulation beschränkt sich allerdings auf die Overlay-Konstruktion (Join, Leave, Fail). Eine einfache zugrunde liegende Underlay-Topologie kann modelliert werden.

ONSP [Wu et al., 2004] führt Simulationen in einem Hochleistungscluster durch. ONSP lehnt sich an das Paradigma der diskreten Ereignissimulation an, in der die Simulationsschritte taktweise ausgeführt werden. Der Vorteil dieser Vorgehensweise gegenüber einer Echtzeitsimulation liegt darin, dass die Ergebnisse bei der gleichen Eingabe bzw. bei gleichen Ausgangsvoraussetzungen reproduzierbar sind. In ihrer Arbeit haben die ONSP-Entwickler Chord- und Pastry-Overlays implementiert und evaluiert.

P2PRealm [Kotilainen et al., 2006] ist ein weiterer Overlay-Netzwerk-Simulator, welcher ebenfalls einen Cluster-Support anbietet. P2PRealm verfügt über eine Visualisierungskomponente namens *P2PStudio*.

[17] `http://overlayweaver.sourceforge.net/`
[18] `http://peersim.sourceforge.net/`

Eine Gegenüberstellung von P-Grid und *FreeNet* [Clarke et al., 2002] Overlay-Netzwerken ist in [Aberer et al., 2003] vorgestellt. Dabei wurden beide Implementierungen auf der Basis einer realistischen Netzwerk-Topologie aufgesetzt. Während der Simulation haben die Autoren die Netzwerkbelastung bei der Overlay-Konstruktion genauso wie den jeweiligen Speicherbedarf erfasst.

Weitere detaillierte Studien, die sich mit den entsprechenden Simulationsumgebungen befassen, können in [Naicken et al., 2006b] und [Naicken et al., 2006a] eingesehen werden.

2.3 Multicast-Kommunikation und DVEs

In verteilten Anwendungen ergibt sich sehr oft die Notwendigkeit, bestimmte Informationen gleichzeitig an mehrere Empfänger zu versenden. In virtuellen Umgebungen müssen zum Beispiel die Zustandsaktualisierungen eines Avatars an alle Nutzer in derselben AoI verteilt werden. Würde hier ein Nutzer eine Nachricht an jeden Empfänger einzeln versenden, so würde daraus ein unnötig hohes Nachrichtenaufkommen resultieren, weil identische Nachrichten zum Teil mehrmals über gleiche Pfade auf dem Weg zu ihren Empfängern geroutet werden.

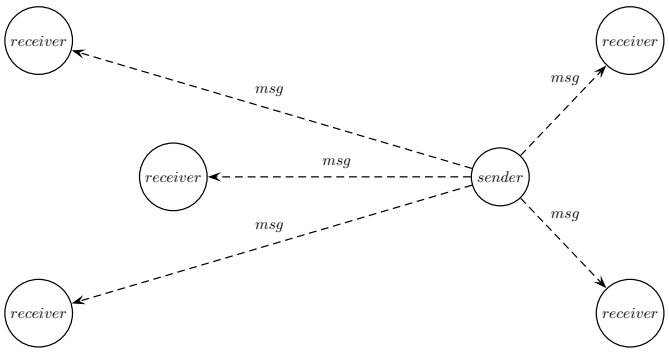
Um diesem Problem entgegenzuwirken, wurde der *IP-Multicast*[19] entwickelt. Dabei sollte der Sender genau eine Nachricht an eine Empfängergruppe versenden. Auf dem Weg zu den Empfängern sollte die Nachricht dann von den entsprechenden Routern vervielfältigt werden. In der Realität ist IP-Multicast nur bedingt einsetzbar, weil die *Internet-Service-Provider* (ISP) keinen IP-Multicast-Dienst für private Anwender zur Verfügung stellen [Schraeder, 2004]. Somit scheidet IP-Multicast als Technologie für die Verteilung von Daten in virtuellen Umgebungen aus.

Um die Unannehmlichkeiten von IP-Multicast zu umgehen, wird der Multicast-Gedanke in die Anwendungsschicht verlagert. Beim *Anwendungsschicht-Multicast* wird über die Netzwerkschicht eine logische Overlay-Schicht gelegt, die dann für die effiziente Verteilung von Informationen Sorge trägt. Dabei wird großer Wert auf die Minimierung der Netzwerkbelastung und des Nachrichtenaufkommens gelegt. Die Abbildung 2.7 zeigt eine Gegenüberstellung zwischen dem Unicast und dem Anwendungsschicht-Multicast. Während beim Unicast mehrere Nachrichten denselben Pfad nehmen, um zu den räumlich benachbarten Empfängern in einer Region zu gelangen, wird beim Anwendungsschicht-Multicast eine Nachricht versendet, die dann in dieser Region verteilt wird.

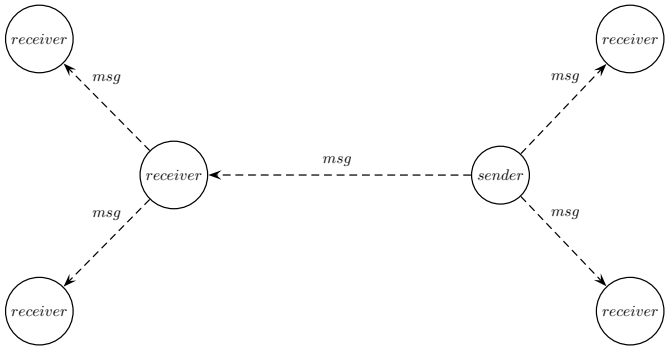
Im Folgenden werden einige Ansätze zur Gewährleistung des Anwendungsschicht-Multicasts vorgestellt.

NARADA [Chu et al., 2000] ist eine der ersten Anwendungsschicht-Multicast-Infrastrukturen. NARADA kann für die Gruppenkommunikation in den Bereichen der Audio- und Videokonferenzen, aber auch virtuellen Klassenzimmern und MMOGs eingesetzt werden. Dabei geht man von einer Gruppengröße von ca. 10

[19] `http://tools.ietf.org/html/rfc1075`



(a) Unicast



(b) Anwendungsschicht-Multicast

Abbildung 2.7. Unicast vs. Anwendungsschicht-Multicast

bis 500 Nutzern aus. Der entsprechende Multicast-Baum wird ausgehend von den Entfernungen (Latenzen) zwischen den Knoten aufgebaut.

NICE [Banerjee et al., 2002] ist eine weitere Anwendungsschicht-Multicast-Infrastruktur, die für Anwendungen mit vielen Empfängern und einem geringen Bandbreitenbedarf ausgelegt ist. Als mögliche Anwendungsszenarien wurden das Internetradio und Börsenkurs- bzw. Nachrichtenticker berücksichtigt. Im Gegensatz zu NARADA ist NICE hierarchisch und strukturiert aufgebaut, das heißt, über die Netzwerk-Topologie werden mehrere logisch und hierarchisch strukturierte Cluster darübergelegt. Ein Knoten in der übergeordneten Ebene ist dabei für die Weiterleitung der Nachrichten an ein darunterliegendes Cluster zuständig.

Im Gegensatz zur NARADA-Infrastruktur, in der ein Baum auf der Basis einer Graph-Abstraktion und Netzwerklatenz als Kantengewichtung aufgebaut wird, und zum hierarchisch strukturierten Cluster-Aufbau von NICE, befassen sich viele Forschungsgruppen [Gallager et al., 1983, Chin und Ting, 1985, Gafni, 1985, Awerbuch, 1987, Garay et al., 1993, Kutten und Peleg, 1995] mit der Berechnung eines *minimalen aufspannenden Baums* (MST, engl. Minimum Spanning Tree) und seiner Verwendung für das Anwendungsschicht-Multicast. Meistens wird hierbei, wie zum Beispiel in ALMI (Application Level Multicast Infrastructure) [Pendarakis et al., 2001], eine zentrale Instanz für die Berechnung des MST eingesetzt. Alle MST-basierten Ansätze müssen jedoch mit einem hohen

Verwaltungsaufwand zurechtkommen, weil das Beitreten bzw. das Ausscheiden von Knoten eine komplette Neuberechnung erfordert. Einige weitere Forschungsgruppen [Peleg und Rubinovich, 2000], [Elkin, 2004b] und [Elkin, 2004a] haben die Konstruktion von suboptimalen aufspannenden Bäumen untersucht. In [Khan und Pandurangan, 2008] wird die Konstruktion eines *Nearest-Neighbor-Tree* (NNT) anstatt eines MST vorgeschlagen, was zum Teil die oben erwähnten Vorteile mit sich bringt. Diese Idee ist auch in dem Peer-to-Peer-Netzwerk *JXTA* wiederzuerkennen. Um die akzeptablen Multicast-Kosten und einen vergleichsweise geringen Verwaltungsaufwand zu garantieren, verbindet sich ein neuer JXTA-Knoten mit dem Knoten, der die geringste Latenz aufweist.

In [Ratnasamy et al., 2002] schlagen die Autoren einen *Binning*-Algorithmus vor, bei dem die benachbarten Knoten, abhängig von ihrer *Round-Trip-Time* (RTT) zu den ausgezeichneten *Landmark-Servern*, zu so genannten Bins zusammengefasst werden. Dabei misst ein Knoten die RTT-Distanz zu den Landmark-Servern und sortiert die Entfernungen in einer aufsteigenden Reihenfolge. Knoten mit der gleichen Reihenfolge gehören zu demselben Bin und sind per Definition näher zueinander als die Knoten mit unterschiedlichen Reihenfolgen. Dieser Ansatz reduziert die anfallende Kommunikation auf ein Minimum (RTT-Anfragen zu den Servern) und unterstützt die Konstruktion eines Multicast-Baums auf der Basis der gewonnenen Informationen.

Eine wichtige Rolle bei der Konstruktion eines Multicast-Baums auf der Anwendungsschicht spielt die Berücksichtigung der darunterliegenden Netzwerk-Topologie und die Anpassung der Kanten des Overlays an die effizienteren Pfade der Netzwerkschicht. In [Choffnes und Bustamante, 2008] ist ein Ansatz vorgestellt, der den teuren Nachrichtenverkehr zwischen unterschiedlichen ISP-Knoten auf der Netzwerkebene und somit auch die Latenzzeiten reduziert, ohne dabei die Performanz des Systems auf der Anwenderebene zu beeinflussen.

In [Kubiatowicz, 2003] hat der Autor betont, dass die Berücksichtigung der topologischen Knoten-Lokalität bei der Konstruktion von Peer-to-Peer-Overlays den Schlüssel zur effizienten Kommunikation darstellt. Die Berücksichtigung der Beschaffenheit des physikalischen Netzes auf der Anwendungsebene würde die Leistungsfähigkeit des Systems verbessern und die Knotenverfügbarkeit steigern, weil die Wahrscheinlichkeit für einen Verbindungsverlust mit kürzeren Distanzen ebenfalls fällt.

[Feldmann und Aggarwal, 2008] behaupten, dass das Routing der meisten Peer-to-Peer-Systeme, das gegenwärtig auf der Anwendungsebene stattfindet, zum größten Teil unabhängig von der Beschaffenheit der darunterliegenden Netzwerk-Topologie erfolgt. Dies führt dazu, dass Nachrichtenverkehr aus der Sicht der physikalischen Netzwerkschicht sehr ineffizient ist und die ISPs unnötig belastet. Schuld daran ist die zufällige Nachbarauswahl. Einige Studien zeigen, dass Peer-to-Peer-Verkehr oftmals unnötigerweise mehrmals die Grenzen von ISPs überschreitet. Das kann dazu führen, dass beispielsweise ein Knoten in Berlin seine Daten von einem Knoten in Sydney bezieht, obwohl exakt die gleichen Daten auch in Frankfurt verfügbar sind. Die Autoren folgern, dass die Beschaffenheit der Netzwerk-Topologie beim Routing auf der Anwendungsebene berücksichtigt werden sollte.

Die *P4P-Architektur* [Xie et al., 2008] zielt ebenfalls auf die Reduzierung des Netzwerkverkehrs durch eine gezielte Wahl von Verbindungen in einem Peer-to-Peer-Overlay und durch die Berücksichtigung der Netzwerkeigenschaften. Dadurch kann unter anderem die Hop-Anzahl einer Nachricht signifikant reduziert und eine nennenswerte Steigerung der Effizienz erreicht werden.

2.4 Konsistenzaspekte und DVEs

In verteilten Systemen ist es oftmals aus Gründen der Effizienz notwendig, Kopien (Replikate) eines Objektes auf die einzelnen Clients oder Server zu verteilen. Laut [Malte, 1997] sind mit der Replikation von Daten immer zwei gegensätzliche Ziele verbunden: die Erhöhung der Verfügbarkeit und die Sicherung der Konsistenz der Daten. Die Form der Konsistenzsicherung bestimmt dabei, inwiefern das eine Kriterium erfüllt und das andere dementsprechend nicht erfüllt ist (Trade-off zwischen Verfügbarkeit und Konsistenz der Daten). Stark konsistente Daten sind stabil, das heißt, falls mehrere Kopien der Daten existieren, dürfen keine Abweichungen auftreten. Die Verfügbarkeit der Daten ist hier jedoch stark eingeschränkt, weil ein lesender oder schreibender Zugriff auf stark konsistente Daten einen hohen algorithmischen Aufwand verursacht und somit viel Zeit in Anspruch nehmen würde. Je schwächer die Konsistenz wird, desto mehr Abweichungen können zwischen verschiedenen Kopien einer Datei auftreten, wobei die Konsistenz nur an bestimmten Synchronisationspunkten gewährleistet wird. Dafür steigt aber die Verfügbarkeit der Daten, weil sie sich leichter replizieren lassen, beim Lesen oder Schreiben keinen hohen Synchronisationsaufwand erfordern und dadurch wesentlich kürzere Zugriffszeiten ermöglichen.

Diese traditionelle Auffassung der Konsistenz adressiert die Konsistenz auf der Systemebene. Aber wie soll diese Definition auf der Anwendungsebene in einer virtuellen Umgebung interpretiert werden? Übertragen auf die virtuellen Umgebungen impliziert für [Gautier et al., 1999] die Konsistenz, dass *zu jedem Zeitpunkt alle Teilnehmer in einer AoI idealerweise dieselbe Information zur selben Zeit sehen sollten.* Diese Ansicht wird in dem Zusammenhang als die *absolute Konsistenz* bezeichnet. Eine formale Definition dieses Modells ist beispielsweise in [Zhou et al., 2003] angegeben:

Definition 2.2 (Absolute Konsistenz). *Sei $o_j,\ j = 1 \dots m$ ein dynamisches Objekt in einer AoI und m die Anzahl der Objekte in dieser AoI. Sei $V_i^{o_j}(t),\ i \in \{1, \dots, n\}$ die Sicht (View) des Nutzers i auf das Objekt o_j zum Zeitpunkt t, wobei n die Anzahl der Nutzer in der AoI ist. Der Zustand einer virtuellen Umgebung heißt* ***absolut konsistent****, wenn für alle Objekte o_j in der entsprechenden AoI zu jedem Zeitpunkt die Gleichung*

$$\forall j,\ t:\ V_1^{o_j}(t) = V_2^{o_j}(t) = \dots = V_n^{o_j}(t)$$

gilt. Das heißt, alle Nutzer haben zu jedem Zeitpunkt eine identische Ansicht.

□

In einer virtuellen Umgebung ist die absolute Konsistenz aufgrund der inhärenten Netzwerklatenz, die zwangsläufig bei der Informationsverteilung anfällt, jedoch nicht erreichbar [Delaney et al., 2006]. In vielen Anwendungsszenarien in virtuellen Umgebungen ist eine absolute Konsistenz nicht erforderlich und es können Inkonsistenzen zu einem bestimmten Grad toleriert werden, solange die Nutzer das subjektive Gefühl haben, in einer konsistenten Umgebung zu interagieren.

So wird zum Beispiel in [Chandler und Finney, 2005] der Grad der regulierten (engl. managed) Inkonsistenz vorgestellt, der den Schwellwert der zulässigen lokalen Zustandsdivergenz darstellt, die nicht als störend empfunden wird. Als Zugewinn profitiert das Verfahren von einer hohen Interaktivität. Dabei werden auch die so genannten Zielzustände an die Teilnehmer propagiert, die durch die Anwendung einer bestimmten Anpassungsroutine erreicht werden.

In [Yu und Vahdat, 2000] haben die Autoren unterschiedliche Anwendungsszenarien in verteilten Systemen untersucht und festgestellt, dass es eine Klasse von Anwendungsszenarien gibt, die einen gewissen Grad an Inkonsistenz tolerieren und vom Zugewinn an Performanz profitieren können. In ihrer Arbeit stellen die Autoren eine Metrik-basierte Middleware vor, die je nach Einstellungen unterschiedliche Konsistenzmodelle unterstützt und eine signifikante Performanz-Steigerung mit sich bringt.

Eine Infrastruktur für das Konsistenz-Management in *Massively Multiuser Virtual Environments* MMVEs wurde in [Schiele et al., 2008] vorgestellt. Hier plädieren die Autoren für den Einsatz von unterschiedlichen Konsistenz-Plugins, die Konsistenz zu unterschiedlichen Graden garantieren können. Die Wahl eines solchen Plugins hängt von dem jeweiligen Anwendungsszenario ab.

In [Lu et al., 1999] wurde ein weiterer Ansatz für das Konsistenz-Handling in virtuellen Umgebungen mit vielen Nutzern vorgestellt. Dieser Ansatz umfasst drei Konsistenzstufen, die sich in dem garantierbaren Grad der Konsistenz bzw. Performanz unterscheiden. Abhängig von der Anzahl der Benutzer bzw. der Systemauslastung wird von einer (strengeren) auf die andere (schwächere) Konsistenzstufe umgeschaltet, um die geforderten Systemantwortzeiten einzuhalten. Um den Grad der Inkonsistenz formal erfassen zu können, wird in [Zhou et al., 2003] die folgende Metrik vorgeschlagen:

Definition 2.3 (Time-Space-Inkonsistenz-Metrik). *Die Time-Space-Inkonsistenz-Metrik Ω wird als Produkt der tatsächlichen anwendungsspezifischen Abweichung δ und der Beobachtungszeit dieser Abweichung τ[20] definiert:*

$$\Omega = \delta \cdot \tau$$

□

Basierend auf der Definition der Time-Space-Inkonsistenz-Metrik definieren [Zhou et al., 2003] das Modell der *Time-Space-Konsistenz* wie folgt:

Definition 2.4 (Time-Space-Konsistenz). *Es seien ε die kleinste Abweichung, die ein Nutzer erkennen kann (anwendungsspezifisch), und ι die Reaktionszeit eines menschlichen Nutzers. Eine virtuelle Umgebung ist* ***Time-Space-konsistent****,*

[20] τ ist die Zeitspanne in der sich die Client-Zustände um δ unterscheiden.

wenn die Bedingung

$$\frac{\Omega}{\varepsilon \cdot \iota} < 1$$

erfüllt ist. Andernfalls ist die Umgebung inkonsistent.

□

Die Festlegung der Grenzwerte ε und ι ist anwendungsabhängig. Die Beobachtungszeit τ der Abweichung kann zur Laufzeit ermittelt werden. Zur Abschätzung der Abweichung δ ist in [Zhou et al., 2003] ein Verfahren angegeben. Im späteren Verlauf der Arbeit in Abschnitt 6.1 wird ein alternatives Konsistenzmodell definiert, welches Bezug auf die Time-Space-Konsistenz bzw. auf die Time-Space-Inkonsistenz-Metrik nimmt.

Die Vielzahl und die Heterogenität der Anwendungsszenarien, in denen die obengenannten Konsistenzaspekte zum Tragen kommen, führen dazu, dass es viele unterschiedliche Konsistenzverfahren und Konsistenzalgorithmen gibt. Im Folgenden werden einige dieser Verfahren vorgestellt, die für diese Arbeit von Belang sind.

In einem verteilten System bzw. einer virtuellen Umgebung kann es vorkommen, dass zwei Prozesse (Nutzer) konkurrierend (nebenläufig) auf ein und dasselbe Objekt zugreifen und dieses verändern wollen. Diese Prozesse müssen synchronisiert werden, um Inkonsistenzen zu vermeiden. Eine wichtige Klasse der Synchronisationsalgorithmen stellen die Algorithmen des verteilten wechselseitigen Ausschlusses dar. Diese stellen sicher, dass bei einer Menge an Prozessen zu jedem Zeitpunkt höchstens einer auf die gemeinsam genutzten Daten zugreift. Ein verteilter Algorithmus für den wechselseitigen Ausschluss wird in Abschnitt 2.4.1 vorgestellt.

Eng verwandt mit dem wechselseitigen Ausschluss sind die *Transaktionen*. Innerhalb einer Transaktion können mehrere lesende oder schreibende Zugriffe auf gemeinsam genutzte Daten erfolgen. Anders als beim wechselseitigen Ausschluss kann eine Transaktion jedoch u.a. abgebrochen und der ursprüngliche Zustand wiederhergestellt werden. Somit wird eine Transaktion entweder erfolgreich oder gar nicht ausgeführt. Die Transaktionen werden in Abschnitt 2.4.2 behandelt.

Schließlich wird in Abschnitt 2.4.3 ein kurzer Überblick über das Konsistenz-Handling in DVEs gegeben. Dabei werden einige Ansätze, die eine schwächere Form der Konsistenz zu Gunsten einer besseren Interaktivität garantieren, sowie einige *Latenzkompensierungstechniken*[21] vorgestellt.

2.4.1 Wechselseitiger Ausschluss

Ein wichtiger Ansatz zur Wahrung der Konsistenz in verteilten Systemen ist *wechselseitiger Ausschluss*. Wechselseitiger Ausschluss kommt bei verteilten Systemen zum Einsatz, in denen mehrere Prozesse auf einer gemeinsam genutzten Datenstruktur arbeiten, d.h. lesen oder schreiben. Um die Konsistenz der gemeinsam genutzten Daten nicht zu verletzen, muss ein Prozess, der auf die Daten lesend

[21] Latenzkompensierungstechniken maskieren die Auswirkungen der Netzwerklatenz, die sich nachteilig auf die Interaktivität und Konsistenz in DVEs auswirkt.

oder schreibend zugreifen möchte, zuerst in einen *kritischen Abschnitt* eintreten, in dem sich zu einem Zeitpunkt höchstens ein Prozess befinden darf.

In Client-Server-Systemen, d.h. in Systemen mit einer autorisierten Kontrollinstanz, können Eintritte in den kritischen Abschnitt vom Server z.B. mithilfe von Semaphoren synchronisiert werden. In dezentralisierten Peer-to-Peer-Systemen müssen zu diesem Zwecke verteilte Algorithmen zum Tragen kommen, bei denen sich Clients, die gleichzeitig in den kritischen Abschnitt eintreten wollen, untereinander darauf einigen, wer zuerst eintreten darf.

Im Folgenden wird ein verteilter Wahl-Algorithmus von Maekawa [Maekawa, 1985] betrachtet, der zur Realisierung des wechselseitigen Ausschlusses in Systemen ohne zentrale Koordinatorinstanz verwendet wird. Dabei vertritt Maekawa die Ansicht, dass ein Prozess nicht von allen beteiligten Prozessen eine Zustimmung braucht, um in den kritischen Abschnitt eintreten zu können. Um dies zu erreichen, unterteilt Maekawa die Prozessmenge in Prozess-Teilmengen. Diese Prozess-Teilmengen werden als *Wählermengen* (engl. Voting Sets) bezeichnet. Dabei wird vorausgesetzt, dass die Wählermengen zweier beliebiger Prozesse sich überlappen. Prozesse in der Schnittmenge dieser Wählermengen stellen sicher, dass höchstens ein Prozess in den kritischen Abschnitt eintreten kann, indem sie ihre Stimme nur an einen Prozess geben. Die Abbildung 2.8 veranschaulicht die Idee des Algorithmus. Die Wählermengen der Prozesse 3 und 6 enthalten die Prozesse 2, 3, 5 und 8 bzw. 4, 5, 6 und 9. Prozess 5 liegt in der Schnittmenge der beiden Wählermengen und ist somit für die Sicherstellung des wechselseitigen Ausschlusses verantwortlich.

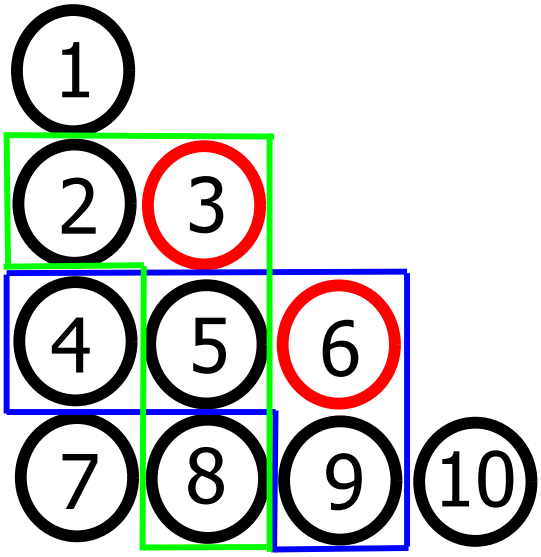


Abbildung 2.8. Überschneidung von Wählermengen

Die theoretische untere Grenze für die Größe einer Wählermenge beträgt $\sqrt{N}$. Allerdings ist die Bestimmung solcher Mengen nicht trivial. In [Maekawa, 1985] ist eine Approximation für die Bestimmung der Wählermengen angegeben, in der

die Knoten in einer quadratischen Matrix $m \times m$ mit $m = \left\lceil \sqrt{N} \right\rceil$ angeordnet sind. Die Wählermenge setzt sich dabei für jeden Knoten $K_{i,j}$ mit $0 \leq i, j \leq m$ aus den Knoten der i-ten Zeile und der j-ten Spalte zusammen. Die Größe der approximierten Wählermenge ist $2\sqrt{N} - 1$. Beim Dreiecks-Verfahren [Luk und Wong, 1997] werden die Knoten in einem Dreieck angeordnet (Abbildung 2.8). Dabei beträgt die Größe der Wählermenge $\sqrt{2}\sqrt{N}$. Im weiteren Verlauf der Arbeit wird für die Größe der Wählermengen $\sqrt{2}\sqrt{N}$ angenommen. Jedem Prozess $p_i,\ (i = 1, 2, \ldots, N)$ wird eine Wählermenge V_i zugeordnet. Für eine Wählermenge V_i gilt:

$$V_i \subseteq \{p_1, p_2, \ldots, p_N\}$$

Somit enthält eine Wählermenge V_i eine Teilmenge von Prozessen. Für die Mengen $V_i,\ (i = 1, 2, \ldots, N)$ gilt:

- $\forall i\ p_i \in V_i$
- $\forall i, j = 1, 2, \ldots, N\ V_i \bigcap V_j \neq \emptyset$ – es existiert mindestens ein gemeinsames Element in zwei beliebigen Teilmengen
- $\forall i\ |V_i| = K$ – alle Prozesse haben gleich große Wählermengen
- jeder Prozess p_i ist in M Wählermengen enthalten

Der Algorithmus von Maekawa ist ebenso wie die Berechnung der optimalen Wählermengen nicht trivial. In der Literatur (u.a. in [Coulouris et al., 2002]) wird deshalb sehr oft eine vereinfachte Variante des Algorithmus angegeben. Leider kann es bei dieser Variante des Algorithmus zu Deadlocks kommen. Um die Deadlock-Freiheit zu garantieren, kann der Algorithmus so angepasst werden, dass die Anfragen in der Reihenfolge ihres Zeitstempels abgearbeitet werden [Sanders, 1987].

Der Ablauf des vereinfachten Algorithmus von Maekawa wird im Folgenden beschrieben. Als Voraussetzung wird festgehalten, dass jeder Prozess p_i einen Zustand $state \in \{RELEASED,\ WANTED,\ HELD\}$ und eine boolesche Variable $voted \in \{false,\ true\}$ besitzt. Die boolesche Variable gibt an, ob einem anderen Prozess die Erlaubnis, in den kritischen Abschnitt einzutreten, erteilt wurde. $RELEASED$ ist der Initialzustand jedes Prozesses. Will ein Prozess in den kritischen Abschnitt eintreten, so wechselt er in den Zustand $WANTED$. Im kritischen Abschnitt befindet sich ein Prozess im Zustand $HELD$.

- **Initialisierung:**
  - $state := RELEASED$
  - $voted := false$
- **Prozess p_i will in den kritischen Abschnitt eintreten:**
  - $state := WANTED$
  - Anfrage an alle Prozesse in $V_i \setminus \{p_i\}$ senden
  - Warte auf $K - 1$ Antworten
  - $state := HELD$ // p_i tritt in den kritischen Abschnitt ein
- **Prozess p_j empfängt eine Anfrage von Prozess p_i, mit $j \neq i$:**
  - WENN $state == HELD$ ODER $voted == true$ DANN

    - Anfrage in Warteschlange gemäß [Sanders, 1987] einsortieren, ohne zu antworten
  - SONST
    - Sende eine Antwort an p_i
    - $voted := true$
- **Prozess p_i will den kritischen Abschnitt verlassen:**
  - $state := RELEASED$
  - Freigabe an alle Prozesse in $V_i \setminus \{p_i\}$ senden
- **Prozess p_j empfängt eine Freigabe von Prozess p_i, mit $j \neq i$:**
  - WENN die Warteschlange nicht leer ist DANN
    - Entnehme den ersten Prozess p_k aus der Warteschlange
    - Sende eine Antwort an p_k
    - $voted := true$
  - SONST
    - $voted := false$

Auch in DVEs werden oftmals Prozess- und Objektsynchronisationsverfahren eingesetzt, wie die nachfolgenden Beispiele zeigen. Um die Verfügbarkeit von Objekten in DVEs zu steigern, werden diese Objekte unter den entsprechenden Clients repliziert. Die Entwickler stehen hier vor einem Dilemma: Der Einsatz eines strengen Synchronisationsverfahrens würde zwar die Inkonsistenzen ausschließen, dafür aber auch die Interaktivität der Umgebung enorm einschränken. Die Abschwächung der Synchronisation würde die Interaktivität steigern, allerdings müssten dann die Inkonsistenzen in Kauf genommen werden.

Im Folgenden werden einige Synchronisationsverfahren, die ihre Anwendung in virtuellen Umgebungen finden, vorgestellt.

Gleichschritt-Synchronisation (engl. Lockstep Synchronization) ist ein pessimistisches Synchronisationsverfahren, das die Konsistenz auf Kosten der Antwortzeiten garantiert, dafür aber die nachträglichen Zustandskorrekturen ausschließt [Funkhouser, 1995].

Erzwungene Globale Konsistenz (engl. Imposed Global Consistency) ist ein weiteres Synchronisationsverfahren, bei dem die Ausführung und die Darstellung von eigenen und fremden Aktionen solange verzögert wird, bis die obere Latenzschranke erreicht ist [Gautier et al., 1999]. Die Berücksichtigung der Latenzschranke hat den Vorteil, dass nun ausreichend viel Zeit zur Verfügung steht, um die Aktionen, die von den anderen Clients getriggert wurden, zu übertragen und in den lokalen Zustand zu integrieren. Auch hier wird die Konsistenz der globalen Zustände auf Kosten der System-Interaktivität bzw. Antwortzeit erreicht.

Verzögerte Globale Konsistenz (engl. Delayed Global Consistency) [Qin, 2002] unterscheidet sich von der erzwungenen globalen Konsistenz dadurch, dass hier eine asynchrone Konsistenz zulässig ist. Das heißt, die Nutzer sehen zwar dieselben Zustände, aber möglicherweise nicht zur selben Zeit.

In [Roberts und Sharkey, 1997] wird ein vorhersagenbasiertes Synchronisationsverfahren vorgestellt, das darauf abzielt, eine effiziente Nebenläufigkeitskontrolle zu gewährleisten. Dabei geht es darum, den nächsten Eigentümer eines Ob-

jektes vorherzusagen und die Objekt-Anfragen gezielt an diesen Eigentümer zu senden. Leider kann es auch hier bei falsch getroffenen Annahmen zu Reaktionsverzögerungen kommen.

In Abschnitt 6.3 dieser Arbeit wird ein alternativer Ansatz untersucht, der auf dem Maekawa-Algorithmus basiert und die Fragestellung des effizienten konkurrierenden Zugriffs auf ein Objekt von mehreren Nutzern behandelt.

2.4.2 Verteilte Transaktionen

Verteilte Transaktionen kommen in vielen verteilten Anwendungen zum Einsatz und spielen auch bei Interaktionen in virtuellen Umgebungen eine wichtige Rolle. Im Folgenden wird der Transaktionsbegriff definiert und seine charakteristischen Eigenschaften beschrieben. Darüber hinaus werden zwei Arten von Transaktionen vorgestellt.

Definition 2.5 (Transaktion). *Eine Transaktion ist eine logische Folge von Lese- und Schreiboperationen auf einer Ansammlung von Objekten/Daten unter Berücksichtigung der Integrität [*Andre *et al., 1985].*

Die Integrität wird durch die folgenden vier Eigenschaften charakterisiert:

- **Atomarität (Atomicity):**
  - Transaktionen werden entweder erfolgreich ausgeführt oder abgebrochen.
  - Nach dem Abbruch erfolgt ein Zurücksetzen nicht nur der letzten fehlerhaften Operation, sondern aller Operationen, die zu der Transaktion gehören.
- **Konsistenz (Consistency):**
  - Systeminvarianten werden eingehalten.
  - Keine inkonsistenten Lese- oder Schreiboperationen.
- **Isolation (Isolation):**
  - Keine störenden Wechselwirkungen bei parallel ablaufenden Transaktionen.
  - Die Wirkung der parallel ablaufenden Transaktionen soll so sein, als wären diese in einer *serialisierbaren* Reihenfolge abgelaufen.
- **Dauerhaftigkeit (Durability):**
  - Ist eine Transaktion erfolgreich durchgelaufen, so bleiben die Änderungen dauerhaft bis zur nächsten Modifikation bestehen.

Eine weitere, eng mit der Isolation von Transaktionen verwandte Eigenschaft ist die Serialisierbarkeit [Coulouris et al., 2002].

Definition 2.6 (Serialisierbarkeit). *Unter Serialisierbarkeit von Transaktionen versteht man die Eigenschaft, dass mehrere Transaktionen gleichzeitig ausgeführt werden können, während sie ständig isoliert bleiben. Das heißt, nach dem parallelen Ablauf von mehreren Transaktionen soll das Endergebnis dasselbe sein, als wären sie nacheinander in irgendeiner Reihenfolge ausgeführt worden.*

Bei den Transaktionen wird zwischen *pessimistischen* und *optimistischen* Verfahren unterschieden. Der älteste und bekannteste pessimistische Ansatz ist das Zwei-Phasen-Sperrverfahren (2-Phase-Locking) [Bernstein et al., 1987]. Dabei

fordert ein Prozess, der eine Transaktion durchführt, während der ersten Phase Objekt-Sperren an, um auf die Objekte zugreifen zu können. In der zweiten Phase gibt der Prozess die Sperren wieder frei, wobei nach der ersten Freigabe keine weitere Objekt-Sperre mehr angefordert werden darf. Dieses Verfahren garantiert die Serialisierbarkeit, kann aber bei Abbruch von Transaktionen zu *Dirty-Read*-Problemen führen. Beim *strengen* Zwei-Phasen-Sperrverfahren werden die Sperren erst beim Transaktionsende oder beim Abbruch der Transaktion freigegeben. Dieses Verfahren hat gegenüber dem *einfachen* Zwei-Phasen-Sperrverfahren den Vorteil, dass eine Transaktion immer einen Wert liest, der von einer anderen Transaktion festgeschrieben wurde. Deshalb sind keine kaskadierten (hintereinander geschalteten) Rollback-Aktionen mehr erforderlich. Dieses Verfahren weist jedoch bei einer hohen Anzahl von Prozessen hohen Kommunikationsaufwand auf, was sich folglich in hohen Latenzzeiten niederschlägt. Sowohl das einfache als auch das strenge Verfahren kann zu Verklemmungen (Deadlocks) führen. Verfahren zur Deadlock-Erkennung sind ebenfalls in [Coulouris et al., 2002] beschrieben.

Ein Nachteil der betrachteten „pessimistischen“ Sperrverfahren liegt darin, dass das Setzen und Freigeben von Sperren mit einem hohen Aufwand verbunden ist und zu sehr die Parallelität von Prozessen einschränkt. Aus diesem Grund werden die pessimistischen Ansätze für den Einsatz in DVEs nicht berücksichtigt. Einen anderen Ansatz verfolgen die optimistischen Transaktionsverfahren [Kung und Robinson, 1981]. Bei den optimistischen Verfahren geht man „optimistisch“ davon aus, dass es keine konkurrierenden Zugriffe auf die Kopien eines Objektes gibt. Daher lassen sich Zugriffe auf gemeinsam genutzte Objekte ohne eine Überprüfung auf mögliche Konflikte durchführen. Wegen des sofortigen Zugriffs wird eine hohe Interaktivität ermöglicht. Kommt es aufgrund von Nebenläufigkeit zu Inkonsistenzen, müssen diese verarbeitet werden, um einen konsistenten Zustand wiederherzustellen.

Das *Time-Warp*-Verfahren, das auf dem Prinzip der optimistischen Transaktionen basiert, wird im folgenden Abschnitt kurz vorgestellt.

2.4.3 Konsistenz-Handling in DVEs

Anders als die meisten Anwendungen in verteilten Systemen stellen die virtuellen Umgebungen sehr hohe Anforderungen an die Interaktivität und erfordern eine sofortige Systemrückmeldung. Das heißt, direkt nach der Eingabe einer Benutzeraktion, die dann als Befehl an die Kontrollinstanz (Server oder Management-Overlay) gesendet wird, erwartet der Benutzer eine sofortige Rückmeldung des Systems. Diese Erwartung ist durch die Tatsache bedingt, dass die DVEs das Gefühl einer physikalischen und sozialen Nähe vermitteln. Erwartungskonform zur Kommunikation bzw. Interaktion in der Realität ist in einer virtuellen Umgebung aufgrund dieses Gefühls eine sofortige Rückmeldung erwünscht. Wegen der inhärenten Netzwerklatenz bzw. der Zeit, die für die Berechnung oder Bearbeitung der Benutzeraktion anfällt, ist eine sofortige Rückmeldung des Systems unmöglich. Denn im Gegensatz zu den konkurrierenden Prozessen, die nebenläufig auf einem Rechner gemeinsam genutzte Objekte verändern, müssen die Aktualisierungen zuerst ei-

ne bestimmte Entfernung in einem Netzwerk überwinden. Laut [Lamport, 1978] zeichnet sich ein verteiltes System dadurch aus, dass die Zeit, die für die Nachrichtenübertragung anfällt, im Vergleich zu einem System, das auf einem Rechner abläuft, signifikant höher ist. Somit enthalten die Aktualisierungen in DVEs fast immer bereits veraltete Informationen.

In [Delaney et al., 2006] stellen die Autoren die These auf, dass die Netzwerklatenz der entscheidende Faktor ist, der in DVEs zu Inkonsistenzen führt. Sie wirkt sich besonders spürbar auf die Anwendungen mit hohen Ansprüchen an die Interaktivität aus. Laut [Mauve et al., 2004] werden Latenzzeiten unter 120 Millisekunden von Nutzern einer virtuellen Umgebung nicht wahrgenommen. Latenzzeiten, die diesen Schwellwert unwesentlich übersteigen, werden von Nutzern zwar wahrgenommen, aber nicht als störend empfunden. Erst, wenn die Latenzzeiten 250 Millisekunden übersteigen, ist laut [Pantel und Wolf, 2002] keine erwartungskonforme Interaktivität mehr möglich. Es ist also erstrebenswert, dass Anwendungsszenarien mit sehr hohen Anforderungen an die Interaktivität innerhalb von 120 Millisekunden eine Systemrückmeldung erhalten. Für Szenarien mit weniger strengen Anforderungen können Latenzzeiten bis 250 Millisekunden toleriert werden. Latenzzeiten, die diesen Schwellwert übersteigen, können nicht mehr akzeptiert werden. Im weiteren Verlauf der Arbeit wird dieser Schwellwert $\iota = 250$ Millisekunden verwendet, um eine Aussage darüber zu treffen, ob ein Konsistenzverfahren den Interaktivitätsanforderungen einer Anwendung genügt oder nicht.

Um den Konsistenzanforderungen gerecht zu werden, versucht man in den virtuellen Umgebungen die Auswirkungen der Netzwerklatenz mittels unterschiedlicher Techniken zu verbergen. Im Folgenden werden einige dieser Techniken vorgestellt.

