Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3060

Laufzeitrekonfiguration modularisierter Overlaynetze

Christoph Schlameuß

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Kurt Rothermel
Betreuer:	Dr. Boris Koldehofe
begonnen am:	22. Juli 2010
beendet am:	21. Januar 2011
CR-Klassifikation:	C.2.2, C.2.4, C.2.5

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Begriff	**Erklärung**
Anwendungsschicht	Oberste Schicht der Internetarchitektur.
API	Application Programming Interface: Die Programmierschnittstelle einer Anwendung zur Anbindung anderer Programme.
CAN	Content Addressable Network: Ein strukturiertes P2P-System auf Basis eines d-dimensionalen kartesischen Identifierraums.
Chord	Ringbasiertes strukturiertes P2P-System.
DHT	Distributed Hash Table: Konzept zur verteilten Speicherung von Informationen. Hierbei werden auf jedem Knoten mehrere Schlüssel/Wert-Paare hinterlegt. Um einen bestimmten Schlüssel zu finden muss zum zuständigen Knoten geroutet werden, der den entsprechenden Wert liefern kann.
Gossip	Verfahren zum Informationsaustausch zwischen Peers. Hierbei werden Informationen auf jeweiligem Peer aggregiert und an Nachbarn/-Kontakte weitergegeben (epidemische Verbreitung von Informationen).
Grid-Computing	Ein Netzwerk, in dem die Ressourcen auf verschiedene Standorte verteilt sind.
Hash	Unidirektionale Abbildung mit fester Länge beliebiger Eingabewerte, meist in Form von Bitstrings, dienen zur effizienten Unterscheidung von einzelnen Werten.
Knoten	Bezeichnet einen Teilnehmer innerhalb eines Netzwerkes (Node).
Lookup	Eine Suchoperation nach Informationen, Schlüsseln oder Werten innerhalb eines Netzwerkes.
MP	Message Passing: Kommunikation mehrerer Teilnehmer eines Netzwerkes durch Senden und Empfang von Nachrichten.
MQ	Message Queuing: Nachrichtenverteilsystem in Netztwerken.
Netzwerkschicht	Schicht der Internetarchitektur, die die Kommunikation über andere Knoten und somit indirekte Kommunikation erlaubt.
Netzwerklast	Belastung eines Netzwerkes durch Informationsaustausch in Form von Suchfunktionen.
P2P	Peer-to-Peer: Kommunikationsparadigma, bei dem die Endsysteme in Netzwerken direkt miteinander kommunizieren und jeder Teilnehmer die gleichen Rollen übernimmt.
Partitionierung	Aufteilung eines Raums oder Systems in verschiedene, abgeschlossene Bereiche ohne Verbindung untereinander.
Publish/Subscribe	Kommuikationsparadigma zur Nachrichtenweiterleitung basierend auf dem Observer Pattern.
Routing	Vorgang innerhalb eines Netzwerkes, die Verbindung zweier Teilnehmer zu finden und Pakete weiterzuleiten.
RPC	Remote Procedure Call: Ausführung von Prozeduren auf entfernten Systemen.

Begriff	**Erklärung**
SPoF	Single Point Of Failure: Komponente eines verteilten Systems, welche, bei Ausfall der Komponente, die Funktion des gesamten Systems unterbricht.
SON	Structured Overlay Network: Overlaynetzwerk mit durch einen verteilten Algorithmus festgelegten Verbindungen.
Takeover Node	Teilnehmer eines P2P-Systems, der die Funktion eines ausfallenden Knotens übernimmt, so bald dessen fehlen erkannt wird.
Vermittlungsschicht	Schicht der Internetarchitektur, welche Verbindungen schaltet und einzelne Datenpakete zwischen verbundenen Knoten weiterleitet.

Zusammenfassung

Viele aktuelle Publish/Subscribe-Systeme verwenden Overlaynetze um ihre Funktion zu erfüllen. Hierbei ändert sich die Belastung der Systeme im Betrieb. In dieser Arbeit werden Verfahren entwickelt, die die Topologie strukturierter Overlaynetze transformieren, um die Leistungsfähigkeit der überliegenden Services zu verbessern. Hierzu wird eine dies ermöglichende Systemarchitektur vorgestellt, in die alle benötigten Module eingebettet werden. Diese Umfassen eine Indirektionsmechanismus, eine Adresstransformation sowie die tatsächliche Transformation. Konkret wird das Konzept am Beispiel der Transformation zwischen Chord- und CAN-Netzwerken erarbeitet. Für die Adresstranformation werden hierfür verschiedene raumfüllende Kurven, auf ihre Eignung zur Abbildung zwischen den verschiedenen Id-Räumen untersucht. Im Kernteil der Arbeit wird eine Transformation mit Hilfe eines Hybridnetzes erarbeitet, die die Transformation ohne Unterbrechung der Arbeitsfähigkeit und nur geringer Erhöhung des Arbeitsaufwands leisten kann. Um die Leistungsfähigkeit dieses Ansatzes nachzuweisen wird dieser einer nebenläufigen Neukonstruktion gegenübergestellt. Abschließend wird die grundsätzliche Umsetzbarkeit der erarbeiteten und vorgestellten Verfahren in einer experimentellen Evaluation nachgewiesen.

Kapitel 1

Einleitung

Im heutigen Internet haben viele Funktionalitäten wie Multicast oder Quality of Service (QoS) keine ausreichende Verbreitung erreicht, um effektiv von Applikationen genutzt zu werden. Es besteht jedoch die Möglichkeit, entsprechende Services auf der Applikationsschicht zu implementieren. Um diese Services betreiben zu können, sind keine Investitionen in die Infrastruktur notwendig, da die im heutigen Internet durchgehend angebotenen verlustbehafteten, paketorientierten Kommunikationskanäle hierfür ausreichend sind. Zudem können diese Services ohne Modifikationen an der verbindenden Netzwerkstruktur zugänglich gemacht werden, da die für die Services benötigten Protokolle nur in der Applikationsdomäne auf den teilnehmenden Endsystemen implementiert werden müssen. Somit können viele verschiedenartige Systeme ausgebracht, erprobt und als Produktivsysteme verwendet werden, ohne vorher langwierige Standardisierungsprozesse durchlaufen zu müssen. Die hierbei entstehenden Overlaynetze (ON) sind nicht auf das Client/Server-Paradigma begrenzt, sondern beruhen oft auf dem P2P-Paradigma oder hybriden Formen der beiden Paradigmen. Diese Overlaynetze können zudem durch spezielle Protokolle in ihrer Formgebung beeinflusst werden und so strukturierte Overlaynetze (SON) bilden.

Die hierbei gebildete Netzwerktopologie ist ausschlaggebend für ihre Eigenschaften in Hinblick auf Weiterleitung, Lastmanagement und Fairness. Je nach Verwendungszweck des Netzwerkes werden spezifische Vorzüge der vorliegenden Topologie ausgenutzt. Die hierbei ausschlaggebenden Vorzüge können beispielsweise der lokale Zustand, der zu erwartende Netzwerkdurchmesser oder das Clustering sein. Ändert sich während der Nutzung eines Netzwerkes die Art der Nutzung, ist es von Vorteil, die Topologie des Netzwerkes anzupassen, um so andere Eigenschaften der Topologie hervor zu bringen oder zuvor bestehende Eigenschaften zum Tragen zu bringen. Auslöser für die Änderung der Nutzungsart sind beispielsweise das Hinzufügen oder Entfernen von Inhalten, Verschiebungen in deren Popularität, Änderungen in der Menge der beteiligten Knoten, Mobilität von Knoten oder Änderungen in den überlagerten Services oder Applikationen.

Publish/Subscribe-Systeme dienen dazu, Nachrichten von N Sendern an M Empfänger ohne zeitliche Kopplung weiterzuleiten. Einige dieser Systeme setzen wiederum auf strukturierten Overlaynetzwerken auf. Hierbei wirkt sich die Netzwerkstruktur stark auf die Effizienz überlagerter Publish/Subscribe-Systeme aus. Die entgegengesetzte Einflussnahme

des Publish/Subscribe-Systems auf das Overlaynetz gestaltet sich hierbei jedoch schwierig, da diese durch das verwendete Protokoll vorgegeben ist. Hierzu muss folglich die Variabilität der Overlaynetze erhöht werden, in dem entweder dynamisch rekonfigurierbare Protokolle zum Aufbau verwendet werden oder das verwendete Protokoll selbst müsste austauschbar sein.

In Verlauf dieser Arbeit werden Lösungsansätze für eine Topologietransformation von Overlaynetzen erarbeitet und evaluiert, die auf dem Austausch der unterliegenden Protokolle beruhen. Konkret wird hierbei gezeigt, wie unterschiedliche Overlaystrukturen im Betrieb ineinander überführt werden können, um die Effizienz aufgesetzter Services, wie etwa Publish/Subscribe-Systemen, zu erhöhen.

Neben einer Architektur für diesen Zweck und verschiedenen naiven Ansätzen, mit denen eine solche Transformation durchgeführt werden kann, wird die Nutzung eines hybriden Netzes zur Transformation erläutert, den primitiven Ansätzen gegenübergestellt und bewertet. Jedoch soll hierbei nicht nur eine rekonfigurierbare Topologie entstehen, sondern vielmehr ein vollständiges Umschalten auf ein anderes Overlayprotokoll ermöglicht werden. Besondere Rücksicht wird hierbei auf die Erhaltung der natürlichen Lokalität der Knoten genommen, indem eine geeignete Adresstransformation verwendet wird. Dies vermindert zum Einen den Änderungsbedarf der verwalteten Daten und zum Anderen die Zahl der aufrechtzuerhaltenden Verbindungen zu Nachbarknoten. Weiterhin wird aufgezeigt, welche Gegebenheiten hierbei für die Erleichterung und Beschleunigung der Transformation genutzt werden können.

Als Ausgangspunkte der Transformationen dienen die ursprünglichen Vorschläge für SONs, auf denen der Großteil der heute verwendeten strukturierten Overlaynetze basiert. Diese sind Pastry [RD01], Tapestry [ZKJ01], Chord [SMK+01] und das Content-Addressable Network (CAN) [RFH+01]. Während Pastry, Tapestry und Chord im Wesentlichen auf einer Ringstruktur basieren, ist CAN auf einem d-dimensionalen Raum aufgebaut. Dieser strukturelle Unterschied zwischen Ring und Raum bietet die komplexesten Unterschiede in Hinsicht auf die Organisation des Id-Raums und somit der daraus folgenden Nachbarschaftsbeziehungen im Overlaynetz. Hiermit bietet die Transformation zwischen den, durch diese beiden Protokolle gebildeten Topologien, die größte Komplexität und somit ebenfalls das größte Optimierungspotential. Zusätzlich bieten diese Topologien das größte Potential für einen überlagerten Service, die resultierenden Änderungen in der Netzwerktopologie, zur Verbesserung der Serviceperformance.

Weitere Optimierungsmöglichkeiten ergeben sich durch die Partitionierung des Schlüsselraums der SONs in verschiedenartige Bereiche mit jeweils eigenem Routing. So kann die Lokalität und das Clustering in den einzelnen Bereichen gezielter als bisher gesteuert werden. Um hierbei eine jederzeit eindeutige Zuständigkeit zu gewährleisten, wird eine statische, bidirektionale Abbildung zwischen den verschiedenen Schlüsselräumen genutzt.

Im Kernteil dieser Arbeit wird eine Systemarchitektur vorgeschlagen, die einen variableren Umgang mit der aufgebauten Topologie erlaubt. In diese Architektur werden Schrittweise die, für die geplante Transformation benötigten, Elemente integriert. Die entwickelten

Elemente sind vorrangig eine Adresstransformation mit Hilfe von verschiedenen raumfüllenden Kurven und einige Transformationsmethoden. Hierbei werden zunächst einige naive Transformationsmethoden betrachtet. Diese umfassen ein direktes überführen der Knoten in ein anderes Protokoll und die Zerlegung des Netzes und Lösung der Transformation und Nachbarsuche in kleinen, später wiederzuvereinenden Teilnetzen. Außerdem aus den weiter vertieften Ansätzen, der sequenziellen Neukonstruktion des Zielnetzes, nebenläufig zum Ausgangsnetz und dem hier unterstützten Ansatz, die Knoten zunächst in eine hybride Zwischenstruktur zu überführen, in dieser alle Informationen für das Zielnetz zusammenzustellen und zu diesem überzugehen.

Zum Abschluss der Arbeit werden die erarbeiteten und vorgeschlagenen Verfahren und Algorithmen experimentell validiert und bewertet.

Aufbau der Arbeit

In Kapitel 2 werden einige thematische Grundlagen gelegt, sowie ein Zusammenhang zu bereits bestehenden Arbeiten hergestellt und zusätzlich in Kapitel 3 weiter abgegrenzt. In Kapitel 4 werden Grundlagen für die Transformation gelegt. Im Anschluss beschreibt Kapitel 5 das prinzipielle Vorgehen bei der Transformation von verschiedenen Overlays. Hierbei werden auch konkrete Kriterien und Vorgehensweisen für die Transformation erörtert. Kapitel 6 beschreibt die Evaluationsmethoden und -ergebnisse der erarbeiteten Verfahren. Kapitel 7 beinhaltet eine allgemeine Bewertung der praktischen Eignung der erarbeiteten Verfahren sowie einen Ausblick auf eine mögliche Weiterführung dieser Arbeit.

Kapitel 2

Grundlagen und Stand der Technik

In diesem Abschnitt werden die grundlegenden Begrifflichkeiten und Systeme, die im Zusammenhang mit dieser Arbeit stehen, erläutert.

2.1 Overlaynetze

Overlaynetze sind virtuelle Netzwerke, die oberhalb der Netzwerkschicht definiert sind. Hierbei sind die Netzwerkknoten nur durch das von der Netzwerkschicht bereitgestellte Netzwerk verbunden. Auf der Schicht des Overlaynetzwerks werden eigenständige Nachbarschaftsbeziehungen definiert, die das Overlaynetz bilden. Somit wird das Netzwerk der Netzwerkschicht zum Underlaynetzwerk. Bei der Traversierung der Verbindungen im Overlaynetz werden hierdurch im Normalfall mehrere Verbindungen auf der Netzwerkschicht traversiert. Ein einfaches Overlaynetz und das zugehörige Underlaynetzwerk sind in Abbildung 2.1 dargestellt. Die verschiedenen Overlaynetze sind hierbei durch die Art ihres Aufbaus und ihre daraus resultierende Struktur definiert.

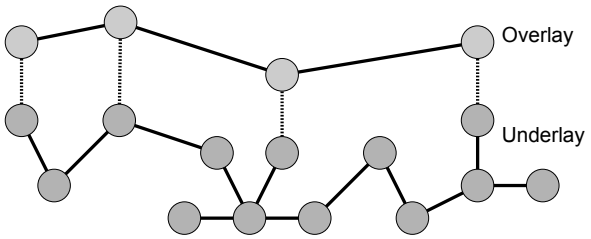

Abbildung 2.1: Overlaynetz mit zugehörigem Underlaynetzwerk

2.1.1 Overlaystruktur

Es gibt verschiedene Möglichkeiten, funktionsfähige Overlaystrukturen zu erzeugen. Hierbei werden unstrukturierte (ON) und strukturierte (SON) Overlaynetze unterschieden. Der

entscheidende Unterschied besteht in der Regulierung des Aufbaus des Overlaynetzes. Hieraus lassen sich verschiedene Topologien mit spezifischen Eigenschaften erzeugen. Die wichtigsten der erzeugten Eigenschaften sind der Netzwerkdurchmesser, die Verteilung des Knotengrades im Netzwerk und die Identifier-Zuordnung der Knoten.

In unstrukturierten ONs werden die Verbindungen zufällig durch den Aufbau des Netzwerkes bestimmt. Soweit keine weiteren Maßnahmen zur Strukturierung des Netzwerks ergriffen werden, entsteht hierdurch eine Potenzverteilung[1] des Knotengrades der Knoten [BA99]. Diese zeichnet sich durch viele Knoten mit niedrigem Grad und wenige Knoten mit hohem Grad aus. Der Grad eines Knotens ist durch die von ihm gehaltenen Verbindungen zu anderen Knoten bestimmt. Durch den zufälligen Aufbau der ONs entsteht zwar typischerweise ein relativ geringer Netzwerkdurchmesser, dieser kann jedoch nachweislich nicht garantiert werden. Weiterhin kann in ONs keine Zuständigkeit der Knoten für bestimmte Schlüssel festgelegt werden, da ein einzelner Knoten seine Position im Overlaynetz nicht ohne weiteres exakt bestimmen kann. Somit können auch keine verlässlichen Aussagen über den Suchaufwand beziehungsweise den Sucherfolg getroffen werden. Abbildung 2.2 zeigt ein solches unstrukturiertes Netzwerk.

Im Gegensatz hierzu sind der Knotengrad und die Position im Overlaynetz der einzelnen Knoten in SONs generell festgelegt. Dies gelingt, indem streng kontrolliert wird, zu welchen anderen Knoten im Netz Verbindungen auf- oder abgebaut werden. Hierdurch werden gezielt Overlaynetze geschaffen, die einen niedrigen Netzwerkdurchmesser und ein hohes Clustering aufweisen. Hierdurch kann, unter Verwendung dies ausnutzender Routingmechanismen, die Terminierung von Nachrichtenzustellungen, wie beispielsweise Suchanfragen, garantiert werden kann.