Wie in jeder anderen verteilten Anwendung existiert auch in DVEs eine Vielzahl gemeinsam genutzter Objekte, deren Zustand konsistent gehalten werden muss. Darüber hinaus sind die Clients in einer virtuellen Umgebung immer am aktuellen Zustand anderer Clients, zum Beispiel an den Positionen von deren Avataren, interessiert. Aufgrund der hohen Interaktivität in DVEs können sich diese Zustände ständig ändern. Zum Beispiel verändert ein Avatar, der sich in einer virtuellen Region bewegt, kontinuierlich seine Position und somit seinen Zustand. Da das ständige Versenden von Positionsupdates einen zu hohen Aufwand bedeutet und das latenzbedingte Warten auf die Systemrückmeldung das Empfinden der Nutzer stört, können Avatar-Positionen mithilfe des *Dead-Reckoning*-Verfahrens zwischen zwei Update-Nachrichten interpoliert werden. Bei diesem Verfahren sagt ein Client die Positionen bzw. die Bewegungen der Avatare in seiner AoI anhand der vorhandenen Informationen (aktuelle Position, Geschwindigkeit, Richtung, etc.) voraus. Obwohl Dead-Reckoning schon vor langer Zeit zur Positionsbestimmung für Schiffe und Flugzeuge verwendet wurde, wurde das Verfahren erst im Jahr 1992 im Zusammenhang mit DVEs in der *VERN*-Testumgebung [Blau et al., 1992] implementiert und ein Jahr später in dem *DIS-Protokoll* [IEEE-Std-1278-1993, 1993] formal definiert. Seitdem wurde das Verfahren im Kontext der DVEs und MMOGs unter anderem in [Aronson, 1997, Mauve, 1999, Mauve, 2000] ausführlich behandelt. Zu den Vorteilen von Dead-Reckoning zählt die Tatsache, dass die Anwendung dieser Technik es möglich macht, sofort auf Benutzereingaben zu reagieren,

ohne auf die Antwort des Systems zu warten. Das Verfahren hat jedoch auch Nachteile. In dem Intervall zwischen zwei Aktualisierungen kann es vorkommen, dass der Mitspieler seine Bewegungsrichtung oder die Geschwindigkeit ändert, was zu einer inkonsistenten Zustandsrepräsentation führt. Nach dem Erhalt der nächsten Aktualisierung muss die Position des Avatars berichtigt werden. Die nachträgliche Korrektur der vorhergesagten Positionen führt jedoch oftmals zu einer ruckeligen Darstellung, welche das Nutzerempfinden negativ beeinträchtigt. Mittlerweile gibt es Verfahren (z.B. *Lineare Konvergenz* [Hamza-Lup, 2004]), welche die Positionskorrektur „sanft“ durchführen, indem sie den Übergang von der vorhergesagten zu der tatsächlichen Position nicht abrupt, sondern verzögert gestalten.

In [Aggarwal et al., 2005] haben die Autoren die Anwendung des Dead-Reckoning-Verfahrens in einem verteilten System mit heterogenen Netzwerklatenzen untersucht und haben festgestellt, dass die Nutzer mit einer guten Netzanbindung bzw. einer geringen Latenz einen Vorteil gegenüber den Nutzern mit einer schlechteren Netzanbindung haben. Je besser die Netzanbindung war, desto genauer wurden die Positionen der Mitspieler vorhergesagt. Um die Fairness wiederherzustellen, wurde ein Verfahren vorgeschlagen, das die Update-Nachrichten zu unterschiedlichen Zeiten versendet, um den Empfangszeitpunkt an dem Knoten mit der größten Latenz zu synchronisieren. Nach der Anwendung des Verfahrens wurde ein vergleichbarer Vorhersagefehler bei allen Teilnehmern, unabhängig von deren Netzanbindung, beobachtet.

Des Weiteren existieren zwei Latenzkompensierungstechniken: *Time-Delay* und *Time-Warp* [Mauve, 2000]. Bei der erstgenannten Technik verzögert der Server die Verarbeitung von eingehenden Updates, um auf diese Weise die Latenzzeiten von unterschiedlichen Clients auszugleichen. Vergleichbar zur Time-Delay-Technik ist in [Kim et al., 2003] ein Verfahren vorgestellt, in dem die eingehenden Update-Nachrichten zuerst gepuffert und dann nach dem Ablauf einer vordefinierten Verzögerung ausgegeben werden. Dieses Verfahren basiert auf den globalen Zeitstempeln von [Kopetz, 1997].

Bei dem optimistischen Transaktionsverfahren Time-Warp werden Benutzeraktionen direkt ausgeführt, ohne auf die möglichen Inkonsistenzen zu prüfen. Kommt es dabei zu Inkonsistenzen, so hat der Server die Möglichkeit, die Ereignisse bis zu dem Zeitpunkt der Benutzereingabe rückgängig zu machen. Solche nachträglichen Zustandskorrekturen sollten jedoch nach Möglichkeit vermieden werden, weil sie das Nutzerempfinden negativ beeinflussen.

3

DVE-Framework HERA

In den vorhergehenden Kapiteln wurde bereits festgestellt, dass die große Popularität von virtuellen Umgebungen und die daraus resultierende große Nutzergemeinde die gegenwärtig eingesetzten Client-Server-Infrastrukturen leistungsmäßig an ihre Grenzen bringen. Deshalb werden in dieser Arbeit Ansätze untersucht, die zur Verbesserung der Skalierbarkeit und der Lastverteilung in virtuellen Umgebungen beitragen sollen. Dazu zählen sowohl die Verwendung eines strukturierten Management-Overlays anstelle eines Servers als auch die Übertragung von bestimmten Aufgaben (Gruppenkommunikation und Konsistenz-Handling) an die Client-Schicht. Dabei sollten bei dem clientbasierten Konsistenz-Handling vor allem die besonderen Interaktivitäts- und Konsistenzanforderungen, die Nutzer an die virtuellen Umgebungen stellen, berücksichtigt werden.

Die Abbildung 3.1 stellt prototypisch eine mögliche Realisierung des Infrastrukturwechsels hin zu verteilten dezentralisierten Peer-to-Peer-Ansätzen sowohl in der Management- als auch in der Client-Schicht dar. Dabei ist zu sehen, dass die Umgebung in unterschiedliche Partitionen (AoIs) aufgeteilt ist und jede AoI bzw. alle Objekte in dieser AoI von einem Knoten der Management-Schicht verwaltet werden. Die Knoten der Management-Schicht bilden untereinander ein Peer-to-Peer-System. Des Weiteren ist in der Abbildung zu sehen, dass die Clients innerhalb einer AoI zusammen mit dem für diese AoI zuständigen Management-Knoten ebenfalls ein (*autonomes*) Peer-to-Peer-System bilden. Die Autonomie in diesem Kontext bedeutet, dass Peer-to-Peer-Systeme unterschiedlicher AoIs völlig unabhängig voneinander existieren können. Somit kann die Client-Schicht aus mehreren autonomen Peer-to-Peer-Systemen (ein System pro AoI) bestehen. Ein autonomes Client-System wird im weiteren Verlauf auch als Client-Overlay bezeichnet.

Aufgrund unterschiedlicher Aufgaben sind in der Management- und in der Client-Schicht unterschiedliche Aspekte von Belang. Die Clients innerhalb einer AoI kommunizieren direkt miteinander[1] und sind für das Konsistenz-Handling verantwortlich. Damit diese Kommunikation effizient ablaufen kann, müssen die Clients innerhalb einer AoI ein dynamisches Multicast-Overlay auf der Anwendungsebene aufbauen. Dynamik an dieser Stelle bedeutet, dass das Multicast-Overlay

[1] Die Kommunikation zwischen den Clients in unterschiedlichen AoIs muss über die Management-Schicht erfolgen.

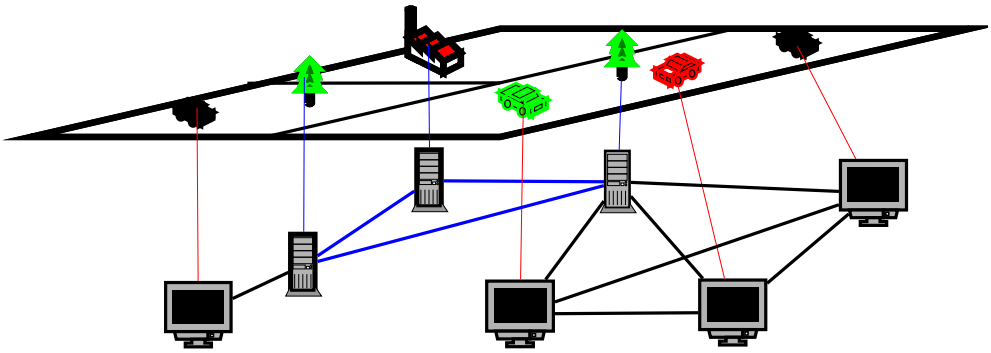


Abbildung 3.1. Peer-to-Peer-basierte virtuelle Umgebung

mit einer hohen Fluktuationsrate, bedingt durch häufige AoI-Wechsel von Clients[2], zurechtkommen muss.

In der Management-Schicht ist hingegen eine effiziente Suchfunktionalität erforderlich. Wenn beispielsweise ein Avatar seine AoI wechselt, fragt der entsprechende Client die Informationen der neuen AoI bei dem zuständigen Management-Knoten an. Falls dieser Knoten diese Informationen nicht hat, muss er die Möglichkeit haben, diese effizient über die Management-Schicht zu beschaffen.

Um u.a. der Frage auf den Grund zu gehen, ob verteilte Peer-to-Peer-basierte Ansätze als Basis-Infrastruktur in DVEs fungieren können, wurde in der Arbeitsgruppe die verteilte virtuelle Umgebung *HyperVerse* [Botev et al., 2008a] entwickelt, die im Folgenden in Abschnitt 3.1 vorgestellt wird. Damit bestimmte Algorithmen im Vorfeld in einer einfachen Umgebung implementiert und getestet werden können, bevor sie ihren endgültigen Einsatz in HyperVerse finden, wurde das leichtgewichtige *HERA-Framework*[3] [Bettinger et al., 2009b] implementiert. Das HERA-Framework wird ausführlich in Abschnitt 3.2 beschrieben. Das auf der Basis des HERA-Frameworks zu Demonstrations- und Untersuchungszwecken entwickelte Online-Spiel *HERATRONos* [Baumann, 2009] wird in Abschnitt 3.3 vorgestellt. Anhand dieses Spiels sollte die Machbarkeit aller in dieser Arbeit beschriebenen Ansätze und Konzepte im praktischen Einsatz evaluiert werden.

3.1 HyperVerse

Der HyperVerse-Browser (Abbildung 3.2) emuliert mögliche Anwendungsszenarien des Internets der Zukunft, in denen die Information nicht wie heute üblich durch Webseiten-Navigation, sondern durch das Bewegen eines Avatars in dreidimensionalen virtuellen Welten recherchiert wird. Diese Avatar-basierte Interaktion und Kommunikation ist ein Bestandteil des Konzeptes von HyperVerse. Jegliche Art der Interaktion mit Inhalten wie virtuellen Objekten, Online-Shops, Auktions-

[2] Ein AoI-Wechsel aus Sicht eines Clients bedeutet, dass der Client aus dem „alten“ Overlay austreten und dem neuen Overlay beitreten muss.

[3] HERA steht für HyperVerse Experimental Research Applications.

plattformen oder mit anderen Benutzern wird mittels eines Avatars durchgeführt. Jeder HyperVerse-Nutzer sollte die Möglichkeit haben die virtuelle Umgebung mitzugestalten, indem er Inhalte generiert bzw. verändert. Außerdem können mehrere HyperVerse-Welten (zum Beispiel Phantasie-Welten, E-Learning-Plattformen, etc.) zur gleichen Zeit koexistieren. Für Avatare gilt dabei das *Freedom of Travel-Paradigma*, welches besagt, dass ein Client ein einziges HyperVerse-Zugangskonto benötigt, um in alle ihm offenstehenden HyperVerse-Welten eintreten zu dürfen.

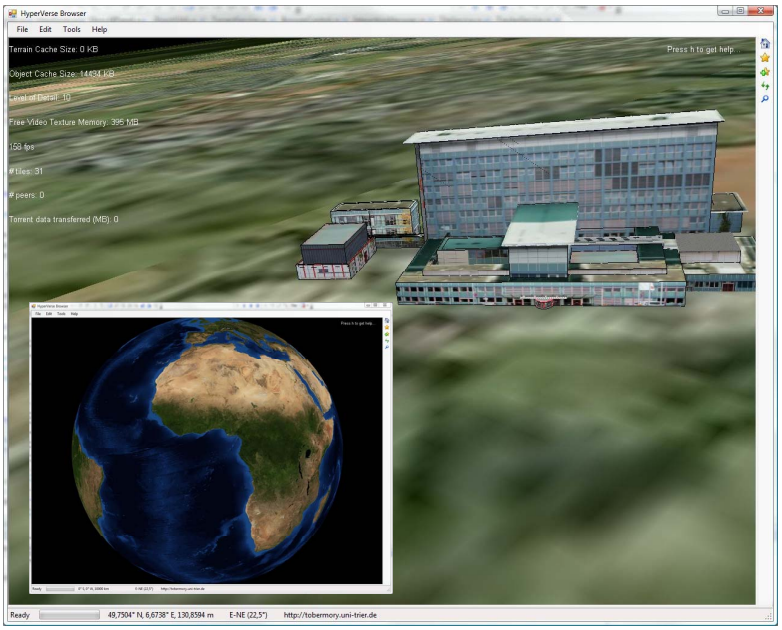


Abbildung 3.2. HyperVerse-Browser (Quelle: [Botev et al., 2008a])

Beim Entwurf von HyperVerse ist man davon ausgegangen, dass der HyperVerse-Client (HyperVerse-Browser) in Analogie zu den heutigen Internet-Browsern genutzt wird. Das heißt, ein Nutzer öffnet den Browser bei Bedarf und schließt ihn, sobald er sein Anliegen erledigt hat. Deshalb musste bei der Konstruktion des Client-Overlays die Restriktion berücksichtigt werden, dass ein einzelner Client evtl. nur eine relativ kurze Zeit online ist. Angesichts dessen würde die Verwendung eines strukturierten Client-Overlays einen hohen Rechen- und Kommunikationsaufwand verursachen. Aus diesem Grund basiert HyperVerse auf einer hierarchischen zweischichtigen Peer-to-Peer-Infrastruktur, die eine strukturierte *Public-Server*-Backbone-Schicht und eine unstrukturierte Client-Schicht umfasst. Die Welt-Metadaten und virtuellen Objekte in der HyperVerse-Umgebung werden initial von der Backbone-Schicht verbreitet. Ähnlich wie bei dem BitTorrent-Protokoll [Cohen, 2003] werden die Clients, die sich in einer bestimmten virtuellen Region (AoI) befinden, in die Verteilung von Weltdaten und Objekten involviert. Das im HyperVerse eingesetzte Public-Server-Overlay basiert auf dem strukturierten *GP3*-Protokoll [Esch et al., 2008a], welches eine effektive Datenverteilung ermöglicht und relativ geringe Konstruktionskosten aufweist [Esch et al., 2009b].

Die Knoten des Public-Server-Overlays teilen dabei die Verantwortung für die virtuellen Regionen der Umgebung auf. Tritt ein Client in eine neue Region ein, so sendet er eine Anfrage an das GP3-Public-Server-Backbone und bekommt alle relevanten Objekt- und Avatar-Informationen von dem zuständigen Public-Server. Ist die Avatar-Dichte in einer AoI groß genug (d.h. die AoI ist vollständig von anderen Avataren überdeckt), so erhält der entsprechende Client, der in eine solche Region eintritt, diese Informationen direkt von den anderen Clients, ohne das GP3-Backbone in Anspruch zu nehmen [Esch et al., 2009a]. Das Backbone wird dadurch spürbar entlastet. Es wird nur dann angesprochen, wenn die Daten nicht vollständig von anderen Clients geladen werden können.

Durch die Übertragung der Verwaltungsaufgaben an das Public-Server-Backbone bzw. an die Clients kann die HyperVerse-Infrastruktur hohe Nutzerzahlen bedienen, ohne dabei Skalierbarkeitsprobleme zu erfahren. Eine zu hohe Avatar-Dichte in einer AoI kann jedoch die Nutzbarkeit des Systems aus der Sicht eines einzelnen Nutzers enorm einschränken, weil die Umgebung unüberschaubar wird. Aus diesem Grund liegt ein weiterer Schwerpunkt, der vom HyperVerse-Team adressiert wird, auf der Informations-Aggregation, die auf der sozialen Nähe eines Nutzers bzw. Avatars basiert. Diese Aggregation ist notwendig, um in Regionen mit hoher Nutzerdichte eine zweckmäßige Avatar-Interaktion zu ermöglichen. Sobald die Avatar-Dichte in einer bestimmten Region der virtuellen Welt einen festgelegten Wert überschreitet, soll eine Aggregation der visuellen Darstellung anhand der sozialen Nähe eines Avatars ausgeführt werden. Dabei werden je nach Dichte weniger relevante Avatare ausgeblendet oder in einer „Bevölkerungswolke“ zusammengefasst. Eine direkte Interaktion mit anderen Avataren, die eine gewisse soziale Nähe zueinander aufweisen, wird somit wieder möglich.

Es ist äußerst schwierig, eine kritische Masse an reellen Nutzern zu erreichen oder für einen Feldversuch zu gewinnen, um die Durchführbarkeit der erarbeiteten Ansätze zu validieren. Deshalb wird das Nutzerverhalten oftmals mithilfe von entsprechenden Simulationsumgebungen simuliert. Anhand dieser Ergebnisse versucht man bestimmte Schlussfolgerungen über das Nutzer- und Systemverhalten zu ziehen. Das simulierte Nutzerverhalten unterscheidet sich jedoch oft von dem reellen Verhalten. Deshalb sind solche Schlussfolgerungen nicht immer objektiv. In [Botev et al., 2009] und [Botev et al., 2010] ist der Ansatz beschrieben, die virtuelle Umgebung HyperVerse und die Simulationsumgebung *TopGen* [Scholtes et al., 2008a] zu kombinieren. Auf diese Weise können reelle und simulierte Nutzer in einer Umgebung miteinander interagieren bzw. kommunizieren. Die so erzielten Simulationsergebnisse spiegeln das Nutzerverhalten viel realistischer wider.

Die Konsistenzaspekte, die im Zusammenhang mit HyperVerse betrachtet wurden, werden im Laufe dieser Arbeit in Kapitel 6 detailliert beschrieben. Weitere Aspekte, u.a. die Evaluation der HyperVerse-Infrastruktur, werden in [Botev et al., 2008b] ausführlich behandelt.

3.2 HERA-Framework

Die meisten der im Kapitel 2 bereits vorgestellten DVEs (HYMS, SimMud, etc., aber auch HyperVerse) basieren auf einem strukturierten Overlay, das deterministisch aufgebaut wird. Der deterministische Aufbau bedeutet, dass eine bestimmte, von der Anzahl der existierenden Overlay-Knoten abhängige Nachrichtenanzahl notwendig ist, bis das Beitreten (Join) zum Netzwerk bzw. das Verlassen (Leave) des Netzwerks abgeschlossen ist. In großen Overlays führt dies dazu, dass die Konstruktions- bzw. Instandhaltungskosten eines Overlays inakzeptabel hoch sind.

Im Gegensatz dazu basiert die Management-Schicht des HERA-Frameworks [Bettinger et al., 2009b] auf dem zufallsgesteuerten P-Grid-Ansatz, wodurch die Kosten für das Beitreten bzw. das Verlassen gering gehalten werden. Diese Tatsache ermöglicht auch ein effizientes Einfügen von zusätzlichen Overlay-Knoten in die stark beanspruchten virtuellen Regionen. Damit wird die Last effizient und dynamisch auf mehrere Knoten des Overlays verteilt. In deterministischen Overlays wäre dieses Einfügen mit einem enormen Nachrichtenaufwand verbunden. Die Gegenüberstellung zwischen einem deterministischen und einem zufallsgesteuerten strukturierten Overlay wird in Abschnitt 4.3 vorgestellt.

Während bei HyperVerse der Fokus u.a. auf eine ansprechende dreidimensionale Repräsentation sowie neuartige Anwendungsszenarien und innovative Fragestellungen (z.B. Informations-Aggregation) gelegt wurde, sollte das HERA-Framework eher einen prototypischen Charakter mit einer einfachen zweidimensionalen Darstellung haben, um bestimmte Algorithmen im Vorfeld in einer einfachen Umgebung implementieren und testen zu können, bevor sie ihren endgültigen Einsatz in HyperVerse finden. Das leichtgewichtige HERA-Framework wurde in einer Zusammenarbeit der Universität Trier und der Fachhochschule Trier entworfen und entwickelt. Das Framework ist modular und generisch aufgebaut. Dadurch können die eingesetzten Verfahren und Technologien unkompliziert durch andere ersetzt werden, ohne dabei das gesamte System anpassen zu müssen.

Das HERA-Framework basiert auf einer dreischichtigen Architektur (siehe Abbildung 3.3). Die oberste Schicht ist das so genannte *Application Layer*. Diese Schicht dient als Schnittstelle für die Implementierung von nutzerspezifischen Anwendungen, DVEs oder MMOGs. Gegenwärtig ist auf der Basis dieser Schnittstelle das zweidimensionale HERATRONos-Online-Spiel implementiert, welches im weiteren Verlauf dieses Kapitels in Abschnitt 3.3 näher betrachtet wird.

Die Anwendungen im HERA-Framework bauen auf einer Infrastrukturschicht, dem so genannten *Topology Layer*, auf. Diese Schicht ist für die Verwaltungsaufgaben, das heißt für das AoI-Management und die Verteilung von Weltdaten und Objekten, zuständig. Momentan basiert die HERATRONos-Spielanwendung auf dem zufallsgesteuerten Peer-to-Peer-Overlay P-Grid.

Anders als in HyperVerse sind HERA-Clients/Nutzer nicht an der Verteilung der Weltdaten beteiligt. Stattdessen ist das Client-Overlay für die Gruppenkommunikation und das Konsistenz-Handling zuständig. Dazu können die Clients auf das *Consistency Provider*-Modul des HERA-Frameworks zurückgreifen.

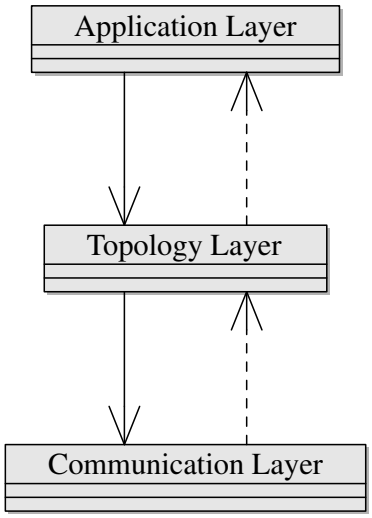


Abbildung 3.3. HERA-Architektur

Auf der untersten Ebene der HERA-Architektur befindet sich das so genannte *Communication Layer*, welches für die Kommunikation verschiedener Knoten untereinander und der dazugehörigen Datenübermittlung über das TCP-Protokoll verantwortlich ist. Bezogen auf den generischen Aufbau von HERA kann hier auch jedes andere Kommunikationsprotokoll verwendet werden.

Im Folgenden wird die HERATRONos-Spielanwendung vorgestellt, die auf dem HERA-Framework basiert.

3.3 HERATRONos-Spiel

Im Vorfeld der Entwicklung einer HERA-Anwendung wurden folgende Anforderungen gestellt, die diese Anwendung erfüllen sollte:

- Die Anwendungsinhalte (Informationen über Objekte und Avatare, die sich in der Umgebung befinden, etc.) sollten von einer Peer-to-Peer-Schicht verwaltet und verteilt werden (verteiltes AoI-Management).
- Die Anwendung sollte Szenarien unterstützen, bei denen ein konkurrierender Zugriff auf gemeinsam genutzte Objekte sowie auch die Wahrung eines gemeinsamen Zustandes eine Rolle spielen (clientbasiertes Konsistenz-Handling).

Um diesen Anforderungen gerecht zu werden, wurde das Multiplayer-Online-Spiel HERATRONos [Baumann, 2009] entwickelt und implementiert. Die Handlung in HERATRONos basiert auf dem klassischen *Tron*-Prinzip, welches 1982 von *Midway Games*[4] vorgestellt wurde.

Abbildung 3.4 zeigt einen Screenshot-Ausschnitt des Spielgeschehens. Die Spielumgebung besteht aus mehreren Räumen, die von den Knoten des Management-Overlays verwaltet werden. Ein Spieler wird durch seinen Avatar – ein futuristisches Fahrzeug – repräsentiert, das sich in der Spielumgebung nach links, rechts,

[4] `http://www.midway.com`

oben oder unten bewegen kann. Während seiner Bewegung zieht das Fahrzeug eine Mauer hinter sich hoch, die nun neben den bereits existierenden Raumwänden als ein weiteres Hindernis für alle Fahrzeuge in der Spielumgebung dient.

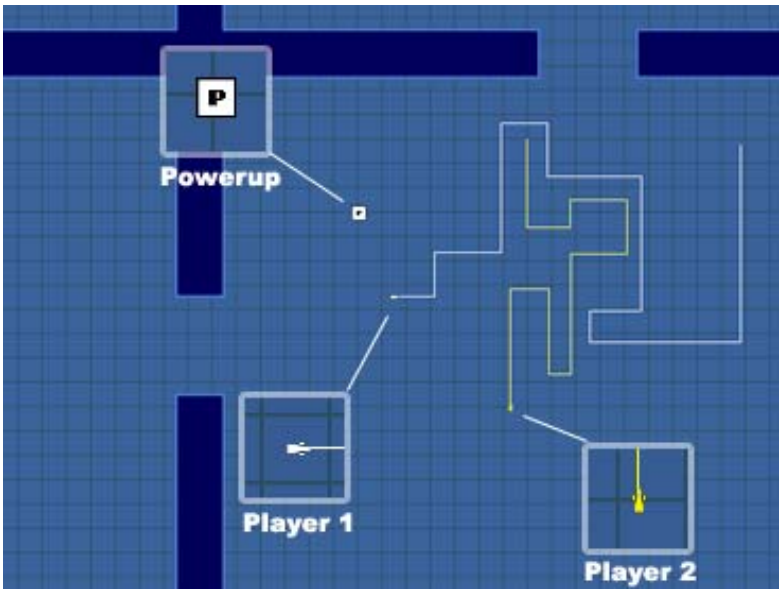


Abbildung 3.4. HERATRONos-Screenshot (Quelle: [Bettinger et al., 2009b])

Sobald mindestens zwei Spieler sich einloggen, beginnt das Spiel. Das Ziel des Spiels ist es die Bewegungsmöglichkeiten der Gegenspieler einzuschränken und somit eine Kollision mit einem Hindernis zu erzwingen. Eine Kollision führt zum sofortigen Verlieren des Spiels. Um möglichst lange im Spiel zu bleiben, muss ein Fahrzeug Kollisionen vermeiden und Hindernisse umfahren. Dazu kann es seine Fahrtrichtung in 90°-Schritten verändern. Der Spieler, der bis zuletzt Hindernisse umfahren kann, wird vom System zum Sieger erklärt. Spieler, die direkt gegeneinander antreten, müssen über konsistente Zustandsrepräsentationen verfügen, um eindeutig entscheiden zu können, wer zuerst eine Kollision mit einem Hindernis hatte. Die Positionen der Wände bzw. der Fahrzeuge stellen somit den gemeinsamen Zustand dar, der konsistent gehalten werden muss.

Um das Spielgeschehen zeitgemäß zu gestalten, wurden in HERATRONos einige neue Features eingeführt, die es in Tron noch nicht gab. Dazu zählen unter anderem positive und negative Power-Ups. Ein positives Power-Up kann beispielsweise einem Spieler dazu verhelfen, eine Wand „durchzubrechen“ und somit aus einer kritischen Situation herauszukommen. Negative Power-Ups wirken sich nachteilig auf das Spielverhalten der Spieler aus. Beispielsweise kann ein negatives Power-Up bewirken, dass das Fahrzeug sich zwei Sekunden lang nicht steuern lässt und nur geradeaus fährt, was die Wahrscheinlichkeit einer Kollision erhöht. Für die Spieler ist es äußerlich nicht ersichtlich, ob es sich bei einem Power-Up um ein positives oder negatives Power-Up handelt. Somit ist das Einsammeln von Power-Ups immer mit einem gewissen Risiko verbunden. Um die Konsistenz in der Umge-

bung nicht zu verletzen, soll vermieden werden, dass mehrere Avatare gleichzeitig ein und dasselbe Power-Up-Objekt aufheben. Das heißt, ein konkurrierender Zugriff auf gemeinsam genutzte Objekte (Power-Ups) soll vorzugsweise unter einem wechselseitigen Ausschluss stattfinden.

Nach der Anmeldung wird das Fahrzeug eines Spielers in einem zufällig gewählten Raum platziert. Ein Raum verfügt über vier Ausgänge: jeweils ein Ausgang in jede Bewegungsrichtung. Der Spieler muss nicht die ganze Spieldauer in seinem Raum bleiben, sondern kann den Raum (soweit möglich) durch seine Ausgänge verlassen. Da ein Spieler keine Informationen über das Spielgeschehen in den angrenzenden Räumen besitzt, kann ein Raumwechsel durchaus mit Gefahren verbunden sein. Beispielsweise kann die Bewegungsfreiheit in dem angrenzenden Raum bereits durch die Hindernisse der Gegenspieler eingeschränkt sein. Sind alle Raumausgänge durch Hindernisse versperrt, ist ohne den Einsatz von positiven Power-Ups kein Ausweg mehr möglich. In so einem Fall muss der Spieler versuchen, durch geschickte Bewegungsabläufe möglichst lange eine Kollision hinauszuzögern, um evtl. das Spiel doch noch gewinnen zu können.

Im weiteren Verlauf dieser Arbeit werden die in diesem Abschnitt angesprochenen Aspekte detailliert betrachtet. Das Overlay-basierte AoI-Management in HERA wird in Kapitel 4 diskutiert. Dort wird unter anderem erläutert, wie die Zuordnung von HERATRONos-Räumen zu den Knoten des Management-Overlays abläuft. Die clientbasierte Kommunikation in HERA wird in Kapitel 5 besprochen. Dabei werden zwei während dieser Arbeit entwickelte Multicast-Ansätze vorgestellt. Schließlich wird in Kapitel 6 das clientbasierte Konsistenz-Handling in HERA, dem ein neuartiges Konsistenzmodell zu Grunde liegt, beschrieben.

4

Overlay-basiertes AoI-Management in HERA

Die relevanten Informationen einer virtuellen Umgebung sind im Wesentlichen auf die Metadaten und Eigenschaften der dynamischen Objekte, das heißt der Gegenstände (Häuser, Bäume, Power-Ups, etc.) sowie der Avatare, beschränkt. Viele DVEs bieten den Nutzern die Möglichkeit, die Umgebung mitzugestalten, indem sie es erlauben, die bereits existierenden Objekte zu verändern. Des Weiteren können Nutzer Gegenstände selbst generieren und in der Umgebung platzieren. Diese benutzergenerierten Inhalte (engl. *User Generated Content*) verleihen einer virtuellen Umgebung eine gewisse Dynamik. Deshalb macht es keinen Sinn, die komplette virtuelle Umgebung beim Erwerb auf einem Medium auszuliefern. Besser ist es, die erforderlichen Informationen effizient und für den Nutzer transparent bei Bedarf auf den Client-Rechner zu laden. Tritt ein Nutzer mit seinem Avatar in eine neue virtuelle Region – AoI – ein, müssen ihm bereits beim Eintritt die entsprechenden relevanten Objekt-Informationen vorliegen, damit keine Lücken in der Darstellung entstehen können. Also ist es erforderlich, die Objekt-Informationen aus der neuen AoI im Voraus effizient erfragen zu können. Daraus ergibt sich die folgende Anforderung an die Basis-Infrastruktur:

Anforderung 4.1 (Effiziente Informationsbeschaffung). Um in einer virtuellen Umgebung eine lückenlose Darstellung zu ermöglichen, ist ein effizienter Zugriff auf relevante Informationen sowie eine effiziente Veröffentlichung von neuen Informationen erforderlich. Die entsprechende Funktionalität muss von der Basis-Infrastruktur zur Verfügung gestellt werden.

In einer Client-Server-Infrastruktur ist diese Anforderung ohne Weiteres effizient realisierbar. Wenn eine AoI eindeutig identifizierbar ist, dann stellt der Client eine Anfrage an den Server und übergibt dabei die entsprechende AoI-ID als Argument. Daraufhin liefert der Server die entsprechenden Informationen. Die Laufzeitkomplexität beträgt dabei gemäß der *O-Notation* $O(1)$.

In einem verteilten Peer-to-Peer-Overlay werden die einzelnen Regionen auf die zuständigen Knoten des Overlays abgebildet. Clients, die eine neue AoI betreten, senden eine Anfrage an das Management-Overlay und bekommen alle relevanten Informationen von dem Overlay-Knoten, der für diese AoI zuständig ist. Verständlicherweise können das Speichern/Veröffentlichen, das Zugreifen und das Löschen der generierten Inhalte verteilt in einem Peer-to-Peer-Overlay nicht so effizient wie

in einer Client-Server-Infrastruktur ausgeführt werden. Die Laufzeitkomplexität $O(log(n))$, die für das Speichern von Informationen (Publish), zum Zugriff auf Informationen (Lookup) und zum Löschen von Informationen (Delete) in einem Overlay-Netzwerk anfällt, ist jedoch akzeptabel. Deshalb wird in dieser Arbeit die Meinung vertreten, dass strukturierte Overlay-Netzwerke aufgrund einer besseren Skalierbarkeit im Vergleich zu Client-Server-Ansätzen als DVE-Basis-Infrastruktur (Management-Schicht) dienen können. Allerdings muss in diesem Fall abgewogen werden, mit welchen Konstruktionskosten die Sucheffizienz erkauft wird.

Aus einer Vielzahl von Overlays wurden Tapestry und P-Grid für den Vergleich in die engere Wahl gezogen. Diese wurden aufgrund ihrer gegensätzlichen Natur ausgewählt, um die Vor- bzw. Nachteile der deterministischen und zufallsgesteuerten Ansätze zu unterstreichen. Das etablierte Tapestry-Overlay dient bereits erfolgreich als Basis-Infrastruktur für das OceanStore-Speichersystem. P-Grid sticht durch seine zufallsgesteuerte Struktur, die anders als die meisten Overlays auf dem Prinzip des Präfixbaums basiert, hervor. Die Evaluation von Tapestry und P-Grid sollte auf der Basis einer Netzwerk-Topologie simuliert werden, die das Internet möglichst gut modelliert. Deshalb wurde die Entscheidung getroffen, die Simulationsumgebung *SimCon* [Esch et al., 2008b] auf Basis des Topologie-Generators TopGen zu entwickeln. Die Simulationsumgebung SimCon wird in Abschnitt 4.1 kurz beschrieben.

Um eine objektive Entscheidung treffen zu können, welches Overlay die Anforderungen einer virtuellen Umgebung am besten erfüllt, werden diese Overlays bezüglich aussagekräftiger Metriken (Konstruktionskosten, Publish/Lookup-Performanz) miteinander verglichen. Diese Metriken werden in Abschnitt 4.2 definiert.

Die gewonnenen Evaluationsergebnisse sollen eine Overlay-Infrastruktur identifizieren, die für den Einsatz als Management-Schicht, zu deren Aufgaben u.a. die Verwaltung und Verteilung von Weltdaten zählt, am besten geeignet ist. Die identifizierte Infrastruktur soll nach Möglichkeit den Spagat zwischen geringen Konstruktionskosten und einer akzeptablen Publish- und Lookup-Funktionalität schaffen. Die Evaluationsergebnisse werden in Abschnitt 4.3 präsentiert und interpretiert.

Die Frage, die in diesem Kapitel behandelt wird, lautet: Wie kann das AoI-Management von einer Peer-to-Peer-Infrastruktur bewerkstelligt werden? Als Antwort auf diese Frage wird das verteilte AoI-Management, das im HERA-Framework implementiert ist, in Abschnitt 4.4 vorgestellt. Hier ist es erforderlich zu klären, wie die Zuordnung von Objekten und Avataren umgesetzt wird bzw. wie die Zuständigkeiten für die Weltregionen unter den Knoten des Management-Overlays aufgeteilt werden. Die in diesem Kapitel gewonnenen Erkenntnisse werden in Abschnitt 4.5 zusammengefasst.

4.1 SimCon-Simulator

Bei der Vielzahl existierender Peer-to-Peer-Ansätze ist es notwendig genauere Untersuchungen anzustellen, um eine für DVEs am besten geeignete Peer-to-Peer-Infrastruktur zu identifizieren. In Abschnitt 2.2.2 wurden bereits einige Simulationsumgebungen für Overlay-Netzwerke vorgestellt. Um möglichst realitätsnahe Simulationsergebnisse bezüglich der Latenz gewinnen zu können, wurde im Rahmen dieses Promotionsvorhabens eine Simulationsumgebung namens SimCon auf der Basis des Topologie-Generators TopGen entwickelt. Mithilfe von SimCon können Overlay-Netzwerke auf ihre Eignung für bestimmte Anwendungsszenarien evaluiert werden.

SimCon ist generisch aufgebaut und verfügt über folgende vier Programmierschnittstellen:

- `IApp` – stellt die Schnittstelle für die Entwicklung von Anwendungsszenarien dar, die auf der Basis einer Overlay-Implementierung laufen sollten.
- `ITopology` – ist die Schnittstelle für die Overlay-Implementierungen.
- `IRecorder` – bietet eine Schnittstelle zum Erfassen von Evaluations- und Simulationsergebnissen an.
- `IView` – ist die Schnittstelle zur Overlay-Visualisierung.

Aufgrund des generischen Aufbaus von SimCon können unterschiedliche Module, die das Verhalten einer Anwendung emulieren, auf Basis der IApp-Schnittstelle implementiert werden. Diese Module können dann auf den unterschiedlichen Overlay-Netzwerken, die auf der ITopology-Schnittstelle basieren, ausgeführt werden. Bei der Ausführung können die benutzerdefinierten IRecorder-Implementierungen das Anwendungsverhalten aufzeichnen und persistent abspeichern, während die Visualisierungskomponente (IView) den entsprechenden Ablauf graphisch darstellt.

Bei der Evaluation definiert man aussagekräftige Metriken, mit deren Hilfe die relevanten Eigenschaften von Overlays beschrieben werden können. Diese Metriken werden dann im Laufe der Simulationen erfasst und anschließend ausgewertet. Durch die Auswertung der Evaluationsergebnisse ist es möglich zu entscheiden, welche Overlay-Implementierung für das berücksichtigte Anwendungsszenario am besten geeignet ist.

Abbildung 4.1 zeigt die Benutzeroberfläche des SimCon-Simulators. Diese Oberfläche besteht aus zwei Teilen: einem Managementframe und einem Visualisierungsframe. Über den Managementframe können die oben besprochenen Module geladen sowie die Anzahl der Overlay-Knoten und die Simulationsdauer festgelegt werden.

Im ersten Simulationsschritt wird der Netzwerk-Graph geladen (Abbildung 4.1 (1)). Um möglichst realitätsnahe Simulationsergebnisse bezüglich der Latenz zu gewinnen, verwendet SimCon die Internet-Graphen von TopGen.

Danach muss das Anwendungsmodul geladen werden (Abbildung 4.1 (2)). Die Anwendung, die auf der Abbildung dargestellt wird, besteht darin, in einem Overlay-Netzwerk Objekte zu veröffentlichen (Publish) und zu suchen (Lookup).