Auf den so entstehenden Overlaynetzen können wiederum verschiedene Services aufgesetzt werden. Abschnitt 2.3 beschreibt einige dieser Anwendungen. Diese sind beispielsweise Distributed-Hash-Tables, mit deren Hilfe große Informationsmengen gespeichert werden können, oder Publish/Subscribe-Systeme, die der effizienten Informationsweiterleitung dienen. Abbildung 2.3 zeigt exemplarisch ein solches strukturiertes Netzwerk. Konkrete Implementierungen hiervon sind beispielsweise Chord (siehe Abschnitt 2.2.1) und CAN (siehe Abschnitt 2.2.2).

Die besonderen Eigenschaften von Overlaynetzen sind im Folgenden näher beschrieben.

2.1.2 Lastbalancierung

Da in P2P-Systemen jeder Teilnehmer an der Aufrechterhaltung der Funktion des Netzwerkes beteiligt ist, ist es wichtig, diesen Aufwand auf die einzelnen Teilnehmer zu verteilen. Hierbei existieren im Allgemeinen zwei Ansätze dies zu tun. Zum einen kann der Aufwand gleichmäßig auf alle Knoten des Netzwerkes aufgeteilt werden. Durch die gleichmäßige Verteilung der Aufgaben und Inhalte wird ein Netzwerk toleranter gegen Ausfälle einzelner Teilnehmer. Dies gilt ebenso für gezielte Angriffe auf das Netzwerk, da kein Knoten eine

[1] Power-law-distribution

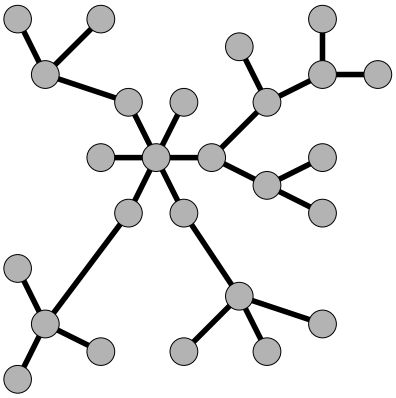

Abbildung 2.2: Unstrukturiertes Overlaynetz (ON)

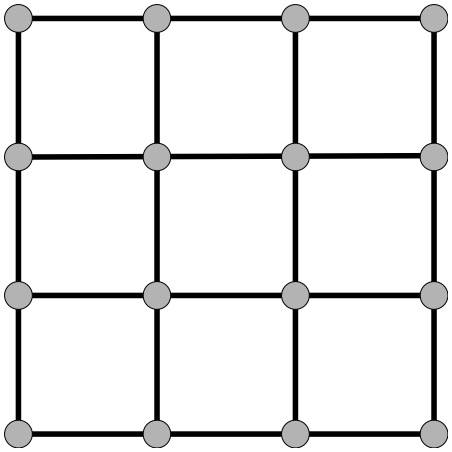

Abbildung 2.3: Strukturiertes Overlaynetz (SON)

besonders wichtige Rolle für das Netzwerk übernimmt, kann dem Netzwerk durch das Ausschalten einzelner Knoten kein ernsthafter Schaden entstehen.

Zum anderen kann der Aufwand fair, entsprechend den Ressourcen, die den jeweiligen Knoten zur Verfügung stehen, verteilt werden. Hierdurch wird dafür gesorgt, dass Knoten, die beispielsweise über eine bessere Anbindung verfügen, mehr Transferaufwand tragen. Dieses Konzept lässt sich ebenso auf andere Ressourcen, wie etwa die verfügbare Rechenleistung oder den verfügbaren Speicherplatz, übertragen. In einigen Systemen kommen hierbei auch geografische Aspekte zum Einsatz.

Der Begriff der Netzwerklast kann in Hinsicht auf verschiedene Gesichtspunkte betrachtet werden, wie die folgenden Abschnitte zeigen.

Knotengrad

In strukturierten Overlaynetzwerken ist der Knotengrad im Allgemeinen direkt durch das verwendete Protokoll vorgegeben. Abweichungen hiervon treten nur in geringem Umfang in SONs auf und sind im Allgemeinen durch unregelmäßige Id-Verteilungen verursacht. Hierbei unterhält ein Knoten üblicherweise mehr Verbindungen zu seiner direkten oder nahen Nachbarschaft im Id-Raum. Auf diesem Verhalten begründet sich die Funktionsweise der in den Overlays üblicherweise verwendeten Routingmechanismen. Dies steht im Gegensatz zu unstrukturierten Overlaynetzen, in denen sich durch das Wachsen des Netzes typischerweise Knoten mit sehr vielen Nachbarn, also sehr vielen Verbindungen im Overlaynetz, herausbilden.

Verteilung der Inhalte

Jeder sich, am Overlaynetz beteiligte Knoten sollte für ähnlich viele Inhalte verantwortlich sein, um die eingesetzten Ressourcen gleichmäßig zu verteilen. Um dies zu gewährleisten, wird eine Gleichverteilung der Inhalte über den gesamten verwendeten Schlüsselraum angenommen. Somit kann eine Gleichverteilung der Inhalte erreicht werden, in dem jeder Teilnehmer für einen ähnlich großen Teil des Schlüsselraums verantwortlich ist. Als Folge hiervon verbessert sich die Robustheit des Systems, indem durch den Ausfall einzelner Knoten nicht übermäßig viele Inhalte verloren gehen. Auch im oben beschrieben Sinne faire Systeme können durch die Größe des verwalteten Schlüsselraums beeinflusst werden. So können besser ausgestatteten Knoten größere Anteile des Schlüsselraums unterstellt werden.

Belastung durch Suche

Die Belastung durch Suchoperationen (Lookups), Nachrichtenweiterleitung und Routingaufgaben sollte ebenfalls auf die teilnehmenden Knoten verteilt werden. Zusätzlich sollten an den einzelnen Operationen möglichst wenig verschiedene Knoten beteiligt sein. Hierdurch wird ebenfalls aktiv Überlastungen einzelner Knoten vorgebeugt, da nicht das gesamte Netz von Suchanfragen betroffen ist. In unstrukturierten Overlaynetzen ist dies oft der am wenigsten beeinflussbare Gesichtspunkt, wenn garantiert werden soll, dass alle im Netz enthaltenen Ergebnisse gefunden werden.

2.2 P2P-Systeme

Peer-to-Peer-Systeme (P2P-Systeme) basieren auf der direkten Kommunikation der Teilnehmer untereinander. Dies steht im Gegensatz zur Kommunikation in Client/Server-Systemen, in denen nur zwischen Client und Server kommuniziert wird. Das P2P-Paradigma ermöglicht es somit, Ressourcen der teilnehmenden Endsysteme zu nutzen. Das Ziel hierbei ist es, den Aufwand und somit die Kosten zum Betrieb des Systems automatisch auf die einzelnen Teilnehmer zu verteilen. Die Fairness und Selbstorganisation stellen neben der Aufrechterhaltung der Funktionsfähigkeit und Konnektivität des Netzwerkes die zentralen Attribute der verschiedenen P2P-Systeme dar. Dies wird unter anderem auch durch Konzepte wie das Grid-Computing erreicht, P2P-Systeme sind im Gegensatz hierzu jedoch auf wesentlich höhere Ausfallraten ausgelegt. Sie sind also von Grund auf fehlertolerant aufgebaut. Hierdurch können, in Hinsicht auf Anbindung, Uptime und andere Ressourcen, auch unzuverlässige Peers als Teilnehmer genutzt werden. Hierzu werden in den einzelnen P2P-Systemen Redundanzen vorgesehen bzw. durch Replikation hinzugefügt.

P2P-Systeme bilden selbstständig fehlertollerante und vielen Fällen sich selbst entwickelnde Overlaynetzwerke. Hierbei sind von vorn herein alle Teilnehmer gleichgestellt (Peers). Dies wird nur in einigen hierarchischen P2P-Systemen aufgebrochen, in denen stärkere und zuverlässige Peers mehr Verantwortung für das Netz übernehmen.

Heutige P2P-Systeme bieten je nach ihrem Einsatzzweck eine Vielfalt von Eigenschaften, wie etwa effiziente Suche nach Inhalten, Rangequeries, Persistenz der gespeicherten Daten, Namensauflösung, Authentifizierung und Anonymität. Sie können für die Umsetzung verschiedenster Konzepte, wie etwa Nachrichtenaustausch (Message Passing), Remote Procedure Call (RPC), Nachrichten Warteschlangen (Message Queuing), DHT und Publish/Subscribe, eingesetzt werden.