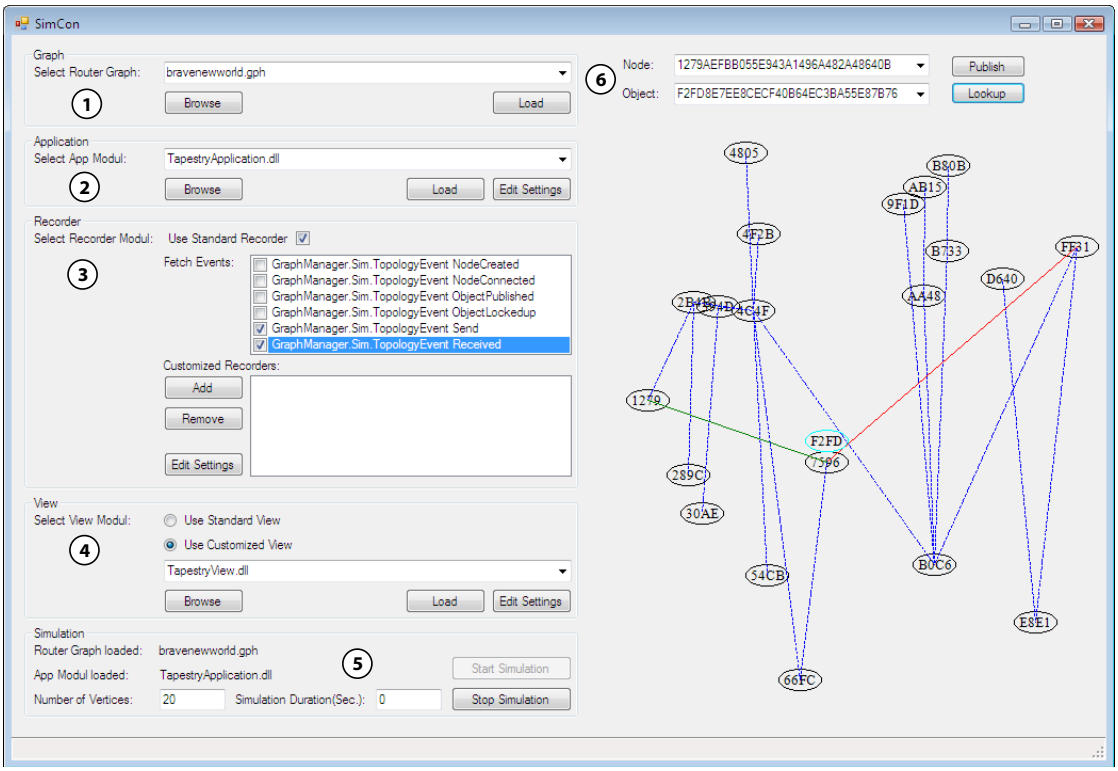


Abbildung 4.1. Simulationsumgebung SimCon

Jede Anwendung kann bestimmte (benutzerdefinierte) Ereignisse auslösen, die von dem Standard-Rekorder und/oder von weiteren benutzerdefinierten Rekordern erfasst werden. Die Ereignisse, die erfasst werden sollen, können auf der Oberfläche durch Anklicken ausgewählt werden (Abbildung 4.1 (3)).

Des Weiteren kann ein Visualisierungsmodul geladen werden (Abbildung 4.1 (4)). Die abgebildete Visualisierungskomponente ist sehr einfach, sie beinhaltet zwei Auswahllisten, zwei Buttons und eine Zeichenfläche (Abbildung 4.1 (6)). Im Kontext der Evaluation ist die Visualisierung zweitrangig, im Kontext der Lehre kann ein Visualisierungsmodul jedoch sehr sinnvoll sein.

Nachdem alle Simulationsmodule ausgewählt und die nötigen Voreinstellungen vorgenommen wurden, kann die Anzahl der zu simulierenden Overlay-Knoten sowie die Simulationsdauer festgelegt werden (Abbildung 4.1 (5)).

Nach Ablauf der Simulation werden alle Ergebnisse persistent auf die Festplatte geschrieben und können anschließend interpretiert werden.

Eine Gegenüberstellung und Evaluation der Overlay-Netzwerke Tapestry und P-Grid mittels SimCon ist in [Bettinger et al., 2009a] detailliert beschrieben. Im Folgenden werden die verwendeten Metriken und die wichtigsten Erkenntnisse, die aus deren Evaluation gewonnen wurden, präsentiert.

4.2 Metriken

Um einen aussagekräftigen Vergleich zwischen Tapestry und P-Grid durchführen zu können, müssen zuerst passende Metriken definiert werden, welche objektive Bewertungen bezüglich der Lookup-Performanz, Konstruktions- und Kommunikationskosten und Speicheranforderungen liefern.

Die Metrik $M(N)$ stellt die Anzahl der Nachrichten dar, die notwendig sind, um einem Overlay-Netzwerk mit N Knoten beizutreten. Weiterhin wird die relative Anzahl an Nachrichten, die für den Beitritt zu einem Netzwerk mit N Knoten anfällt, definiert als:

$$A(N) = \frac{M(N)}{N}$$

Die nächste Metrik $MEM(N)$ repräsentiert die durchschnittliche Anzahl von Routing-Tabellen-Einträgen über alle Knoten in einem Netzwerk. Die Routing-Tabellen von Tapestry- und P-Grid-Knoten enthalten l Stufen, die ihrerseits eine Menge von Einträgen s_l enthalten. Sei nun s_j^i eine Menge von Einträgen auf der j-ten Stufe des i-ten Knotens. Dann wird $MEM(N)$ wie folgt definiert:

$$MEM(N) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{l_i} |s_j^i|}{N}$$

Um eine bessere Vergleichbarkeit der Publish- und Lookup-Operationen von P-Grid und Tapestry zu erzielen, werden die Hop-Anzahl und die Latenz nach dem Veröffentlichen bzw. Aufsuchen von mehreren Objekten ermittelt. Der Einfluss der Zufallsfaktoren im Vergleich zum Veröffentlichen bzw. Aufsuchen eines einzigen Objektes wird dabei reduziert.

Die Metriken $P(O, N)$ bzw. $L(O, N)$ repräsentieren die Anzahl von Hops, die für das Veröffentlichen (Publish) bzw. das Auffinden (Lookup) von O Objekten in einem Netzwerk der Größe N notwendig sind.

Zur Messung der Latenz wird die Metrik $P_{Lat}(O, N)$ definiert, welche die Zeit in Millisekunden darstellt, die für das Veröffentlichen von O Objekten in einem Netzwerk der Größe N verstreicht. Sei M_{P_O} die Nachrichtenmenge, welche für das Veröffentlichen von O Objekten in einem Netzwerk der Größe N notwendig ist und $Lat(m)$ ($m \in M_{P_O}$) die Latenz einer einzigen Nachrichtenübertragung. $P_{Lat}(O, N)$ wird definiert als:

$$P_{Lat}(O, N) = \sum_{m \in M_{P_O}} Lat(m)$$

Analog dazu wird die Metrik $L_{Lat}(O, N)$ für das Auffinden von Objekten definiert:

$$L_{Lat}(O, N) = \sum_{m \in M_{L_O}} Lat(m)$$

wobei M_{L_O} die Nachrichtenmenge, welche für das Veröffentlichen von O Objekten notwendig ist, darstellt.

Darüber hinaus werden die durchschnittlichen Kosten, die für das Veröffentlichen und Aufsuchen von O Objekten in einem Netzwerk der Größe N anfallen, definiert als

$$A(P_{Lat}(O,N)) = \frac{\sum_{m \in M_{P_O}} Lat(m)}{|M_{P_O}|}$$

und

$$A(L_{Lat}(O,N)) = \frac{\sum_{m \in M_{L_O}} Lat(m)}{|M_{L_O}|}.$$

Die entsprechenden Latenzinformationen werden aus dem Internet-Graphen entnommen, der von TopGen generiert und als Netzwerkmodul in SimCon eingebunden wurde. Die gewonnenen Simulationsergebnisse werden im nächsten Abschnitt diskutiert.

4.3 Evaluationsergebnisse

In der ersten Simulationsphase wurden Tapestry- und P-Grid-Overlays nacheinander mit 100, 300, 500, 1000 und 2000 Knoten konstruiert und die Metriken $M(N)$, $A(N)$, $P(O,N)$, $L(O,N)$ und $MEM(N)$ erfasst. Dabei wurden jeweils 20 Publish- und Lookup-Operationen durchgeführt.

Wie die Zahlen in den Tabellen 4.1 und 4.2 zeigen, steigt die Anzahl der Nachrichten $M(N)$, die zur Konstruktion eines Tapestry-Overlays benötigt werden, wesentlich stärker an als die zur Konstruktion eines P-Grids (siehe Abbildung 4.2). Ein P-Grid-Knoten benötigt eine geringe und nahezu konstante durchschnittli-

N	M(N)	A(N)	P(20, N)	L(20, N)	MEM(N)
100	1701	17.01	63	60	65
300	8976	29.92	64	60	82
500	20280	40.56	69	61	93
1000	51923	51.92	75	62	128
2000	118160	59.08	83	64	196

Tabelle 4.1. Evaluationsergebnisse: Tapestry

che relative Nachrichtenanzahl $A(N)$, um einem Netzwerk beizutreten. Während seiner Lebenszeit tauscht sich dieser Knoten immer wieder mit anderen Knoten aus und spezialisiert dadurch seinen Zuständigkeitsbereich. Beim Tapestry-Overlay werden weniger Nachrichten für das Auffinden der 20 Objekte $L(20,N)$ als für deren Veröffentlichung $P(20,N)$ benötigt. Dies wird darauf zurückgeführt, dass Tapestry-Knoten Objektzeiger speichern und so ein Zeiger oftmals gefunden wird, bevor die Anfrage den Objekt-Root-Knoten erreicht (siehe Abschnitt 2.2.1). Die

N	M(N)	A(N)	P(20, N)	L(20, N)	MEM(N)
100	694	6.94	64	58	81
300	2094	6.98	71	71	238
500	3494	6.99	75	79	390
1000	6994	6.99	65	66	789
2000	13994	6.99	88	92	1549

Tabelle 4.2. Evaluationsergebnisse: P-Grid

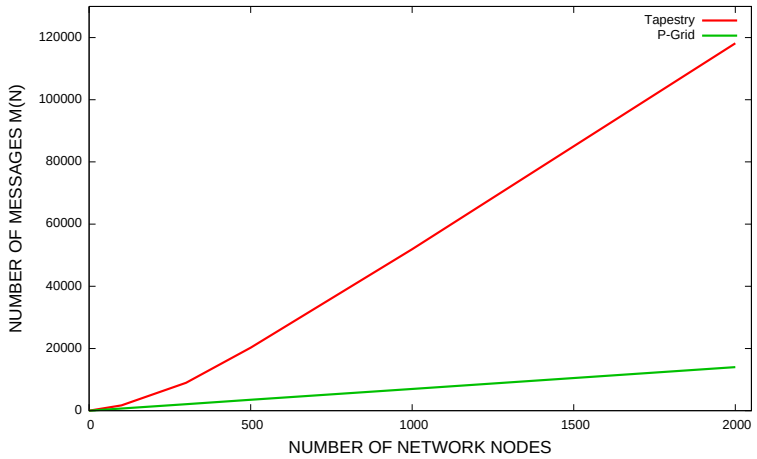


Abbildung 4.2. Gegenüberstellung von M(N)

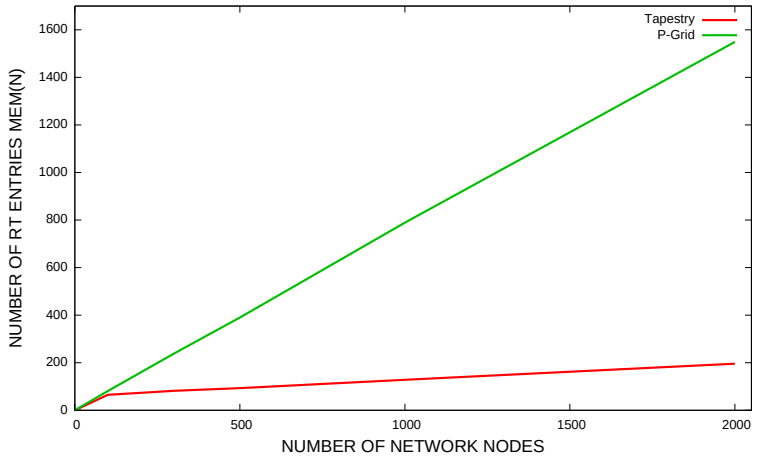


Abbildung 4.3. Gegenüberstellung von MEM(N)

Anzahl der Routing-Tabellen-Einträge $MEM(N)$ im Tapestry-Netzwerk ist geringer als in P-Grid (siehe Abbildung 4.3).

In der nächsten Simulationsphase wurden Tapestry- und P-Grid-Overlays bezüglich der Latenz-Metriken $P_{Lat}(O, N)$, $L_{Lat}(O, N)$, $A(P_{Lat}(O, N))$ und $A(L_{Lat}(O, N))$ verglichen. Dabei wurden beide Netzwerke mit jeweils 1000 Knoten erzeugt und jeweils 1000 Objekte veröffentlicht und anschließend gesucht. Die Nachrichtenpfade wurden aufgezeichnet und anhand dieser Aufzeichnungen die Latenzkosten ermittelt (siehe Tabelle 4.3).

Metrik	P-Grid	Tapestry
$P_{Lat}(1000, 1000)$	248302.93	106011.25
$\|M_{P_{1000}}\|$	4245	2442
$A(P_{Lat}(1000, 1000))$	58.49	43.41
$L_{Lat}(1000, 1000)$	257540.89	103372.25
$\|M_{L_{1000}}\|$	4281	2525
$A(L_{Lat}(1000, 1000))$	60.16	40.94

Tabelle 4.3. Latenzkosten

Wie die Zahlen zeigen, verhält sich Tapestry in Bezug auf Latenzzeiten besser als das P-Grid-Overlay, das die Knoten-Lokalität im physikalischen Netzwerk nicht berücksichtigt. Weil der P-Grid-Ansatz, aufgrund seiner zufallsgesteuerten Struktur, einen aus der Sicht der DVEs besseren Trade-off zwischen Sucheffizienz und Konstruktionskosten darstellt, fiel letztendlich die Entscheidung zu Gunsten des P-Grid-Ansatzes aus. Die erste Umsetzung eines P-Grid-basierten Avatar- und Objekt-Managements in der HERATRONos-Spielanwendung wird im folgenden Abschnitt 4.4 diskutiert.

4.4 Overlay-basiertes AoI-Management

In diesem Abschnitt wird das P-Grid-basierte AoI-Management der HERATRONos-Spielanwendung beschrieben. Dabei wird auch explizit darauf eingegangen, wie die Publish- und Lookup-Funktionalität eines Overlays beim AoI-Management eingesetzt werden kann bzw. wie die Zuordnung von Power-Up-Objekten und Avataren an den zuständigen Overlay-Knoten abläuft.

Die HERATRONos-Spielwelt wird vom ersten Overlay-Knoten anhand der getroffenen Einstellungen generiert. Wenn ein Overlay-Knoten beim Startvorgang seine Kontaktknoten nicht erreichen kann, wird der Nutzer über das interaktive Menü dazu aufgefordert, die Welt selbst zu initialisieren. Eine der möglichen Einstellungen, die vorgenommen werden können, ist zum Beispiel die Anzahl der Räume, die innerhalb der virtuellen Umgebung generiert werden sollen. Jeder Raum enthält Power-Up-Objekte, welche nach dem Zufallsprinzip im Raum verteilt werden.

Der Knoten, der die Spielumgebung initialisiert, ist von jetzt an für die gesamte Welt, das heißt für alle Objekte und beitretende Avatare, zuständig (vergleichbar

mit einem einzigen Server). Der binäre Pfad dieses Knotens ist noch leer. Wenn ein zweiter Knoten dem Netzwerk beitritt, führt er eine *Spezialisierung* mit dem bereits existierenden Knoten gemäß dem P-Grid-Algorithmus durch. Bei der Spezialisierung erweitern beide Knoten den gemeinsamen Pfad (hier ist der gemeinsame Pfad leer) um eine 0 bzw. 1, aktualisieren ihre Routing-Tabellen, teilen die Zuständigkeitsbereiche unter sich auf und melden die eventuell angemeldeten Objekte, die jetzt in einen anderen Zuständigkeitsbereich fallen, um. Jeder der beiden Knoten ist nun für seinen Teil des Suchraums verantwortlich. Da möglicherweise ein anderer Knoten bereits einen identischen Pfad wie der nun neu hinzugekommene besitzt, findet zwischen diesen beiden ein weiterer Austausch gemäß dem P-Grid-Algorithmus statt und der gemeinsame Schlüsselraum wird weiter unterteilt. Die exakte Definition der *Spezialisierungs-Prozedur* ist in [Aberer et al., 2002] angegeben. Abbildung 4.4 (Quelle: [Bettinger et al., 2009b]) zeigt ein Beispiel, in dem die Knoten A und B die Gesamtverantwortung für die Weltobjekte aufteilen – genauer gesagt, der Knoten A überträgt einen Teil der Verantwortung an den Knoten B –, indem sie ihre Pfade auf 0 bzw. 1 erweitern. Dabei entstehen zwei AoIs, die von nun an von den Knoten A und B verwaltet werden.

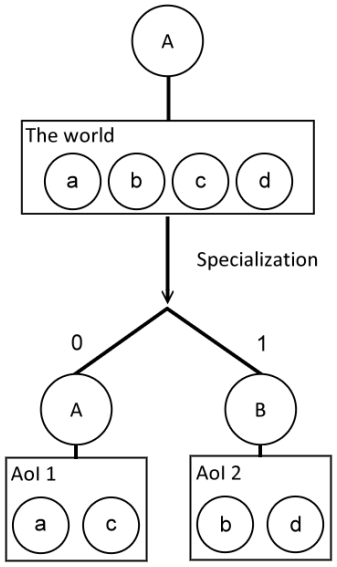


Abbildung 4.4. Aufteilung der Verantwortung nach der Spezialisierung

Power-Up-Objekte und Avatare in der HERATRONos-Spielanwendung werden durch ihre Kennung (Objekt-ID) identifiziert, die anhand der aktuellen Position, d.h (x,y)-Koordinaten mit dem Wertebereich $0 \leq x, y \leq 999$, berechnet wird. Die Zuordnung von Power-Up-Objekten und Avataren an den zuständigen Overlay-Knoten findet auf der Basis dieser koordinatenabhängigen Objekt-ID statt. Abbildung 4.5 (Quelle: [Bettinger et al., 2009b]) zeigt ein Beispielszenario, in dem die Management-Knoten A, B und C die Spielwelt sowie die Avatare a, b, c und d verwalten. Dabei ist der Knoten A für die Verwaltung der linken Bildschirmhälfte (Avatare a und b) zuständig. Die Knoten B und C teilen die Verwaltung der rech-

ten Bildschirmhälfte derart unter sich auf, dass der Knoten B für die obere Hälfte (Avatar c) und der Knoten C für die untere Hälfte (Avatar d) zuständig sind.

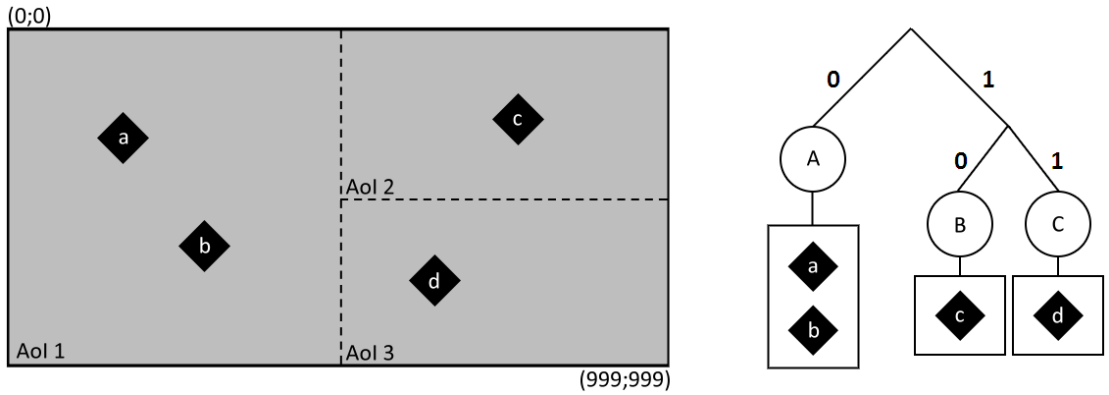


Abbildung 4.5. Koordinatenabhängiges AoI-Management

Die Abbildung von (x,y)-Koordinaten auf eine binäre Objekt-ID erfolgt folgendermaßen: Die Koordinaten, die weniger als drei Stellen aufweisen, werden mit den führenden Nullen ergänzt (z.B. $x = 5, y = 24 \Rightarrow x = 005, y = 024$). In den nachfolgenden Schritten werden abwechselnd die x- und y-Koordinaten beginnend mit dem ersten Zeichen betrachtet und die ID anhand der entsprechenden Wertigkeiten aufgebaut (siehe Listing 4.1). Durch diese Abbildung wird erreicht, dass die Objekte, die in derselben AoI liegen, auch auf denselben verantwortlichen Knoten abgebildet werden.

```
1  ...
2  String id = "";
3  if(x[0] < 5) id+="0"; else id+="1";
4  if(y[0] < 5) id+="0"; else id+="1";
5  if(x[1] < 5) id+="0"; else id+="1";
6  if(y[1] < 5) id+="0"; else id+="1";
7  if(x[2] < 5) id+="0"; else id+="1";
8  if(y[2] < 5) id+="0"; else id+="1";
9  ...
```

Listing 4.1. Abbildung von x- und y-Koordinaten auf eine binäre ID

Um die Veröffentlichung des Objektes im Overlay zu veranlassen, wird die Publish-Methode des P-Grid-Overlays aufgerufen. Da das Objekt anhand seiner Koordinaten identifiziert wird, fällt es in den Zuständigkeitsbereich des Knotens, der für die AoI, in der sich das Objekt befindet, zuständig ist. Somit muss der zuständige Overlay-Knoten das neue Objekt nur bei sich selbst veröffentlichen, da das Objekt sich in seinem Zuständigkeitsbereich befindet. Bei der Spezialisierung kann diese Zuständigkeit jedoch an einen anderen Knoten übertragen werden.

Aufgrund von Bewegungsabläufen verändert ein Avatar in der HERATRONos-Spielanwendung ständig seine Position, was gelegentlich zu einem Wechsel der AoI

und somit des Zuständigkeitsbereichs eines Overlay-Knotens führt. Um Verzögerungen in der Darstellung zu vermeiden und den Spielfluss nicht zu unterbrechen, müssen die Informationen über die Objekte und Avatare, die sich in der neuen AoI befinden, im Voraus beim Überschreiten einer Schwelldistanz zur AoI-Grenze angefordert werden. Dabei wird eine AoI-Lookup-Aufforderung von dem entsprechenden Client an den gegenwärtig zuständigen Overlay-Knoten versendet. Der zuständige Overlay-Knoten leitet die Anfrage anhand der AoI-Koordinaten, genauer gesagt der daraus berechneten AoI-ID, gemäß dem P-Grid-Algorithmus weiter. Der Client bekommt die angeforderten Objekt- und Avatar-Informationen von dem Knoten, der für die Verwaltung der neuen AoI zuständig ist. Nach dem Erhalt der neuen Informationen baut der Client Verbindungen zu allen in der AoI befindlichen Clients auf, um eine direkte Kommunikation zu ermöglichen. In der nächsten Version der HERATRONos-Anwendung soll diese Kommunikation über eine Multicast-Infrastruktur erfolgen (siehe Kapitel 5).

Das in diesem Abschnitt beschriebene AoI-Management wurde in der HERATRONos-Spielanwendung implementiert und im praktischen Einsatz getestet. Für weitere Details wird an dieser Stelle auf [Baumann, 2009] verwiesen.

4.5 Schlussfolgerung

Wie die Evaluationsergebnisse in Abschnitt 4.3 zeigen, verhält sich Tapestry aufgrund seiner deterministischen Struktur effizienter als P-Grid in Bezug auf den Speicherbedarf und die Publish- bzw. Lookup-Performanz bzw. Latenz. Somit ist Tapestry für die Anwendungsszenarien, in denen das aufgebaute Overlay über längere Zeiträume unverändert bleibt, besser geeignet als P-Grid.

In Peer-to-Peer-basierten virtuellen Umgebungen kann es dazu kommen, dass eine virtuelle Region zu sehr von den Benutzern beansprucht wird. Aufgrund der zu hohen Belastung kann der dafür vorgesehene Knoten diese Region nicht mehr allein verwalten. Hier ist es erforderlich, zusätzliche Ressourcen in Form von einigen Overlay-Knoten der Basis-Infrastruktur auf einen Schlag und ohne großen Konstruktionsaufwand hinzuzufügen. Hierfür ist P-Grid aufgrund seiner zufallsgesteuerten Struktur, die verhältnismäßig geringe Konstruktionskosten erfordert, besser geeignet als Tapestry.

Da außerdem in P-Grid das Beitreten zum Netzwerk bzw. das Verlassen des Netzwerks (im Gegensatz zu DHT-basierten Pendants) keinen hohen Berechnungs- bzw. Kommunikationsaufwand zur Folge hat, wird in dieser Arbeit das P-Grid-Overlay als Basis-Infrastruktur für das HERA-Framework favorisiert. Die Wahl des zufallsgesteuerten P-Grid-Ansatzes bringt die gewünschte Verbesserung der Skalierbarkeit und der Lastverteilung im Vergleich zu einer Client-Server-Infrastruktur mit sich, weil jeder Knoten im Overlay nur für die Verwaltung einer ihm zugeteilten virtuellen Region zuständig ist.

Die in Abschnitt 4.4 vorgestellte koordinatenbasierte Methode des Mappings von Objekten und Avataren auf den zuständigen Knoten des Overlays zeigt, wie das AoI-Management von einem Peer-to-Peer-Overlay umgesetzt werden kann.

5

Clientbasierte Multicast-Kommunikation in HERA

Bei einem Infrastrukturwechsel von Client-Server-basierten zu Peer-to-Peer-basierten Ansätzen muss das Augenmerk nicht nur auf die Management-Schicht gelegt werden, sondern auch auf die Client-Schicht. Es muss also überlegt werden, auf welche Art und Weise die Clients miteinander verbunden werden, welche Einschränkungen aus der Dynamik der Client-Schicht resultieren und welche Funktionalität in der Client-Schicht erforderlich ist.

Im Gegensatz zu den Knoten des Management-Overlays, die ziemlich lange online bleiben und nur sehr selten das Overlay verlassen, sieht es mit den Clients anders aus. Zum einen können die Clients mehrmals täglich on- und offline gehen. Zum anderen sind mit einer kontinuierlichen Bewegung eines Avatars auch häufige AoI-Wechsel verbunden. Aufgrund dieser Dynamik kommt der Einsatz eines strukturierten Overlays für die Clients nicht in Frage, weil die hohe Fluktuationsrate die Konstruktions- bzw. die Instandhaltungskosten eines solchen Overlays in die Höhe treiben würde. Der Einsatz eines nicht-strukturierten Peer-to-Peer-Overlays in der Client-Schicht würde keine hohen Konstruktionskosten nach sich ziehen und eine direkte Kommunikation bzw. Interaktion zwischen Clients in einer AoI ermöglichen, was zu einer weiteren Entlastung des Management-Overlays führen würde.

Welche Funktionalität ist nun in der Client-Schicht erforderlich? Wenn sich ein Client in einer dicht besiedelten virtuellen Region fortbewegt, ist es wesentlich effizienter, seine Zustandsänderungen mittels einer Multicast-Infrastruktur an die Clients in derselben AoI zu propagieren, als sie an jeden Client einzeln zu versenden. Dasselbe gilt auch für jegliche andere Art der Gruppenkommunikation innerhalb einer AoI. Daraus resultiert die folgende Anforderung:

Anforderung 5.1 (Effiziente Gruppenkommunikation). Um eine effiziente Gruppenkommunikation innerhalb einer AoI zu ermöglichen, beispielsweise um Zustandsänderungen effizient an die Clients in derselben AoI propagieren zu können, soll eine Anwendungsschicht-Multicast-Unterstützung in der Client-Schicht bereitgestellt werden.

Die Verwendung eines Multicast-Protokolls anstelle eines Unicast-Protokolls minimiert die Netzbelastung. Jedoch ist die Konstruktion einer Multicast-Infrastruktur immer zwangsläufig mit einem zusätzlichen Berechnungs- und Kommuni-

kationsaufwand verbunden (siehe Abschnitt 2.3). Der Zugewinn an Effizienz muss gegen diesen Mehraufwand abgewogen werden, um entscheiden zu können, ob sich der Einsatz einer Multicast-Infrastruktur für das entsprechende Anwendungsszenario lohnt. Für eine objektive und qualitative Bewertung sind Kenngrößen und Metriken notwendig, die auch einen Vergleich unterschiedlicher Infrastrukturen erlauben. In Abschnitt 5.1 werden solche Metriken hergeleitet und definiert.

Bei der Realisierung eines Multicast-Protokolls in Peer-to-Peer-Systemen existieren einige Besonderheiten, die beachtet werden müssen. In Peer-to-Peer-Systemen kennt ein Knoten in der Regel nicht alle anderen Knoten, sondern nur eine Teilmenge davon. Außerdem sind die Entfernungen zwischen den Knoten unbekannt. In Abschnitt 5.2 wird *Hierarchical and Multicast Cost Optimized Peer-to-peer System* (HiOPS) [Schloss et al., 2008b] vorgestellt, das für den Einsatz in Peer-to-Peer-basierten Gruppen-Chat- und File-Sharing-Anwendungen, unter Berücksichtigung der obengenannten Besonderheiten, entwickelt wurde. Aufgrund seiner hierarchischen Aufteilung in Rendezvous- und Peer-Knoten erzielt HiOPS einen vertretbaren Trade-off zwischen den Konstruktionskosten und der Effizienz. Zu den Schwächen von HiOPS zählen jedoch die Tatsachen, dass für die Konstruktion von HiOPS die Messung der Paketumlaufzeit (RTT) erforderlich ist und die Bei- bzw. Austritte von Rendezvous-Knoten zu einer Restrukturierung der Infrastruktur führen.

Das Ziel beim Entwurf und der Entwicklung der *CARMA based Multicast Infrastructure* (CARMIn), die ihrerseits auf der *Combined*[1] *Affinity Reconnaissance Metric Architecture* (CARMA) basiert, war es, diese Schwächen zu umgehen [Poryev et al., 2010]. Es sollte keine zusätzliche Kommunikation in Form von Ping-Nachrichten erforderlich sein, um die Entfernungen zu anderen Knoten abzuschätzen. Des Weiteren sollten die Bei- bzw. Austritte von Knoten nur die direkten Nachbarn betreffen und keine Auswirkungen auf die gesamte Infrastruktur haben. Die CARMIn-Infrastruktur wird in Abschnitt 5.3 beschrieben.

In Abschnitt 5.4 werden CARMIn- und HiOPS-Infrastrukturen sowie einige Ansätze, die in Abschnitt 2.3 vorgestellt wurden, bezüglich der definierten Metriken verglichen und die gewonnenen Ergebnisse diskutiert. Die für künftige Arbeiten relevanten Erkenntnisse werden in Abschnitt 5.5 zusammengefasst.

5.1 Definition von Multicast-Metriken

Beim Anwendungsschicht-Multicast soll der Sender genau eine Nachricht an eine Empfängergruppe versenden. Dabei kommt es darauf an, dass die Nachricht möglichst schnell alle Empfänger erreicht. Somit stellt die Zeit, die verstreichen würde, bis der letzte Empfänger die Nachricht erhält, ein Gütemaß dar. Eine optimale Lösung bezüglich dieses Gütemaßes würde aus der Sicht jedes Knotens (Senders) ein Baum der *kürzesten Wege* [Dijkstra, 1959] darstellen. Hierbei wird

[1] Ab der nächsten Version soll die CARMA-Metrik mit zusätzlichen Informationen (z.B. Hop-Count) kombiniert werden.

ein Baum auf der Basis des Netzwerkgraphen berechnet, der ausgehend von einem Startknoten (Sender) alle anderen Knoten (Empfänger) im Graphen auf den kürzesten Pfaden (Wegen) erreicht.

Um die Vorteile des Baumes der kürzesten Wege gegenüber einem minimal aufspannenden Baum aufzuzeigen, wurde ein Beispielszenario betrachtet, in dem ein Sender zu allen Empfängern die Entfernung 10 Millisekunden besitzt. Alle benachbarten Empfänger besitzen die Entfernung 1 Millisekunde zueinander. Beim Versenden einer Multicast-Nachricht an N Empfänger über den Baum der kürzesten Wege, würden alle Empfänger aufgrund der Parallelität die Nachricht in 10 Millisekunden empfangen (siehe Abbildung 5.1). Beim Versenden einer Multicast-Nachricht an N Empfänger über den minimalen aufspannenden Baum, würde der letzte Empfänger die Nachricht in $10 + (N - 1)$ Millisekunden empfangen, was insbesondere bei sehr großen N weniger effizient ist (siehe Abbildung 5.2).

Da aber ein Baum der kürzesten Wege für jeden (Start-)Knoten im Netzwerk berechnet und konstruiert werden müsste, ist diese Lösung in großen und dynamischen Netzwerken nicht praktikabel. Stattdessen wird die Verwendung eines minimal aufspannenden Baumes als ein Kompromiss von vielen Multicast-Infrastrukturen vorgeschlagen (siehe Abschnitt 2.3). Im Gegensatz zum Baum der kürzesten Wege muss ein MST nur einmal für das gesamte Netzwerk berechnet bzw. approximiert werden.

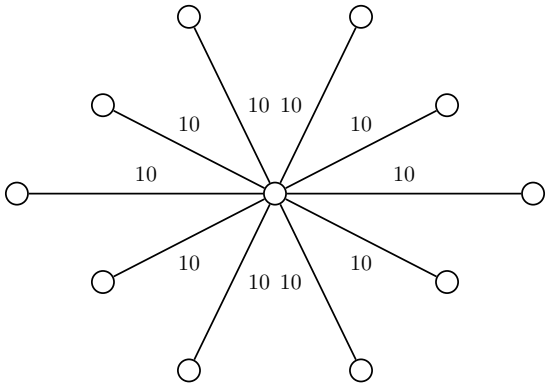


Abbildung 5.1. Baum der kürzesten Wege

Wie bereits in Abschnitt 2.3 in Abbildung 2.7 gezeigt wurde, liegt der Vorteil eines MST gegenüber reiner Unicast-Kommunikation darin, dass während beim Unicast mehrere Nachrichten denselben Pfad nehmen, um zu den räumlich benachbarten Empfängern in einer Region zu gelangen, wird beim MST nur eine Nachricht versendet, die dann in dieser Region verteilt wird.

Das MST-Problem ist eines der bekanntesten und wichtigsten Probleme in den Themengebieten der Graphentheorie und der verteilten Algorithmen bzw. der verteilten Netzwerke. Im Gegensatz zu den theoretischen Modellen in der Graphentheorie, in der eine Menge an Knoten und Kanten sowie die zugehörigen Kantengewichte bekannt sind, existiert in realen Netzwerken – wie zum Beispiel dem

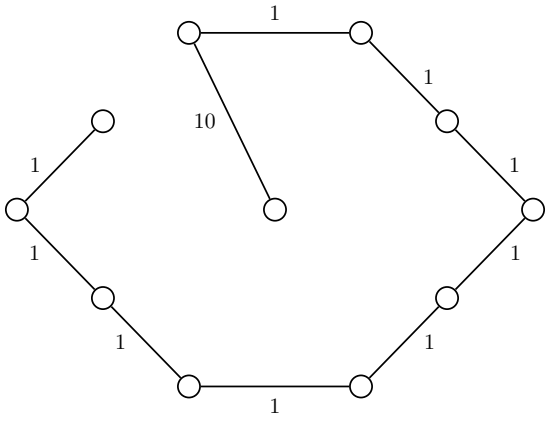


Abbildung 5.2. Minimal aufspannender Baum

Internet – keine globale Sicht. Das heißt, ein Netzwerkknoten kennt nicht alle, sondern nur einen Teil der anderen Netzwerkknoten. Außerdem sind die Entfernungen zwischen den Knoten meistens unbekannt. Um zu einer globalen Sicht zu gelangen, muss ein hoher Kommunikations- und Aktualisierungsaufwand betrieben werden. Denn jedes Mal, wenn sich diese Sicht ändert, müssen die beteiligten Knoten darüber informiert werden. Auch die Erfassung von Entfernungen zwischen den Knoten mittels Ping-Nachrichten, welche die Paketumlaufzeit (RTT) messen, kann in großen Netzwerken einen zu hohen Kommunikationsaufwand verursachen und sollte deshalb vermieden werden. Daraus lässt sich folgern, dass das Aufspannen eines minimalen Baumes in realen Netzwerken nur mit einem unverhältnismäßig hohen Aufwand zu realisieren ist. Deshalb gehen die meisten Ansätze (u.a. Hi-OPS und CARMIn) dazu über, einen suboptimalen Baum mit einem vertretbaren Aufwand zu konstruieren, der einen guten Kompromiss zwischen den Konstruktionskosten bzw. dem Nachrichtenaufwand und der Effizienz darstellt.

Um nun diese Ansätze miteinander objektiv und qualitativ vergleichen zu können, müssen die Kenngrößen wie Effizienz (Kommunikationskosten) bzw. Konstruktionskosten formal definiert werden. Dabei wird ein Netzwerk als ungerichteter und zusammenhängender Graph $G = (V, E, w)$ modelliert, auf dessen Basis der aufspannende Baum berechnet wird. Hierbei stellt V die Menge von Netzwerkknoten $v_i,\ 1 \leq i \leq |V|$, E die Menge der Kanten $e_{i,j} = \{v_i, v_j\}$, welche die logischen Verbindungen der Anwendungsschicht repräsentieren, und $w : E \to \mathbb{N}$ eine Gewichtsfunktion, die einer Kante ihr Gewicht (Latenz) zuordnet, dar. Des Weiteren sei $T = (V, E_T, w)$ ein aufspannender Baum von G und $E_T \subseteq E$ die Kantenmenge von T. Das Gewicht der Kantenmenge[2] E_T wird als die Summe aller Kantengewichte im Multicast-Baum berechnet

$$C(E_T) = \sum_{e \in E_T} w(e) \tag{5.1}$$

und repräsentiert dabei die Summe der Latenzen, die eine Nachricht auf ihrem Weg zu allen anderen Mitgliedern in der Gruppe erfährt. $C(E_T)$ wird im Laufe

[2] Synonym: Gewicht des aufspannenden Baums T.

dieser Arbeit als *Kommunikationskosten-Metrik* (engl. *Communication Cost Metric*) oder auch als *Multicast-Kosten-Metrik* bzw. *Effizienz* bezeichnet. Abbildung 5.5 auf Seite 68 verdeutlicht den Zusammenhang zwischen der Art des aufspannenden Baums und den Multicast-Kosten (Kantengewichten).

Um die Konstruktionskosten zu erfassen, werden zwei weitere Metriken eingeführt. Zum einen wird die Laufzeitkomplexität $O(T)$, die bei der Konstruktion des Multicast-Baums T aus dem Netzwerkgraphen $G = (V, E, w)$ anfällt, gemäß der O-Notation erfasst. Zum anderen wird der Wissensanteil der Netzwerkknoten $K(V) = |V'|$ mit $V' \subseteq V$, die bekannt sein müssen, um den entsprechenden Multicast-Baum zu konstruieren, bei der Erhebung der Konstruktionskosten berücksichtigt[3].

5.2 HiOPS-Infrastruktur

In [Schloss et al., 2008b] ist ein Ansatz beschrieben, wie man einen Trade-off zwischen Multicast- und Konstruktionskosten mithilfe einer MST-Approximation erzielen kann. Da die Konstruktion eines optimalen aufspannenden Baums mit sehr vielen Knoten zu aufwändig ist, baut die HiOPS-Infrastruktur auf einer zweischichtigen Hierarchie auf, in der nur die so genannten Rendezvous-Knoten als MST organisiert werden (Abbildung 5.3). Die anderen Knoten – Peers genannt – bauen lediglich eine Verbindung zu dem Rendezvous-Knoten mit der geringsten Latenz auf. Das Verhältnis der Rendezvous- zu den Peer-Knoten stellt somit einen Trade-off zwischen Kommunikations- und Konstruktionskosten dar. Je nach Anforderungen kann das Verhalten der HiOPS-Infrastruktur über das Verhältnis der Anzahl der Rendezvous-Knoten zur Gesamtanzahl der Knoten gesteuert werden. Besteht ein Netz nur aus Rendezvous-Knoten, so ist es optimal im Sinne der $C(E_T)$-Metrik. Gleichzeitig ist die Konstruktion des aufspannenden Baumes bei einer hohen Anzahl der Knoten laufzeit- und kommunikationsintensiv. Reduziert man im Verhältnis die Anzahl der Rendezvous-Knoten, so werden auch die Konstruktionskosten reduziert[4]. Gleichzeitig ist die Infrastruktur jedoch nicht optimal bezüglich $C(E_T)$. Die Multicast-Nachrichten werden über die so entstandene Infrastruktur verteilt.

Ein Problem, das inhärent in Peer-to-Peer-Netzwerken existiert, ist die Abwesenheit einer zentralen Instanz, die über das globale Wissen aller im Netzwerk existierenden Knoten verfügt. Die Knoten müssen anhand ihres lokalen Wissens einer Teilmenge von aktiven Netzwerkknoten eine möglichst gute MST-Approximation generieren, um einen – im Hinblick auf die Netzbelastung – effizienten Multicast gewährleisten zu können.

Um das *Bootstrapping-Problem*[5] zu umgehen, führt jeder HiOPS-Knoten eine IP-Adress-Liste mit den Rendezvous-Knoten, die online sein sollten. Die Knoten

[3] K steht für Knowledge.

[4] Ein HiOPS-System mit einem einzigen Rendezvous-Knoten ist vergleichbar mit einem Client-Server-System (Stern-Abstraktion).

[5] Beim Startvorgang muss ein neuer Knoten Informationen über einen Zugangspunkt im Netzwerk haben. Bei einem Client-Server-System ist die IP-Adresse des Servers statisch und kann in einer

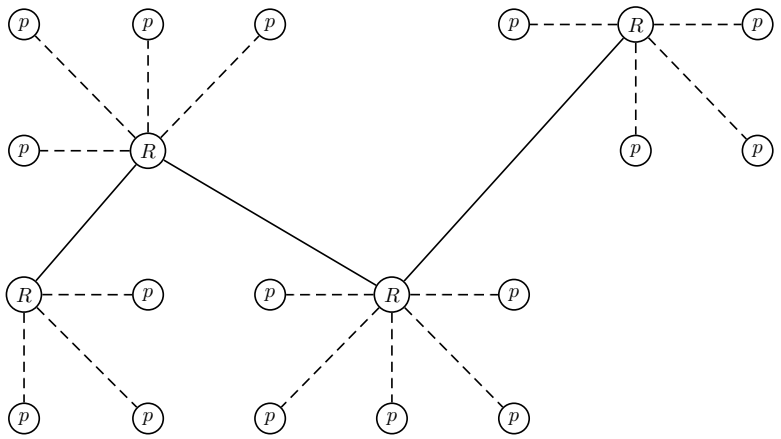


Abbildung 5.3. HiOPS-Infrastruktur

aus dieser Liste dienen als Eintrittspunkt für das Netzwerk und stellen die Menge der Knoten dar, die initial einem neuen Knoten bekannt sind. Das heißt, das Wissen eines Knotens ist auf die Knoten aus der Bootstrapping-Liste begrenzt.

Sei r_l ein verfügbarer Rendezvous-Knoten aus der Bootstrapping-Liste. Wenn nun ein neuer Rendezvous-Knoten r_n dem HiOPS-Netzwerk beitreten will, sendet er eine Anfrage an r_l, um eine Liste mit gegenwärtig aktiven Rendezvous-Knoten zu erhalten. Um Inkonsistenzen zu vermeiden, die durch den gleichzeitigen Beitritt von zwei Rendezvous-Knoten verursacht werden, findet das Beitreten von Rendezvous-Knoten unter wechselseitigem Ausschluss statt. Hierbei sendet r_n eine *Acquire*-Nachricht an den Koordinator-Knoten[6] r_c. Falls gegenwärtig ein anderer Knoten r_x dem Netzwerk beitritt, wartet r_c auf die *Release*-Nachricht von r_x und bearbeitet erst dann die Anfrage von r_n. Andernfalls antwortet der Koordinator r_c sofort und sendet an r_n eine Rendezvous-Graph-Abstraktion $G_R = (V_R, E_R, w)$, die nur aus Rendezvous-Knoten und entsprechenden Verbindungen besteht. r_n modifiziert die Graph-Abstraktion, indem er sich selbst in die Menge der Rendezvous-Knoten V_R einfügt ($V_R = V_R \bigcup \{r_n\}$) und die Kantenmenge E_R um seine Verbindungen zu allen aktiven Knoten erweitert ($E_R = E_R \bigcup\limits_{v_i \in V_R} \{r_n, v_i\}$). Zur Ermittlung der Entfernungen wird hier die Paketumlaufzeit erfasst, was nicht sehr effizient ist. Nach der Modifikation wird die Graph-Abstraktion an alle aktiven Rendezvous-Knoten propagiert. Mit dem Versenden einer *Release*-Nachricht an den Koordinator verlässt r_n den kritischen Abschnitt. Alle betroffenen Rendezvous-Knoten müssen nun ihre Verbindungen gemäß den neuen Gegebenheiten anpassen. Sie berechnen lokal einen MST anhand des neuen Graphen und aktualisieren deren Verbindungen entsprechend. Auch hier wird die Konsistenz der Graph-Abstraktion mit einem hohen Reorganisationsaufwand erkauft.

Das Verlassen des Netzwerks von einem Rendezvous-Knoten geschieht auf identische Art und Weise. Der Knoten tritt in den kritischen Abschnitt ein, entfernt sich

Konfigurationsdatei ohne weiteres abgespeichert werden. In einem Peer-to-Peer-System kann dagegen nicht angenommen werden, dass ein bestimmter Peer ständig online ist.

[6] Koordinator ist der Knoten mit der kleinsten ID von allen aktiven Rendezvous-Knoten.

selbst sowie seine Verbindungen aus seiner lokalen Graph-Repräsentation, versendet die modifizierte Graph-Abstraktion an die anderen Rendezvous-Knoten und verlässt den kritischen Abschnitt. Die anderen Rendezvous-Knoten müssen ihre Verbindungen gemäß der neuen Graph-Abstraktion anpassen.

Der Beitritt von Peers ist vergleichsweise einfach. Ein neuer Peer p_n sendet eine Anfrage an den Rendezvous-Knoten r_l, um eine Liste der aktiven Rendezvous-Knoten zu erhalten. Anschließend misst der Peer die RTT-Zeit zu allen aktiven Rendezvous-Knoten und verbindet sich zu dem Rendezvous-Knoten mit der geringsten Paketumlaufzeit. Um das Netzwerk wieder zu verlassen, baut der Peer die Verbindung einfach wieder ab.

Alle HiOPS-Knoten unterstützen eine Ausfall- und Wiederherstellungsroutine. Jedes Mal, wenn ein Knoten erkennt, dass ein anderer Knoten ausgefallen ist, führt er das Abmelden dieses Knotens durch. Wenn beispielsweise ein Rendezvous-Knoten den Ausfall eines anderen Rendezvous-Knotens erkennt, dann tritt er in den kritischen Abschnitt ein, aktualisiert die Graph-Abstraktion, indem er den ausgefallenen Knoten löscht, versendet die modifizierte Graph-Abstraktion und verlässt den kritischen Abschnitt. Wenn ein Peer den Ausfall seines Rendezvous-Knotens erkennt, dann führt er einfach die Beitrittsroutine erneut durch.

Der HiOPS-Ansatz besitzt einige Schwächen. So sind beispielsweise Messungen der Paketumlaufzeit sehr ineffizient und sollten vermieden werden. Auch die Tatsache, dass die Bei- bzw. Austritte von Rendezvous-Knoten zu einer Restrukturierung des Netzes führen, macht den Einsatz der HiOPS-Infrastruktur in großen Netzwerken unpraktikabel. Außerdem wird der Koordinator-Knoten bei einer hohen Fluktuationsrate von Rendezvous-Knoten überlastet und stellt einen *Single Point of Failure* dar.

5.3 CARMIn-Infrastruktur

Das Ziel, das beim Entwurf und der Entwicklung der CARMIn-Infrastruktur verfolgt wurde, war es, die Schwachstellen von HiOPS aufzugreifen und zu beheben. Das heißt, es sollte zum einen ganz auf die Messung der Paketumlaufzeit zum Zweck der Distanzerfassung verzichtet werden. Zum anderen sollten die Bei- bzw. Austritte von Knoten nur die direkten Nachbarn betreffen und keine Auswirkungen mehr auf die gesamte Infrastruktur haben.

Ein Problem, welches in der HiOPS-Infrastruktur zwar adressiert, aber nicht sehr effizient gelöst wurde, ist die Tatsache, dass die Entfernungen zwischen den Knoten in einem realen Netzwerk, im Gegensatz zu den graphtheoretischen Modellen, unbekannt sind. Die Messung der Paketumlaufzeit bewirkt nicht nur einen enormen Nachrichtenverkehr, sondern ist auch nicht sehr zuverlässig, weil die Netzwerkeigenschaften sich mit der Zeit ändern. Deshalb verändert sich die Latenz zwischen zwei Knoten aufgrund der Netzwerkbelastung ständig. Eine physikalische Verbindung, die zum Zeitpunkt t_0 noch schnell war, kann aufgrund der geänderten Netzbelastung zu einem späteren Zeitpunkt t_1 möglicherweise nicht mehr die erwartete Paketumlaufzeit garantieren. Hier nutzt die CARMIn-Infrastruktur die

CARMA-Architektur, die anhand von IP-Adressen zweier Knoten eine Metrik für die Verbindungsgüte liefert. Dabei beschreibt die CARMA-Metrik das Lokalitätsverhältnis der beiden Knoten. Das heißt, die Metrik sagt aus, ob die beiden Knoten zu demselben Subnetz, demselben Netz, demselben *autonomen System* (AS) oder demselben *Internet-Exchange-Point* (IX) gehören. Unter der Annahme, dass die Latenz zwischen zwei Knoten im selben Subnetz geringer ist als die Latenz zwischen zwei Knoten im selben Netz, bzw. dass die Latenz zwischen zwei Knoten im selben Netz geringer ist als die Latenz zwischen zwei Knoten im selben AS usw., wird die gelieferte Metrik als eine Art Distanz-Substitut zwischen zwei Knoten behandelt.

Eine weitere Schwachstelle von HiOPS ist die starre Graph-Komponente, deren Veränderung aus Konsistenzgründen nur unter einem wechselseitigen Ausschluss stattfinden darf. Aus diesem Grund skaliert das HiOPS-Verfahren bei einer großen Knotenanzahl nicht. Außerdem zieht jede Veränderung der Graph-Komponente eine partielle Reorganisation der Verbindungsinfrastruktur nach sich, was ebenfalls vermieden werden sollte. Deshalb sollte die CARMIn-Infrastruktur so einfach wie möglich gehalten werden, sodass die Bei- bzw. Austritte von Knoten nur für die benachbarten Knoten von Bedeutung sind.

Bei der Konstruktion der CARMIn-Infrastruktur wurde das *Nearest-Neighbor-Prinzip* angewandt, wobei ein neuer CARMIn-Knoten eine Verbindung zu dem Knoten aus der Bootstrapping-Liste mit der geringsten CARMA-Distanz aufbaut. Auf Seite 57 wurde die Baum-Abstraktion $T = (V, E_T, w)$ zur Berechnung der Kommunikationskosten-Metrik $C(E_T)$ eingeführt. Bei der Konstruktion der CARMIn-Multicast-Infrastruktur wird eine modifizierte Gewichtsfunktion w_c verwendet, die einer Kante $e \in E_T$ (Verbindung zwischen zwei Knoten) die entsprechende CARMA-Distanz zuordnet. Im Folgenden wird die Konstruktion der CARMIn MST Approximation T erläutert.

Will ein neuer Knoten n der CARMIn-Infrastruktur beitreten, so ordnet er zuerst die Knoten aus seiner Bootstrapping-Liste BS in Abhängigkeit von ihrer CARMA-Distanz an[7]. Anschließend wird eine Verbindung zu dem Knoten mit der geringsten Distanz aufgebaut. Der entsprechende Algorithmus wird im Folgenden formal dargestellt (Algorithmus 1).

Will ein Knoten die Infrastruktur verlassen, so identifiziert er zuerst einen Knoten n_n aus der Menge seiner direkten Nachbarn N mit der geringsten CARMA-Distanz. Anschließend versendet er eine Abmeldungsnachricht an alle anderen Nachbarn $(N \setminus n_n)$ und teilt ihnen jeweils die IP-Adresse des Knotens n_n mit, zu dem eine Verbindung aufgebaut werden soll, um den Zusammenhang der Infrastruktur zu garantieren. Der Algorithmus ist nachfolgend formal zusammengefasst (Algorithmus 2).

Die Definitionen der Join- und Leave-Operationen garantieren, dass die CARMIn-Infrastruktur zu jedem Zeitpunkt *zusammenhängend* und *kreisfrei* ist[8]. Wenn jedoch ein Knoten ausfällt, können diese Eigenschaften verletzt werden. So kann es

[7] Die Klassifizierung der Distanzwerte wird später in diesem Abschnitt vorgestellt.
[8] Eigenschaften eines aufspannenden Baumes.

Algorithmus 1: Join-Operation

Eingabe : Knoten n, MST-Approximation $T = (V, E_T, w_c)$ und Bootstrapping-Liste $BS \subseteq V$;
Ergebnis : $T' = (V', E'_T, w_c)$ einschließlich n;
Beginn
  wenn $V \neq \emptyset$ **dann**
    Ordne $v \in BS$ so an, dass $\forall\ v_i \in BS : w(v_i, n) \leq w(v_{i+1}, n)$ gilt;
    $E'_T = E_T \bigcup \{n, v_0\}$, wobei v_0 der erste Knoten in BS ist;
  $V' = V \bigcup \{n\}$;

Algorithmus 2: Leave-Operation

Eingabe : Knoten n, MST-Approximation $T = (V, E_T, w_c)$ und Nachbarschaftsmenge $N \subseteq V$;
Ergebnis : $T' = (V', E'_T, w_c)$ ohne n;
Beginn
  wenn $Grad(n) > 1$ **dann**
    Identifiziere den Knoten $n_n \in N$, der die Bedingung $\forall\ v \in N : w_c(n_n, n) \leq w_c(v, n)$ erfüllt;
    Weise alle $v \in (N \setminus \{n_n\})$ an eine Verbindung zu n_n aufzubauen;
  für alle $v \in V$ *mit* $\{v, n\} \in E_T$ **tue**
    $E'_T = E_T \setminus \{v, n\}$;
  $V' = V \setminus \{n\}$;

vorkommen, dass der Baum in zwei oder mehrere Komponenten zerfällt und somit nicht mehr zusammenhängend ist. Beim Versuch den Zusammenhang durch einen erneuten Verbindungsaufbau von betroffenen Knoten wiederherzustellen, kann es zur Kreisbildung im Graph kommen, was ebenfalls zu vermeiden wäre.

Um die geforderten Eigenschaften immer einhalten zu können, führen die Knoten der CARMIn-Infrastruktur eine *Backup*-Routine durch. Die Idee dieser Routine besteht darin, das Wissen eines Knotens auf seine *2-Hop-Nachbarschaft* auszuweiten. Dafür teilt jeder Knoten allen seinen Nachbarn seine komplette Nachbarschaftsmenge inklusive der entsprechender CARMA-Distanzen mit und benachrichtigt seine Nachbarn über jede Änderung dieser Menge. Auf diese Art und Weise kennt ein CARMIn-Knoten alle direkten und alle 2-Hop-Nachbarn bzw. die entsprechenden CARMA-Distanzen.

Wird der Ausfall (*Failure*) des Knotens n_f erkannt, so können seine Nachbarn den Knoten n_n identifizieren, der die geringste CARMA-Distanz zu n_f hatte. Nachdem der Knoten n_n identifiziert wurde, bauen die restlichen Nachbarn von n_f eine Verbindung zu n_n auf, wie dies der Fall bei der Anwendung der Leave-Operation von n_f wäre.

Die Ermittlung der Verbindungsgüte (CARMA-Distanz) anhand von IP-Adressen wird nachfolgend ausführlich beschrieben.

CARMA-Architektur

Das Internet ist ein äußerst heterogenes Netzwerk. Abbildung 5.4 zeigt prototypisch die Internetstruktur, die von einem Internet-Exchange-Point (IX) abgedeckt wird. Dabei bilden die Endbenutzer mit ihren unterschiedlichen Anbindungen und Zugängen die unterste Schicht dieses Modells. Diese spannen untereinander ein IP-Netz oder ein IP-Subnetz auf. Ein autonomes System (AS) kann mehrere IP-Netze umfassen und wird üblicherweise von einem *Internet-Service-Provider* (ISP) verwaltet. Einige autonome Systeme haben eine direkte Anbindung an den Exchange-Point. Einige andere können den IX-Point nur über weitere autonome Systeme erreichen. Die Exchange-Points werden untereinander durch leistungsfähige Links verbunden.

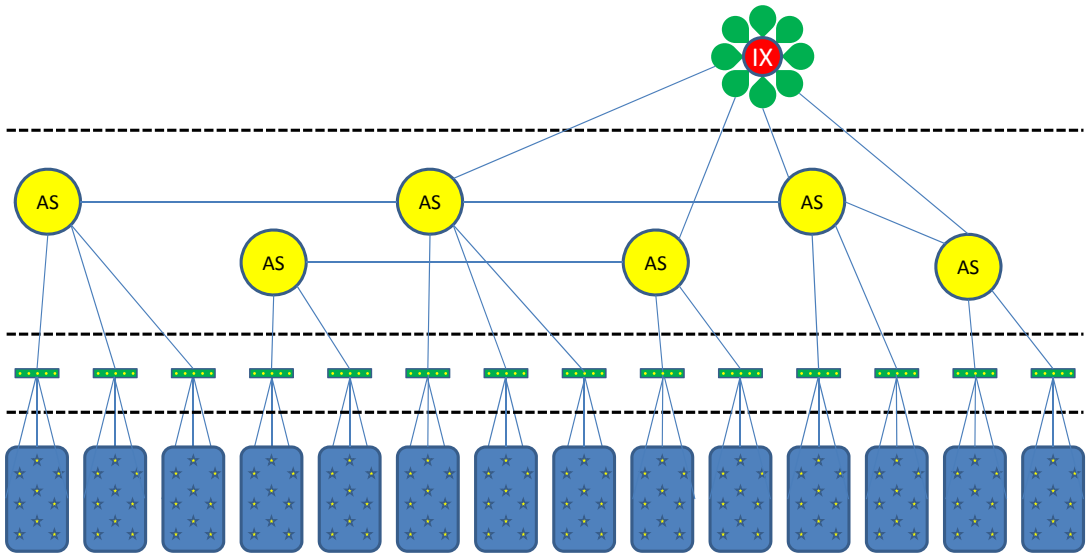


Abbildung 5.4. IX-Internet-Segment (Quelle: [Poryev et al., 2010])

Die meisten Peer-to-Peer-Ansätze betrachten die Knoten-Lokalität auf der Netzwerkebene nicht. Diejenigen, die es doch tun, führen entweder Latenz-Messungen durch oder ermitteln den Hop-Count (d.h. die Anzahl der Zwischenstationen) zwischen dem Sender und dem Empfänger [Lucas et al., 2003]. Die dafür eingesetzte *Trace-Route-Methode* ist laut [Augustin et al., 2006], genauso wie die Ping-Methode, nicht ausreichend zuverlässig und hängt stark von der Netzwerkbelastung ab. Um eine Aussage über die Knoten-Lokalität machen zu können, ohne auf die Ping- oder die Trace-Route-Methode zurückzugreifen, wurde in [Poryev et al., 2010] die CARMA-Architektur vorgestellt, die lokal anhand der IP-Adressen eine Distanz ermittelt. Dabei werden alle erforderlichen Informationen aus den öffentlichen *Regional Internet Registries* (RIRs) extrahiert:

- *AfriNIC* für Afrika,

- *ARIN* für Kanada, die USA und Teile der Karibik,
- *APNIC* für Asien, Australien und Umgebung,
- *LACNIC* für Lateinamerika und weitere Teile der Karibik,
- *RIPE NCC* für Europa, den Mittelosten und Zentralasien.

Die folgenden Informationen aus diesen RIRs sind für die Berechnung der CARMA-Metrik von Bedeutung:

IPv4-Range stellt eine Teilmenge des IPv4-Adressraumes dar und wird durch die Range-IP-Adresse sowie die maximale Anzahl der Host-Rechner im entsprechenden Adressbereich und das Registrierungsdatum definiert. Diese Information kann aus der Datenbankdatei *delegated-ripencc-latest* herausgelesen werden. Ein Ausschnitt ist in Listing 5.1 dargestellt.

```
1  ripencc|EU|ipv4|143.65.0.0|65536|19900326|assigned
2  ripencc|EU|ipv4|143.93.0.0|65536|19940413|assigned
3  ripencc|NO|ipv4|143.97.0.0|65536|20070104|assigned
4  ripencc|EU|ipv4|143.99.0.0|65536|19900907|assigned
```

Listing 5.1. Dateiausschnitt: delegated-ripencc-latest

Autonomes System (AS) ist in [Hawkinson und Bates, 1996] definiert. Autonome Systeme werden in der Datei *delegated-ripencc-latest* durch die Angabe der ISO-Länderkennung, der numerischen ID und des Registrierungsdatums definiert. Die Datei enthält aber keine Informationen über die Beziehungen zwischen den autonomen Systemen und IP-Adressbereichen. Um diese Informationen zu erhalten, muss die Datei *ripe.db.route.gz* (siehe Listing 5.2) analysiert werden. Diese Datei besteht aus mehreren Definitionsbereichen. Zeilen 1 bis 4 enthalten mehrere AS-Definitionen. Zeilen 6 bis 11 beinhalten Informationen über das IPv4-Range 143.93.192.0/18, das zum autonomen System AS2857 gehört. Auf diese Art und Weise kann ein IP-Adressbereich einem autonomen System zugeordnet werden. Diese Zuordnung ist nicht eindeutig, da ein IP-Range zu mehreren autonomen Systemen gehören kann.

```
1  ripencc|EU|asn|2857|1|19931227|allocated
2  ripencc|EU|asn|2858|1|19940112|allocated
3  ripencc|SE|asn|2859|1|19940127|allocated
4  ripencc|EU|asn|2860|1|19940118|allocated
5
6  route:          143.93.192.0/18
7  descr:          FH-RPL-NET
8  origin:         AS2857
9  mnt-by:         AS2857-MNT
10 changed:        weiss@uni-mainz.de 20001220
11 source:         RIPE
```

Listing 5.2. Datei-Ausschnitt: ripe.db.route.gz

IPv4-Subrange ist eine Teilmenge des IPv4-Adressraums, die durch die Angabe der Netz-IP-Adresse sowie der letzten IP-Adresse des Netzes definiert ist. Diese

Information ist in der Datei *ripe.db.inetnum.gz* aufgelistet (siehe Listing 5.3). Im Gegensatz zu den IPv4-Ranges werden die IPv4-Subranges nicht direkt einem autonomen System zugeordnet. Grundsätzlich ist ein Subrange kleiner als ein Range, der wiederum mehrere Subranges enthalten kann. Von daher ist es möglich eine Beziehung von Subranges zu Ranges aufzustellen.

```
1  inetnum:        143.93.32.0 - 143.93.63.255
2  netname:        FH-RPL-NET
3  descr:          Fachhochschule Trier
4  descr:          Rechenzentrum
5  descr:          Schneidershof
6  descr:          D-54293 Trier
7  country:        DE
8  admin-c:        KM624-RIPE
9  tech-c:         RB373-RIPE
10 status:         ASSIGNED PI
11 mnt-by:         TRANSKOM-MNT
12 changed:        hostmaster@transkom.net 20050207
13 source:         RIPE
```

Listing 5.3. Dateiausschnitt: ripe.db.inetnum.gz

AS-Set kann mehrere autonome Systeme umfassen und wird in der Datenbank mit den alphanumerischen Bezeichnern versehen. Autonome Systeme innerhalb eines AS-Sets verfügen über eine gute Konnektivität. Im Kontext von CARMA werden Internet-Exchange-Points (IXs) als AS-Sets mit vielen (hunderten) von autonomen Systemen angesehen. Die AS-Sets sind in der Datei *ripe.db.as-set.gz* definiert (Listing 5.4).

```
1  as-set:         AS-DECIX-CONNECTED
2  descr:          ASN of DE-CIX members
3  descr:          DE-CIX, the German Internet Exchange
4  admin-c:        AN6695-RIPE
5  tech-c:         WT6695-RIPE
6  tech-c:         DM6695-RIPE
7  tech-c:         SJ6695-RIPE
8  notify:         notify@de-cix.net
9  mnt-by:         DECIX-MNT
10 source:         RIPE
11 changed:        auto-upd@de-cix.net 20091011
12 members:        AS42
13 ...
14 members:        AS2828
15 members:        AS2857
16 members:        AS2914
17 ...
18 members:        AS65333
```

Listing 5.4. Dateiausschnitt: ripe.db.as-set.gz

Nach dem Traversieren der Datenbankdateien reflektiert der resultierende Graph die Internet-Topologie so gut wie es ohne den Zugriff auf die BGP-Informationen[9] möglich ist. Dabei liegt der Fokus von CARMA nicht darauf das Internet zu modellieren, sondern die Distanz zwischen zwei Knoten im Internet anhand der oben beschriebenen Informationen abzuschätzen. Eine Auflistung der CARMA-Distanzen *(Flavors)* wird im Folgenden angegeben.

1. *Distanz 0* deutet darauf hin, dass sich beide IP-Adressen in demselben IPv4-Subrange gemäß der Datenbankspezifikation aus der Datei *ripe.db.inetnum.gz* befinden könnten. Diese Distanz bedeutet, dass beide Knoten mit großer Wahrscheinlichkeit an den gleichen Router angeschlossen sind (z.B. im Universitäts-Netzwerk).
2. *Distanz 1* deutet darauf hin, dass sich beide IP-Adressen entsprechend der Spezifikation aus den Dateien *delegated-* oder *ripe.db.route.gz* in dem gleichen IPv4-Range befinden.
3. *Distanz 2* weist darauf hin, dass beide Adressen innerhalb des Adressraums desselben autonomen Systems liegen, gemäß der Definition in *ripe.db.route.gz*. Diese Distanz sagt aus, dass die beiden Knoten zwar nicht mehr innerhalb des gleichen IPv4-Ranges sind und deshalb eine größere Latenz aufweisen könnten, dennoch wird die Kommunikation zwischen den beiden Knoten mit großer Wahrscheinlichkeit die Grenzen eines ISPs nicht verlassen.
4. *Distanz 3* bedeutet, dass die beiden IP-Adressen innerhalb desselben AS-Sets liegen. Daraus kann abgeleitet werden, dass die Kommunikation die Grenzen eines Internet-Exchange-Points (IX) nicht überschreitet.
5. *Distanz 4* bedeutet, dass keine Ähnlichkeit der IP-Adressen festgestellt werden konnte. Somit wird angenommen, dass die Knoten unterschiedlichen IXs zugeordnet werden und die Kommunikation mit entsprechend hohen Latenzen verbunden ist.

Um ein Gefühl für die Güte der CARMA-Distanz-Schätzungen zu bekommen, wurde ein kleiner Testversuch durchgeführt, bei dem die Entfernungen zwischen der Fachhochschule Trier und der Universität Trier, der Universität Mainz, der University of Limerick und der University of California, Berkeley, mittels Ping-Nachrichten ermittelt und mit den entsprechenden CARMA-Distanzen verglichen wurden. Tabelle 5.1 stellt die ermittelten Entfernungen gegenüber.

Einrichtung	CARMA-Distanz	Latenz [ms]
Universität Trier	2 – AS	1 ms
Universität Mainz	2 – AS	4 ms
University of Limerick	3 – IX	35 ms
University of California	4 – Distant	159 ms

Tabelle 5.1. Gegenüberstellung von CARMA-Distanzen und Latenzen

[9] Das Routingprotokoll des Internets wird als *Border Gateway Protocol* (BGP) bezeichnet. Dieses Protokoll beschreibt, wie Router intern die Kommunikation zwischen den Netzen autonomer Systeme regeln.

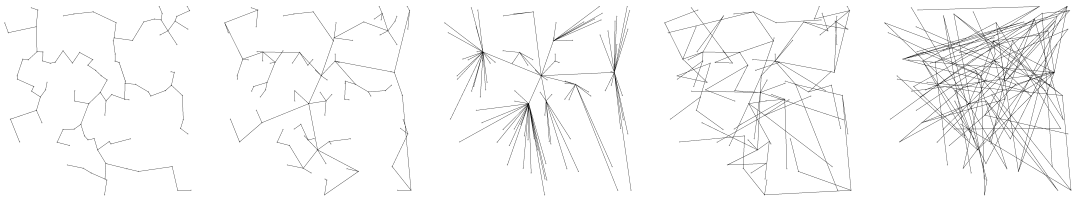
Wie die Werte zeigen, werden sowohl die Universität Trier als auch die Universität Mainz dem gleichen autonomen System wie die Fachhochschule Trier zugeordnet. Somit sind die beiden Standorte gemäß der CARMA-Distanz gleich weit von der Fachhochschule Trier entfernt. Die Messung der Latenzzeit liefert hier eine genauere Differenzierung. Die Entfernungen zu der University of Limerick bzw. der University of California werden mit den beiden Verfahren vergleichbar eingeordnet. Das heißt, die Ergebnisse der beiden Verfahren lassen darauf schließen, dass die University of Limerick näher zu der Fachhochschule Trier liegt, als die University of California. Das Fazit lautet also: Latenz-Messungen können zwar differenziertere Ergebnisse liefern, erfordern dafür aber einen (für große Netzwerke) inakzeptablen Kommunikationsaufwand.

CARMA bietet hingegen eine Möglichkeit, Entfernungen in fünf Distanz-Klassen einzuteilen und kommt gänzlich ohne zusätzliche Kommunikation aus. Außerdem besitzt CARMA einen weiteren signifikanten Vorteil gegenüber RTT-Messungen. Mithilfe von Ping-Nachrichten kann ein Knoten nur seine eigene Entfernung zu einem anderen Knoten im Internet messen, wohingegen die Verwendung von CARMA es erlaubt, die Entfernungen zwischen beliebigen Knoten-Paaren im Internet abzuschätzen und auf diese Art und Weise Aussagen über die vorliegende Netzwerk-Topologie zu machen.

5.4 Evaluationsergebnisse

Um nun aussagekräftige Ergebnisse über das Effizienz-Konstruktionskosten-Verhältnis zu erhalten, wurden CARMIn- und HiOPS-Infrastrukturen mit anderen existierenden Peer-to-Peer-Ansätzen, die einen Multicast auf Anwendungsebene unterstützen, verglichen. Für den Vergleich wurden die JXTA-Infrastruktur und ALMI (Application Level Multicast Infrastructure) [Pendarakis et al., 2001] herangezogen. Um die Multicast-Kosten gering zu halten, wird der JXTA-Multicast-Baum nach dem *Nearest-Neighbor*-Prinzip aufgebaut. Das heißt, ein neuer Knoten baut eine Verbindung zu dem Knoten mit der geringsten Latenz auf. In ALMI wird hingegen auf eine Zentralinstanz zugegriffen, um einen MST zu berechnen. Der Vergleich wurde um einen *RANDOM*-Ansatz erweitert, in dem ein neuer Knoten eine Verbindung zu einem beliebigen Knoten im Netzwerk aufbaut. Abbildung 5.5 stellt die obengenannten Ansätze beispielhaft in einem Netzwerk mit 100 Knoten anschaulich dar.

Um diese Ansätze miteinander zu vergleichen, wurde eine einfache Simulationsumgebung entwickelt, die eine vordefinierte Anzahl an Netzwerkknoten generiert und diese entsprechend des gewählten Algorithmus (ALMI, JXTA, HiOPS, CARMIn oder RANDOM) zu einer Multicast-Infrastruktur verbindet. Mehrere Simulationsläufe (mit 10, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500 und 5000 Knoten) wurden durchgeführt. In jedem Simulationsdurchlauf wurden für alle Ansätze die Multicast-Kosten $C(E_T)$ (in Millisekunden) sowie die Konstruktionskosten $O(T)$ und der notwendige Wissensanteil $K(V)$ ermittelt und gegenübergestellt.



(a) ALMI (b) JXTA (c) HiOPS (d) CARMIn (e) RANDOM

Abbildung 5.5. MST-Approximationen in einem Netzwerk mit 100 Knoten

Abbildung 5.6 zeigt die Gegenüberstellung der betrachteten Ansätze bezüglich der Kommunikationskosten $C(E_T)$ in Abhängigkeit von der Knotenanzahl $|V|$. Erwartungsgemäß erfordern ALMI- und JXTA-Ansätze die geringsten Kommunikationskosten, was durch deren Struktur bedingt ist. Der RANDOM-Ansatz verursacht bereits bei einer kleinen Knotenanzahl (ca. 1000) einen zu hohen Kommunikationsaufwand und kann deshalb nicht als Multicast-Infrastruktur berücksichtigt werden. HiOPS und CARMIn haben ungefähr die gleichen Kommunikationskosten.

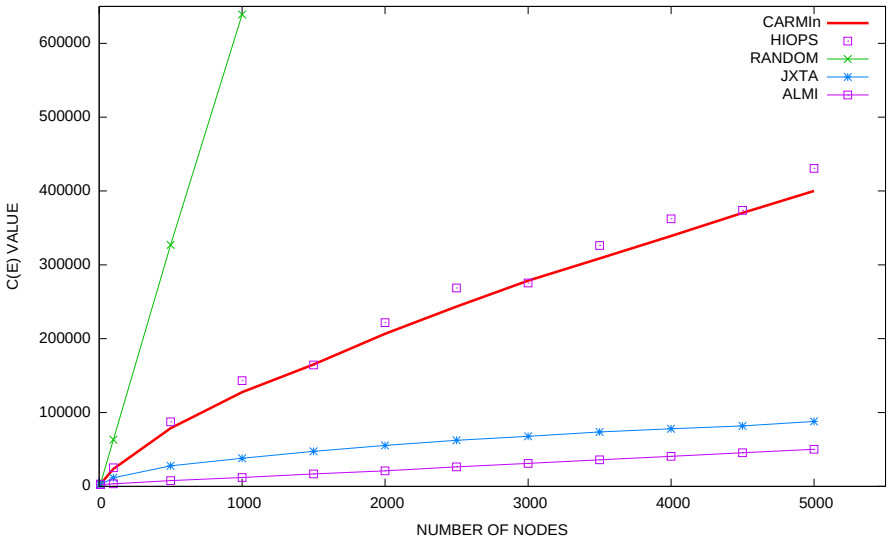


Abbildung 5.6. Kommunikationskosten $C(E_T)$ in Abhängigkeit von $|V|$

Der Wissensanteil der Knoten, die bei der Konstruktion bekannt sein müssen, wird durch die Metrik $K(V)$ ausgedrückt. Für die Konstruktion eines MST müssen alle Knoten bekannt sein. Somit beträgt $K(V) = |V|$ für die ALMI-Infrastruktur. Damit ein neuer Knoten im JXTA-Netzwerk entscheiden kann, welcher existierende Netzwerkknoten zu ihm am nächsten ist, erfordert er ebenfalls ein globales Wissen $K(V) = |V|$. In HiOPS müssen lediglich alle Rendezvous-Knoten bekannt sein: $K(V) = |V_R| = \sqrt{|V|}$. Im Worst-Case beträgt die Größe der Bootstrapping-Liste in CARMIn ebenfalls $K(V) = \sqrt{|V|}$. Aus diesen Knoten wird gemäß dem beschriebenen Verfahren der Knoten mit der kleinsten CARMA-Distanz ausgewählt. Ein neuer Knoten in der RANDOM-Infrastruktur muss lediglich einen einzigen

Kontaktknoten kennen: $K(V) = 1$. Die Abbildung 5.7 stellt die erforderlichen Wissensanteile gegenüber.

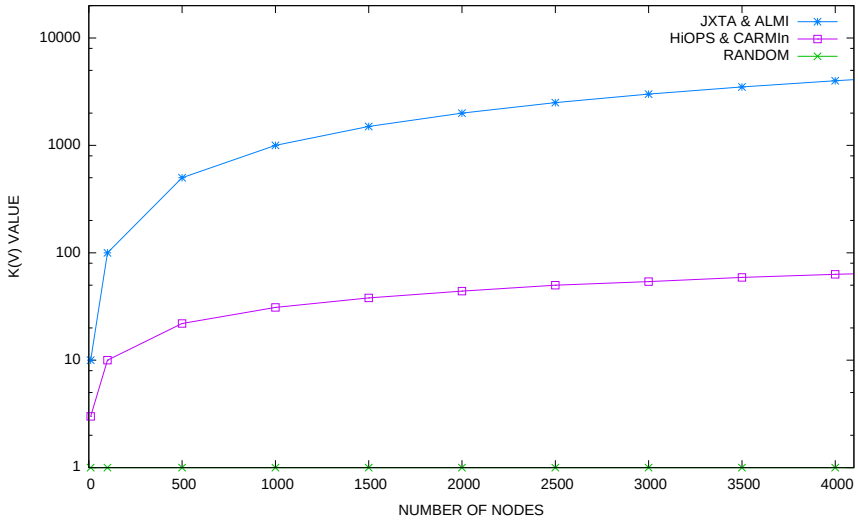


Abbildung 5.7. Wissensanteile K(V) in Abhängigkeit von $|V|$

Die Laufzeitkomplexität für die Konstruktion eines ALMI-MST bei Anwendung des Kruskal-Algorithmus [Kruskal, 1956] beträgt $O(log(|V|)|E|)$, wobei diese hauptsächlich durch die zum Sortieren der Kanten benötigte Laufzeit bestimmt wird. Bei der Konstruktion eines NNT müssen die Kanten ebenfalls sortiert werden, um entscheiden zu können, welcher Knoten die geringste Entfernung aufweist. Somit beträgt die Laufzeit für die Konstruktion der JXTA-Infrastruktur auch $O(log(|V|)|E|)$.

Da bei HiOPS nur die Rendezvous-Knoten als ein MST organisiert sind und der Anteil der Rendezvous-Knoten ca. $V_R = \sqrt{|V|}$ beträgt, beläuft sich die Laufzeitkomplexität hier auf $O(log(|V_R|)|E_R|)$, wobei die Gültigkeit der Implikation $|V_R| = \sqrt{|V|} \Rightarrow |E_R| = \sqrt{|E|}$ angenommen wurde. In CARMIn müssen die Knoten der Bootstrapping-Liste anhand der CARMA-Distanzen sortiert werden. Da in der Simulation davon ausgegangen wurde, dass sich die Größe der Bootstrapping-Liste auf $\sqrt{|V|}$ beläuft, beträgt der Konstruktionsaufwand hier – ebenfalls bedingt durch das Sortieren – $O(log(\sqrt{|V|})\sqrt{|E|})$. Hier ist noch anzumerken, dass sowohl das Verhältnis der Rendezvous-Knoten in HiOPS als auch die Größe der Bootstrapping-Liste in CARMIn einen Trade-off zwischen den Kommunikations- und Konstruktionskosten darstellen. Je größer das Verhältnis der Rendezvous-Knoten in HiOPS bzw. je größer die Bootstrapping-Liste in CARMIn ist, desto geringer sind die Kommunikationskosten und desto höher die Konstruktionskosten, und umgekehrt.

Der RANDOM-Ansatz baut eine MST-Approximation durch zufällige Verbindungen zu bestehenden Netzwerkknoten auf und erfordern keine nennenswerten Konstruktionskosten ($O(1)$), weil keine Sortierung o.ä. erforderlich ist. Abbildung 5.8 stellt alle Ansätze bezüglich deren Konstruktionskosten gegenüber. Wie man

unschwer erkennen kann, ist die Laufzeitkomplexität bei der Verwendung eines MST (ALMI) oder eines NNT (JXTA) für realistische Anwendungsszenarien zu groß.

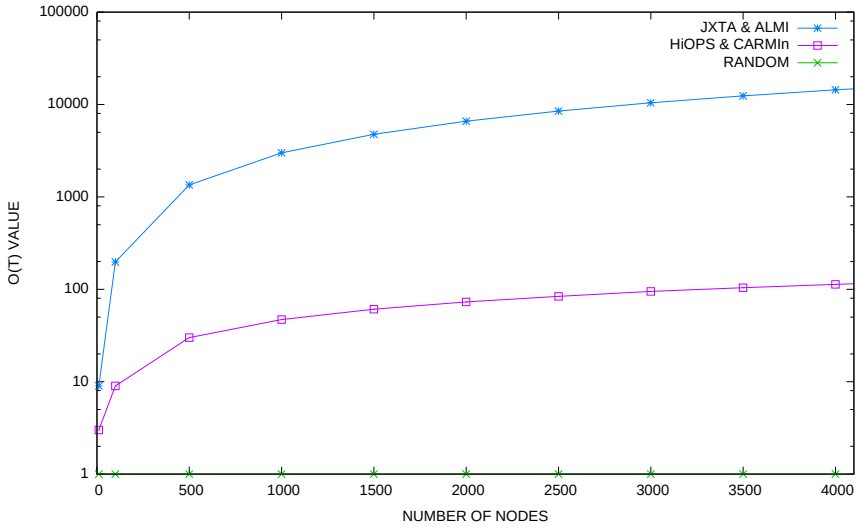


Abbildung 5.8. Konstruktionskosten $O(T)$ in Abhängigkeit von $|V|$

Wie die Abbildungen 5.6, 5.8 und 5.7 zeigen, haben die JXTA- und ALMI-Ansätze die geringsten Kommunikationskosten, benötigen dafür aber ein globales Wissen über alle sich in den Netzwerken befindenden Knoten und verursachen darüber hinaus zu hohe Konstruktionskosten. Aus diesem Grund scheiden die beiden Verfahren für den Einsatz als Multicast-Infrastruktur im HERA-Framework aus.