P2P-Systeme können aus nur wenigen Teilnehmern oder aus mehreren tausend Knoten mit Teilnehmern in der ganzen Welt bestehen, sind also besonders skalierbar. Sie werden dazu genutzt, Netzwerkservices auf der Anwendungsschicht zu realisieren. Jedoch bringt dies einen erhöhten Kommunikationsaufwand gegenüber der Realisierung auf der Vermittlungsschicht mit sich. Mit der ansteigenden Zahl solcher Systeme, die simultan betrieben werden, steigt auch der nötige Kommunikationsaufwand, um die Funktion der Overlaynetze aufrecht zu erhalten. Dies ist vor allem der Fall, wenn mehrere Overlaynetze vom gleichen Endsystem genutzt werden sollen. Es kann also von Vorteil sein, diese recht kleinen Netzwerke und somit auch die enthaltenen/verwalteten Informationen miteinander zu verschmelzen. Eine Möglichkeit hierfür wäre es, auf dem Endsystem Informationen über potentielle Nachbarknoten und Verbindungen in lokalen Repositories zu verbinden und in mehreren Overlaynetzwerken simultan zu nutzen. Dieser Ansatz wird von Waldhost et al. in [WBH+08] verfolgt.

Neben den Bemühungen, die verwalteten Informationen und somit den lokalen Zustand zu minimieren, wird versucht, die verwendeten Overlaynetze in sich wandlungsfähiger zu gestalten. Im Allgemeinen unterscheiden sich die Overlaynetze in der Topologie, die durch das verwendete Protokoll gebildet und aufrecht erhalten wird.

Der Begriff des P2P-Systems beinhaltet keine Aussage, ob es über eine definierte Struktur verfügt oder nicht. Jedoch lassen sich, wie in Abschnitt 2.1.1 bereits erwähnt, belastbare Aussagen über die Netzqualitäten oft nur in strukturierten P2P-Systemen treffen.

Im Folgenden werden die für diese Arbeit relevanten P2P-Systeme Chord und CAN detaillierter vorgestellt. Dies sind zwei der ursprünglich vier Vorschläge für strukturierte P2P-Systeme, die um das Jahr 2001 entwickelt wurden. Diese sind Pastry [RD01], Tapestry [ZKJ01], Chord [SMK+01] und das Content-Addressable Network (CAN) [RFH+01]. Im Anschluss wird die Allgemeine von solchen Systemen bereitgestellte Schnittstelle erklärt.

2.2.1 Chord

Chord [SMK+01] ist ein ringbasiertes strukturiertes P2P-System, dass 2001 von Stoica et al. vorgestellt wurde. Hiermit ist Chord eines der ersten strukturierten P2P-Systeme, dass den Aufbau effizienter Overlaynetze ermöglicht. Es stehen Implementierungen in verschiedenen Sprachen und mit verschiedensten Erweiterungen zur Verfügung. Hierdurch hat Chord einen sehr hohen Bekanntheitsgrad und somit eine weite Verbreitung gefunden.

Die Position auf dem Ring wird durch einen für jeden Knoten einmaligen Identifikator (auf dem Intervall $[0, K[)$ bestimmt. Hierbei unterhält jeder Knoten Verbindungen zu seinen direkten Nachfolgern auf dem Ring. Der direkte Nachfolger wird zwingend benötigt und stellt die minimale Randbedingung zum Routing im Netz dar. Die weiteren Nachfolger dienen zur Absicherung gegen Knotenausfälle der direkten Nachfolger, also zur Stabilisierung des Rings nach Knotenausfällen. Eine zusätzliche Verbindung wird zum Vorgänger auf dem Ring gehalten. Um die Suchpfade zu verkürzen, unterhält jeder Knoten zusätzlich Verbindungen (Shortcuts) zu $logN$ weiteren Knoten in quadratisch steigender Entfernung im Identifierraum. Hierbei ist jeder Knoten für seinen eigenen sowie alle bis zu seinem Nachfolger folgenden Identifier zuständig.

Während die Existenz und die Korrektheit der direkten Nachfolger streng kontrolliert werden muss, um ein korrektes Routing zu gewährleisten, muss die Korrektheit der Shortcuts nicht sonderlich kontrolliert werden, da diese nur zur sicheren Verkürzung des Routingpfades verwendet werden, das Netz aber auch ohne sie Routingfähig bleibt. Somit führen Fehler in der Distanz der Shortcuts nicht zu Routingfehlern, solange das Überspringen des gesuchten Identifiers verhindert wird.

Üblich sind Identifierlängen von $N = 128$ Bit. Hieraus resultieren 2^N, also 2^{128} Identifier und somit maximal ebenso viele Teilnehmer.[2]

Abbildung 2.4 zeigt die Ringstruktur (links) sowie einen einfachen Lookup (rechts) auf dem Chord-Ring.

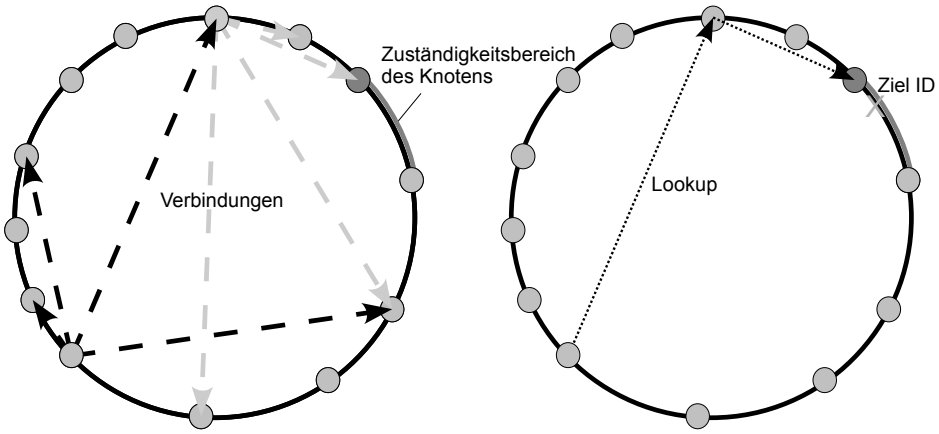

Abbildung 2.4: Struktur eines Chord-Netzwerks

Routing

Nachrichten werden von jedem Knoten an den dem Ziel am nächsten liegenden Knoten weitergeleitet, ohne hierbei über den gesuchten Identifier hinaus zu springen. Setzt man nun

[2] ca. 340 Sextillionen oder $340 * 10^{36}$

voraus, dass jeder Knoten nur seinen direkten Nachfolger kennt, würde sich eine mittlere Distanz von $N/2$ für das Routing auf dem Ring ergeben. Zudem stellt das Routing auf der Oberfläche des Rings die grundlegende Funktionalität des Routings sicher. Durch die Nutzung der bereits erwähnten Shortcuts lässt sich die Routingdistanz im Mittel auf $logN$ verkürzen.

Stabilisierung

Jeder Knoten führt zyklisch einen Stabilisierungsalgorithmus aus, der in Abbildung **??** dargestellt ist. Dieser gewährleistet die Integrität des Rings, indem überprüft wird, ob sich zwischen dem jeweiligen Knoten und seinem Nachfolger ein neuer Knoten befindet. Ist dies der Fall, wird dieser in den Ring integriert in dem der neue Knoten in beiden Protokollinstanzen als *successor* beziehungsweise *predecessor* abgelegt wird.

2.2.2 Content Addressable Network (CAN)

Das Content Addressable Network (CAN) [RFH+01] wurde von Ratnasamy et al. im Jahr 2001 vorgestellt. Hiermit wurde es etwa zeitgleich mit Chord vorgestellt und wird seit dem erweitert und weiterentwickelt.

CANs fassen den Id-Bereich als d-dimensionalen kartesischen Koordinatenraum auf. In diesem Bereich unterhalten die Teilnehmer im Optimalfall $2 * D$ Verbindungen zu Nachbarknoten. Jeweils eine Verbindung in jeder Richtung in jeder Dimension. Dennoch handelt es sich um ein strukturiertes P2P-System mit niedrigem und nahezu konstantem Knotengrad. Nicht alle Knoten verfügen über diesen konstanten Knotengrad, da die Zahl der Nachbarn einiger Bereiche durch unterschiedliche Partitionierung der Bereiche nicht konstant ist, also eine ungleichmäßige Verteilung der Teilnehmer im Id-Bereich.

Abbildung 2.5 zeigt beispielhaft die Struktur eines CAN-Netzwerkes in zwei Dimensionen, das mit acht Protokollinstanzen besetzt ist, sowie die Teilung des Id-Bereiches eines Teilnehmers, wie sie beim Hinzufügen von Knoten auftritt. Hierbei wird der Id-Bereich bei jeder Teilung in einer weiteren Dimension aufgeteilt. Das Optimum stellen hierbei ein quadratischer ($D = 2$), kubischer ($D = 3$) usw. Id-Bereich dar. Ist dies nicht der Fall, steigt die Zahl der Nachbarn entsprechend der Zahl der aneinander grenzenden Id-Bereiche an.

Um einen neuen Knoten dem Netzwerk hinzuzufügen, tritt ein Knoten des Netzwerkes die Hälfte seines Zuständigkeitsbereichs und seiner Daten an den neuen Knoten ab. Die Nachbarschaft, also die Informationen über die Nachbarn des Knotens, kann ebenfalls übernommen werden. Im Anschluss hieran müssen die Nachbarn über den neuen Knoten und somit die neuen Zuständigkeiten benachrichtigt werden, um die Konsistenz des Netzwerkes zu erhalten.

Durch die so gebildete Teilung des Id-Raums in d-dimensionale Bereiche eignet sich CAN besonders für Bereichsabfragen über mehrere Dimensionen. Hierdurch werden im Vergleich

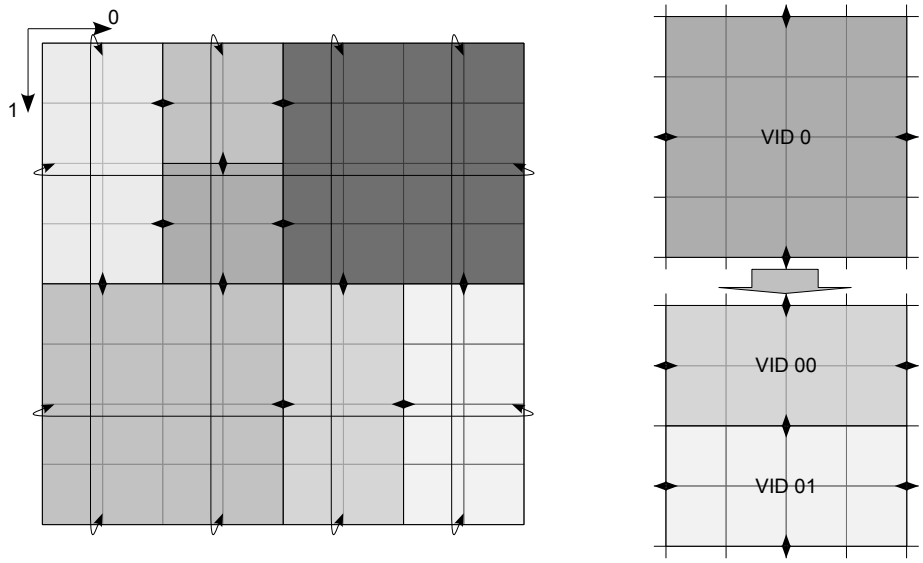

Abbildung 2.5: Struktur eines Content Addressable Network in zwei Dimensionen

zu anderen P2P-Systemen relativ wenige Knoten betroffen, was den gesamten Kommunikationsaufwand reduziert.

Knotenausfälle

Damit das Netzwerk routingfähig bleibt, wenn Teilnehmer das Netzwerk verlassen, ist für jeden Bereich ein *Takeover Node* definiert. Dieser übernimmt den frei gewordenen Id-Bereich, sobald das Fehlen des ausgefallenen Knotens festgestellt wird. Der *Takeover Node* ist als der Knoten mit der geringsten Differenz in der Virtual Node Id (VID) des ausgefallenen Knotens definiert. Die VID ist bestimmt durch die vorhergehenden Partitionen, durch die der jeweilige Id-Bereich zustande gekommen ist. Die VIDs bilden den Partitionierungsbaum des CAN.

Hat ein Knoten zuvor einen Bereich von einem anderen Knoten übernommen und soll nun den Beitritt eines neuen Knotens behandeln, wird einer der Bereiche abgegeben, anstatt den Bereich weiter zu partitionieren.

Routing

In CAN-Netzwerken können verschiedene Routingverfahren angewandt werden. Im Allgemeinen werden Nachrichten im Netzwerk geometrisch in Richtung des Ziels weitergeleitet. Hierdurch wird in einem gleichmäßig belegten CAN eine durchschnittliche Länge der Routingpfade von $(d/4)(n^{1/d})$ erreicht. Jedoch bestehen auch bei Fehlstellen, aufgrund von Knotenausfällen, im Netzwerk oft mehrere alternative Routingpfade zum jeweiligen

Zielknoten. CAN verfügt also bereits über eine integrierte Redundanz, was den Bedarf an zusätzlicher Redundanz etwa durch Replikation des Netzwerkes reduziert.

Eine genauere Betrachtung der möglichen Routingverfahren findet sich in [PEFK09].

2.2.3 P2P-System API

Alle P2P-Systeme stellen höheren Schichten relativ einfache Programmierschnittstellen (APIs) zur Verfügung, um über das System zu kommunizieren, oder Services darauf aufzusetzen. Diese stellen somit eine Abstraktion des erbrachten Services dar.

Die bereitgestellte Schnittstelle besteht aus mindestens drei Methoden. Dieses beinhaltet in jedem Fall eine Methode zum Senden von Nachrichten über das P2P-System, sowie eine Methode zum Empfangen von Nachrichten. Die Argumente der Methoden sind jeweils eine zu sendende oder zu empfangene Nachricht (*message*) sowie eine Id (*id*) beziehungsweise einen Schlüssel als Ziel oder Ursprung der Nachricht. Zum Senden einer Nachricht kann ein optionaler Netzknoten (*nextHop*) angegeben werden, falls bereits ein guter Kandidat, für den anstehenden Routingvorgang bekannt ist. Die dritte Methode wird vor allem von komplexeren Services genutzt. Diese wird jeweils aufgerufen, bevor eine Nachricht zu einem anderen Knoten weitergeleitet wird. Als Argumente werden die zu weiterleitende Nachricht (*message*), die Ziel-Id (*id*) und Informationen (*nextHop*) über den nächsten Knoten übergeben. Sie dient einerseits dazu Informationen über weitergeleitete Nachrichten an überlagerte Schichten weiterzuleiten, andererseits kann an dieser Stelle von höheren Schichten Einfluss auf die versendeten Nachrichten genommen werden, wie etwa deren Inhalt oder Ziel zu verändern.

Abbildung 2.1 zeigt die minimale Schnittstelle eines P2P-Systems.

```
route( id, message, nextHop )

forward( id, message, nextHop )

deliver( id, message )
```

Listing 2.1: Minimale Schnittstelle eines P2P-Systems

Zusätzliche Methoden sind je nach verwendetem P2P-System möglich. Hierzu sei auf die weiterführende Lektüre verwiesen [DZD+03]. Unter anderm unternehmen Dabek et al. hierbei bemühungen, ebenfalls eine einheitliche Schnittstelle für den Zugriff auf spezifischere Routinginformationen zu nehmen.

Die Empfangs- sowie Weiterleitungsmethoden sind hierbei über Events oder CallBacks implementiert, werden also im Allgemeinen nicht direkt aus dem überliegenden Service aufgerufen, sondern vor Gebrauch registriert und später von Seiten des unterliegenden P2P-Systems aufgerufen. Alternativ können diese auch als blockierende Methoden ausgeführt werden, die die Ausführung erst nach dem Empfang einer Nachricht fortführen.

Eine Besonderheit bietet hierbei die Empfangsfunktion. Diese liefert alle Nachrichten, für die der jeweilige Knoten verantwortlich ist, also nicht nur die direkt an ihn adressierten Nachrichten, an den überliegenden Service aus.

Wird die *route*-Methode aufgerufen, wird die hierbei übergebene Nachricht an das P2P-System weitergeleitet. Dieses ermittelt daraufhin entsprechend des genutzten Routingverfahrens den nächsten Knoten auf dem Routingpfad des P2P-Systems. Vor der Weiterleitung der Nachricht über die unterliegenden Netzwerkschichten wird die *forward*-Methode aufgerufen. Sofern die Ausführung weitergeführt wird, wird die Nachricht an den nächsten Knoten weitergeleitet. Auf diesem wird wiederum der nächste Knoten auf dem Routingpfad ermittelt, die *forward*-Methode aufgerufen und die Nachricht schlussendlich weitergeleitet. Dies wird so lange wiederholt, bis der für die Ziel-Id zuständige Knoten erreicht ist. Auf diesem wird die *deliver*-Methode aufgerufen und somit die Nachricht an den zuständigen Knoten zugestellt.

In Abschnitt 2.3.2 wird beschrieben, wie die hier vorgestellte Schnittstelle verwendet werden kann, um ein einfaches Publsih/Subscribe-System aufzubauen.

2.3 Anwendungen von P2P-Systemen

Auf den verschiedenen P2P-Systemen und somit auf den hierdurch gebildeten Overlaynetzen können verschiedene Services oder Anwendungen aufgesetzt werden. Einige hiervon werden in diesem Abschnitt beschrieben.