Im Gegensatz zu JXTA und ALMI, die einen gewissen Berechnungsaufwand bei der Konstruktion erfordern und über globales Infrastruktur-Wissen verfügen müssen, fallen beim Aufbau der RANDOM-Infrastruktur nur sehr geringe Konstruktionskosten an, da ein neuer Knoten lediglich einen anderen kennen muss, um dem Netzwerk beizutreten. Die Kommunikationskosten sind bei diesem Ansatz jedoch inakzeptabel hoch. Eine RANDOM-Infrastruktur ist somit ebenfalls ungeeignet für Multicast.

Die CARMIn- und HiOPS-Infrastrukturen verursachen beinahe dieselben Kommunikations- und Konstruktionskosten und erfordern denselben Anteil an Wissen. Da im Gegensatz zu HiOPS bei der Konstruktion von CARMIn keine zusätzliche Kommunikation anfällt und die Bei- und Austritte von Knoten sich nur lokal auswirken, erzielt CARMIn einen akzeptablen Trade-off zwischen den Kommunikations- und Konstruktionskosten und entscheidet den Vergleich der unterschiedlichen Ansätze für sich.

5.5 Schlussfolgerung

In diesem Kapitel wurden zwei im Rahmen dieser Arbeit entwickelte Multicast-Infrastrukturen vorgestellt und mit den existierenden Ansätzen bezüglich bestimmter Metriken verglichen.

Im Gegensatz zu den Ansätzen, die über globales Wissen verfügen müssen, baut der favorisierte CARMIn-Ansatz eine MST-Approximation anhand des lokalen Wissens einer Teilmenge von Netzwerkknoten. Darüber hinaus erfordert CARMIn keine Kommunikation zur Erfassung der Knotendistanzen. Stattdessen werden die Knotenentfernungen auf der Basis der Datenbankeinträge aller fünf RIRs geschätzt. Die Häufigkeit der Datenbankaktualisierungen kann in der CARMA-Architektur eingestellt werden (zum Beispiel einmal täglich), sodass man davon ausgehen kann, dass die Datenbankinformationen immer aktuell sind.

Wie die Evaluationsergebnisse zeigen, kann bereits ein einfacher *Nearest-Neighbor*-Algorithmus (in dem ein neuer Knoten eine Verbindung zu dem Knoten mit der geringsten CARMA-Distanz aufbaut) akzeptable Kommunikationskosten $C(E_T)$ garantieren. Da beim Beitreten oder Austreten von Knoten auch keine Reorganisation der gesamten CARMIn-Infrastruktur anfällt, stellt dieser Ansatz insgesamt einen guten Kompromiss zwischen den Konstruktions- und Kommunikationskosten dar und kann daher in dem dynamischen Client-Overlay als Multicast-Infrastruktur fungieren.

Die Tatsache, dass Clients innerhalb einer AoI miteinander über die CARMIn-Infrastruktur kommunizieren und die Management-Schicht dafür nicht beanspruchen, kann eine weitere Entlastung der Basis-Infrastruktur mit sich bringen und somit zur Verbesserung der Skalierbarkeit und der Lastverteilung in DVEs beitragen.

6

Konsistenzaspekte in HERA

Virtuelle Umgebungen können sowohl auf einer zentralisierten Client-Server- als auch auf einer dezentralisierten Peer-to-Peer-Infrastruktur aufbauen. Die Wahl der Basis-Infrastruktur muss dabei nicht zwangsläufig die Art des Konsistenz-Handlings beeinflussen. Die beiden Arten des Konsistenz-Handlings (server- oder clientbasiert) können sowohl mit einer zentralisierten (Server) als auch mit einer verteilten (Peer-to-Peer-Overlay) Infrastruktur kombiniert werden. Das heißt, auch bei der traditionellen Client-Server-Infrastruktur kann das Konsistenz-Handling an die Clients übertragen werden. Umgekehrt kann ein Knoten in einem Peer-to-Peer-Overlay die Rolle einer zentralen Kontrollinstanz übernehmen, die für das Konsistenz-Handling in der entsprechenden AoI auf die traditionelle Art und Weise zuständig ist. Grundsätzlich ist die Zusicherung der Konsistenz basierend auf einer Kontrollinstanz (serverbasiertes Konsistenz-Handling) einfacher als im verteilten Fall. Eine Kontrollinstanz kann die Zugriffe auf das gemeinsam genutzte Objekt einfacher synchronisieren und konkurrierende Zustandsaktualisierungen einfacher ordnen. Doch aufgrund der genannten Skalierbarkeits- und Lastverteilungsprobleme müssen Alternativen überlegt werden, die gänzlich oder zum großen Teil ohne diese zentrale Instanz auskommen. Deshalb wird in dieser Arbeit die Übertragung von Konsistenz-Handling-Aufgaben an die dezentralisierte Client-Schicht vorgeschlagen. Dadurch soll die gewünschte Entlastung der Management-Schicht und die daraus resultierende Verbesserung der Skalierbarkeit und der Lastverteilung erreicht werden.

Um die besonderen Anforderungen der virtuellen Umgebungen an die Konsistenz besser berücksichtigen zu können, werden in Abschnitt 6.1 die Modelle der *elastischen* bzw. *statistisch elastischen Konsistenz* definiert. Eng gekoppelt mit dem Konsistenzbegriff in DVEs ist der Begriff der Interaktivität bzw. der *Systemantwortzeit* (*System Responsiveness*). Die Systemantwortzeit ist die Zeit, die zur Bearbeitung der Benutzereingabe und der Ergebnisdarstellung anfällt. Virtuelle Umgebungen stellen sehr hohe Anforderungen an Systemantwortzeiten. Deshalb wird in Abschnitt 6.2 der Einfluss von unterschiedlichen Konsistenzmodellen auf die Systemantwortzeit (Interaktivität) der Umgebung verdeutlicht.

Die Einhaltung der beiden gegensätzlichen Kriterien – Interaktivität und Konsistenz – wird in dieser Arbeit in Form eines *Consistency-Responsiveness-Trade-off*-Parameters (CRT-Parameters) ausgedrückt. Dieser wird zur Anpassung des

Verhaltens von Konsistenzalgorithmen eingesetzt, um bei einer geforderten Systemantwortzeit den höchstmöglichen Grad an Konsistenz zu garantieren. Um die Verwendung des CRT-Parameters zu veranschaulichen, wird in Abschnitt 6.3 eine parametrisierte Erweiterung des Maekawa-Algorithmus vorgestellt, die den statistisch elastischen wechselseitigen Ausschluss sicherstellt. In Abschnitt 6.4 wird dieser Ansatz mit dem Algorithmus von Maekawa bezüglich der Konsistenzwahrscheinlichkeiten und der Systemantwortzeiten verglichen.

Oftmals ist es erforderlich bei der Durchführung einer bestimmten Interaktion eine Konsistenzmindestwahrscheinlichkeit zu garantieren, ohne die Systemantwortzeitanforderung zu verletzen. Aus diesem Grund wird in Abschnitt 6.5 untersucht, ob der Wert des CRT-Parameters und somit auch die benötigte Systemantwortzeit aus der geforderten Konsistenzwahrscheinlichkeit zurückberechnet werden kann.

Um die Allgemeingültigkeit und die Anwendbarkeit der vorgestellten Konsistenzmodelle zu demonstrieren, wird in Abschnitt 6.6 ein weiteres DVE-Szenario betrachtet, bei dem das Modell der statistisch elastischen Konsistenz Anwendung finden kann. Die in diesem Kapitel gewonnenen Erkenntnisse werden in Abschnitt 6.7 zusammengefasst.

6.1 Definition der elastischen Konsistenz

Die Idee der *elastischen Konsistenz* wurde erstmals in [Schloss et al., 2008a] vorgestellt. Da die Zusicherung der absoluten Konsistenz in DVEs an den hohen Interaktivitätsanforderungen scheitert, wurde in der Arbeit ein dreistufiges Konsistenzmodell vorgeschlagen, welches anwendungsabhängig das vorliegende Konsistenzproblem einer Stufe zuordnet und entsprechend der Stufenspezifikation algorithmisch behandelt.

In diesem Abschnitt werden die Konsistenzmodelle *der elastischen bzw. der statistisch elastischen Konsistenz* formal definiert. Hierbei wird auf eine explizite Klassifizierung von Konsistenzproblemen in diskrete Konsistenzstufen verzichtet.

Bei der Betrachtung der Time-Space-Inkonsistenz-Metrik in Abschnitt 2.4 wurden bereits die Kenngrößen ε und ι als die kleinste Abweichung, die ein Nutzer erkennen kann (anwendungsspezifisch) bzw. als die Reaktionszeit eines menschlichen Nutzers oder die Verzögerung, die von einem Nutzer nicht als störend empfunden wird, vorgestellt [Zhou et al., 2003]. Die Time-Space-Inkonsistenz-Metrik Ω wird dabei als Produkt der tatsächlichen Abweichung δ und der Zeitdauer τ, während der sich die Client-Zustände unterscheiden, definiert (Definition 2.3).

Die Verwendung des mathematischen Produktes für die Verknüpfung der beiden relevanten Größen ε und ι zu einem Konsistenz-Schwellwert ist jedoch nicht plausibel. Auch wenn die Beobachtungszeit τ bzw. die Reaktionszeit ι eindeutig in Sekunden angegeben werden können, so bleibt die Einheit der Abweichung δ und des entsprechenden Grenzwertes ε undefiniert. Aus diesem Grund ist auch die Einheit des Produktes undefiniert. Deshalb wird in dieser Arbeit eine andere Verknüpfung der beiden Größen vorgeschlagen.

Mit Sicherheit kann Folgendes gesagt werden: Wenn die tatsächliche Abweichung kleiner ist als die Abweichung, die von einem Nutzer wahrgenommen werden kann, und die Beobachtungszeit der tatsächlichen Abweichung kleiner ist als die Reaktionszeit des Nutzers, d.h. wenn gilt

$$\delta < \varepsilon \wedge \tau < \iota,$$

dann ist die Umgebung konsistent. Ebenfalls plausibel ist die Folgerung, dass, wenn die Bedingung

$$\delta > \varepsilon \wedge \tau > \iota$$

erfüllt ist, die Umgebung inkonsistent ist.

Ist die Umgebung aber immer noch konsistent, wenn die tatsächliche Abweichung kleiner als die zulässige Abweichung und die Beobachtungszeit größer als die Reaktionszeit ist ($\delta < \varepsilon \wedge \tau > \iota$), bzw. wenn die tatsächliche Abweichung größer als die zulässige Abweichung und die Beobachtungszeit kleiner als die Reaktionszeit ist ($\delta > \varepsilon \wedge \tau < \iota$)?

$\delta < \varepsilon \wedge \tau > \iota$ ist unkritisch, da die Abweichung so klein ist, dass sie entweder nicht wahrgenommen oder nicht als störend empfunden wird. Auch bei längerer Beobachtungszeit fällt diese Abweichung nicht negativ ins Gewicht.

Die Situation, bei der eine signifikante Abweichung nur eine kurze Zeit beobachtet wird, d.h. $\delta > \varepsilon \wedge \tau < \iota$, ist ebenfalls unkritisch, da die Beobachtungszeit der Abweichung so klein ist, dass sie von einem menschlichen Benutzer nicht wahrgenommen oder nicht als störend empfunden wird. Das heißt, dass das System innerhalb der Zeitschranke ι wieder in einen konsistenten Zustand übergeht.

Daraus lässt sich folgern, dass, wenn die tatsächliche Abweichung **oder** ihre Beobachtungszeit kleiner als der entsprechende Grenzwert ist

$$\delta < \varepsilon \vee \tau < \iota, \tag{6.1}$$

die Umgebung konsistent ist.

Um nun anhand der obengenannten Folgerungen und Feststellungen ein Konsistenzmodell definieren zu können, müssen die Größen δ, ε, τ und ι genauer spezifiziert werden. Wie in Abschnitt 2.4.3 beschrieben, liegt der Schwellwert für die Verzögerung, die ein Nutzer tolerieren kann, zwischen 120 und 250 Millisekunden ($120 \leq \iota \leq 250$). Die tatsächliche Beobachtungszeit einer signifikanten Abweichung ($\delta > \varepsilon$) darf somit diesen Schwellwert nicht übersteigen ($0 \leq \tau < \iota$). Der Grenzwert ε für die Abweichung ist anwendungsabhängig und muss für jedes Szenario explizit gewählt und angegeben werden. Hierbei ist noch anzumerken, dass ε eine subjektive Größe ist, die auf das jeweilige Nutzerempfinden zurückzuführen ist und deshalb formal nur mit einer begrenzten „Genauigkeit“ erfasst werden kann.

Was bedeutet nun die tatsächliche Abweichung δ und wie wird diese ermittelt? Im weiteren Verlauf dieses Abschnitts gehen wir dieser Frage auf den Grund und geben formale Definitionen der elastischen bzw. statistisch elastischen Konsistenz an. Zuvor ist jedoch eine formale Definition des Client-Zustandes erforderlich.

Definition 6.1 (Client-Zustand). *Es seien $\mathbb{C}$ eine Menge von Clients in derselben AoI, $\mathbb{Z}$ der entsprechende Zustandsraum und t die Zeit. Dann stellt Z eine zweistellige Operation dar, die $\forall i, t$ jedem Client $c_i \in \mathbb{C}$ zu jedem Zeitpunkt t einen Zustand $z \in \mathbb{Z}$ zuweist.*

$$Z : \mathbb{C} \times t \rightarrow \mathbb{Z} \tag{6.2}$$

Der Zustand, der nach der Anwendung der Operation Z auf den Client c_i zum Zeitpunkt t vorliegt, wird als $z_i^t = Z(c_i, t)$ gekennzeichnet.

□

Gemäß der Definition beschränkt sich ein Zustandsraum $\mathbb{Z}$ auf die Clients, von denen angenommen wird, dass sie sich in derselben AoI befinden und somit exakt die geleichen Objekte in der virtuellen Umgebung beobachten (lesen) und verändern (schreiben) können.

Ausgehend von der Definition des Client-Zustandes wird δ als die absolute Abweichung (Differenzbetrag) der Client-Zustände z_i^t und z_j^t aus dem Zustandsraum $\mathbb{Z}$ aufgefasst, wobei die exakte mathematische Bestimmung der Differenz selbst, genauso wie die Festlegung des Schwellwertes ε, anwendungsabhängig ist.

Anhand der getroffenen Annahmen und Festlegungen kann jetzt das Modell der *elastischen Konsistenz* definiert werden.

Definition 6.2 (Elastische Konsistenz). *Ein Zustandsraum ist **elastisch konsistent**, wenn für Zustände aller Clients in derselben AoI gilt:*

$$\forall\ t, i, j :\ \delta = \|Z(c_i, t) - Z(c_j, t)\| < \varepsilon \vee\ \exists\ |\tau| < \iota : Z(c_i, t) = Z(c_j, t + \tau) \tag{6.3}$$

Die Gleichung 6.3 kann wie folgt verallgemeinert werden:

$$\forall\ t, i, j\ \exists\ |\tau| < \iota :\ \delta = \|Z(c_i, t) - Z(c_j, t + \tau)\| < \varepsilon \tag{6.4}$$

□

Die Definition sagt aus, dass zu jedem bestimmten Zeitpunkt t die Zustandsabweichung δ den festgelegten Schwellwert ε unterschreiten muss, oder, dass beim Vorliegen größerer Abweichungen ($\delta \geq \varepsilon$) die Zustände sich im Zeitintervall τ wieder ausgleichen. Im Folgenden werden unterschiedliche Beispiele für die Erfüllung und Verletzung der elastischen Konsistenz betrachtet.

Beispiel 6.3 (Elastische Konsistenz ist verletzt (1)). Es existiert ein Spielfeld mit Figuren, die sich in jedem Schritt auf ein benachbartes Feld bewegen können. Es dürfen auch mehrere Figuren auf demselben Feld stehen. Der Schwellwert für die Zustandsabweichung ε sei definiert als: *höchstens eine Figur darf höchstens um ein Feld in den unterschiedlichen Client-Repräsentationen abweichen.* Wie die Abbildung 6.1 zeigt, kann bei einer beliebig langen Laufzeit keine elastische Konsistenz garantiert werden, weil die tatsächliche Abweichung den festgelegten Schwellwert übersteigt.

□

Um zu zeigen, dass es Anwendungsszenarien gibt, die dem Modell der elastischen Konsistenz genügen, werden im Folgenden zwei positive Beispiele vorgestellt.

Abbildung 6.1. Screenshots von zwei Clients

Beispiel 6.4 (Elastische Konsistenz ist erfüllt (1)). Das vorhergehende Beispielszenario (Beispiel 6.3) wird um einen Token erweitert, welcher den wechselseitigen Ausschluss sicherstellt, so dass nur ein Spieler seine Figur bewegen kann. Der Spieler mit dem Token kann seine Figur um ein Feld bewegen. Nach jedem Zug versendet der Spieler Aktualisierungen an alle Mitspieler und wartet anschließend auf alle Bestätigungen. Nach dem Eintreffen aller Bestätigungen, kann der Spieler entweder den Token freigeben oder den nächsten Schritt machen. Hier wird auch bei einer beliebig langen Laufzeit die elastische Konsistenz eingehalten, weil für alle Spieler der Schwellwert der Abweichung ε nicht überschritten wird, obschon das Warten auf alle Antworten u.a. lange dauern kann ($\tau \geq \iota$).

□

Das Beispiel 6.4 veranschaulicht den ersten Teil der Definition der elastischen Konsistenz ($\|Z(c_i, t) - Z(c_j, t)\| < \varepsilon$). Das nachfolgende Beispiel adressiert hingegen den zweiten Teil der Definition ($\exists\ |\tau| < \iota : Z(c_i, t) = Z(c_j, t + \tau)$).

Beispiel 6.5 (Elastische Konsistenz ist erfüllt (2)). In diesem Szenario kann jeder Spieler seine Figur beliebig (auch in zweier Schritten) bewegen, falls er im Besitzt des Tokens ist. Unter der Annahme, dass jede Bewegung an alle anderen Mitspieler in einer beschränkten Zeit $\tau < \iota$ übertragen und auf dem Bildschirm dargestellt wird, kann auch hier die elastische Konsistenz garantiert werden.

□

In einer virtuellen Umgebung würde ein Spieler jedoch oftmals nicht warten, bis er den Token erhält, um seine Figur ziehen zu können, sondern schon vor der Token-Freigabe ziehen wollen. Um bei dem betrachteten Beispielszenario eine gewisse Interaktivität zu erlauben, bedarf es einer weiteren Erweiterung, die im nächsten Beispiel vorgestellt wird.

Beispiel 6.6 (Elastische Konsistenz ist verletzt (2)). Um die Interaktivität (zumindest eingeschränkt) zu erlauben, soll es möglich sein mit einer Wahrscheinlichkeit p eine Token-Kopie weiter zu geben. Der Spieler, der die Kopie erhält, macht mit der Wahrscheinlichkeit q einen Schritt und ist dann gezwungen die Kopie zu verwerfen. Da sich ansonsten nichts ändert, bewegt sich der Token-Eigentümer exakt genau so wie in den Beispielen davor. Aus diesem Grund kann in diesem Beispiel die elastische Konsistenz mit einer Wahrscheinlichkeit $P = pq$ verletzt werden.

□

Dieses Beispiel verletzt zwar die elastische Konsistenz, eröffnet aber gleichzeitig den Zugang zu einem weiteren Konsistenzmodell – der *statistisch elastischen Konsistenz.*

Definition 6.7 (Statistisch elastische Konsistenz). *Ein Zustandsraum ist* ***statistisch elastisch konsistent****, wenn für Zustände aller Clients in derselben AoI gilt:*

$$\forall\ t, i, j\ \exists\ |\tau| < \iota,\ 0 < \alpha \leq 1 :\ P\left(\delta = \|Z(c_i, t) - Z(c_j, t+\tau)\| < \varepsilon\right) \geq \alpha \quad (6.5)$$

□

Gemäß dieser Definition wird die Einhaltung der Bedingungen der elastischen Konsistenz nicht mehr gefordert, sondern lediglich mit einer festgelegten Mindestwahrscheinlichkeit erwartet ⇒ statistisch elastische Konsistenz.

6.2 Konsistenzmodelle und die Interaktivität

In diesem Kapitel werden an einem Beispielszenario die Eigenschaften unterschiedlicher Konsistenzverfahren gegenübergestellt. Dabei soll insbesondere der Einfluss des gewählten Konsistenzmodells auf die Interaktivität hervorgehoben werden.

Die besonderen Interaktivitätsanforderungen in den virtuellen Umgebungen liegen darin, dass direkt nach der Eingabe einer Benutzeraktion eine Systemrückmeldung erwartet wird. Die Systemantwortzeit, die von dem Zeitpunkt der Benutzereingabe bis zur Darstellung des Ergebnisses verstreicht, wird im weiteren Verlauf durch die Variable T ausgedrückt. Aus Sicht eines Nutzers ist die Systemantwortzeit T sogar wesentlich immersiver als die Beobachtungszeit τ einer Abweichung δ. Deshalb soll T ähnlich wie τ die Interaktivitätsschranke ι nicht überschreiten.

Im Folgenden wird der Zusammenhang zwischen der Wahl eines Konsistenzmodells und der entsprechenden Systemantwortzeit T am Beispiel des wechselseitigen Ausschlusses erklärt. Beim wechselseitigen Ausschluss wird gefordert, dass sich zu jeder Zeit höchstens ein Nutzer im kritischen Abschnitt befindet (ε).

Das strengste in dieser Arbeit vorgestellte Konsistenzmodell ist die absolute Konsistenz (Definition 2.2). Überträgt man diese Definition auf die Notation der elastischen Konsistenz, so wird sowohl $\tau = 0$ als auch $\delta = 0$ gefordert. Das heißt, jeder Nutzer hat zu jedem Zeitpunkt exakt die gleiche Zustandsrepräsentation. Um die Forderung $\delta = 0$ einhalten zu können, wäre ein unverhältnismäßig großer Synchronisationsaufwand notwendig, welcher die Interaktivität der Umgebung merklich einschränken würde. Da darüber hinaus, aufgrund der inhärenten Netzwerklatenz, Zustandsaktualisierungen nicht gleichzeitig ($\tau = 0$) über viele Knoten im Internet verteilt werden können, spielt dieses Modell in der Praxis keine Rolle.

Das strengste in virtuellen Umgebungen anwendbare Konsistenzmodell ist die elastische Konsistenz. Die Intention des Modells der elastischen Konsistenz liegt darin, den Algorithmen-Designern durch die Zulassung einer tolerierbaren Abweichung ($\delta < \varepsilon$) bzw. durch die kurzzeitige ($\tau < \iota$) Zulassung einer großen Abweichung ($\delta \geq \varepsilon$) mehr Spielraum zu geben, um Algorithmen zu entwickeln, die im Vergleich zu den traditionellen Algorithmen bessere Laufzeiten haben und somit den Interaktivitätsanforderungen von virtuellen Umgebungen gerecht werden. Der

Einfluss dieses Konsistenzmodells auf die Interaktivität wird am Beispiel des verteilten Algorithmus von Maekawa erläutert, obwohl dieser strenger ist als von der elastischen Konsistenz gefordert[1].

Die Abbildung 6.2 zeigt ein Szenario, bei dem der Client c_1 in den kritischen Abschnitt eintreten will und gemäß dem Algorithmus von Maekawa eine Wählermenge kontaktiert. Nach dem Erhalt aller Antworten tritt der Client in den kritischen Abschnitt ein und benachrichtigt die anderen Clients über seine Zustandsänderung. Die Tatsache, dass diese Benachrichtigung eine Zeit τ in Anspruch nimmt, ist nicht so kritisch, weil die Zustandsabweichung δ[2] die zulässige Schranke ε nicht überschreitet. Jedoch kann die Anwendung eines verteilten Algorithmus nach einer

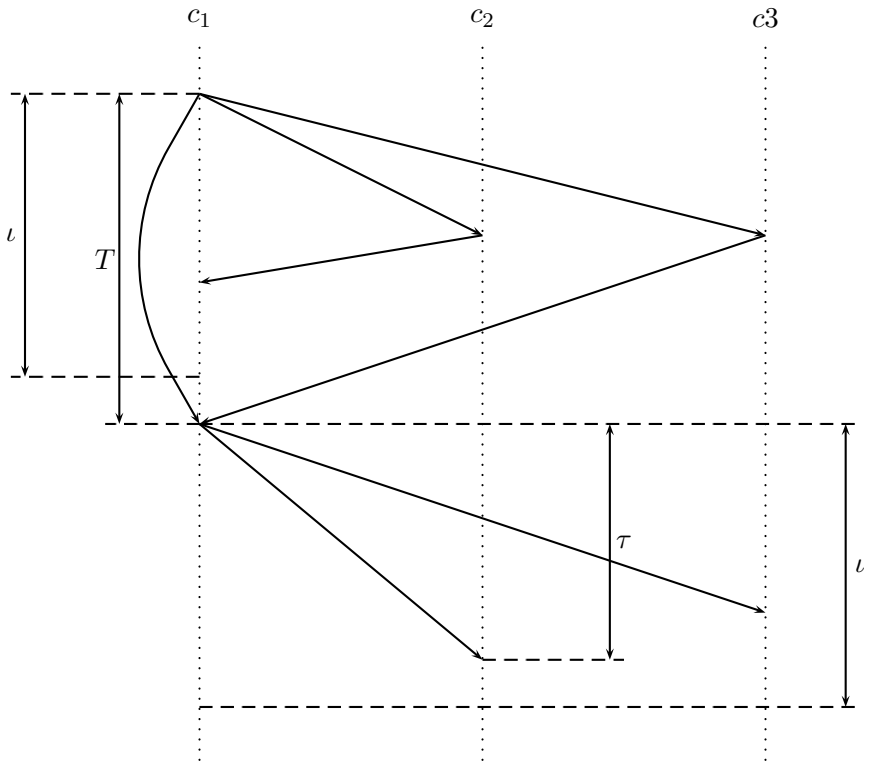


Abbildung 6.2. Elastische Konsistenz und die Systemantwortzeit

Benutzereingabe dazu führen, dass je nach Rahmenbedingungen[3] die Interaktivität der Umgebung merklich eingeschränkt wird $T > \iota$. Zusammenfassend kann man sagen, dass der Einsatz von traditionellen (verteilten) Konsistenzverfahren (z.B. Algorithmus von Maekawa) zwar die elastische Konsistenz garantiert, gleichzeitig aber an den strengen Interaktivitätsanforderungen $T < \iota$ scheitern kann.

[1] Der Algorithmus von Maekawa stellt zu jedem Zeitpunkt sicher, dass höchstens ein Nutzer den kritischen Abschnitt betritt. Die elastische Konsistenz würde kurzzeitig ($\tau < \iota$) zulassen, dass mehrere Nutzer den kritischen Abschnitt betreten.

[2] c_1 ist im kritischen Abschnitt und c_3 weiß nichts davon.

[3] Die Netzwerkeigenschaften und die Größe der Wählermenge sind die Faktoren, die für die Systemantwortzeit hier von der Bedeutung sind.

Um die Interaktivität einer virtuellen Umgebung nicht zu verletzen, wird in der Praxis oft das optimistische Transaktionsverfahren Time-Warp angewandt. Dabei werden kritische Operationen ohne Konflikt-Checks ausgeführt, um das geforderte Antwortzeitverhalten einzuhalten. Die Abbildung 6.3 zeigt ein Szenario, bei dem ein Nutzer, der in den kritischen Abschnitt eintreten will, die Aktion lokal ausführt und seinen Mitspielern die Zustandsänderung mitteilt. Die schnelle

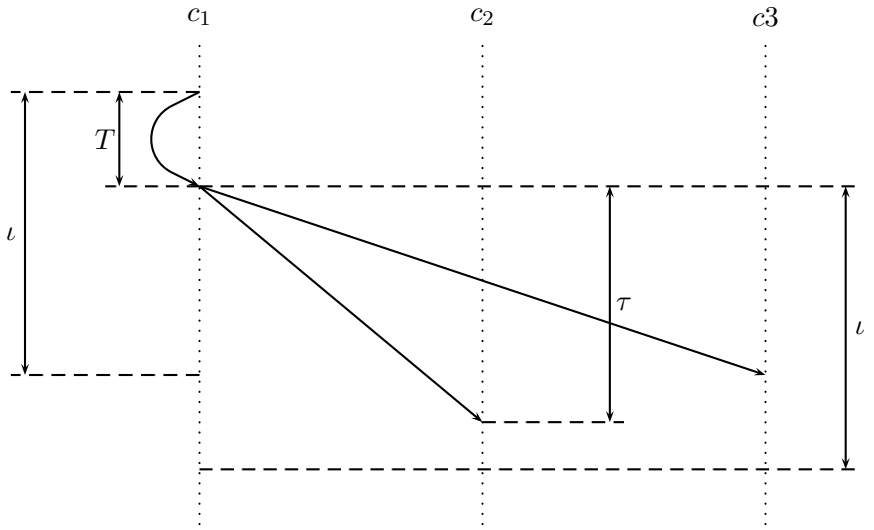


Abbildung 6.3. Optimistische Transaktionen und die Systemantwortzeit

lokale Ausführung von Zustandsänderungen stellt sicherlich einen Vorteil von optimistischen Verfahren dar. Ein großer Nachteil dieser Verfahren liegt allerdings darin, dass aufgrund fehlender Konflikt-Überprüfungen mehrere Nutzer gleichzeitig in den kritischen Abschnitt eintreten können. Ein optimistisches Verfahren kann somit keine elastische Konsistenz garantieren, da es weder die tatsächliche Abweichung ($\delta < \varepsilon$) noch die Beobachtungszeit ($\tau < \iota$) einschränken kann. Kommt es dabei zu einer Konsistenzverletzung, wird ein Rollback-Verfahren angewandt, das die Konsistenz durch das Zurücksetzen der Zustände wiederherstellt. Da solche Zustandskorrekturen von Nutzern deutlich wahrgenommen und als störend empfunden werden, sollten diese nach Möglichkeit gänzlich vermieden werden oder zumindest seltener auftreten.

Im Gegensatz zu den optimistischen Verfahren, die Benutzeraktionen ohne Konsistenzprüfung direkt ausführen, wird bei den Verfahren, die die statistisch elastische Konsistenz garantieren, ein vertretbarer Aufwand in die Konsistenzkontrolle investiert und dadurch die Wahrscheinlichkeit für das Auftreten von inkonsistenten Zuständen und somit die Häufigkeit der Zustandskorrekturen wie gewünscht reduziert. Darüber hinaus ermöglichen die Verfahren der statistisch elastischen Konsistenz eine kontinuierliche Einhaltung der Interaktivitätsschranke $T < \iota$, weil die Einhaltung der Konsistenzbedingungen $\delta < \varepsilon \vee \tau < \iota$ nur mit einer Mindestwahrscheinlichkeit erwartet und nicht wie bei der elastischen Konsistenz immer gefordert wird.

Abbildung 6.4 zeigt ein Szenario, bei dem der Client c_1 vor dem Eintritt in den kritischen Abschnitt einen Algorithmus anstößt und die Entscheidung einzutreten innerhalb der Interaktivitätsschranke $T < \iota$ trifft. Das Ziel bei der Definition bzw.

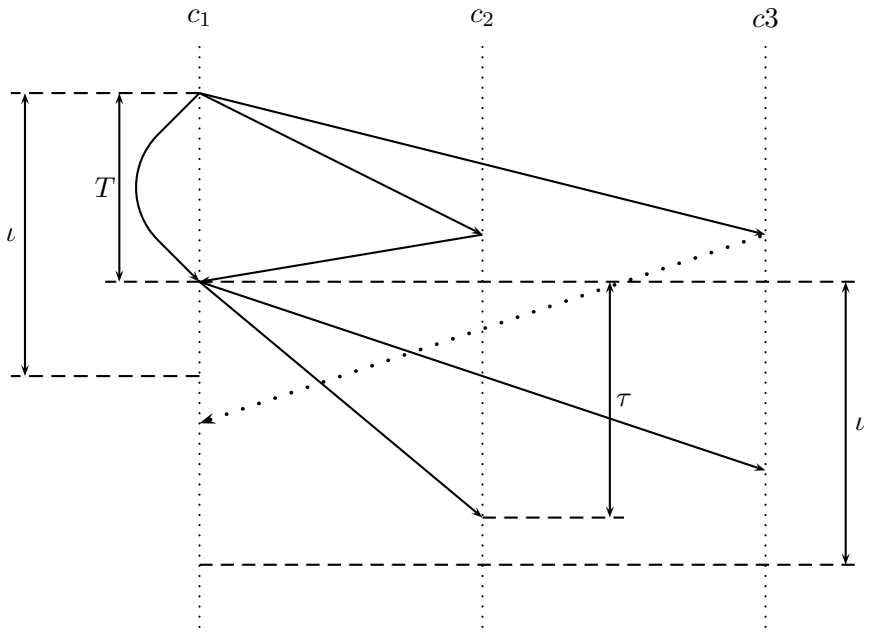


Abbildung 6.4. Statistisch elastische Konsistenz und die Systemantwortzeit

beim Entwurf von Verfahren und Algorithmen, die die statistisch elastische Konsistenz implementieren, sollte also sein, die geforderte Interaktivität zu gewährleisten und dabei eine festgelegte statistische Mindestwahrscheinlichkeit α für die statistisch elastische Konsistenz zu garantieren.

Das Laufzeit- bzw. Antwortzeitverhalten von Konsistenzalgorithmen in virtuellen Umgebungen hängt von vielen Faktoren ab, wie z.B. Nutzer- bzw. Objekt-Dichte. Aber auch technische Aspekte wie die Rechnerleistung bzw. die Netzwerklatenz können das Antwortzeitverhalten beeinflussen. Es ist also sinnvoll, diese Aspekte schon im Entwurf von entsprechenden Konsistenzverfahren zu berücksichtigen.

Die Faktoren, die auf das Antwortzeitverhalten Einfluss nehmen können, werden in Form eines CRT-Parameters mit einem stetigen Wertebereich $0 \leq CRT \leq 1$ ausgedrückt. Die Berechnung dieses Parameters auf der Basis unterschiedlicher Faktoren mittels eines Fuzzy-Logik-Systems ist im Anhang in Kapitel A beschrieben. Das Verhalten von Konsistenzalgorithmen, die die statistisch elastische Konsistenz garantieren sollen, soll also in Abhängigkeit vom Wert des CRT-Parameters an die Gegebenheiten angepasst werden.

$CRT = 1$ würde bedeuten, dass das Verfahren innerhalb der vorgegebenen Zeitschranke und mit der Wahrscheinlichkeit $P = 1$ die elastische Konsistenz garantieren kann. Somit wäre das Verfahren elastisch konsistent.

$CRT = 0$ würde hingegen heißen, dass innerhalb der vorgegebenen Zeitschranke weder elastische noch statistisch elastische Konsistenz garantiert werden können. Das Algorithmus-Verhalten könnte mit dem einer optimistischen Transaktion verglichen werden, bei der die Ausführung direkt stattfindet, ohne jeglichen algorithmischen Aufwand bzw. ohne jegliche Konfliktüberprüfung.

$CRT = \psi$ mit $0 < \psi < 1$ würde somit ein Verfahren darstellen, das die statistisch elastische Konsistenz garantieren soll und das sein Verhalten in Abhängigkeit vom Wert ψ des CRT-Parameters anpasst. Somit verhält sich beispielsweise dasselbe Verfahren bei $CRT = 0.4$ anders als bei $CRT = 0.6$. Bei $CRT = 0.6$ wird das Verfahren mit einer höheren Wahrscheinlichkeit die statistisch elastische Konsistenz einhalten als bei $CRT = 0.4$. Clients innerhalb einer AoI können aufgrund unterschiedlicher Randbedingungen unterschiedliche CRT-Werte berechnen und somit unterschiedliches Algorithmus-Verhalten beobachten. Der Zusammenhang zwischen dem Wert des CRT-Parameters $CRT = \psi$ und der Mindestwahrscheinlichkeit α wird in Abschnitt 6.5 betrachtet.

In dieser Arbeit wird die These vertreten, dass die Algorithmen der statistisch elastischen Konsistenz kürzere Laufzeiten im Vergleich zu den Algorithmen der elastischen Konsistenz haben und dadurch für den Einsatz in den virtuellen Umgebungen besser geeignet sind. Diese Annahme wird im nächsten Abschnitt mithilfe von Simulationen evaluiert.

6.3 Statistisch elastischer wechselseitiger Ausschluss

In der HERATRONos-Spielanwendung können Avatare konkurrierend auf Power-Up-Objekte zugreifen, um sich dadurch bestimmte Vorteile zu verschaffen. Damit aber ein Power-Up nur ein einziges Mal und (nach Möglichkeit) nur von einem Avatar verwendet werden kann, müssen die Zugriffe darauf synchronisiert werden, das heißt unter wechselseitigem Ausschluss stattfinden.

In diesem Kapitel wird eine Erweiterung des Maekawa-Algorithmus vorgeschlagen, die basierend auf dem Modell der statistisch elastischen Konsistenz den wechselseitigen Ausschluss garantiert.

Im Gegensatz zu Techniken wie Time-Warp, die üblicherweise in Anwendungsszenarien mit hohen Interaktivitätsanforderungen zum Einsatz kommen und die Benutzeraktionen ohne Konsistenzprüfung direkt ausführen, wird bei dem CRT-Parameter-basierten wechselseitigen Ausschluss, den wir fortan λ_{CRT}-Heuristik nennen, ein vertretbarer Aufwand in die Konsistenzkontrolle investiert und dadurch eine Mindestwahrscheinlichkeit α für die statistisch elastische Konsistenz garantiert.

Sollte es beim CRT-basierten wechselseitigen Ausschluss zu Inkonsistenzen kommen, so ist die Behandlung dieser Inkonsistenzen anwendungsabhängig. Beispielsweise können hier – wie bei Time-Warp – die Ereignisse bis zum Zeitpunkt der Benutzereingabe rückgängig gemacht werden. Der Grund, warum beim clientbasierten Konsistenz-Handling ein vertretbarer Aufwand in die Vermeidung von Inkonsistenzen investiert werden sollte, liegt darin, dass die Wiederherstellung eines

konsistenten Zustandes verteilt weitaus schwieriger ist als mit einer zentralen Kontrollinstanz. Deshalb sind solche Verfahren wie Time-Warp, die Aktionen optimistisch ohne Konfliktprüfung ausführen und beim Vorliegen einer kritischen Situation zur Verletzung der elastischen Konsistenz ohne jegliche Konsistenzgarantie führen, für den verteilten Fall weniger geeignet.

Im Gegensatz zu den bekannten Algorithmen des verteilten wechselseitigen Ausschlusses geht es beim parametrisierten Algorithmus (λ_{CRT}-Heuristik) darum, einen schnellen Zugriff zu ermöglichen, der mit großer Wahrscheinlichkeit konsistent ist. Das heißt, mit großer Wahrscheinlichkeit befindet sich nur ein einzelner Avatar im kritischen Abschnitt und greift auf das Power-Up-Objekt zu. Da aber nicht garantiert werden kann, dass sich kein zweiter Avatar im kritischen Abschnitt befindet, kann man in diesem Zusammenhang nicht mehr vom wechselseitigen Ausschluss sprechen. Stattdessen wird versucht, einen statistisch elastischen wechselseitigen Ausschluss zu erreichen.