Die einzelnen Services können hierbei verschiedenste Komplexitäten aufweisen. Dies beginnt bei relativ simplen Speicherlösungen, wie den weiter unten erläuterten Distributed Hash Tables. Es lassen sich aber auch aufwändigere Services, wie Publish/Subscribe-Systeme in verschiedenen Ausprägungen, realisieren.

2.3.1 Distributed Hash Tables (DHT)

Einer der direktesten Services, der sich mit P2P-Systemen bereitstellen lässt sind die Distributed Hash Tables (DHT). Sie repräsentieren einen der grundlegenden Services, die von P2P-Netzwerken zur Verfügung gestellt werden können und auch wiederum als Substrat für verschiedene andere Services dienen kann. Sie dienen dazu, Daten in Form von Schlüssel/Wert-Paaren in verteilten Systemen zu speichern.

Um Werte aus der DHT abzurufen, muss der entsprechende Schlüssel bekannt sein. Um die Schlüssel festzulegen, unter denen die Nutzdaten oder Werte abgelegt werden, gibt es verschiedene Verfahrensweisen. Es kann der Hash der Nutzdaten berechnet werden, um diese zu hinterlegen. Um diese abzurufen, muss jedoch entweder der Hash oder der abgelegte Wert und die verwendete Hashfunktion bekannt sein. Hierdurch wird eine direkte Suche nach Inhalten erschwert. Wird diese Funktionalität benötigt, kann beispielsweise der Hash über den Dateinamen oder über Tags der Datei gebildet werden. Damit die Nutzdaten

unter mehreren Begriffen gefunden werden können, ist es gegebenenfalls notwendig, sie mehrfach unter mehreren Hashes in die DHT einzufügen oder im Fall der Suche mehrere Suchanfragen zu stellen.

Auf welchem Netzwerkknoten ein Schlüssel/Wert-Paar abgelegt wird, ist abhängig vom verwendeten Schlüssel. Hierzu wird der Schlüssel als Id im, unter der DHT liegenden, P2P-Netzwerk aufgefasst. Beim Einfügen wird das Schlüssel/Wert-Paar, entsprechend der Regeln des P2P-Netzwerkes, an den zuständigen Knoten weitergeleitet und auf diesem gespeichert.

Erreicht nun im Rahmen der Suche eine Anfrage den zuständigen Knoten, kann der den zugehörigen Wert beziehungsweise die Nutzdaten an den anfragenden Knoten übermitteln. Auf diese Weise können nahezu beliebige Daten in den Systemen hinterlegt werden. Abbildung 2.6 zeigt exemplarisch die Anordnung von Schlüssel/Wert-Paaren in einer DHT.

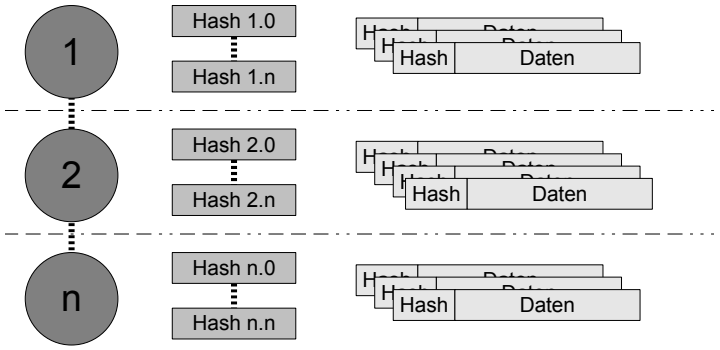

Abbildung 2.6: Anordnung von Schlüssel/Wert-Paaren in einer DHT

2.3.2 Publish/Subscribe-Systeme

Einen komplexeren Service als die DHTs stellen die Publish/Subscribe-Systeme dar. Diese realisieren einen zeitlich transparenten Nachrichtendienst. Hierbei können beliebige Sender Nachrichten an Gruppen von Empfängern zustellen. Der Leistungsumfang ähnelt hiermit einem zeitlich entkoppelten Multicast. Gemeinsam haben die beiden Systeme, dass sie auf den gleichen von den P2P-Systemen bereitgestellten Schnittstellen (API) aufbauen.

In Publish/Subscribe-Systemen wird zwischen den Sendern, die die Nachrichten senden, den Klienten, die die Nachrichten empfangen, und den Brokern, die ein Netzwerk bilden, um die Nachrichten an die Klienten weiter zu leiten, unterschieden. Die Nachrichten, die über das System verteilt werden sollen werden auf verschiedene Themen verteilt. Damit ein Klient Nachrichten zu einem Thema empfangen kann, muss er sich hierzu beim Broker registrieren. Abbildung 2.7 zeigt exemplarisch ein Publish/Subscribe-System. Genauer können die Systeme nach der Art der möglichen Subscriptions an den Brokern unterschieden werden.

Die einfachste Form von Publish/Subscribe-Systemen sind themenbasierte Systeme (Channel-based). In diesen Systemen können verschiedene Kanäle abonniert werden und schließlich auf dem Klienten gefiltert werden. Hierbei werden unter Umständen zu viele Nachrichten an die Klienten weitergeleitet, dies kann durch die mangelnde Feingranularität nicht verhindert werden.

Feinere Abstufungen können in inhaltsbasierten Systemen (Content-based) über zusätzliche Filter auf den Brokern realisiert werden. Dies ist vor allem von Bedeutung, wenn nicht ein einzelner Broker sondern ein Netzwerk von kooperierenden Brokern verwendet wird. Der Vorteil dieser Technik liegt in den Optimierungsmöglichkeiten für die Nachrichtenweiterleitung zwischen den Teilnehmern.

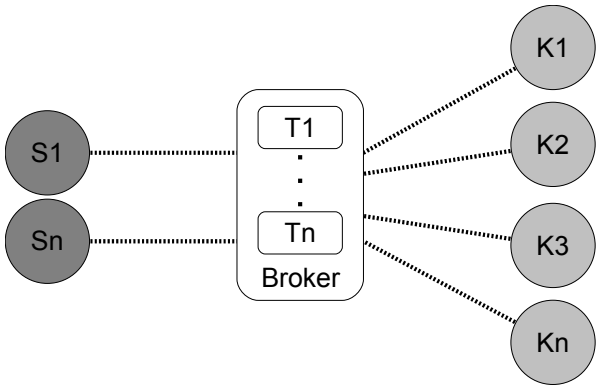

Abbildung 2.7: Publish/Subscribe-System mit Sendern (S), Themen (T) und Klienten (K)

Es gibt zwei unterschiedliche Ansätze, die vorgestellten Publish/Subscribe-Systeme dynamischer zu gestalten. Hierzu wird davon ausgegangen, dass nicht nur ein Broker im System vorhanden ist, sondern eine Vielzahl von Brokern, die einen gemeinsamen Dienst zur Verfügung stellen. Die Broker sind hierzu untereinander und mit den Klienten in einer verteilten Struktur miteinander verbunden. Diese kann im primitiven Fall von außen durch statische Verbindungen vorgegeben sein. Eleganter ist jedoch ein dynamischer Ansatz, bei dem die Broker ihre Kommunikationsbeziehungen eigenständig anpassen. Zwei verschiedene Ansätze spielen hierbei eine Rolle.

Ausgehend von dem beschriebenen Verbund von Brokern kann die Administration der Verbindungen und Weiterleitungen unter den Brokern automatisiert werden. Hierzu wird im einfachsten Fall ein Verteilbaum unter den Brokern aufgebaut und ständig schrittweise optimiert. In komplexeren Systemen kommen auch dynamischere Netzwerke zum Einsatz, durch deren Nutzung der erzeugte Nachrichtenoverhead weiter verringert wird, da weniger Nachrichten an daran nicht interessierten Knoten weitergeleitet werden. Die hierbei fälschlich übertragenen und zugestellten Nachrichten werden als False Positives bezeichnet. Es wird stets versucht, diese auf ein Minimum zu begrenzen. Dies ist möglich durch den Abgleich der auf den Brokern vorliegenden Abonnements (Subscriptions) auf den verschiedenen Kanälen. Verschiedene Möglichkeiten hierfür werden in [JMW+08] aufgezeigt. Je nach der verwendeten Granularität lässt sich dies weiter optimieren. Hierdurch steigt jedoch

zwangsläufig der Administrationsaufwand beziehungsweise der Kommunikationsaufwand für die Administration an.

Andere Ansätze machen sich die hiervon unabhängig entwickelten Overlaynetze zunutze. Zunächst werden einige Möglichkeiten hierzu vorgestellt. Mit Hilfe der bereits in Abschnitt 2.3.1 vorgestellten und von den Overlaynetzwerken bereitgestellte DHT-Funktionalität lassen sich ebenfalls Publish/Subscribe-Systeme realisieren. Ebenso ist es möglich, ein Publish/Subscribe-System auf der vorgestellten P2P-System API aufzusetzen. Dieses Vorgehen wird in Abschnitt 2.3.2 erläutert. Abschnitt 2.3.2 zeigt wie auch ein inhaltsbasierter Publish/Subscribe-Dienst auf strukturierten P2P-Systemen gestaltet werden kann.