Der erste Ansatz, Consistency-Responsiveness-Trade-off anzuwenden, ist in [Schloss et al., 2009] beschrieben. Dabei wird ein klassischer Algorithmus zum wechselseitigen Ausschluss von Maekawa [Maekawa, 1985] mit einem statistisch elastischen Algorithmus (λ_{CRT}-Heuristik) bezüglich der Konsistenzgüte und Latenzzeiten verglichen. Wie in Abschnitt 2.4.1 beschrieben, sendet ein Nutzer, der in den kritischen Abschnitt eintreten will, bei der Anwendung des Maekawa-Algorithmus eine Anfrage an $\sqrt{2}\sqrt{N}$ andere Nutzer. Nutzer, die sich selbst im kritischen Abschnitt befinden oder Informationen haben, dass ein anderer Nutzer sich dort befindet, puffern die Nachricht und senden eine Bestätigung erst dann, wenn der Abschnitt wieder verlassen wurde. Nutzer, die keine Informationen darüber haben, dass der kritische Abschnitt belegt ist, senden sofort eine Bestätigung. Erst nach Erhalt aller Bestätigungen darf der Nutzer den kritischen Abschnitt betreten. Aus Sicht eines Nutzers sind die Latenzzeiten gemäß dem *Power-Law* verteilt [Zhang et al., 2005]. Diese Verteilung besagt, dass es von einem Nutzer aus gesehen viele Nutzer mit kurzen Latenzzeiten und nur wenige mit langen Latenzzeiten gibt. Eine Auswahl von $\sqrt{2}\sqrt{N}$ Nutzern kann unter Umständen Nutzer mit langen Latenzzeiten enthalten. Deshalb kann das Warten auf alle $\sqrt{2}\sqrt{N}$ Bestätigungen Verzögerungen der Systemantwortzeiten verursachen.

Im Folgenden wird die λ_{CRT}-Heuristik genauer beschrieben. Entsprechend der hohen Anforderungen an die Interaktivität wurde der Algorithmus von Maekawa zu der λ_{CRT}-Heuristik wie folgt modifiziert: Es werden weiterhin Anfragen an $\sqrt{2}\sqrt{N}$ Nutzer versendet. Nutzer, die sich selbst im kritischen Abschnitt befinden oder Informationen haben, dass ein anderer Nutzer sich dort befindet, versenden eine Ablehnungsnachricht (*nein*). Eine Ablehnungsnachricht veranlasst den Nutzer seine Aktion abzubrechen[4] und evtl. zu einem späteren Zeitpunkt einen erneuten Versuch zu starten. Nutzer, die keine Informationen darüber haben, dass der kritische Abschnitt belegt ist, senden direkt eine Bestätigung (*ja*). Eine Eintrittsentscheidung wird bereits nach Erhalt der ersten $CRT \cdot \sqrt{2}\sqrt{N}$ Nachrichten getroffen. Befindet sich darunter keine Ablehnungsnachricht, so tritt der Nutzer

[4] Das heißt, ein inkonsistenter Zugriff wurde verhindert.

in den kritischen Abschnitt ein, andernfalls nicht. Durch die Reduktion der Anzahl der Antworten, die zur Entscheidungsfindung notwendig sind, werden auch die Verzögerungen der Systemantwortzeiten reduziert, weil man so die Nutzer mit langen Antwortzeiten aus der Menge der Bestätigungen (Reply Set) ausschließt. Die Abbildung 6.5 macht deutlich, dass wenn der Prozess 5, der in der Schnittmenge der beiden Wählermengen der Prozesse 3 und 6 liegt, lange Antwortzeiten aufweist, seine Antwort aufgrund der Skalierung mit dem CRT-Parameter gar nicht berücksichtigt wird. Aus diesem Grund kann es zur Verletzung des wechselseitigen Ausschlusses kommen.

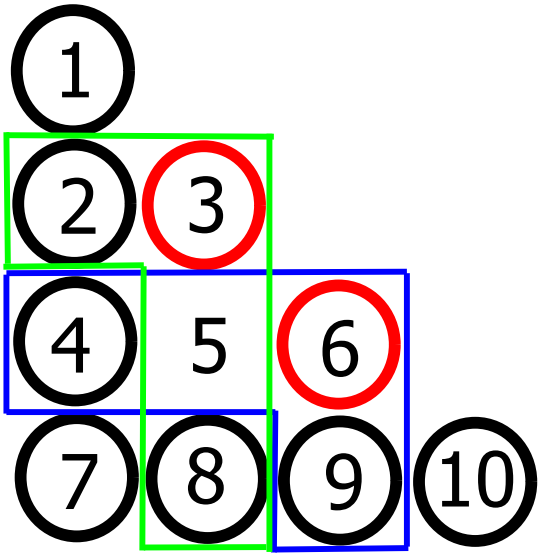


Abbildung 6.5. Ausschluss aus der Antwortmenge aufgrund langer Antwortzeiten

Sei nun $\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon$ (i,j,t,τ gemäß der Definition 6.7) ein *inkonsistentes* Ereignis, bei dem mindestens zwei Benutzer sich gleichzeitig im kritischen Abschnitt befinden. Im Folgenden wird untersucht, wie sich die Inkonsistenzwahrscheinlichkeit

$$P\left(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon\right)$$

bei der Anwendung der λ_{CRT}-Heuristik verhält, wenn die Anzahl der Nachrichten, die für den Eintritt in den kritischen Abschnitt notwendig sind, mit dem CRT-Wert skaliert wird. In diesem Fall entspricht die Inkonsistenzwahrscheinlichkeit der Wahrscheinlichkeit, $CRT \cdot \sqrt{2}\sqrt{N}$ Bestätigungen $(ja's)$ zu erhalten, obwohl mindestens eine der ausstehenden $\sqrt{2}\sqrt{N} - (CRT \cdot \sqrt{2}\sqrt{N})$ Antworten eine Ablehnungsnachricht $(nein)$ gewesen wäre.

Die Konsistenzwahrscheinlichkeit – der *Konsistenzgrad* – lässt sich aus der Inkonsistenzwahrscheinlichkeit wie folgt berechnen:

$$P(\|Z(c_i,t) - Z(c_j,t+\tau)\| < \varepsilon) = 1 - P(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon) \qquad (6.6)$$

Um die Inkonsistenzwahrscheinlichkeit $P(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon)$ zu berechnen, wird die *hypergeometrische Verteilung* angewandt, welche eine diskrete Wahrscheinlichkeitsverteilung für die Anzahl von erfolgreichen Ziehungen aus einer Stichprobe ohne Wiederholung beschreibt. Bei der Belegung $\{ja, ja, nein, ja, nein\}$ besteht die Antwortmenge aus $R = 5$ Antworten und davon sind $M = 3$ Bestätigungsnachrichten $(ja's)$. Die Stichprobe der Größe n enthält dann m Bestätigungsnachrichten mit der Wahrscheinlichkeit $P(X = m)$, die sich mit der hypergeometrischen Verteilung $H(R, M, n)$ wie folgt berechnen lässt:

$$P(X = m) = \frac{\binom{M}{m} \cdot \binom{R-M}{n-m}}{\binom{R}{n}} \tag{6.7}$$

Bei dem Versuch, die Wahrscheinlichkeit für einen inkonsistenten Zugriff zu bestimmen, liegen folgende Größen vor:

- $R = \sqrt{2}\sqrt{N}$ ist die Anzahl von Antworten, die vom Maekawa-Algorithmus erwartet werden.
- M ist die Anzahl der Bestätigungsnachrichten $(ja's)$ in der Probe. Somit beträgt die Anzahl der Ablehnungsnachrichten $(nein's)$ $R - M$.
- $n = CRT \cdot \sqrt{2}\sqrt{N}$ entspricht der Anzahl der Antworten, die von der λ_{CRT}-Heuristik zum Treffen einer Entscheidung benötigt werden.
- m ist die Anzahl der Bestätigungen $(ja's)$, die von der λ_{CRT}-Heuristik erhalten wurden.

Wie bereits beschrieben, kommt es dann zu Inkonsistenzen, wenn ein Nutzer $m = n$ Bestätigungsnachrichten $(ja's)$ erhält, obwohl unter den verbleibenden $R - n$ Nachrichten mindestens eine Ablehnungsnachricht (also $nein$) vorhanden war. Um dies zu modellieren, müssen daher alle Werte von M (Anzahl der Bestätigungen $(ja's)$ in der Menge aller Antworten) betrachtet werden, für die gilt: $n \leq M < R$. Das heißt, mindestens eine, jedoch maximal $R - n$ Ablehnungsnachrichten hätte ein Nutzer erhalten, wenn er auf alle Antworten gewartet hätte.

Unter der Annahme, dass alle Nutzer voneinander unabhängig in den kritischen Abschnitt eintreten möchten, enthält eine Antwortmenge der Länge R[5] M Bestätigungen mit der Wahrscheinlichkeit $P(Y = M)$, die sich mit der Binomialverteilung $B(R, p)$ als

$$P(Y = M) = \binom{R}{M} p^M (1-p)^{R-M}$$

berechnen lässt. Dabei repräsentiert die Zufallsgröße Y die Anzahl der „Erfolge“, bei denen eine Bestätigungsnachricht (ja) empfangen wurde, M steht für M-malige Wiederholung des zweipunktverteilten Versuchs und $p = \frac{M}{R}$ stellt die Erfolgswahrscheinlichkeit der Verteilung dar (z.B. 3 von 5 $ja's$ $p = 0.6$).

[5] Für große R kann die Binomialverteilung durch die Normalverteilung approximiert werden, weil gemäß dem *Satz von Moivre und Laplace* eine mit den Parametern R und p binomialverteilte Zufallsgröße X für große R näherungsweise normalverteilt ist, mit $EX = Rp$ und $Var(X) = Rp(1-p)$.

Betrachtet man die hypergeometrische Verteilung und die Binomialverteilung zusammen, so beträgt die *totale* Wahrscheinlichkeit, $m = n$ Bestätigungen ($ja's$) zu erhalten, obwohl es in der Menge der verbleibenden Nachrichten mindestens eine und höchstens $R - n$ Ablehnungsnachrichten gab:

$$\begin{aligned} P(X = (m = n)) = {} & P(X = (m = n)|Y = (M = R - 1)) \cdot P(Y = (M = R - 1)) \\ & + P(X = (m = n)|Y = (M = R - 2)) \cdot P(Y = (M = R - 2)) \\ & + \ldots + P(X = (m = n)|Y = (M = n)) \cdot P(Y = (M = n)). \end{aligned}$$

Die *bedingte* Wahrscheinlichkeit dafür, $m = n$ Bestätigungen aus einer Antwortmenge mit exakt M Bestätigungen und $R-M$ Ablehnungsnachrichten zu erhalten, kann mittels der hypergeometrischen Verteilung (Formel 6.7) berechnet werden. Wegen $m = n$ und $n \leq M < R$ ist hier sogar die folgende Vereinfachung möglich:

$$\begin{aligned} P(X = (m = n)|Y = M) &= \frac{\binom{M}{n} \cdot \binom{R-M}{n-n}}{\binom{R}{n}} = \frac{\binom{M}{n} \cdot \binom{R-M}{0}}{\binom{R}{n}} \\ &= \frac{\binom{M}{n} \cdot 1}{\binom{R}{n}} = \frac{M!}{n!(M-n)!} \cdot \frac{n!(R-n)!}{R!} \\ &= \frac{M!(R-n)!}{R!(M-n)!} \end{aligned} \tag{6.8}$$

Damit kann die Wahrscheinlichkeit $P(\|Z(c_i, t) - Z(c_j, t + \tau)\| \geq \varepsilon)$ wie folgt berechnet werden:

$$\begin{aligned} P(\|Z(c_i, t) - Z(c_j, t + \tau)\| \geq \varepsilon) &= P(X = (m = n)) \\ &= \sum_{j=n}^{R-1} \frac{j!(R-n)!}{R!(j-n)!} \cdot \binom{R}{j} \cdot \left(\frac{j}{R}\right)^j \cdot \left(1 - \frac{j}{R}\right)^{R-j} \end{aligned} \tag{6.9}$$

Wenn man bei einem Anwendungsszenario davon ausgeht, dass der kritische Abschnitt belegt ist, dann ist $P(\|Z(c_i, t) - Z(c_j, t + \tau)\| < \varepsilon)$ die Wahrscheinlichkeit dafür, dass der Nutzer die Belegung des kritischen Abschnittes erkennt und somit keinen inkonsistenten Zugriff durchführt. Um diese Berechnung besser nachvollziehen zu können, soll sie an dem folgenden Beispiel erörtert werden.

Beispiel 6.8 (Berechnung der Inkonsistenzwahrscheinlichkeit). Angenommen, es befinden sich 100 Nutzer in einer AoI. Beim Versuch, in den kritischen Abschnitt einzutreten, würde ein Nutzer $R = \sqrt{2}\sqrt{100} = 14.14 \approx 14$ andere Nutzer fragen und entsprechend dem Maekawa-Algorithmus auf genauso viele Antworten warten. Der Wert des CRT-Parameters (Abbildung A.5) bei einer Nutzeranzahl $N = 100$ beträgt ca. $CRT \approx 0.81$. Somit muss ein Nutzer gemäß dem modifizierten Algorithmus auf $n = CRT \cdot R = 0.81 \cdot 14 = 11.34 \approx 12$ Antworten warten. Gemäß Formel 6.9 wird die Berechnung folgendermaßen durchgeführt:

$$
\begin{aligned}
P(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon) &= P(X = 12) \\
&= \sum_{j=12}^{13} \frac{j!(14-12)!}{14!(j-12)!} \cdot \binom{14}{j} \cdot \left(\frac{j}{14}\right)^{j} \cdot \left(1 - \frac{j}{14}\right)^{14-j} \\
&= \frac{12!(14-12)!}{14!(12-12)!} \cdot \binom{14}{12} \cdot \left(\frac{12}{14}\right)^{12} \cdot \left(1 - \frac{12}{14}\right)^{14-12} \\
&+ \frac{13!(14-13)!}{14!(13-13)!} \cdot \binom{14}{13} \cdot \left(\frac{13}{14}\right)^{13} \cdot \left(1 - \frac{13}{14}\right)^{14-13} \\
&= \frac{2}{14 \cdot 13} \cdot \frac{14 \cdot 13}{2} \cdot 0.16 \cdot 0.02 + \frac{1}{14} \cdot 14 \cdot 0.38 \cdot 0.07 \\
&= 0.0032 + 0.027 \approx 0.03.
\end{aligned}
$$

Das heißt, würde man auf zwei Antworten weniger warten, wäre in diesem Fall die Inkonsistenzwahrscheinlichkeit 3%. Im Vergleich dazu würde ein Zugriff unter Anwendung eines Verfahrens, das keine statistisch elastische Konsistenz garantiert (z.B. Time-Warp), auf jeden Fall in einem inkonsistenten Zustand resultieren ($P(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon) = 100\%$).

□

6.4 Evaluationsergebnisse

Im vorhergehenden Abschnitt wurde gezeigt, wie der Algorithmus von Maekawa parametrisiert werden kann, um die Systemrückmeldung bzw. die Interaktivität, durch die Skalierung der Antwortmenge mit dem CRT-Parameter, zu verbessern.

Im speziellen Fall der Anpassung des Maekawa-Algorithmus gibt es eine weitere einfache Art und Weise, die Menge der Bestätigungen zu reduzieren, um bei akzeptablen Antwortzeiten noch einen vertretbaren Grad an Konsistenz zu garantieren. Die Idee hierbei liegt darin, immer auf eine konstante Anzahl an Antworten weniger zu warten, als man Anfragen versendet hat. Allgemein kann man sagen, dass bei $\sqrt{2}\sqrt{N}$ Anfragen ein Nutzer auf $\sqrt{2}\sqrt{N} - K$ Antworten mit $K \geq 1$ wartet. Diese Modifikation wird fortan als λ_{-K}-Heuristik bzw. „Minus K“-Ansatz bezeichnet. Bei der Evaluation wurde der Wert von $K = 5$ verwendet, wobei die Konstante K erst dann abgezogen wird, wenn $\sqrt{2}\sqrt{N} > K$ gilt.

Im weiteren Verlauf dieses Abschnitts werden der Algorithmus von Maekawa sowie die beiden Heuristiken λ_{CRT} und λ_{-K} bezüglich der Systemantwortzeit und Konsistenzwahrscheinlichkeit gegenübergestellt.

Im Laufe der angestellten Untersuchungen wurden zuerst die Inkonsistenz- bzw. die Konsistenzwahrscheinlichkeit für die beiden Heuristiken gemäß den Formeln 6.6 bzw. 6.9 für die variablen Nutzerzahlen $0 \leq N \leq 1000$ berechnet. Vollständigkeitshalber wurde auch die Konsistenzwahrscheinlichkeit $P(\|Z(c_i,t) - Z(c_j,t+\tau)\| < \varepsilon) = 1$ des Maekawa-Algorithmus[6] in der Abbildung 6.6 aufgeführt.

[6] Der Maekawa-Algorithmus garantiert, dass sich zu einem Zeitpunkt immer nur ein einziger Nutzer im kritischen Abschnitt befindet.

Wie man in der Abbildung 6.6 sehen kann, liegt bei kleinen Nutzerzahlen die Wahrscheinlichkeit, die Inkonsistenz zu erkennen, bei Anwendung der λ_{CRT}-Heuristik zwischen 80% und 95%. Im Gegensatz dazu fällt diese Wahrscheinlichkeit bei der λ_{-K}-Heuristik an der Unstetigkeitsstelle[7] auf ca. 11%, was nicht akzeptabel ist.

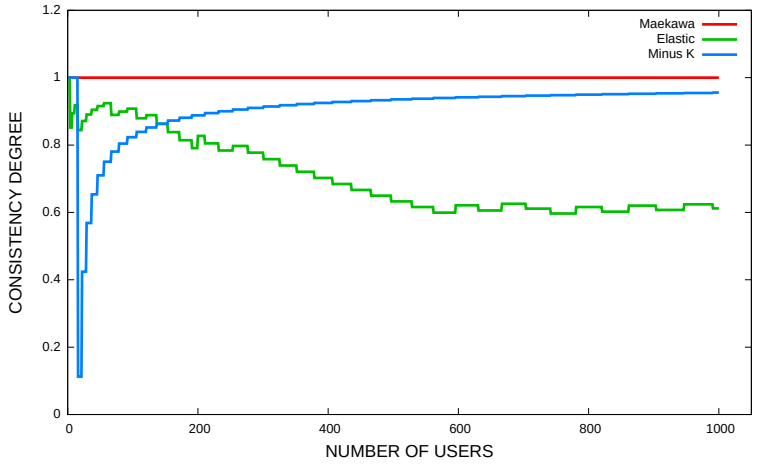


Abbildung 6.6. Gegenüberstellung von Konsistenzwahrscheinlichkeiten

Die Wahl eines kleineren Wertes für K würde zwar die Wahrscheinlichkeit an der Unstetigkeitsstelle etwas erhöhen, gleichzeitig würden aber auch die Antwortzeiten, insbesondere bei großen Nutzerzahlen, ansteigen, sodass die Antwortzeitschranke ι überschritten werden würde.

Bei großen Nutzerzahlen fällt die Konsistenzwahrscheinlichkeit bei der λ_{CRT}-Heuristik ab, bei der λ_{-K}-Heuristik steigt sie an. Da aber das Konsistenzverhalten beim „Elastic“-Ansatz eher der Erwartung der Nutzer entspricht[8], ist solches Konsistenzverhalten nach Meinung des Autors vorzuziehen.

Bei Anwendung der λ_{CRT}-Heuristik pendelt sich die Konsistenzwahrscheinlichkeit bei großen Nutzerzahlen bei ca. 60% ein. Das heißt, in dem betrachteten Szenario würde der Konsistenz-Schwellwert für λ_{CRT}-Heuristik bei $\alpha = 0.6$ und für λ_{-K}-Heuristik bei lediglich $\alpha = 0.11$ liegen. Optimistische Verfahren, die auf Interaktivität ausgelegt sind (Time-Warp), können überhaupt keine elastische Konsistenz garantieren ($\alpha = 0$). Somit liegt der Vorteil des proaktiven Konsistenz-Handlings, bei dem ein vertretbarer Aufwand vor der Ausführung einer Aktion investiert wird, auf der Hand.

Interessant sind in diesem Zusammenhang die Fragen, ob der Mehraufwand, der durch den λ_{CRT}-Ansatz hervorgerufen wird, angesichts der strengen Anforderungen an die Systemantwortzeiten gerechtfertigt ist und ob die beiden Heuristiken die Antwortzeitanforderungen erfüllen und somit statistisch die elastische Konsistenz garantieren. Diese Fragen werden im Folgenden beantwortet.

[7] Stelle, an der zum ersten Mal von der Kardinalität der Antwortmenge die Konstante K abgezogen wird.

[8] Nutzer würden bei großen Nutzerzahlen Inkonsistenzen eher tolerieren als bei kleinen.

Zur Gegenüberstellung der Systemantwortzeiten wurde zuerst die Anzahl der Antworten verglichen, die sowohl vom Originalalgorithmus als auch von den beiden Modifikationen erwartet werden, um in den kritischen Abschnitt eintreten zu dürfen. Wie die Abbildung 6.7 zeigt, brauchen die modifizierten Algorithmen weniger Nachrichten zur Entscheidungsfindung als der Originalalgorithmus. Darüber hinaus kann man erkennen, dass die λ_{-K}-Heuristik bei kleinen Nutzerzahlen weniger Antworten als die λ_{CRT}-Heuristik erwartet, was auch die geringe Konsistenzwahrscheinlichkeit erklärt (siehe Abbildung 6.6). Bei steigenden Nutzerzahlen fällt die Anzahl der Nachrichten, die von der λ_{CRT}-Heuristik erwartet werden und somit auch die Konsistenzwahrscheinlichkeit zu Gunsten einer konstant kleinen Systemantwortzeit (siehe Abbildung 6.8). Um eine Aussage über die

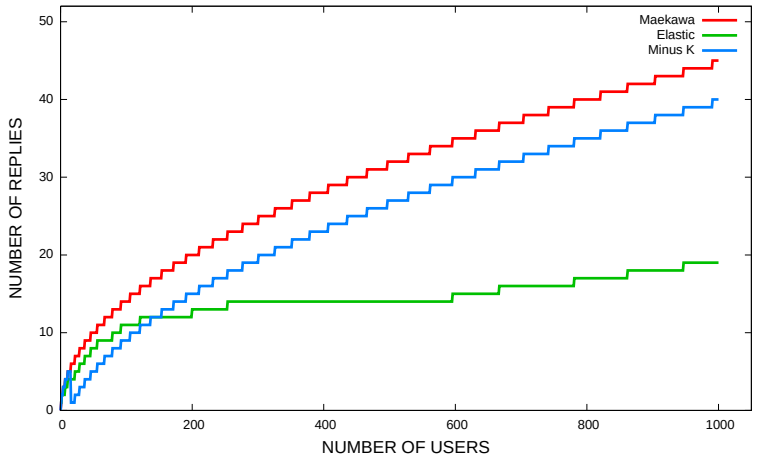


Abbildung 6.7. Benötigte Nachrichten

Systemantwortzeit-Vorteile der modifizierten Verfahren gegenüber dem Originalalgorithmus machen zu können, ist eine qualitative Messung bzw. Simulation der Latenzzeiten erforderlich. Dazu ist eine genaue Untersuchung der Internet-Topologie und der Latenzzeitverteilung notwendig.

Laut [Faloutsos et al., 1999] haben 75% der Knoten im Internet einen Grad an ausgehenden Kanten $d_{out} \leq 2$. Daraus folgt, dass der Internet-Router-Graph eine verhältnismäßig geringe Anzahl an Kanten besitzt. Da (wie bereits gesagt) aus der Sicht eines Knotens die Latenzzeiten gemäß dem Power-Law verteilt sind, gibt es viele Knoten mit geringen Latenzzeiten und nur einige wenige mit großen Latenzzeiten. Es soll also gemessen werden, wie viel Zeit man durch den Ausschluss der latenten Knoten gewinnt. Um die anfallenden Latenzzeiten möglichst genau zu messen, wurde die Latenzverteilung zwischen den Knoten im Internet simuliert und bei der Erzeugung von zufälligen Power-Law-verteilten Latenzzeiten die Rechenvorschrift von *Aaron Clauset* [Clauset et al., 2007] verwendet:

$$x = x_{min} \cdot (1 - r)^{-\frac{1}{(\alpha - 1)}}$$

Dabei ist r eine gleichverteilte Zufallszahl mit dem Wertebereich $0 \leq r \leq 1$ und $\alpha = 2.0$ eine Konstante. $x_{min} = 15$ ist die minimal angenommene Latenzzeit in Millisekunden.

Da r eine Zufallszahl ist, wäre eine einzige Messung nicht aussagekräftig. Aus diesem Grund wurden 100 Simulationsläufe durchgeführt und dabei jeweils die Latenzen ermittelt, die beim Warten auf die von dem entsprechenden Ansatz benötigte Anzahl von Antworten anfallen. Zum Schluss wurde das arithmetische Mittel aller Latenzen, die beim Warten auf diese bestimmte Anzahl von Nachrichten angefallen sind, ermittelt. Diese gemittelten Latenzen sind in der Abbildung 6.8 aufgeschlüsselt nach dem gewählten Ansatz dargestellt.

Die Abbildung 6.8 zeigt deutlich, dass die Reduzierung der Anzahl der zur Entscheidungsfindung benötigten Bestätigungsnachrichten zu einer wesentlichen Reduzierung der Latenzzeiten führt. Die beiden Modifikationen schneiden deutlich besser ab als der Originalalgorithmus. Bei einer Nutzeranzahl $N = 1000$ benötigt die λ_{CRT}-Heuristik 41 Millisekunden und ist somit fast 64-mal schneller als Maekawa (2586 Millisekunden) und immer noch um den Faktor 5 schneller als die λ_{-K}-Heuristik, die ihrerseits 211 Millisekunden benötigt.

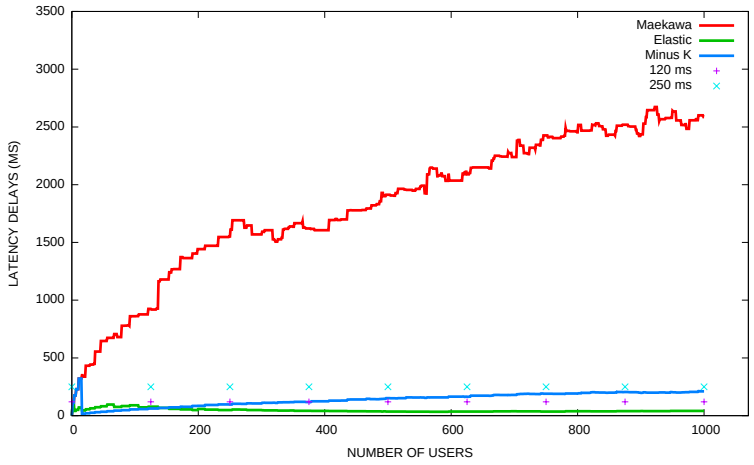


Abbildung 6.8. Gegenüberstellung von Latenzzeiten

Die λ_{-K}-Heuristik hat im Hinblick auf die Antwortzeiten den Nachteil, dass sie mit steigenden Nutzerzahlen nicht konstant bleibt, sondern steigt. Abgesehen davon steigt die ermittelte durchschnittliche Systemantwortzeit bei 15 Nutzern (Unstetigkeitsstelle) auf 321 Millisekunden, womit auch die Antwortzeitschranke ι überschritten wird. Da darüber hinaus an der Unstetigkeitsstelle auch nur eine geringe Mindestwahrscheinlichkeit $\alpha = 0.11$ garantiert werden kann, ist der Einsatz der λ_{-K}-Heuristik für das Konsistenz-Handling in virtuellen Umgebungen nicht geeignet, obwohl diese Heuristik den Anforderungen der statistisch elastischen Konsistenz genügt.

Die Systemantwortzeit der λ_{CRT}-Heuristik liegt unterhalb 120 Millisekunden und fällt mit steigenden Nutzerzahlen (bei konstant bleibender Konsistenzwahr-

scheinlichkeit) sogar leicht ab[9]. Die λ_{CRT}-Heuristik terminiert innerhalb der vorgegebenen Zeitschranke und kann die Einhaltung der elastischen Konsistenz mit einer akzeptablen Mindestwahrscheinlichkeit $\alpha = 0.6$ garantieren. Daraus folgt, dass die λ_{CRT}-Heuristik die Bedingung der statistisch elastischen Konsistenz erfüllt. Daher ist die λ_{CRT}-Heuristik für die Realisierung des clientbasierten wechselseitigen Ausschlusses geeignet.

Im nächsten Abschnitt wird die Bestimmung des CRT-Parameters ausgehend von einer festgelegten Mindestwahrscheinlichkeit angesprochen.

6.5 Zusammenhang zwischen CRT und α

Im Laufe des vorhergehenden Abschnitts wurde deutlich, wie sich der Wert des CRT-Parameters auf die Konsistenzwahrscheinlichkeit und das Antwortzeitverhalten auswirkt. Große CRT-Werte haben einen höheren Konsistenzgrad und längere Antwortzeiten zu Folge. Kleine hingegen führen dazu, dass zwar eine Entscheidung schnell getroffen werden kann, die Konsistenzwahrscheinlichkeit aber fällt. Im vorhergehenden Abschnitt wurde ein Szenario betrachtet, in dem beim Einsatz der λ_{CRT}-Heuristik die Mindestwahrscheinlichkeit $\alpha = 0.6$ garantiert und die Systemantwortzeit 120 Millisekunden nicht überschritten wurde. Am Beispiel der λ_{-K}-Heuristik wurde gezeigt, dass selbst bei großen Nutzerzahlen (N = 1000) die Konsistenzwahrscheinlichkeit von 95.0% garantiert werden kann und die Antwortzeitschranke $\iota = 250$ Millisekunden nicht überschritten wird. Also kann zur Steigerung der Konsistenzwahrscheinlichkeit unter Umständen auf weitaus mehr Antworten gewartet werden, als es bei der λ_{CRT}-Heuristik der Fall war, ohne die Systemantwortzeitbedingung zu verletzen.

Das Antwortzeit- und das Konsistenzgradverhalten der λ_{CRT}-Heuristik erlaubt es, anwendungsabhängig eine Mindestwahrscheinlichkeit (z.B. $\alpha = 0.95$) im Vorfeld festzulegen. Ausgehend davon kann der Wert des CRT-Parameters durch die Bildung einer Umkehrfunktion berechnet und zur Bestimmung der Mindestanzahl der Antworten, auf die gewartet werden muss, verwendet werden.

Die Frage lautet also, wie kann die folgende zweistellige Funktion nach dem CRT-Parameter umgestellt werden?

$$\begin{aligned} P(\|Z(c_i,t) - Z(c_j,t+\tau)\| \geq \varepsilon) &= F(R, CRT) \\ &= \sum_{j=R\cdot CRT}^{R-1} \frac{j!(R - R\cdot CRT)!}{R!(j - R\cdot CRT)!} \cdot \binom{R}{j} \cdot \left(\frac{j}{R}\right)^j \cdot \left(1 - \frac{j}{R}\right)^{R-j} \end{aligned}$$

Wie man in der Abbildung 6.6 erkennen kann, ist diese Funktion F nicht *injektiv*. Das heißt, aus der Gleichheit der Funktionswerte kann nicht auf die Gleichheit der Eingabewerte geschlossen werden. Somit ist die Funktion auch nicht *bijektiv* und die Bildung einer Umkehrfunktion auf dem analytischen Wege ist ausgeschlossen [Bartsch, 1999]. Die Frage, die sich in diesem Zusammenhang stellt

[9] Mit wachsender Wählermenge steigt die Wahrscheinlichkeit, Nutzer mit geringeren Latenzzeiten zu fragen.

und die in weiteren Arbeiten untersucht werden könnte, ist: Kann die Umkehrfunktion approximiert werden? Gegenwärtig wird davon ausgegangen, dass die Umkehrfunktion entweder mit Mitteln der numerischen Mathematik (Newton-Verfahren [Bartsch, 1999]) oder mit Mitteln der Linearen Algebra (LSI-Problem [Lawson und Hanson, 1995]) approximiert werden kann.

Eine solche Anpassung, hätte den Vorteil, dass man sowohl bei kleinen als auch bei großen Nutzerzahlen einen hohen Grad an Konsistenz garantieren könnte, ohne dabei die Antwortzeitbedingung zu verletzen. Die so bestimmte λ_{CRT}-Heuristik hätte keine Unstetigkeitsstelle, wie es bei der λ_{-K}-Heuristik der Fall war, und würde die Bedingung der statistisch elastischen Konsistenz stets erfüllen.

6.6 Statistisch elastischer Dead-Reckoning

Neben dem betrachteten Problem der Sicherstellung eines wechselseitigen Ausschlusses, existieren in DVEs weitere Konsistenzprobleme, auf die das Modell der statistisch elastischen Konsistenz angewendet werden kann. Bei der Implementierung der Verfahren, welche die statistisch elastische Konsistenz in DVEs garantieren, geht es nicht darum die aufgetretenen Inkonsistenzen zu behandeln[10], sondern darum die Inkonsistenzwahrscheinlichkeit zu minimieren. In diesem Abschnitt wird die Allgemeingültigkeit der Definitionen der elastischen bzw. der statistisch elastischen Konsistenz durch die Anwendung auf ein weiteres Konsistenzproblem nachgewiesen.

In einer virtuellen Umgebung kann es Anwendungsszenarien geben (z.B. ein Pferderennen), in denen Nutzer gegenseitig an aktuellen Positionen anderer Nutzer interessiert sind. Da es sich hierbei um dynamische Objekte (Teilnehmer des Rennens) handelt, die sich ständig bewegen und somit ihren Zustand kontinuierlich verändern, würde das ständige Versenden von Positionsupdates einen zu hohen Aufwand bedeuten. Außerdem würde auch das latenzbedingte Warten auf die Systemrückmeldung das Empfinden der Nutzer stören. Daraus resultiert die Notwendigkeit, im Intervall zwischen zwei Updates eines Opponenten seine Position mithilfe des Dead-Reckoning-Verfahrens (siehe Abschnitt 2.4.3) zu interpolieren. Die Aktualisierungsnachrichten werden also nur dann versendet, wenn sich eine für die Interpolation notwendige Größe (z.B. die Geschwindigkeit) ändert.

In LotRo-Forum[11] wird von einem Rennszenario berichtet, in dem es zu folgenden Inkonsistenzen kommt. Jeder Teilnehmer des Rennens führt in seiner lokalen Repräsentation (auf seinem Bildschirm) das Rennen an, weil seine Aktionen direkt und die Aktionen der Opponenten aufgrund der Netzwerklatenz zeitversetzt ausgeführt werden. Aufgrund dieser Tatsache gewinnt auch jeder Teilnehmer in seiner lokalen Ansicht das Rennen (siehe Abbildung 6.9). Da es aber nur einen Sieger geben kann, wird dieser anschließend vom System ermittelt, was dazu führt, dass Teilnehmer, die das Rennen lokal gewonnen haben dieses im Nachhinein doch noch

[10] Sollte es zu Inkonsistenzen kommen, werden diese anwendungsabhängig und mit bereits etablierten Verfahren, wie z.B. die lineare Konvergenz beim Dead-Reckoning, behandelt.

[11] `http://forums.lotro.com/showpost.php?p=3854771&postcount=40`

verlieren. Offensichtlich sagt das im Spiel eingesetzte Dead-Reckoning-Verfahren die Positionen nicht präzise genug voraus. Diese Tatsache trübt die Nutzererfahrung enorm, weil man als Führender in der lokalen Ansicht keine Aktionen ausführen kann, die zum Gesamtsieg (systemweit) verhelfen könnten.

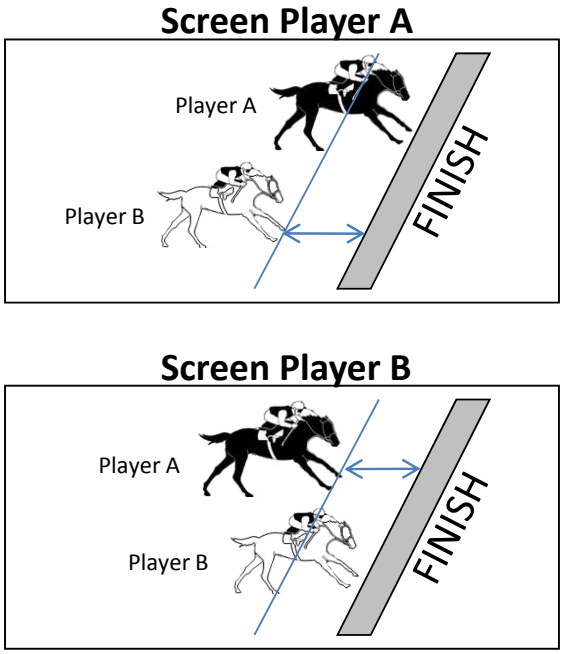


Abbildung 6.9. Bildschirmpräsentationen von zwei Teilnehmern

Im weiteren Verlauf wird eine Erweiterung des Dead-Reckoning-Verfahrens vorgestellt, welche die statistisch elastische Konsistenz mit einer bestimmten Mindestwahrscheinlichkeit garantieren kann.

Dazu müssen zuerst die Faktoren identifiziert werden, welche die Nutzererfahrung beeinflussen. Fakt ist, dass man an dem Vorhandensein der Latenzzeiten nichts ändern kann. Offensichtlich ist auch die Anwendung eines prädiktiven Ansatzes, wie Dead-Reckoning, aufgrund der hohen Interaktivität und der kontinuierlichen Veränderung gerechtfertigt.

Die Unzufriedenheit der Nutzer in dem aktuellen Beispielszenario rührt daher, dass man den eigenen Vorsprung deutlich wahrnimmt und die nachträgliche Degradierung zum Verlierer deshalb nicht hinnehmen will. Nun stellt sich die Frage, ob man das Nutzerempfinden zum Positiven beeinflussen kann, ohne die Invarianten des Modells zu verletzen und somit ohne die Interaktivität des Rennens einzuschränken.

Es ist erwiesen, dass die Menschen eine eingeschränkte Wahrnehmung von Zeit haben. Wenn die Übertragungs- bzw. die Ausbreitungsgeschwindigkeit höher als die Reaktionszeit des Menschen bzw. seine Interaktionsgeschwindigkeit ist, dann nimmt der Mensch keine Inkonsistenzen wahr und empfindet die Umgebung als konsistent. In einem realen Pferderennen ist die Ausbreitungsgeschwindigkeit

höher als die Interaktionsgeschwindigkeit. Das heißt, ein Teilnehmer eines realen Pferderennens nimmt die Bewegungen seiner Opponenten unmittelbar wahr. Das deutliche Wahrnehmen des eigenen Vorsprungs ist in vielen Sportarten u.a. in einem realen Pferderennen eher eine Seltenheit. Meistens kann man mit dem menschlichen Auge keine objektive Entscheidung darüber treffen, wer das Rennen gewonnen hat. In solchen Fällen wird zum Beispiel ein Fotofinish als objektive Entscheidungsinstanz eingesetzt.

In einer virtuellen Welt hingegen ist die Ausbreitungsgeschwindigkeit latenzbedingt geringer als die Interaktionsgeschwindigkeit, deshalb kommen die Bewegungsupdates zeitverzögert an. In einem virtuellen Pferderennen wird einem Teilnehmer aufgrund der inhärenten Netzwerklatenz also ein Gefühl vermittelt den Vorsprung abschätzen zu können, welches in der Wirklichkeit in vergleichbaren Szenarien oftmals gar nicht existiert hätte.

Um einen Teilnehmer eines Rennens nicht irre zu führen, muss in Analogie zum *Kalman-Filter*[12] [Kalman, 1960] unterschieden werden können, ob der Teilnehmer das Rennen tatsächlich anführt oder sein Vorsprung aufgrund der Netzwerklatenz zu Stande kommen könnte. Das vorgeschlagene Verfahren wird im Folgenden erläutert.

Beim Empfang einer Aktualisierungsnachricht, welche den Geschwindigkeitsvektor und den dazugehörigen Zeitstempel enthält, wird zuerst die Position zurückgerechnet, die zum Zeitpunkt des Stempels interpoliert war. Anschließend wird die Abweichung zwischen der interpolierten und der tatsächlichen Position berechnet. Anhand dieser Abweichungen wird der Erwartungswert für die Abweichung von der tatsächlichen Position geschätzt. Die Schätzfunktion für den Erwartungswert EX ist das **arithmetische Mittel** (Stichprobenmittel) $\overline{X}$ der beobachteten Abweichungen X_i:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{6.10}$$

Dann wird die Varianz der Abweichung bestimmt. Für die Varianz $Var(X)$ wird die mittlere quadratische Abweichung der X_i's vom Mittelwert $\overline{X}$ – **die Streuung** S^2 (Stichprobenvarianz) – verwendet:

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1} \tag{6.11}$$

Als Schätzung der Standardabweichung $\sigma = \sqrt{Var(X)}$ wird S verwendet. Für hinreichend große n liefern $\overline{X}$ und S^2 gute Näherungen für μ und σ^2.

Nun ist es so, dass die Latenzzeiten und somit die Abweichungen keineswegs immer konstant gleich sind. Bei der Interpolation in einer realen Anwendung reicht

[12] Der Kalman-Filter wird zum Entfernen der von den Messgeräten verursachten Störungen eingesetzt. Durch seinen Einsatz ist es möglich, aus den fehlerbehafteten Beobachtungen auf den exakten Zustand eines Systems zu schließen.

es also nicht, den Erwartungswert der entsprechenden Abweichung von der interpolierten Position abzuziehen, sondern es muss auch die entsprechende Varianzschätzung σ mitberücksichtigt werden. Das heißt, für die Darstellung ist nicht mehr die interpolierte Position, sondern der so genannte σ-Bereich von Bedeutung[13]. Setzt man $\varepsilon = \|\sigma\|$, dann kann man mit einer Mindestwahrscheinlichkeit $\alpha = 0.68$ erwarten, dass die tatsächliche Position eines Avatars innerhalb des entsprechend errechneten σ-Bereiches liegt:

$$P(\|Z(c_i, t) - Z(c_j, t + \tau)\| < \varepsilon = \|\sigma\|) \geq \alpha = 0.68$$

Bei $\varepsilon = 2\|\sigma\|$ bzw. $\varepsilon = 3\|\sigma\|$ steigt die entsprechende Wahrscheinlichkeit auf 0.95 bzw. 0.99. Somit erfüllt die vorgeschlagene Modifikation die Bedingung der statistisch elastischen Konsistenz.

Liegt eine Überlappung zwischen der eigenen Position und dem σ-Bereich eines anderen Teilnehmers vor, so werden die Teilnehmer auf der gleichen Höhe im Rennen dargestellt. Erst wenn keine Überlappung vorliegt, kann mit großer Wahrscheinlichkeit behauptet werden, dass der Vorsprung nicht latenzbedingt ist. Dieses Szenario ist in der Abbildung 6.10 anschaulich dargestellt.

Falls jetzt mehrere Spieler die Ziellinie gleichzeitig überqueren und ein Spieler das Rennen nicht gewinnt, ist seine Enttäuschung nicht so groß, als wenn er das Rennen lokal gewonnen hätte. Diese Entscheidung wird ähnlich wie beim Fotofinish von den Teilnehmern akzeptiert, weil sie mit ihrer menschlichen Wahrnehmung keine objektive Entscheidung treffen könnten. Leider war es aus Zeitgründen nicht möglich, entsprechende Verifikationen und Evaluationen durchzuführen. Es wird jedoch davon ausgegangen, dass der Verlust eines solchen Rennens nicht so kritisch wahrgenommen wird, als wenn man das Rennen lokal gewinnt, aber global verliert.

6.7 Schlussfolgerung

In der Abbildung 6.11 sind Konsistenzmodelle bezüglich der Konsistenzwahrscheinlichkeit $P(\|Z(c_i, t) - Z(c_j, t + \tau)\| < \varepsilon)$, der Systemantwortzeit der entsprechenden Algorithmen T in Millisekunden und der tolerierbaren Abweichung ε eingeordnet. Die Beobachtungszeit τ wird in dieser Abbildung vernachlässigt.

Das Modell der absoluten Konsistenz (siehe Definition 2.2) umfasst die traditionellen bzw. pessimistischen verteilten Konsistenzverfahren und ist so definiert, dass es zu jedem Zeitpunkt genau die gleiche Ansicht unter allen Nutzern garantiert und somit überhaupt keine Zustandsabweichungen zulässt ($\delta = 0$). Unter bestimmten Umständen (große Nutzerzahlen, etc.) würden die entsprechenden Verfahren aufgrund ihrer Laufzeitkomplexität und der Netzwerklatenz jedoch die vorgegebene Systemantwortzeitschranke überschreiten und sind aus diesem Grund nicht für das Konsistenz-Handling in DVEs geeignet.

[13] Eine Zufallsgröße liegt im 1σ-Bereich mit einer Wahrscheinlichkeit von 68%, im 2σ-Bereich mit einer Wahrscheinlichkeit von 95% und im 3σ-Bereich mit einer Wahrscheinlichkeit von 99%, unabhängig von den eigentlichen Werten von μ und σ.

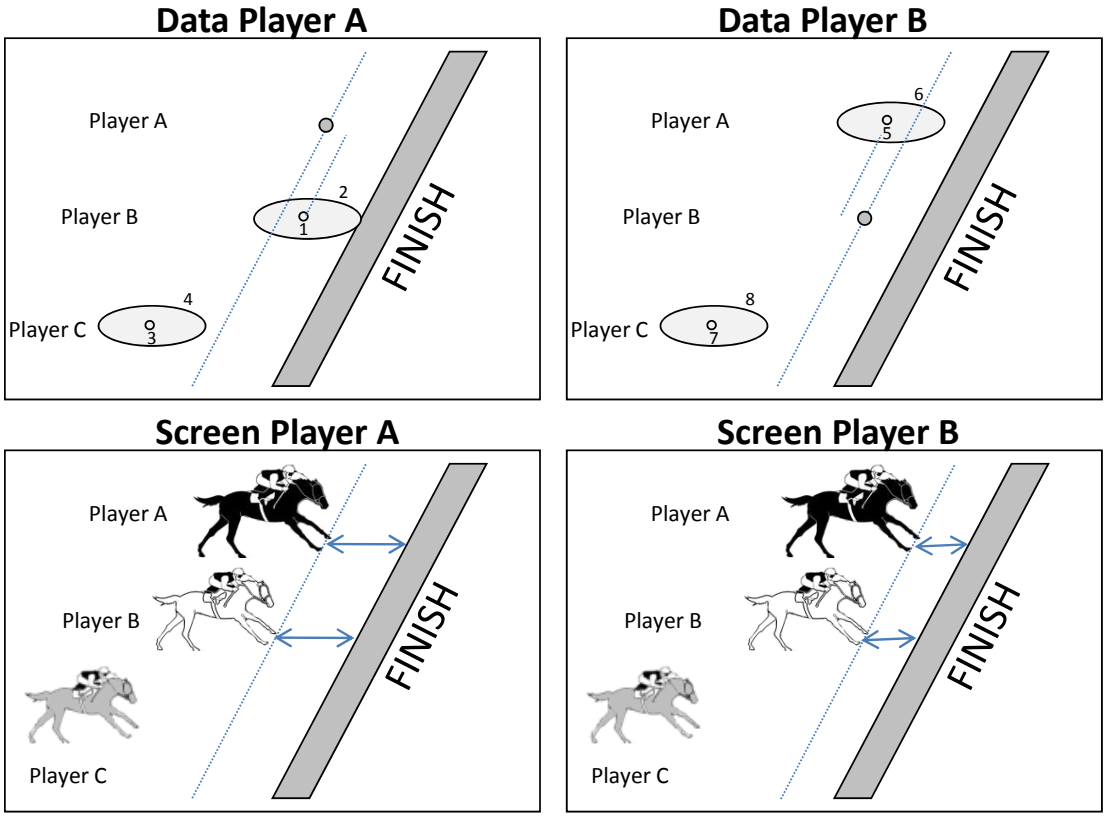


Abbildung 6.10. Bildschirmpräsentationen von zwei Teilnehmern (mit σ)

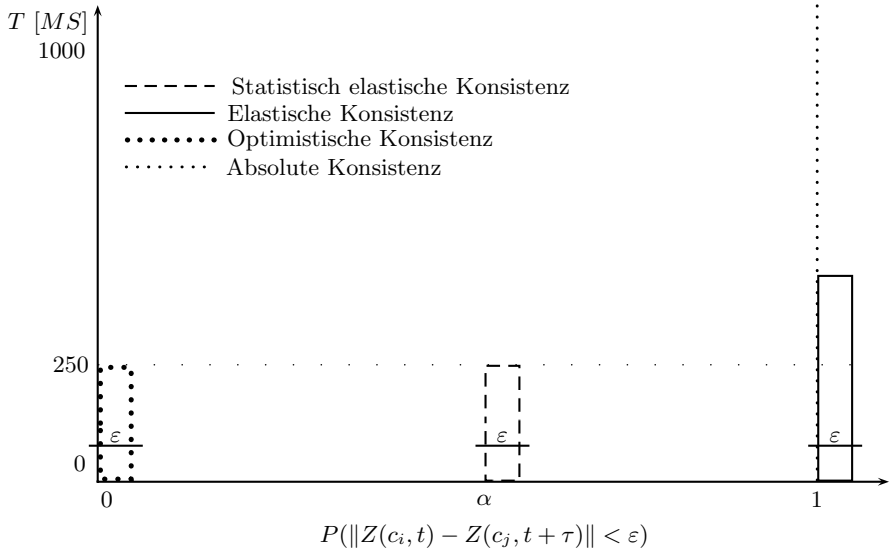


Abbildung 6.11. Garantierbare Konsistenz-Mindestwahrscheinlichkeiten

Die Intention des Modells der elastischen Konsistenz (siehe Definition 6.2) liegt darin, Abweichungen zu einem bestimmten Grad $\delta < \varepsilon$ zu tolerieren und dafür im Gegenzug das Laufzeitverhalten der entsprechenden Algorithmen im Vergleich zu den traditionellen Algorithmen zu verbessern. Die Einhaltung der Interaktivitätszeitschranke $\iota \approx 250$ kann jedoch nicht garantiert werden (Abbildung 6.11). Aus diesem Grund ist der Einsatz von Algorithmen der elastischen Konsistenz nur in den DVE-Szenarien mit weniger strengen Interaktivitätsanforderungen ratsam.

Weil optimistische Verfahren, wie Time-Warp oder Dead-Reckoning, keinerlei algorithmischen Aufwand (Konfliktüberprüfung) vor der Ausführung einer kritischen Operation erfordern, terminieren sie innerhalb der vorgegebenen Zeitschranke unabhängig von äußeren Einflüssen (Nutzerzahlen, Latenz, etc.). Aus diesem Grund kommen die optimistischen Verfahren verstärkt in den virtuellen Umgebungen zum Einsatz. Diese Verfahren werden bei der vorgenommenen Einordnung unter einem Kunstbegriff *optimistische Konsistenz* zusammengefasst. Beim Vorliegen einer kritischen Situation (z.B. zwei Nutzer wollen gleichzeitig in den kritischen Abschnitt eintreten) können optimistische Verfahren jedoch keine Konsistenz-Mindestwahrscheinlichkeit garantieren ($\alpha = 0$) und führen zwangsläufig zu Inkonsistenzen, die ihrerseits unerwünschte Rollback-Aktionen nach sich ziehen können.

Um die Häufigkeit von Zustandskorrekturen im Vergleich zu den optimistischen Verfahren zu minimieren, gleichzeitig aber die Interaktivität im Vergleich zu den elastischen und traditionellen Verfahren zu verbessern, wurde in dieser Arbeit das Modell der statistisch elastischen Konsistenz definiert (siehe Definition 6.7), das eine statistische Mindestwahrscheinlichkeit α für die Einhaltung der elastischen Konsistenz garantiert. Der Vorteil dieses Modells liegt darin, dass die entsprechenden Verfahren auf jeden Fall innerhalb der vorgegebenen Zeitschranke terminieren und mit einer Mindestwahrscheinlichkeit α die Abweichungsgrenze ε nicht übersteigen. Dadurch wird die Inkonsistenzwahrscheinlichkeit und somit die Häufigkeit der Rollback-Aktionen reduziert, was ohne jeden Zweifel die Nutzerzufriedenheit steigert.

In der Tabelle 6.1 sind die Konsistenzmodelle und die entsprechenden Konsistenz-Mindestwahrscheinlichkeiten (in absteigender Reihenfolge) zusammengefasst. Angesichts der Konsistenzwahrscheinlichkeiten und der Systemantwortzeiten der

Konsistenzmodell	Mindestwahrscheinlichkeit
Absolute Konsistenz	$P(\|Z(c_i,t) - Z(c_j,t)\| = 0) = 1$
Elastische Konsistenz	$P(\|Z(c_i,t) - Z(c_j,t+\tau)\| < \varepsilon) = 1$
Statistisch elastische Konsistenz	$P(\|Z(c_i,t) - Z(c_j,t+\tau)\| < \varepsilon) \geq \alpha$
Optimistische Konsistenz	$P(\|Z(c_i,t) - Z(c_j,t+\tau)\| < \varepsilon) \geq 0$

Tabelle 6.1. Konsistenzmodelle und Mindestwahrscheinlichkeiten

betrachteten Modelle kann behauptet werden, dass die Verfahren und Algorithmen, die das Modell der statistisch elastischen Konsistenz implementieren, für das verteilte clientbasierte Konsistenz-Handling in virtuellen Umgebungen am besten geeignet sind.

7

Zusammenfassung

Aufgrund ihrer dreidimensionalen und sehr realistischen Darstellungen vermitteln virtuelle Umgebungen den Nutzern das Gefühl einer sozialen und physischen Nähe und erfreuen sich in den letzten Jahren einer sehr großen Beliebtheit. So verfügt beispielsweise *World of Warcraft* über 11,5 Millionen Abonnenten. Die kontinuierliche Weiterentwicklung von virtuellen Umgebungen, die immer bessere Graphik und die stetig steigende Geschwindigkeit der Internetverbindungen sind nur einige begünstigende Faktoren dafür, dass die Nutzergemeinde von virtuellen Umgebungen weiterhin anwächst. Leider stoßen die virtuellen Umgebungen infrastrukturbedingt bereits heute an ihre Leistungsgrenzen. So kann ein Spiel-Server beispielsweise nur eine begrenzte Anzahl an Spielern bedienen. Außerdem können sich in einer virtuellen Region gleichzeitig nur begrenzt viele Nutzer befinden.

Um den genannten Lastverteilungs- und Skalierbarkeitsproblemen entgegenzuwirken, wurden in dieser Arbeit Aspekte eines möglichen Infrastrukturwechsels untersucht. Außerdem wurde evaluiert, ob die Clients einige Verwaltungsaufgaben eigenständig übernehmen können. Zu solchen Aufgaben zählen zum Beispiel das clientbasierte Konsistenz-Handling und die Bildung einer Multicast-Infrastruktur zur effizienten Gruppenkommunikation innerhalb einer AoI. Beim Wechsel der Basis-Infrastruktur in Richtung der Peer-to-Peer-Ansätze muss zwischen Management- und Client-Schicht unterschieden werden. Aufgrund unterschiedlicher Aufgaben spielen in der Management- und in der Client-Schicht unterschiedliche Aspekte eine Rolle.

Gemäß dem vorgeschlagenen Infrastrukturwechsel teilen die Knoten der Management-Schicht die Verwaltungsbereiche unter sich auf. Nach der Aufteilung der Verantwortung ist jeder Knoten in der Management-Schicht nur für einen Teil der Welt zuständig. Dementsprechend sind die Objekt- und Avatar-Informationen in einem anderen Zuständigkeitsbereich für einen Knoten aus der Management-Schicht nicht direkt sichtbar. Wenn nun ein Client die AoI wechseln will, sendet er eine Anfrage nach den Objekt- und Avatar-Informationen aus der neuen Region an den (gegenwärtig) zuständigen Knoten. Der zuständige Knoten muss die entsprechende Anfrage innerhalb der Management-Schicht weiterleiten, um den Knoten zu identifizieren, der für die neue Region zuständig ist, und um den Client an den betreffenden Knoten verweisen zu können. Daraus ergibt sich die Notwendigkeit einer effizienten Such-Funktionalität innerhalb der Management-Schicht.

Zu diesem Zweck kann die Management-Schicht logisch als ein Overlay-Netzwerk organisiert werden. Overlay-Netzwerke haben den Vorteil, dass sie eine effiziente Suche mit logarithmischer Laufzeit ermöglichen[1]. Gegenwärtig existiert eine Vielzahl von Overlay-Ansätzen, die sich in der Zuordnung von Objekten zu den zuständigen Knoten unterscheiden. Aus der Vielzahl dieser Ansätze sollte ein am besten geeignetes Overlay-Netzwerk anhand bestimmter Kriterien ausgewählt werden. Dafür wurde die Simulationsumgebung SimCon entwickelt (siehe Abschnitt 4.1), mit deren Hilfe unterschiedliche Metriken definiert, erfasst und schließlich ausgewertet werden können. Für einen Vergleich wurden zwei proprietäre Implementierungen der Overlay-Netzwerke Tapestry und P-Grid ausgewählt. Wie die Evaluationsergebnisse in Abschnitt 4.3 zeigen, kann P-Grid die benötigte Sucheffizienz garantieren und erfordert dabei aufgrund seiner zufallsgesteuerten Natur nur geringe Konstruktionskosten. Deshalb würde der Einsatz des P-Grid-Overlays anstelle eines Servers die gewünschte Verbesserung der Skalierbarkeit und der Lastverteilung in virtuellen Umgebungen mit sich bringen. Die in Abschnitt 4.4 vorgestellte koordinatenbasierte Mapping-Methode von Objekten und Avataren auf den zuständigen Knoten des P-Grid-Overlays zeigt, wie das AoI-Management im HERA-Framework umgesetzt worden ist.

Die Client-Schicht kann aus vielen autonomen Peer-to-Peer-Systemen bestehen, die von dem zuständigen Management-Knoten und den Nutzern, die sich innerhalb der entsprechenden AoI befinden, gebildet werden. Die Zustandsaktualisierungen eines Clients müssen innerhalb seines Peer-to-Peer-Systems – das heißt, an alle anderen Clients innerhalb der AoI und an den zuständigen Knoten – weitergeleitet werden.

Die Weiterleitung kann auf drei unterschiedliche Arten erfolgen. Zum einen kann der Management-Knoten als eine Art „Verstärker“ eingesetzt werden, der die Zustandsaktualisierungen empfängt und diese dann weiterleitet. Der Vorteil dieser Variante liegt auf der Hand: Es wird kein zusätzlicher Aufwand in die Konstruktion einer Multicast-Infrastruktur investiert. Da diese Arbeit aber die Entlastung der Management-Schicht befürwortet, wurde von dieser Möglichkeit Abstand genommen.

Eine weitere Möglichkeit besteht darin, dass ein Client nach einer Zustandsänderung selbst alle anderen Clients (einzeln) über die Änderung in Kenntnis setzt. Dieser Ansatz hat zwei Schwachpunkte. Zum einen kann ein Client bei großen Nutzerzahlen rein technisch nicht die Verbindungen zu allen anderen Clients in der jeweiligen AoI pflegen. Zum anderen ist die Weiterleitung an jeden einzelnen Client sehr ineffizient und kann zu einer überflüssigen Netzbelastung führen, weil viele Nachrichten unter Umständen größtenteils über denselben Pfad geroutet werden.

Da ein IP-Multicast für die Privatanwender nicht zur Verfügung steht, bleibt somit nur eine Alternative: den Aufbau einer Multicast-Infrastruktur auf die Anwendungsschicht zu verlagern. Hierfür werden häufig baumartige Strukturen verwendet, die den Vorteil haben, dass nicht mehrere Nachrichten gleichzeitig den-

[1] Vorausgesetzt, dass das gesuchte Objekt gemäß dem entsprechenden Algorithmus veröffentlicht wurde.

selben Pfad entlang geroutet werden. Dadurch wird die Netzbelastung minimiert. Den geringen Kommunikationskosten (Multicast-Kosten) stehen jedoch die Konstruktionskosten der entsprechenden Infrastruktur gegenüber, die auf der Basis des jeweiligen autonomen Peer-to-Peer-Systems (Clients innerhalb einer AoI) aufgebaut wird.

Geringe Konstruktionskosten sind deshalb erstrebenswert, weil die Clients nur für die Dauer ihres Aufenthaltes in einer AoI ein Teil der jeweiligen Infrastruktur sind. Ein Wechsel der AoI führt neben dem Wechsel des zuständigen Management-Knotens ebenfalls zum Wechsel des Peer-to-Peer-Systems. Da in einer dynamischen Umgebung ein Client unter Umständen seine AoI sehr oft wechseln kann, darf dies keinen hohen Konstruktions- und Verwaltungsaufwand nach sich ziehen. Der Trade-off zwischen Konstruktions- und Kommunikationskosten sollte mittels MST-Approximationen, die auf Basis des jeweiligen autonomen Peer-to-Peer-Systems konstruiert werden, erreicht werden. Zu diesem Zweck wurde die HiOPS-Infrastruktur entwickelt (Abschnitt 5.2), die über eine hierarchische Struktur verfügt und einen guten Kompromiss zwischen den Konstruktions- und Kommunikationskosten erzielt. Der Nachteil dieser Infrastruktur liegt unter anderem darin, dass die Entfernungen zwischen den Clients, die für die Konstruktion der entsprechenden Infrastruktur notwendig sind, mittels Ping-Nachrichten ermittelt werden.

Mit der CARMIn-Infrastruktur (siehe Abschnitt 5.3) wurden die Schwächen von HiOPS adressiert. Mittels der CARMA-Architektur wird die Entfernung zwischen zwei Knoten anhand deren IP-Adressen abgeschätzt. Diese Abschätzung erfolgt lokal auf der Basis der öffentlich verfügbaren Informationen über die Internetstrukturen und deren Abhängigkeiten. Somit fällt bei CARMIn keine zusätzliche Kommunikation bei der Konstruktion der Multicast-Infrastruktur an. Da auch Bei- und Austritte von Clients nur die Nachbarknoten betreffen, ist der Ansatz auch in äußerst dynamischen Systemen einsetzbar. Aus diesem Grund wurde CARMIn als Multicast-Infrastruktur für die Client-Schicht ausgewählt. Weil für die clientbasierte Multicast-Kommunikation via CARMIn-Infrastruktur die Management-Schicht nicht in Anspruch genommen werden muss, wird diese dadurch weiterhin entlastet, was die erhoffte Verbesserung der Skalierbarkeit und der Lastverteilung mit sich bringt.

Der Schwerpunkt dieser Arbeit lag auf den Konsistenzaspekten in virtuellen Umgebungen und insbesondere auf dem clientbasierten Konsistenz-Handling. Viele verteilte Anwendungen (z.B. Geld-Transaktionen in einer Bank oder Online-Buchungssysteme) haben strenge Konsistenzanforderungen und müssen zur Erfüllung dieser Anforderungen Wartezeiten tolerieren. Im Gegensatz dazu müssen virtuelle Umgebungen sofort nach der Eingabe einer Benutzeraktion eine Systemrückmeldung liefern, um die Interaktivität der Umgebung nicht zu stören und somit die Erwartungskonformität der Nutzer nicht zu verletzen.

Um die besonderen Konsistenz- und Antwortzeit-Anforderungen beim Konsistenz-Handling in DVEs berücksichtigen zu können, wurde in dieser Arbeit (in Abschnitt 6.1) das Modell der elastischen Konsistenz definiert (Definition 6.2). Die Intention des Modells liegt darin, Abweichungen zu einem bestimmten Grad

zuzulassen und dafür die geforderte Systemantwortzeit wenn möglich einzuhalten. Weil die traditionellen Algorithmen aufgrund ihrer Laufzeitkomplexität die hohen Interaktivitätsanforderungen der elastischen Konsistenz verletzen würden, finden in DVEs oftmals optimistische Verfahren Anwendung. Allerdings können diese Verfahren auch keine elastische Konsistenz garantieren, weil sie keine Konfliktprüfungen durchführen und beim Vorliegen einer kritischen Situation Inkonsistenzen nicht vermeiden können.

Mit der Übertragung des Konsistenz-Handlings an die Client-Schicht wird die nachträgliche Wiederherstellung der Konsistenz im Vergleich zu den serverbasierten Ansätzen jedoch wesentlich erschwert. Deshalb wurde in dieser Arbeit untersucht, ob es möglich ist, durch die Investition in einen vertretbaren algorithmischen Aufwand die Wahrscheinlichkeit für das Auftreten von Inkonsistenzen und somit die Häufigkeit der Zustandskorrekturen zu reduzieren. Dazu wurde im Abschnitt 6.1 das Modell der statistisch elastischen Konsistenz definiert (Definition 6.7). Die Verfahren, die dieses Modell implementieren, terminieren innerhalb der vorgegebenen Zeitschranke und können, im Gegensatz zu den optimistischen Verfahren, elastische Konsistenz mit einer festgelegten Mindestwahrscheinlichkeit garantieren (siehe Abschnitte 6.4 und 6.6).

Diese Arbeit hat gezeigt, dass das Konsistenz-Handling verteilt realisiert werden kann ohne die Interaktivitätsanforderungen einer DVE zu verletzen. Das bedeutet, dass die Übertragung des Konsistenz-Handlings an die Clients grundsätzlich möglich ist, was zur weiteren Entlastung der Management-Schicht beitragen kann.

Zu Demonstrations- und Untersuchungszwecken wurde im Rahmen dieser Dissertation ein einfacher DVE-Framework-Prototyp HERA (siehe Abschnitt 3.2) entwickelt, der die benötigte Kernfunktionalität zur Verfügung stellt. Um die Machbarkeit aller in dieser Arbeit beschriebenen Ansätze und Konzepte im praktischen Einsatz evaluieren zu können, wurde basierend auf HERA das Multiplayer-Online-Spiel HERATRONos implementiert.

8

Ausblick

Die in dieser Arbeit vorgestellten Konsistenz- und Infrastrukturansätze wurden im HERA-Framework implementiert und auf ihre Eignung evaluiert. Die Evaluationsergebnisse zeigen, dass das mit diesen Ansätzen verknüpfte Ziel – bessere Skalierbarkeit und Lastverteilung der Basis-Infrastruktur – erreicht werden kann. Die prototypische Implementierung des clientbasierten Konsistenz-Handlings und des Overlay-basierten AoI-Managements in der HERATRONos-Spielanwendung postuliert deren Einsatzfähigkeit. Nachfolgend wird aufgezeigt, welche lang- und kurzfristigen Erweiterungen im HERA-Framework bzw. in Peer-to-Peer-basierten DVEs im Allgemeinen in Angriff genommen werden sollten.

Cheating

Die Nachrichtenmanipulation (Cheating) ist einer der Aspekte, die in Peer-to-Peer-basierten virtuellen Umgebungen enorm wichtig sind. Dieser Aspekt wurde im Laufe dieser Arbeit jedoch aus Zeitgründen nicht behandelt. Sowohl Manipulation an der Anwendung als auch an den Nachrichten ist bereits heute in serverbasierten Umgebungen ein massives Problem, welches durch die Dezentralisierung noch weiter verstärkt wird. Die Implementierung von (kryptographischen) Maßnahmen, die eine Manipulation von Nachrichten ausschließen und eventuell sogar eine Rückverfolgung der manipulierten Netzwerkpakete zulassen, ist deshalb enorm wichtig. In [Schuster et al., 2009] ist ein Ansatz beschrieben, wie Nachrichten- und Objektmanipulationen in Peer-to-Peer-basierten DVEs minimiert werden können. Es sind jedoch noch weitere Arbeiten auf dem Gebiet erforderlich, um das Cheating-Problem in Peer-to-Peer-basierten DVEs und MMOGs unter Kontrolle zu halten.

Service-Orientierung in DVEs

Weiterhin wäre es interessant zu untersuchen, ob die Neuauslegung der Service-Orientierung, die in den Projekten SISC [Schloss et al., 2007] und SEMPA [Esch et al., 2007] eingeführt wurde, auf massiv verteilte Anwendungsszenarien wie DVEs und MMOGs ausgeweitet werden kann.

SISC ist eine neuartige nachrichtenbasierte serviceorientierte Kommunikationsmiddleware. Der Leitgedanke bei SISC ist die Einführung eines neuen Softwareentwicklungsmodells, bei dem vollständig unabhängige Komponenten mithilfe

der SISC-Middleware zu einem einheitlichen Software-Produkt verknüpft werden. Um dies zu gewährleisten, ermöglicht SISC die Bereitstellung und die Nutzung von lokalen Diensten (Betriebssystem) und entfernten Diensten (Web Services) auf einheitliche Art und Weise über eine homogene Schnittstelle. Dabei werden die SISC-Dienste mittels einer XML-Konfigurationsdatei *(Kontrakt)* beschrieben und konfiguriert. Solche Kontrakte stellen die Einstiegs- bzw. Verknüpfungspunkte zwischen den einzelnen Komponenten dar. Auf diesem Wege wurde eine Entkopplung zwischen unterschiedlichen Softwarekomponenten erreicht, die für darauf aufsetzende Projekte wesentlich ist.

Auf SISC aufbauend wurde ein leichtgewichtiger Middleware-Prototyp für Peer-to-Peer-Anwendungen namens SEMPA entwickelt. Im Gegensatz zu vielen anderen Software-Lösungen zeichnet sich SEMPA dadurch aus, dass die Software eine klare Trennung zwischen Anwendungslogik und Anwendungsrepräsentation ermöglicht. Das heißt, dass ein Client nur die entsprechende Benutzerschnittstelle (Repräsentation) kennen muss, wenn er eine bestimmte Anwendung in Anspruch nehmen will. Die Funktionalität wird in Form von SISC-Services, die sowohl lokal als auch entfernt sein können, von dem Anwendungs-Provider zur Verfügung gestellt und von dem Anwendungs-Consumer über die SEMPA-Benutzerschnittstelle in Anspruch genommen.

Die Frage, die in diesem Zusammenhang geklärt werden muss, ist, ob es je nach Endgerät möglich und sinnvoll wäre, den Nutzern einer DVE einen einfachen Client zur Verfügung zu stellen, der nur die entsprechende Benutzerschnittstelle kennt und die entsprechende Funktionalität über das SISC-Protokoll in Anspruch nimmt.

Automatisierte Optimierung von Fuzzy-Mengen

Ein weiterer Aspekt, der in nachfolgenden Arbeiten behandelt werden sollte, ist die automatisierte Optimierung der Form von Fuzzy-Mengen, die in dem Fuzzy-System zur Bestimmung des CRT-Parameters verwendet werden. Die Definition der Fuzzy-Mengen ist der ausschlaggebende Faktor für das Verhalten eines Fuzzy-Systems und somit auch für den Verlauf der CRT-Funktion.

Wie Abbildung 6.8 zeigt, ist die Interaktivitätsgrenze bei dem CRT-Parameter-basierten wechselseitigen Ausschluss noch nicht ausgereizt. Das heißt, man könnte einen größeren CRT-Wert bei der Skalierung der Antwortmenge nehmen, ohne die Interaktivität des Systems nachteilig zu beeinflussen. Ein größerer CRT-Skalierungsfaktor würde dazu führen, dass die Inkonsistenzwahrscheinlichkeit weiter gesenkt wird.

Es wäre also wünschenswert, die Fuzzy-Mengen-Optimierung dahingehend zu automatisieren, dass durch eine Rückkopplung die Latenzzeiten, die in der Iteration i anfallen, auf die Form der Fuzzy-Mengen Einfluss nehmen. Auf diese Art und Weise könnten die Latenzzeiten bei der Berechnung des CRT-Parameters in der Iteration $i + 1$ berücksichtigt werden.

Erweiterung der CARMIn-Infrastruktur

Die in Abschnitt 5.3 beschriebene CARMA-Architektur sollte in weiteren Arbeiten so erweitert werden, dass nicht nur eine grobe Klassifizierung in fünf Distanz-Klassen vorgenommen wird, sondern auch eine differenzierbare Unterteilung von Knoten innerhalb einer Klasse möglich ist. Nach wie vor soll bei der Erhebung dieser Informationen keine zusätzliche Kommunikation anfallen.

Bei der Konstruktion der CARMIn-Infrastruktur wurde das *Nearest-Neighbor*-Prinzip angewandt, bei dem ein neuer Knoten einfach eine Verbindung zu dem Knoten mit der geringsten CARMA-Distanz aufbaut. Hier kann untersucht werden, ob durch den Aufbau einer unwesentlich komplexeren MST-Approximation[1] die Kommunikationskosten noch weiter gesenkt werden können.

Schlusswort

Im Laufe dieser Arbeit wurden sowohl die Infrastrukturaspekte (z.B. Anwendungsschicht-Multicast) als auch die Konsistenzaspekte (z.B. Konsistenz-Handling) in der Client-Schicht ausführlich betrachtet. In weiteren Forschungsarbeiten sollte untersucht werden, ob es bei der gemeinsamen Betrachtung der Infrastruktur- und Konsistenzaspekte zu Synergie-Effekten kommen kann. Das heißt, es soll geklärt werden, ob sich die Beschaffenheit der Basis-Infrastruktur positiv auf die Zusicherung der Konsistenz auswirken kann.

[1] Der Anstieg der Konstruktionskosten soll vertretbar gering bleiben.

Literatur

Eigene Referenzen

Bettinger et al., 2009a. Bettinger, Christian, M. Esch, H. Schloss, R. Oechsle und P. Sturm (2009a). *Evaluation of P2P Overlays for AoI Management in Distributed Virtual Environments*. In: *The International Conference on Ultra Modern Telecommunications (ICUMT 2009), St. Petersburg, Russia*, S. 1–6.

Bettinger et al., 2009b. Bettinger, Christian, H. Schloss, A. Baumann, F. Hausen und G. Schneider (2009b). *HERA: Design Framework for Decentralized Distributed Virtual Environments and Games*. In: *NCM '09: Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, S. 876–881.

Botev et al., 2008a. Botev, Jean, M. Esch, A. Höhfeld, H. Schloss und I. Scholtes (2008a). *The HyperVerse - Concepts for a Federated and Torrent-Based "3D Web"*. In: *Proceedings of The 1st International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality 2008 (MMVE)*, S. 1–6.

Botev et al., 2008b. Botev, Jean, A. Höhfeld, H. Schloss, I. Scholtes, P. Sturm und M. Esch (2008b). *The HyperVerse - Concepts for a Federated and Torrent-Based "3D Web"*. Int. J. Adv. Media Commun., 2(4):331–350.

Botev et al., 2008c. Botev, Jean, A. Höhfeld, H. Schloss, I. Scholtes, P. Sturm und M. Esch (2008c). *HyperVerse - Towards a Self-Organizing and Sustainable Global-Scale Virtual Environment*. In: *Proceedings of the 4th Microsoft Academic Days*.

Botev et al., 2009. Botev, Jean, M. Esch, H. Schloss, I. Scholtes und I. Scholtes (2009). *HyperVerse: Simulation and Testbed Reconciled*. In: *Proceedings of The 2nd International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality 2009 (MMVE)*.

Botev et al., 2010. Botev, Jean, M. Esch, H. Schloss, I. Scholtes und P. Sturm (2010). *HyperVerse: Simulation and Testbed Reconciled*. Int. J. Adv. Media Commun., 4(2):167–181.

Esch et al., 2007. Esch, Markus, H. Schloss und I. Scholtes (2007). *The SEMPA Prototype - Using XAML and Web Services for Rich Interactive Peer-to-Peer Applications.* In: *CollaborateCom*, S. 271–277. IEEE.

Esch et al., 2008a. Esch, Markus, J. Botev, H. Schloss und I. Scholtes (2008a). *GP3 - A Distributed Grid-Based Spatial Index Infrastructure for Massive Multiuser Virtual Environments*. International Conference on Parallel and Distributed Systems, 0:811–816.

Esch et al., 2008b. Esch, Markus, H. Schloss, I. Scholtes, J. Botev, A. Höhfeld und B. Zech (2008b). *SimCon - a Simulation and Visualization Environment for Overlay Networks and Large-scale Applications*. In: *SIMUTOOLS '08, Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, S. 1–9, ICST, Brussels, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Esch et al., 2009a. Esch, Markus, J. Botev, H. Schloss und I. Scholtes (2009a). *P2P-Based Avatar Interaction in Massive Multiuser Virtual Environments*. Complex, Intelligent and Software Intensive Systems, International Conference, 0:977–982.

Esch et al., 2009b. Esch, Markus, J. Botev, H. Schloss und I. Scholtes (2009b). *Performance Evaluation of GP3 - A Grid-Based Spatial Index Infrastructure*. In: *ITNG '09: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, S. 663–670, Washington, DC, USA. IEEE Computer Society.

Höhfeld et al., 2009. Höhfeld, Alexander, P. Gratz, A. Beck, J. Botev, H. Schloss und I. Scholtes (2009). *Self-organizing Collaborative Filtering in Global-scale Massive Multi-user Virtual Environments*. In: *SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing*, S. 1719–1723, New York, NY, USA. ACM.

Poryev et al., 2010. Poryev, Gennadiy, H. Schloss und R. Oechsle (2010). *CARMA based MST Approximation for Multicast Provision in P2P Networks*. In: *The Sixth International Conference on Networking and Services (ICNS 2010)*, S. 1–6.

Morth et al., 2007. Morth, Thiemo, R. Oechsle, H. Schloss und M. Schwinn (2007). *Automatische Bewertung studentischer Software*. In: *Proceedings der Pre-Conference Workshops der 5. ELearning Fachtagung Informatik (DeLFI 2007)*.

Oechsle et al., 2005. Oechsle, Rainer, H. Schloss und R. Weires (2005). *VLANSim: Simulation of Virtual LANs*. In: *International Conference on Engineering Education, Gliwice, Poland*, S. 1–6.

Schloss und Schmitz, 2006. Schloss, Hermann und H. Schmitz (2006). *Entwicklung eines Systems zur effizienten Lösung von Least-Squares-Problemen mit Nebenbedingungen für die Parameteroptimierung in TSK-Fuzzy-Systemen*. Technischer Bericht, Trier University of Applied Sciences.

Schloss et al., 2007. Schloss, Hermann, I. Scholtes und P. Sturm (2007). *SISC: Providing Efficient XML-based Service-orientation for Core OS Functionality*. In: *SOCP '07: Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches*, S. 45–52, New York, NY, USA. ACM.

Schloss et al., 2008a. Schloss, Hermann, J. Botev, M. Esch, A. Höhfeld, I. Scholtes und P. Sturm (2008a). *Elastic Consistency in Decentralized Distributed Virtual Environments.*. In: *Proceedings of the Fourth International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS 2008)*, S. 249–253, Florence, Italy.

Schloss et al., 2008b. Schloss, Hermann, R. Oechsle, J. Botev, M. Esch, A. Höhfeld und I. Scholtes (2008b). *HiOPS Overlay - Efficient Provision of Multicast in Peer-to-Peer Systems*. In: *16th IEEE International Conference on Networks (ICON 2008)*, S. 1–6, New Delhi, India.

Schloss et al., 2009. Schloss, Hermann, J. Botev, M. Esch, A. Höhfeld, I. Scholtes und P. Sturm (2009). *Fuzzy Logic supported Consistency Management in DDVEs*. In: *Proceedings of the 1st International Workshop on Concepts of Massively Multiuser Virtual Environments at the 16th ITG/GI Kommunikation in verteilten Systemen 2009*, Kassel, Germany.

Scholtes et al., 2008a. Scholtes, Ingo, J. Botev, M. Esch, A. Höhfeld, H. Schloss und B. Zech (2008a). *TopGen - internet router-level topology generation based on technology constraints*. In: *SIMUTOOLS '08, Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, S. 1–10, ICST, Brussels, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Scholtes et al., 2008b. Scholtes, Ingo, J. Botev, M. Esch, H. Schloss und P. Sturm (2008b). *Using Epidemic Hoarding to Minimize Load Delays in P2P Distributed Virtual Environments*. In: *CollaborateCom*, S. 578–593.

Scholtes et al., 2008c. Scholtes, Ingo, J. Botev, A. Höhfeld, H. Schloss und M. Esch (2008c). *Awareness-Driven Phase Transitions in Very Large Scale Distributed Systems*. International Conference on Self-Adaptive and Self-Organizing Systems, 0:25–34.

Allgemeine Referenzen

Aberer, 2001. Aberer, Karl (2001). *P-Grid: A Self-Organizing Access Structure for P2P Information Systems*. Sixth International Conference on Cooperative Information Systems (CoopIS 2001), 2172:179–194.

Aberer et al., 2003. Aberer, Karl, P. Cudré-Mauroux, A. Datta und M. Hauswirth (2003). *PIX-Grid: A Platform for P2P Photo Exchange*. In: *CAiSE Workshops*.

Aberer et al., 2002. Aberer, Karl, M. Punceva, M. Hauswirth und R. Schmidt (2002). *Improving Data Access in P2P Systems*. IEEE Internet Computing, 6(1):58–67.

Aggarwal et al., 2005. Aggarwal, Sudhir, H. Banavar, S. Mukherjee und S. Rangarajan (2005). *Fairness in dead-reckoning based distributed multi-player games*. In: *NetGames '05: Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, S. 1–10, New York, NY, USA. ACM.