Im Folgenden werden einige konkrete Systeme genauer beschrieben.

Scribe

Ein konkretes System, das dies auf Basis von Pastry[RD01] leistet, ist Scribe von Rowstron et al. [RKCD01]. Das Ziel des Systems ist es, einen effizienten und skalierbaren themenbasierten Publish/Subscribe-Dienst zur Verfügung zu stellen. Unterstützt wird hierbei zwar lediglich ein Best-Effort-Dienst, die Durchsetzung von geordneten Nachrichtenströmen kann aber auf höheren Schichten realisiert werden. Realisiert wird dies durch quellbasierte Verteilbäume entlang des Pastry-Netzwerkes, unter Nutzung der bereits vorgestellten P2P-System API.

Voraussetzung ist, dass alle Sender und Klienten dem Overlaynetz beitreten, um über dieses kommunizieren zu können. Die Rolle der Broker wird von allen Knoten im Netzwerk gemeinsam übernommen, das heißt, es werden keine dedizierten Broker-Systeme mehr benötigt. Während des Betriebs des Systems kann jeder Knoten neue Themen erstellen, indem er dieses abonniert oder eine Nachricht an dieses sendet. Hierzu wird dem Thema eine Id zugewiesen und eine entsprechende Nachricht in Richtung des zuständigen Knotens geroutet. Hierbei speichert jeder Knoten, den die Nachricht passiert die Themen-Id sowie seinen direkten Nachbarn, in Richtung des Initiators. Hierzu wird die *forward*-Methode der P2P-System API genutzt. Dieser Vorgang wird weitergeführt, bis der zuständige Knoten oder ein Koten, dem das Thema bereits bekannt ist, erreicht wird. Der zuständige Knoten, und damit die Wurzel des Weiterleitungsbaums, ist, der entsprechend der Pastry-Vorgabe zuständige Knoten, also der Knoten mit der zur Themen-Id ähnlichsten Id. So bildet sich für jedes Thema ein eigenständiger Verteilbaum.

Dieses Vorgehen folgt dem Vorbild des Reverse-Path-Forwarding (RPF), mit dem Multicastbäume aufgebaut werden können. Ebenso lässt sich dieser Ansatz auf andere strukturierte P2P-Systeme übertragen.

CAN-basiertes Publish/Subscribe

Ratnasamy et al. beschreiben eine Möglichkeit, Multicastdienste mit Hilfe von mehreren Instanzen von CAN-Netzwerken bereitzustellen [RHKS01]. Die beschriebene Art des Multicasts ist äquivalent zu themenbasiertem Publish/Subscribe. Anders als die in Abschnitt 2.3.2

vorgestellte Methode werden bei diesem Verfahren keine quellbasierten Verteilbäume aufgebaut. Jedes der Themen im erzeugten Publish/Subscribe-System wird durch eine eigene mini-CAN-Netzwerkinstanz repräsentiert. Um eine Nachricht in einem Themenkanal zu versenden, muss diese lediglich in der entsprechenden mini-CAN-Netzwerkinstanz geflutet werden. Um ein Thema zu abonnieren muss analog hierzu der entsprechenden mini-CAN-Netzwerkinstanz beigetreten werden. Jede mini-CAN-Netzwerkinstanz stellt eine Multicastgruppe dar.

Um einem unbekannten Thema beitreten zu können, wird neben den themenspezifischen Netzwerkinstanzen eine weitere Instanz eingeführt. In dieser werden unter Nutzung der DHT-Funktionalität Verbindungsdaten zu Kontaktknoten in den Themeninstanzen unter den entsprechenden Themen-Ids abgelegt. Somit kann also jeder Knoten, der Teil des Verwaltungsnetzwerkes ist, Knoten in den themenspezifischen Netzwerkinstanzen erreichen und diesen beitreten.

Das hier beschriebene Vorgehen lässt sich, abgesehen von der CAN-spezifischen Optimierung, auf die Nutzung anderer strukturierter P2P-Systeme übertragen.

Optimiertes fluten von CAN-Netzwerken

Für die themenspezifischen CAN-Netzwerkinstanzen wird weiterhin ein optimierter Flutungsalgorithmus vorgeschlagen. Dieser nutzt die kartesische Beschaffenheit des aufgespannten Id-Raums aus.

Der Vorgang folgt den folgenden vier Prinzipien: Der initiale Sender der Nachricht leitet diese an alle seine Nachbarn weiter. Jeder folgende Knoten, der die Nachricht über eine Verbindung in der Dimension d empfängt, leitet diese auf allen Verbindungen in den Dimensionen $< d$ weiter. Eine Nachricht wird maximal über die Hälfte der Ausdehnung einer Dimension in einer Richtung weitergeleitet. Kein Knoten leitet die gleiche Nachricht ein zweites Mal weiter.

Der hier beschriebene Algorithmus stellt eine der topologiespezifischen Besonderheiten dar, durch deren Ausnutzung es von Vorteil sein kann, die Netzwerktopologie zu rekonfigurieren.

Content-basierte Publish/Subscribe-Systeme

Die nächste Verfeinerung, ausgehend von themenbasierten Publish/Subscribe, sind die inhaltsbasierten Publish/Subscribe-Systeme. Diese bieten neben den Themen weitere Verfeinerungsmöglichkeiten der Nachrichtenströme. Wie ein inhaltsbasierter Publish/Subscribe-Dienst auf Basis beliebiger strukturierter P2P-Systeme aufgebaut werden kann beschreibt Baldoni et al. [BMVV05].

Die Klassifikation der einzelnen Nachrichten erfolgt mit Hilfe eines d-dimensionalen Attributvektors. Die enthaltenen Attribute sind durch ihren Namen und ihren Typ definiert.

Abonnements in diesem System bestehen aus einer Konjunktion von Beschränkungen auf den Attributen. Hierbei kann sowohl die Existenz, sowie der Wertebereich der einzelnen Attribute eingeschränkt werden. Werden hierbei disjunkte Beschränkungen verwendet, können diese als getrennte Abonnements aufgefasst werden.

Zum Betrieb des Systems werden nun zwei aufeinander abgestimmte Abbildungsfunktionen benötigt. Die erste bildet jede Nachricht auf eine Menge von Rendezvous-Ids ab, während die zweite die Abonnements auf die Mengen von Rendezvous-Ids abbildet. Mit Hilfe dieser Abbildungen können nun die Abonnements auf den die jeweiligen Rendezvous-Ids verwaltenden Knoten ablegen. Nachrichten werden ihrerseits ebenfalls an ihre betreffenden Rendezvous-Ids weitergeleitet und können auf den zuständigen Knoten mit Hilfe der Abonnements endgültig gefiltert werden. Hierdurch können sie, entsprechend der Abonnements, an die Abonnenten weitergeleitet werden.

Für die zur Zuordnung verwendeten Abbildungen und dem Speichern der Abonnements kommen verschiedene Strategien in Frage. Beim *Atttibute-Split* wählt der Abonnement einen der Rendezvous-Knoten und hinterlegt sein Abonnement nur auf diesem. Hierdurch wird zum einen Mehrfachzustellungen der Nachrichten vorgebeugt zum anderen kann die Last so besser auf die verschiedenen Knoten verteilt werden. Beim *Key Space-Split* werden die verwendeten Ids in kleinere Bereiche aufgeteilt, die jeweils ein Attribut repräsentieren. Hierdurch wird die Zahl der Rendezvous-Knoten stark eingeschränkt. Beim *Selective-Attribute* wird das Attribut mit dem höchsten Einschränkungspotential bestimmt. Im Anschluss wird das Abonnement auf dem entsprechenden Rendezvous-Knoten gespeichert. Der Erfolg dieser Methode beruht auf dem Grad der stärksten Einschränkung innerhalb der einzelnen Abonnements.

Um dies umzusetzen, wird, wie bei anderen Publish/Subscribe-Systemen auch, oberhalb des genutzten P2P-Systems ein zusätzlicher Abstraktionslayer erzeugt. Dieser stellt die Funktionalität überlagerten Systemen zur Verfügung.

Kapitel 3

Problembeschreibung und Abgrenzung zu anderen Arbeiten

In diesem Kapitel wird ein Szenario und eine hierin eingebettete Problemstellung vorgestellt. Hierzu werden die entsprechenden Randbedingungen und Herausforderungen aufgezeigt. Zusätzlich wird eine Abgrenzung zu verwandten Arbeiten vorgenommen.