Andre et al., 1985. Andre, Francoise, J.-S. Banino, C. Betourne, J. Ferrie, D. Herman, C. Kaiser, S. Krakowiak, G. Mazare, J. Mossiere, X. R. de Pina, J. Seguin und J.-P. Verjus (1985). *Distributed Computing Systems. Communication, cooperation, consistency*. Elsevier.

Aronson, 1997. Aronson, Jesse (1997). *Dead Reckoning: Latency Hiding for Networked Games*. `http://www.gamasutra.com/features/19970919/aronson_01.htm`.

Augustin et al., 2006. Augustin, Brice, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien und R. Teixeira (2006). *Avoiding Traceroute Anomalies with Paris Traceroute*. In: *IMC '06: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, S. 153–158, New York, NY, USA. ACM.

Awerbuch, 1987. Awerbuch, B. (1987). *Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election, and Related Problems*. In: *STOC '87: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, S. 230–240, New York, NY, USA. ACM.

Banerjee et al., 2002. Banerjee, Suman, B. Bhattacharjee und C. Kommareddy (2002). *Scalable Application Layer Multicast*. In: *SIGCOMM '02: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, S. 205–217, New York, NY, USA. ACM.

Bartsch, 1999. Bartsch, Hans-Jochen (1999). *Taschenbuch mathematischer Formeln*. Fachbuchverlag Leipzig.

Baumann, 2009. Baumann, Andreas (2009). *Entwurf und Implementierung einer prototypischen Anwendung zur Simulation von Bewegungsabläufen in virtuellen Umgebungen*. Diplomarbeit, Fachhochschule Trier. `http://www.jede-menge-zukunft.de/uploads/tx_rfttheses/Baumann_Andreas_Heratronos.pdf`.

Baumgart et al., 2007. Baumgart, Ingmar, B. Heep und S. Krause (2007). *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*.

Bernstein et al., 1987. Bernstein, Philip A., V. Hadzilacos und N. Goodman (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

Bettinger, 2008. Bettinger, Christian (2008). *Entwurf und Implementierung einer modularen Visualisierungsumgebung für Overlay-Netzwerke sowie eines P-Grid-Moduls*. Diplomarbeit, Fachhochschule Trier.

Blau et al., 1992. Blau, Brian, C. E. Hughes, M. J. Moshell und C. Lisle (1992). *Networked Virtual Environments*. In: *SI3D '92: Proceedings of the 1992 Symposium on Interactive 3D Graphics*, S. 157–160, New York, NY, USA. ACM.

Castro et al., 2002. Castro, Miguel, P. Druschel, A.-M. Kermarrec und A. Rowstron (2002). *Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure*. IEEE Journal on Selected Areas in Communication (JSAC).

Chandler und Finney, 2005. Chandler, Angie und J. Finney (2005). *On the effects of loose causal consistency in mobile multiplayer games*. In: *NetGames '05: Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, S. 1–11, New York, NY, USA. ACM.

Chawathe et al., 2003. Chawathe, Yatin, S. Ratnasamy, L. Breslau, N. Lanham und S. Shenker (2003). *Making gnutella-like P2P systems scalable*. In: *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, S. 407–418, New York, NY, USA. ACM Press.

Chim et al., 2003. Chim, Jimmy H. P., R. W. H. Lau, H. V. Leong und A. Si (2003). *CyberWalk: a web-based distributed virtual walkthrough environment*. IEEE Transactions on Multimedia, 5(4):503–515.

Chin und Ting, 1985. Chin, F. und H. F. Ting (1985). *An almost linear time and O(nlogn+e) Messages distributed algorithm for minimum-weight spanning trees*. In: *SFCS '85: Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, S. 257–266, Washington, DC, USA. IEEE Computer Society.

Choffnes und Bustamante, 2008. Choffnes, David R. und F. E. Bustamante (2008). *Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems*. SIGCOMM Comput. Commun. Rev., 38(4):363–374.

Chu et al., 2000. Chu, Yang-hua, S. G. Rao und H. Zhang (2000). *A case for end system multicast (keynote address)*. In: *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, S. 1–12, New York, NY, USA. ACM.

Clarke et al., 2002. Clarke, Ian, S. G. Miller, T. W. Hong, O. Sandberg und B. Wiley (2002). *Protecting Free Expression Online with Freenet*. IEEE Internet Computing, 6(1):40–49.

Clauset et al., 2007. Clauset, Aaron, C. R. Shalizi und M. E. J. Newman (2007). *Power-law distributions in empirical data*. SIAM Reviews.

Cohen, 2003. Cohen, Bram (2003). *Incentives Build Robustness in BitTorrent*. `citeseer.ist.psu.edu/cohen03incentives.html`.

Coulouris et al., 2002. Coulouris, George, J. Dollimore und T. Kindberg (2002). *Verteilte Systeme: Konzepte und Design*. Addison-Wesley.

Dabek et al., 2004. Dabek, Frank, J. Li, E. Sit, J. Robertson, M. F. Kaashoek und R. Morris (2004). *Designing a DHT for Low Latency and High Throughput.* In: *NSDI*, S. 85–98. USENIX.

Datta et al., 2003. Datta, Anwitaman, M. Hauswirth und K. Aberer (2003). *Updates in Highly Unreliable, Replicated Peer-to-Peer Systems*. In: *In Proceedings of the 23rd International Conference on Distributed Computing Systems*.

Delaney et al., 2006. Delaney, Declan, T. Ward und S. McLoone (2006). *On consistency and network latency in distributed interactive applications: a survey–part I*. Presence: Teleoper. Virtual Environ., 15(2):218–234.

Dijkstra, 1959. Dijkstra, Edsger W. (1959). *A note on two problems in connexion with graphs*. Numerische Mathematik, 1:269–271.

Elkin, 2004a. Elkin, Michael (2004a). *Distributed approximation: a survey*. SIGACT News, 35(4):40–57.

Elkin, 2004b. Elkin, Michael (2004b). *Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem*. In: *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, S. 331–340, New York, NY, USA. ACM.

Elmas und Özkasap, 2004. Elmas, Tayfun und Ö. Özkasap (2004). *Distributed Document Sharing with Text Classification over Content-Addressable Network*. In: *AWCC*, S. 70–81.

Faloutsos et al., 1999. Faloutsos, Michalis, P. Faloutsos und C. Faloutsos (1999). *On power-law relationships of the Internet topology*. In: *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, S. 251–262, New York, NY, USA. ACM.

Fan et al., 2007. Fan, Lu, H. Taylor und P. Trinder (2007). *Mediator: a design framework for P2P MMOGs*. In: *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, S. 43–48, New York, NY, USA. ACM.

Feldmann und Aggarwal, 2008. Feldmann, Anja und V. Aggarwal (2008). *ISP-Aided Neighbor Selection in P2P Systems*. In: *IETF P2P Infrastructure Workshop (P2Pi)*, Berlin, Germany.

Freedman und Vingralek, 2002. Freedman, Michael J. und R. Vingralek (2002). *Efficient Peer-to-Peer Lookup Based on a Distributed Trie*. In: *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, S. 66–75, London, UK. Springer-Verlag.

Funkhouser, 1995. Funkhouser, Thomas A. (1995). *RING: A Client-Server System for Multi-User Virtual Environments*. In: *Symposium on Interactive 3D Graphics*, S. 85–92.

Gafni, 1985. Gafni, Eli (1985). *Improvements in the time complexity of two message-optimal election algorithms*. In: *PODC '85: Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, S. 175–185, New York, NY, USA. ACM.

Gallager et al., 1983. Gallager, R. G., P. A. Humblet und P. M. Spira (1983). *A Distributed Algorithm for Minimum-Weight Spanning Trees*. ACM Trans. Program. Lang. Syst., 5(1):66–77.

Garay et al., 1993. Garay, J. A., S. Kutten und D. Peleg (1993). *A sub-linear time distributed algorithm for minimum-weight spanning trees*. In: *SFCS '93: Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science*, S. 659–668, Washington, DC, USA. IEEE Computer Society.

Garcia et al., 2005. Garcia, Pedro, C. Pairot, R. Mondejar, J. Pujol, H. Tejedor und R. Rallo (2005). *PlanetSim: A New Overlay Network Simulation Framework*. In Journal: Software Engineering and Middleware, Volume 3437/2005:123–136.

Gautier et al., 1999. Gautier, Laurent, C. Diot und J. F. Kurose (1999). *End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet*. In: *INFOCOM*, S. 1470–1479.

Gupta et al., 2004. Gupta, Anjali, B. Liskov und R. Rodrigues (2004). *Efficient routing for peer-to-peer overlays*. In: *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, S. 9–9, Berkeley, CA, USA. USENIX Association.

Gupta et al., 2003. Gupta, Indranil, K. Birman, P. Linga, A. Demers und R. van Renesse (2003). *Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead*. In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*.

Hamza-Lup, 2004. Hamza-Lup, Felix George (2004). *Dynamic Shared State Maintenance in Distributed Virtual Environments*. Doktorarbeit, University of Central Florida.

Hawkinson und Bates, 1996. Hawkinson, J. und T. Bates (1996). *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. RFC 1930 (Best Current Practice).

IEEE-Std-1278-1993, 1993. IEEE-Std-1278-1993 (1993). *IEEE Standard for Information Technology-Protocols for Distributed Interactive Simulation Applications*. Institute for Electrical and Electronics Engineers.

Iyer et al., 2002. Iyer, Sitaram, A. Rowstron und P. Druschel (2002). *Squirrel: A decentralized peer-to-peer web cache*. In: *12th ACM Symposium on Principles of Distributed Computing (PODC 2002)*, S. ?–?

Jovanovic et al., 2001. Jovanovic, M., F. Annexstein und K. Berman (2001). *Scalability issues in large peer-to-peer networks - a case study of gnutella*.

Kaashoek und Karger, 2003. Kaashoek, M. Frans und D. R. Karger (2003). *Koorde: A simple degree-optimal distributed hash table*. In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*.

Kalman, 1960. Kalman, R. E. (1960). *A New Approach to Linear Filtering and Prediction Problems*. Transactions of the ASME: Journal of Basic Engineering.

Karnstedt et al., 2007. Karnstedt, Marcel, K.-U. Sattler, M. Richtarsky, J. Müller, M. Hauswirth, R. Schmidt und R. John (2007). *UniStore: Querying a DHT-based Universal Storage*. In: *ICDE*, S. 1503–1504.

Keller und Simon, 2003. Keller, Joaquín und G. Simon (2003). *Solipsis: A Massively Multi-Participant Virtual World*. In: *PDPTA*, S. 262–268.

Khan und Pandurangan, 2008. Khan, Maleq und G. Pandurangan (2008). *A fast distributed approximation algorithm for minimum spanning trees*. Distributed Computing, 20(6):391–402.

Kim et al., 2003. Kim, Sung-Jin, F. Kuester und K. H. Kim (2003). *Towards enhanced data consistency in distributed virtual environments*. In: Woods, Andrew J., M. T. Bolas, J. O. Merritt und S. A. Benton, Hrsg.: *Stereoscopic Displays and Virtual Reality Systems X*, Bd. 5006, S. 436–444. SPIE.

Kim et al., 2004. Kim, Young-Chul, I. Yeom und J. Lee (2004). *HYMS: A Hybrid MMOG Server Architecture (Internet Systems) (New Technologies and their Applications of the Internet)*. IEICE transactions on information and systems.

Klir und Yuan, 1996. Klir, George J. und B. Yuan, Hrsg. (1996). *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Knutsson et al., 2004. Knutsson, Bjorn, H. Lu, W. Xu und B. Hopkins (2004). *Peer-to-Peer Support for Massively Multiplayer Games*. In: *INFOCOM*, S. 96–107.

Kopetz, 1997. Kopetz, Hermann (1997). *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, USA.

Kotilainen et al., 2006. Kotilainen, N., M. Vapa, T. Keltanen, A. Auvinen und J. Vuori (2006). *P2PRealm - Peer-to-Peer Network Simulator*. In: *11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, S. 93–99.

Kruskal, 1956. Kruskal, Joseph B.jun. (1956). *On the shortest spanning subtree of a graph and the traveling salesman problem.* Proc. Am. Math. Soc., 7:48–50.

Kubiatowicz, 2003. Kubiatowicz, John (2003). *Extracting guarantees from chaos*. Commun. ACM, 46(2):33–38.

Kubiatowicz et al., 2000. Kubiatowicz, John, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells und B. Zhao (2000). *OceanStore: An architecture for global-scale persistent storage*. In: *9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

Kung und Robinson, 1981. Kung, H. T. und J. T. Robinson (1981). *On optimistic methods for concurrency control*. ACM Trans. Database Syst., 6(2):213–226.

Kutten und Peleg, 1995. Kutten, Shay und D. Peleg (1995). *Fast distributed construction of k-dominating sets and applications*. In: *PODC '95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, S. 238–251, New York, NY, USA. ACM.

Lamport, 1978. Lamport, Leslie (1978). *Time, clocks, and the ordering of events in a distributed system*. Commun. ACM, 21(7):558–565.

Lawson und Hanson, 1995. Lawson, Charles L. und R. J. Hanson (1995). *Solving Least Squares Problems.* Philadelphia, SIAM.

Li et al., 2005. Li, Jinyang, J. Stribling, R. Morris, M. F. Kaashoek und T. M. Gil (2005). *A performance vs. cost framework for evaluating DHT design tradeoffs under churn*. In: *Proceedings of the 24th Infocom*, Miami, FL.

Lu et al., 1999. Lu, Tainchi, M.-T. Lin und C. Lee (1999). *Control Mechanism for Large-Scale Virtual Environments*. J. Vis. Lang. Comput., 10(1):69–85.

Lucas et al., 2003. Lucas, Gabriel, A. Ghose und J. Chuang (2003). *On characterizing affinity and its impact on network performance*. In: *MoMeTools '03: Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, S. 65–75, New York, NY, USA. ACM.

Luk und Wong, 1997. Luk, Wai-Shing und T.-T. Wong (1997). *Two New Quorum Based Algorithms for Distributed Mutual Exclusion*. In: *ICDCS '97: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97)*, S. 100, Washington, DC, USA. IEEE Computer Society.

Maekawa, 1985. Maekawa, Mamoru (1985). *A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems*. ACM Trans. Comput. Syst., 3(2):145–159.

Mahlmann und Schindelhauer, 2007. Mahlmann, Peter und C. Schindelhauer (2007). *Peer-to-Peer-Netzwerke: Algorithmen und Methoden*. Springer-Verlag Berlin, 1 Aufl. ISBN 3540339914.

Malte, 1997. Malte, Peter (1997). *Replikation in Mobil Computing*. Seminar No 31/1997, Institut für Telematik der Universität Karlsruhe, Karsruhe.
`http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/1997/31`.

Maniatis et al., 2003. Maniatis, Petros, D. S. H. Rosenthal, M. Roussopoulos, M. Baker, T. Giuli und Y. Muliadi (2003). *Preserving peer replicas by rate-limited sampled voting*. SIGOPS Oper. Syst. Rev., 37(5):44–59.

Manku et al., 2003. Manku, Gurmeet Singh, M. Bawa und P. Raghavan (2003). *Symphony: distributed hashing in a small world*. In: *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA. USENIX Association.

Margery et al., 1999. Margery, D. M., B. Arnaldi und N. Plouzeau (1999). *A General Framework for Cooperative Manipulation in Virtual Environments.* In: *EGVE 99: Eurographics Workshop on Virtual Environments*, S. 169–178.

Mauve, 1999. Mauve, Martin (1999). *Consistency in Continuous Distributed Interactive Media*. Technical Report TR-9-99, Department of Computer Science, University of Mannheim.

Mauve, 2000. Mauve, Martin (2000). *How to Keep a Dead Man from Shooting*. In: *IDMS '00: Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, S. 199–204, London, UK. Springer-Verlag.

Mauve et al., 2004. Mauve, Martin, J. Vogel, V. Hilt und W. Effelsberg (2004). *Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications*. IEEE Transactions on Multimedia, 6(1):47–57.

Maymounkov und Mazieres, 2002. Maymounkov, Petar und D. Mazieres (2002). *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*. In: *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. `http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf`.

Naicken et al., 2006a. Naicken, S., A. Basu, B. Livingston und S. Rodhetbhai (2006a). *A Survey of Peer-to-Peer Network Simulators*. In: *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*.

Naicken et al., 2006b. Naicken, S., A. Basu, B. Livingston, S. Rodhetbhai und I. Wakeman (2006b). *Towards Yet Another Peer-to-Peer Simulator*. In: *Proceedings of The Fourth International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs), Ilkley, UK*.

Pantel und Wolf, 2002. Pantel, Lothar und L. C. Wolf (2002). *On the impact of delay on real-time multiplayer games*. In: *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, S. 23–29, New York, NY, USA. ACM.

Peleg und Rubinovich, 2000. Peleg, David und V. Rubinovich (2000). *A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction*. SIAM J. Comput., 30(5):1427–1442.

Pendarakis et al., 2001. Pendarakis, Dimitrios, S. Shi, D. Verma und M. Waldvogel (2001). *ALMI: an application level multicast infrastructure*. In: *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA. USENIX Association.

Plaxton et al., 1999. Plaxton, C. Greg, R. Rajaraman und A. W. Richa (1999). *Accessing Nearby Copies of Replicated Objects in a Distributed Environment.*. Theory Comput. Syst., 32(3):241–280.

Qin, 2002. Qin, X. (2002). *Delayed Consistency Model for Distributed Interactive Systems with Real-time Continuous Media.*. Journal of Software, Vol.13(No.6):1029–1039.

Ratnasamy et al., 2002. Ratnasamy, S., M. Handley, R. Karp und S. Shenker (2002). *Topologically-aware overlay construction and server selection*. In: *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Bd. 3, S. 1190–1199.

Ratnasamy et al., 2001a. Ratnasamy, Sylvia, P. Francis, M. Handley, R. Karp und S. Schenker (2001a). *A scalable content-addressable network*. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, S. 161–172, New York, NY, USA. ACM.

Ratnasamy et al., 2001b. Ratnasamy, Sylvia, P. Francis, M. Handley, R. Karp und S. Schenker (2001b). *A scalable content-addressable network*. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, S. 161–172, New York, NY, USA. ACM Press.

Rhalibi und Merabti, 2005. Rhalibi, Abdennour El und M. Merabti (2005). *Agents-based modeling for a peer-to-peer MMOG architecture*. Comput. Entertain., 3(2):3–3.

Roberts und Sharkey, 1997. Roberts, David J. und P. M. Sharkey (1997). *Maximising Concurrency and Scalability in a Consistent, Causal, Distributed Virtual Reality System Whilst Minimising the Effect of Network Delays*. In: *WET-ICE '97: Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, S. 161–166, Washington, DC, USA. IEEE Computer Society.

Rowstron und Druschel, 2001a. Rowstron, Antony und P. Druschel (2001a). *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*. Lecture Notes in Computer Science.

Rowstron und Druschel, 2001b. Rowstron, Antony und P. Druschel (2001b). *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*. In: *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, S. 329–350.

Rowstron und Druschel, 2001c. Rowstron, Antony und P. Druschel (2001c). *Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility*. In: *18th ACM Symposium on Operating Systems Principles (SOSP'01)*, S. 188–201.

Sanders, 1987. Sanders, Beverly A. (1987). *The information structure of distributed mutual exclusion algorithms*. ACM Trans. Comput. Syst., 5(3):284–299.

Schaeffer et al., 2005. Schaeffer, Benjamin, P. Brinkmann, G. Francis, C. Goudeseune, J. Crowell und H. Kaczmarski (2005). *Myriad: scalable VR via peer-to-peer connectivity, PC clustering, and transient inconsistency*. In: *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '05)*, S. 68–77, New York, NY, USA. ACM.

Schiele et al., 2008. Schiele, Gregor, R. Sueselbeck, A. Wacker, T. Triebel und C. Becker (2008). *Consistency Management for Peer-To-Peer-based Massively Multiuser Virtual Environments*. In: *Proceedings of The 1st International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality 2008 (MMVE)*.

Schraeder, 2004. Schraeder, Gregor (2004). *Multicast in P2P-Netzwerken*. Diplomarbeit, Lehrstuhl Rechnernetze und Internet, Universität Tübingen.

Schuster et al., 2009. Schuster, Sebastian, A. Wacker und T. Weis (2009). *Fighting Cheating in P2P-based MMVEs with Disjoint Path Routing*. In: *Proceedings of the 1st International Workshop on Concepts of Massively Multiuser Virtual Environments at the 16th ITG/GI Kommunikation in verteilten Systemen 2009*, Kassel, Germany.

Smith et al., 2003. Smith, David A., A. C. Kay, A. Raab und D. P. Reed (2003). *Croquet - A Collaboration System Architecture*. In: *C5*, S. 2–.

Smith, 2000. Smith, Finlay S (2000). *Selection of Fuzzication, Inference and Defuzzication Techniques for Fuzzy Logic Control*. Technischer Bericht, National University of Ireland, Galway.

Snowdon und Munro, 2001. Snowdon, David N. und A. J. Munro (2001). *Collaborative Virtual Environments: Digital Places and Spaces for Interaction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Stoica et al., 2001. Stoica, Ion, R. Morris, D. Karger, M. F. Kaashoek und H. Balakrishnan (2001). *Chord: A scalable peer-to-peer lookup service for internet applications*. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, S. 149–160, New York, NY, USA. ACM Press.

Wu et al., 2004. Wu, Yinghui, M. Li und W. Zheng (2004). *ONSP: Parallel Overlay Network Simulation Platform*. In: *PDPTA*, S. 1147–1153.

Xie et al., 2008. Xie, Haiyong, A. Krishnamurthy, A. Silberschatz und R. Y. Yang (2008). *P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers*. DCIA P2P Market Conference.

Yu und Vuong, 2005. Yu, Anthony und S. T. Vuong (2005). *MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games*. In: *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, S. 99–104, New York, NY, USA. ACM.

Yu und Vahdat, 2000. Yu, Haifeng und A. Vahdat (2000). *Design and evaluation of a continuous consistency model for replicated services*. In: *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, S. 21–21, Berkeley, CA, USA. USENIX Association.

Zadeh, 1965. Zadeh, Lotfi Asker (1965). *Fuzzy Sets*. Information and Control, 8:338–353.

Zech, 2007. Zech, Benjamin (2007). *Ein Public Server-Ansatz für massiv verteilte interaktive Applikationen*. Diplomarbeit, Fachhochschule Trier.

Zhang et al., 2005. Zhang, Hui, A. Goel und R. Govindan (2005). *An empirical evaluation of internet latency expansion*. SIGCOMM Comput. Commun. Rev., 35(1):93–97.

Zhao et al., 2004. Zhao, Ben Y., L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph und J. D. Kubiatowicz (2004). *Tapestry: A Resilient Global-scale Overlay for Service Deployment*. IEEE Journal on Selected Areas in Communications, 22(1):41–53.

Zhou et al., 2003. Zhou, Suiping, W. Cai, S. J. Turner und H. Zhao (2003). *A Consistency Model for Evaluating Distributed Virtual Environments*. In: *CW '03: Proceedings of the 2003 International Conference on Cyberworlds*, S. 85, Washington, DC, USA. IEEE Computer Society.

Index

A

Fuzzy-Logik-basierte Berechnung des CRT-Parameters

Die Intention des in diesem Abschnitt vorgeschlagenen Ansatzes liegt darin, einen Trade-off zwischen der statistischen Konsistenzwahrscheinlichkeit und Systemantwortzeiten auf der Basis der lokal verfügbaren System-Informationen bzw. -Eigenschaften und -Einstellungen zu bestimmen. Dazu zählen unter anderem Nutzer- bzw. Objektdichte sowie die Größe der betrachteten Aol, ebenso wie die zur Verfügung stehenden Ressourcen oder die aktuelle Netzwerkauslastung bzw. Übertragungslatenz. Das Problem liegt darin, diese heterogenen Größen auf einen Wert abzubilden. Eine mathematische Funktion, welche die Korrelation der einzelnen Größen beschreibt, ihnen eine bestimmte Gewichtung zuordnet und durch mathematische Verknüpfungen den Wert des CRT-Parameters bestimmt, existiert nicht. Auf der anderen Seite kann festgestellt werden, dass diese Zusammenhänge sehr einfach umgangssprachlich ausgedrückt werden können. So ist zum Beispiel die Folgerung „Mit steigender Anzahl der Benutzer fällt die Konsistenzwahrscheinlichkeit“ plausibel. Immer dann, wenn eine mathematische Funktion nicht existiert bzw. nur schwierig zu finden ist, eine umgangssprachliche Beschreibung der Zusammenhänge jedoch ohne weiteres möglich und plausibel ist, kann auf die Mittel der *Fuzzy-Logik* [Zadeh, 1965] zurückgegriffen werden.

Das Besondere an der Fuzzy-Logik ist die Definition der Fuzzy-Mengen, bei denen die Zugehörigkeit zu einer Menge durch eine Zugehörigkeitsfunktion angegeben wird, welche einem Element aus der Menge einen reellwertigen Zugehörigkeitsgrad $\mu \in [0, 1]$ zuweist. So gesehen stellt die Fuzzy-Logik eine Verallgemeinerung der klassischen Mengentheorie dar, in der ein Element entweder in einer Menge enthalten ist ($\mu = 1$) oder nicht ($\mu = 0$). Die diskrete Zugehörigkeitsdefinition zu einer Menge widerspricht jedoch unserer menschlichen Wahrnehmung von kontinuierlich veränderlichen Größen und Eigenschaften. Betrachtet man z.B. das menschliche Empfinden der Raumtemperatur, so stellt man fest, dass es keine scharfe Definition von „kalt“ bzw. „warm“ gibt. Darüber hinaus werden in der Fuzzy-Logik die Definitionen der Mengenoperatoren *Vereinigung* $\cup$ und *Durchschnitt* $\cap$ (manchmal auch *UND* und *ODER* genannt) erweitert, sodass diese Operatoren auch auf die Fuzzy-Mengen angewandt werden können.

Auf diese Weise macht es die Fuzzy-Logik möglich, die menschlichen Wahrnehmungen von Zuständen und Eigenschaften in linguistische Aussagen zu fassen, diese formal zu definieren und zu modellieren, mathematisch zu verknüpfen,

Schlussfolgerungen zu ziehen und daraus ein Ergebnis zu berechnen. Für Details bezüglich der Fuzzy-Logik wird an dieser Stelle auf [Klir und Yuan, 1996] und [Zadeh, 1965] verwiesen.

Ein Fuzzy-Logik-System ist prototypisch in Abbildung A.1 dargestellt. Dabei

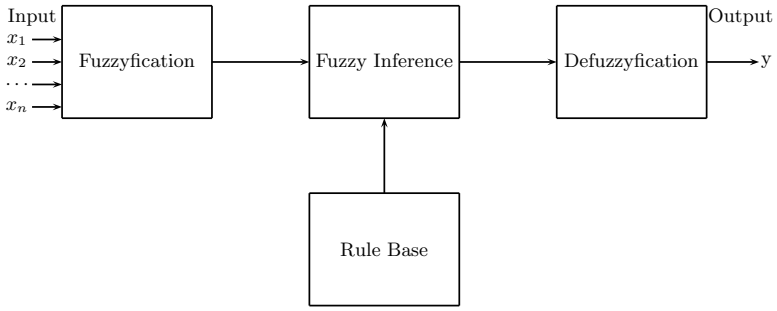


Abbildung A.1. Fuzzy-Inference

werden die reellwertigen Eingangsgrößen zuerst in einen Vektor transformiert (*Fuzzyfizierung*), der Zugehörigkeitsgrade zu den jeweiligen Fuzzy-Mengen enthält. Danach werden die Ausgangsgrößen (als Fuzzy-Mengen) anhand der Erfüllungsgrade der Regeln aus der Regelbasis bestimmt (*Inferenz*). Anschließend werden diese (Fuzzy) Ausgangsgrößen in reellwertige Größen transformiert (*Defuzzyfizierung*).

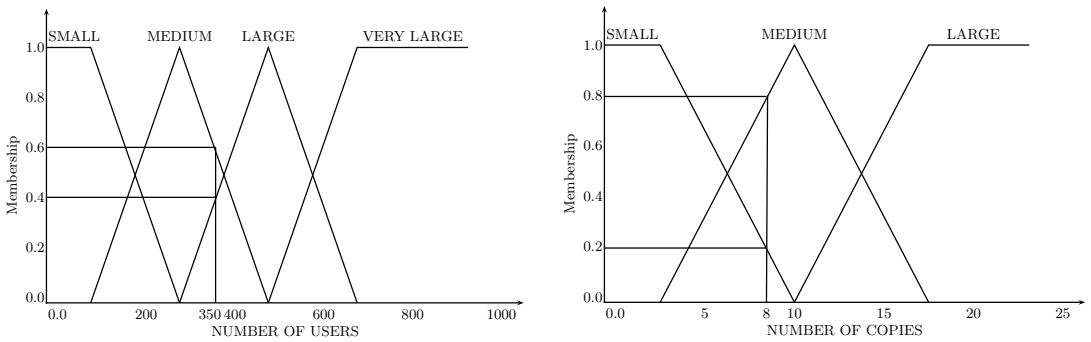
Die Anwendbarkeit des Fuzzy-Logik-Ansatzes zur Berechnung des CRT-Parameters wird an einem einfachen Beispielszenario demonstriert. Hierbei wird die konsistente Haltung von Objekten, die unter Benutzern in einer AoI repliziert sind, betrachtet.

In dem Beispielszenario werden nur zwei Eingangsgrößen – *Benutzeranzahl* (engl. Number of Users) und *Objektanzahl* (engl. Number of Copies) innerhalb einer AoI – für die CRT-Berechnung berücksichtigt. Das heißt, es wird die folgende Abbildung gesucht:

$$CRT : \mathbb{N} \times \mathbb{N} \rightarrow \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$$

Der stetige Wertebereich des CRT-Parameters liegt dabei im Bereich zwischen 0 und 1, d.h. $0 \leq CRT \leq 1$. Der Wert CRT $= 1$ sagt aus, dass innerhalb der geforderten Antwortzeit die elastische Konsistenz mit der Wahrscheinlichkeit $P = 1$ garantiert wird. Der Wert CRT $= 0$ hingegen bedeutet, dass innerhalb der geforderten Antwortzeit weder elastische noch statistisch elastische Konsistenz garantiert werden kann. Der Wert des CRT-Parameters $0 < CRT < 1$ steht für einen Tradeoff zwischen Konsistenzwahrscheinlichkeit und Systemantwortzeit. Dabei dürfen die oberen Schranken α und ι dieser Größen nicht überschritten werden.

Bei der Berechnung werden zuerst zwei linguistische Variablen NUMBEROFUSERS und NUMBEROFCOPIES definiert. Die linguistische Variable NUMBEROFUSERS kann die linguistischen Werte SMALL, MEDIUM, LARGE und VERY LARGE, welche durch Fuzzy-Mengen repräsentiert werden, annehmen (Abbildung A.2(a)). Die Variable NUMBEROFCOPIES kann die Werte SMALL, MEDIUM und LARGE annehmen (Abbildung A.2(b)).



(a) Fuzzyfizierung von 350 Nutzern (b) Fuzzyfizierung von 8 Objekten

Abbildung A.2. Fuzzyfizierung von Eingangsgrößen

Angenommen, es werden 8 Objekte unter 350 Nutzern repliziert. Wie in Abbildung A.1 gezeigt, müssen die beiden Eingangsgrößen zuerst *fuzzyfiziert* werden. Dabei findet eine Zuordnung eines reellen Wertes x_1 zu einer Fuzzy-Menge T^1 mit der Zugehörigkeit μ^1, zu der Fuzzy-Menge T^2 mit der Zugehörigkeit μ^2 usw. statt.

Die Abbildung A.2(a) zeigt die Transformation der Eingangsgröße NUMBEROFUSERS = 350. Dabei ergeben sich die folgenden Zugehörigkeiten:

- $\mu_{SMALL}(350) = 0$
- $\mu_{MEDIUM}(350) = 0.6$
- $\mu_{LARGE}(350) = 0.4$
- $\mu_{VERYLARGE}(350) = 0$

Die Fuzzyfizierung der Variable NUMBEROFCOPIES = 8 ist in Abbildung A.2(b) dargestellt.

Schließlich wird die linguistische Variable CRT VALUE definiert, welche die Ausgangsgröße repräsentiert und die Werte SMALL, MEDIUM und LARGE annehmen kann (Abbildung A.4(a)).

Die Regelbasis (Abbildung A.3) definiert die Abhängigkeit der Ausgangsgröße von den beiden Eingangsgrößen.

In unserem Beispielszenario „feuern“ (treffen zu) die vier folgenden Regeln aus der Regelbasis:

1. when #USERS = MEDIUM and #COPIES = SMALL then CRT VALUE = LARGE
2. when #USERS = MEDIUM and #COPIES = MEDIUM then CRT VALUE = MEDIUM
3. when #USERS = LARGE and #COPIES = SMALL then CRT VALUE = MEDIUM
4. when #USERS = LARGE and #COPIES = MEDIUM then CRT VALUE = SMALL

Um den Erfüllungsgrad α_i der Regel i $(1 \leq i \leq 4)$ zu bestimmen, wird die *min*-Funktion als UND-Operator verwendet.

$$\alpha_1 = \min\left(\mu^{\#USERS}_{MEDIUM}(350), \mu^{\#COPIES}_{SMALL}(8)\right) = \min(0.6, 0.2) = 0.2$$

$$\alpha_2 = \min\left(\mu^{\#USERS}_{MEDIUM}(350), \mu^{\#COPIES}_{MEDIUM}(8)\right) = \min(0.6, 0.8) = 0.6$$

$$\alpha_3 = \min\left(\mu^{\#USERS}_{LARGE}(350), \mu^{\#COPIES}_{SMALL}(8)\right) = \min(0.4, 0.2) = 0.2$$

$$\alpha_4 = \min\left(\mu^{\#USERS}_{LARGE}(350), \mu^{\#COPIES}_{MEDIUM}(8)\right) = \min(0.4, 0.8) = 0.4$$

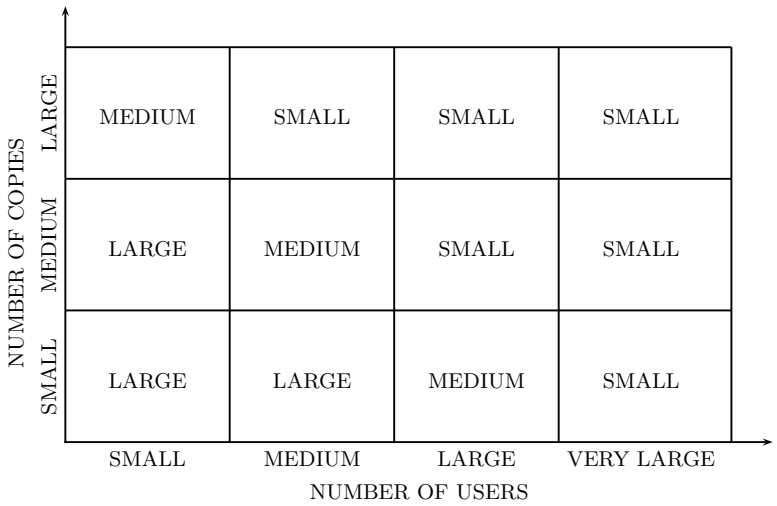


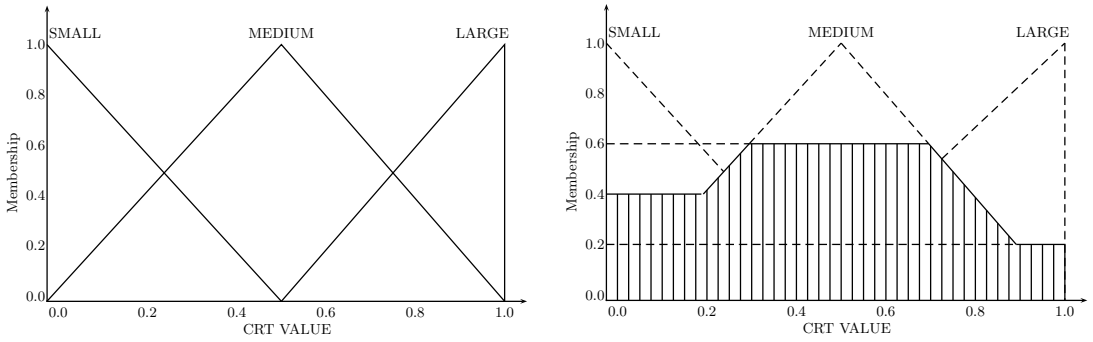
Abbildung A.3. Regelbasis

Der Erfüllungsgrad dieser Regeln wird verwendet, um die Fuzzy-Menge der Ausgangsgröße zu „formen“. Dabei wird diese Fuzzy-Menge in ihrer Höhe durch den Erfüllungsgrad der zutreffenden Regel begrenzt. Genauer gesagt, der Erfüllungsgrad μ der Fuzzy-Menge y für die Regel i ist definiert als

$$\mu_y^i(w)' = \min(\alpha_i, \mu_y^i(w))$$

wobei w die Variable aus dem Definitionsbereich der entsprechenden Zugehörigkeitsfunktion darstellt, in dem sie Werte größer Null annimmt ($\mu > 0$).

Der nächste Schritt ist die Komposition der vier Fuzzy-Mengen, die durch den Erfüllungsgrad der zutreffenden Regel begrenzt sind, zu einer einzigen Ergebnis-Fuzzy-Menge. Dieser Prozess wird als *Aggregation* bezeichnet. Wenn mehrere Regeln auf dieselbe Fuzzy-Menge abbilden (siehe Regeln 2 und 3), dann wird das Maximum der Erfüllungsgrade bei der Aggregation berücksichtigt. Auf diese Weise entsteht die folgende Ergebnis-Fuzzy-Menge (siehe Abbildung A.4(b)).



(a) Variablendefinition (b) Ergebnis-Fuzzy-Menge

Abbildung A.4. Bestimmung der Ergebnis-Fuzzy-Menge CRT VALUE

Die aggregierte Ergebnis-Fuzzy-Menge dient ihrerseits als Eingabe für den *Defuzzyfizierungsprozess*, in dem eine Fuzzy-Menge auf einen „scharfen“ Wert abgebildet wird. Unterschiedliche Defuzzyfizierungstechniken werden in [Smith, 2000] gegenübergestellt. In dem Beispielszenario findet die *Schwerpunkt-Methode* bei der Defuzzyfizierung Anwendung. Dabei ergibt sich der folgende Wert des CRT-Parameters

$$CRT = \frac{\int_0^1 y \cdot \mu_y(y) dy}{\int_0^1 \mu_y(y) dy} \approx 0.42$$

wobei y die Achse der Ausgangsgröße – CRT VALUE-Achse – repräsentiert.

Die Abbildung A.5 stellt das Verhalten der CRT-Funktion in Abhängigkeit von einer konstanten Anzahl von Objekten (8) und einer variablen Benutzeranzahl $0 \leq N \leq 1000$ dar. Aus Abbildung A.5 ist ersichtlich, dass sich der Wert des CRT-Parameters für die Benutzerzahlen $N \geq 380$ nicht mehr ändert. Dieses Verhalten ist auf die Definition der Schwerpunktmethode zurückzuführen und von der Definition der Fuzzy-Mengen abhängig.

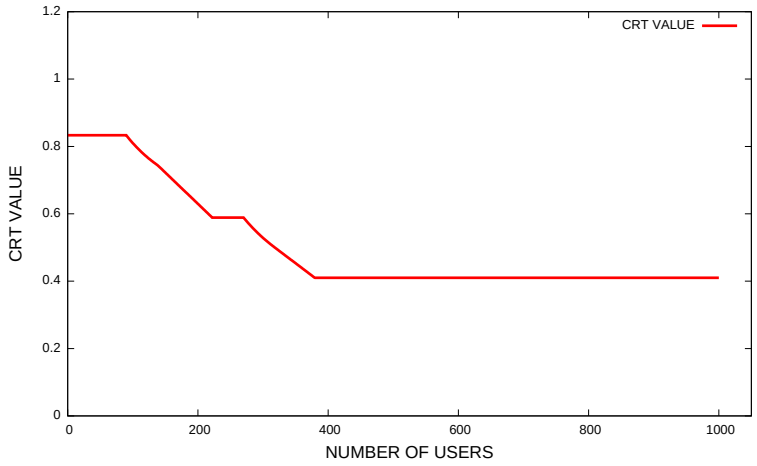


Abbildung A.5. CRT-Parameter

Der berechnete Wert des CRT-Parameters ist allgemeingültig und kann zur Anpassung des Algorithmenverhaltens verwendet werden. Dabei fällt mit dem fallenden Wert des Parameters auch die Konsistenzwahrscheinlichkeit.