3.1 Szenario

Die verschiedenen Varianten der heute genutzten Publish/Subscribe-Systeme nutzen entsprechend ihrem Aufbau verschiedene Vorteile der unterliegenden Topologie. Dies ist darin begründet, dass die Publish/Subscribe-Systeme die Verbindungen unter ihren Teilnehmern, entsprechend der Verbindungen und Nachbarschaftsbeziehungen im genutzten Overlaynetz, aufbauen. So entstehen je nach System entsprechende Verteilbäume oder -netze.

Im Betrieb dieser Systeme werden in jeder Komponente Änderungen vorgenommen. Dies geschieht durch das Beitreten und Austreten von Teilnehmern in und aus den Systemen. Ähnlich wirken sich mobile Geräte aus, die nicht dauerhaft im Netz vertreten sind oder deren Verbindungsparameter sich ändern können. Außerdem ändert sich, während des ausgedehnten Betriebs des Systems, die Popularität der verwalteten Themen. Hier durch ändert sich der Nachrichtenfluss und das Nachrichtenaufkommen im gesamten System.

Des Weiteren soll für die zukünftige Nutzung ermöglicht werden, den Vorgaben aus höheren Schichten und komplexeren Services, zu entsprechen und hierzu Modifikationen an dem verwendeten Overlaynetz vorzunehmen. Hierbei sind insbesondere eventuelle Sicherheitsanforderungen an zukünftige Systeme zu bedenken, um sensible Nachrichtenströme auf bestimmte Teile von Firmennetzwerken zu begrenzen. Zusätzlich benötigt das verwendete Publish/Subscribe-System ein strukturiertes Underlay.

Im Weiteren Verlauf wird davon ausgegangen, dass die oben genannten Änderungen im Umfeld des verwendeten Publish/Subscribe-Systems erkannt werden und hieraus ein Bedarf

abgeleitet wird, die Overlaytopologie zu ändern. Dieser wird dann an das genutzte P2P-System weitergeleitet.

3.2 Herausforderungen

An die beabsichtigten Transformationen werden einige zusätzliche Anforderungen gestellt, die eine optimale Lösung für die Transformation erfüllen muss.

Die Transformation soll zwischen den strukturierten P2P-Systemen Chord und CAN stattfinden. Die wichtigste Herausforderung ist hierbei die durchgehende Funktionsfähigkeit des Systems. Die Transformation soll also für überlagerte Systeme transparent ablaufen. Hierzu gehört auch, dass das Netzwerk während der Transformation nicht partitioniert oder anderweitig über die Maße in Mitleidenschaft gezogen wird.

Um einen möglichst großen Nutzen aus der Transformation zu ziehen muss die Transformation eine möglichst geringe Nachrichtenkomplexität aufweisen. Auch die Dauer muss überschaubar sein, um möglichst schnell einen Vorteil aus der Transformation zu ziehen.

Einhergehend mit der durchgängigen Funktionsfähigkeit des Netzwerkes kann es zusätzlich von Vorteil sein, das verwendete Netzwerk nicht komplett zu transformieren, sondern die Transformation in ihrem Verlauf zu stoppen, um so einzelne Bereiche mit verschiedenen Eigenschaften in einem gemeinsamen Netzwerk zu schaffen. Diese Möglichkeit muss ein optimaler Algorithmus unterstützen.

Die konkreten Lösungsmöglichkeiten für diese Herausforderungen werden in Kapitel 4 beziehungsweise Kapitel 5 behandelt.

3.3 Problembeschreibung

Der Kernteil dieser Arbeit beschäftigt sich mit der Topologieanpassung und somit der Transformation zwischen einem Chord- in ein CAN-Netzwerk. Um dies zu ermöglichen müssen zuerst die signifikanten und relevanten Unterschiede und Gemeinsamkeiten der beiden Topologien ermittelt werden. Hierzu werden zunächst die Vorbedingungen für eine Rekonfiguration des Overlaynetzes erarbeitet. Sind diese ermittelt, muss ein Weg gefunden werden, um Einfluss auf die Konstruktionsmechanismen der durch die P2P-Systeme entstehenden Overlaynetze zu nehmen. Um dies zu ermöglichen wird zusätzlich eine angepasste Systemarchitektur benötigt, die den Austausch, oder das Umschalten zwischen verschiedenen Protokollen auf einer Netzwerkschicht, ohne Unterbrechung der Funktionalität des Systems, gewährleisten kann. Im Anschluss muss die Topologie der Netzwerke angeglichen werden und ein Weg gefunden werden, die in den Netzwerken enthaltenen Daten zu transferieren. Mit dem Transfer der Daten muss ebenfalls die Funktionalität des überliegenden Publish/Subscribe-Systems übertragen werden.

3.3.1 Abhängigkeit vom verwendeten Overlaynetz

Entsprechend dem aufgestellten Szenario werden im Folgenden nur die Overlaybasierten Publish/Subscribe-Systeme näher betrachtet.

Unabhängig davon, ob Verteilbäume, optimiertes Fluten oder andere Mechanismen den einzelnen Publish/Subscribe-Systemen zugrunde liegen spielt die tatsächliche Ausprägung der verwendeten Overlaynetze eine große Rolle. Werden Verteilbäume genutzt, werden diese typischerweise entlang der Verbindungen im Overlaynetz aufgespannt. In Systemen, wie dem in Abschnitt 2.3.2 vorgestellten optimierten CAN-Multicast, werden Besonderheiten der Topologien ausgenutzt.

In Zukunft ist es reizvoll, verschiedene Ansätze abhängig von den aktuellen Gegebenheiten miteinander zu verknüpfen. Um die Flexibilität der Publish/Subscribe-Systeme zu steigern, muss also die Struktur des Overlaynetzes, auf dem sie aufgebaut sind, rekonfiguriert werden. Um dies zu erreichen, können die im Netzwerk enthaltenen Knoten ihre Identifier und somit ihre Position im Overlaynetz anpassen. Hierdurch würden jedoch bereits aufgebaute und genutzte Verteilstrukturen überliegender Services beinträchtigt oder zerstört werden. Der bessere Ansatz ist es, dies nicht durch die Änderung der Ids, sondern durch die Rekonfiguration der gesamten Overlaystruktur, also durch den gezielten Auf- und Abbau von Nachbarschaftsbeziehungen, zu erreichen.

Die hierfür geforderte Flexibilität und Modularität ist ein logischer Schluss aus der Entstehung der heutigen Overlaynetze. Diese wurden erschaffen, um Mängel der unterliegenden Netzwerke auszugleichen und die Kooperation vieler einzelner Netzwerkknoten zu erleichtern. Heute werden die bestehenden Overlaynetze weiter verfeinert. Dies gilt besonders für die Klasse der P2P-Systeme.

3.4 Abgrenzung zu anderen Arbeiten

Da diese Arbeit nicht der erste Versuch ist, Overlays in ihrer Struktur zu beeinflussen werden an dieser Stelle verwandte und bereits bestehende Themen kurz erwähnt und von dem hier verfolgen Ansatz abgegrenzt.

Hierbei ist zu erwähnen, dass aufgrund der relativ hohen Aktualität der Entwicklung von strukturierten Overlaynetzen und Publish/Subscribe-Systemen, nur in begrenztem Umfang relevante Vorarbeiten zur Verfügung stehen. Dies bezieht sich insbesondere auf die Transformation zwischen heterogenen strukturierten Overlaynetzen.

3.4.1 Verschmelzen gleichartiger Overlaynetze

Es wurden bereits diverse Aufwände unternommen um gleichartige Netzwerke zu vereinigen oder zu stabilisieren. Dies ist vor allem nach Störungen der Netzwerke, wie etwa Partitionierungen oder massive Knotenausfälle nötig. Ghodsi et al. zeigen beispielsweise,

wie Gossiping-Algorithmen genutzt werden können, um ringbasierte SONs effizient zu stabilisieren [GHW07]. Erläutert wird dies am Beispiel von Chord. Auch Shafaat et al. behandeln ebenfalls die Stabilisierung und Verschmelzung von ringbasierten SONs nach Partitionierungen [SGH07]. Diese Stabilisierungen können ebenfalls dazu genutzt werden, um zuvor getrennt aufgebaute Netzwerke effizienter zusammenzuführen und zu einem Verbundnetz zu vereinigen. Jedoch wird, durch die hier verwendeten Verfahren, kein Zusammenführen heterogener Netzwerke unterstützt.

Ebenso existieren schon einige Bemühungen, um einzelne Systeme effizienter aufzubauen. Diese Ansätze können durchaus für die Nutzung innerhalb einiger, der später vorgestellten Transformationsmechanismen genutzt werden, um das Verhalten weiter zu verbessern. Aberer et al. beschäftigen sich in diesem Zusammenhang mit der effizienten, selbstorganisierten Erstellung von SONs [ADHS05]. Hierbei werden bessere Ergebnisse erzielt, als durch einen sequenziellen Aufbau der jeweiligen Systeme.

3.4.2 Overlaynetze mit variabler Struktur

In der Vergangenheit wurden Overlaynetze oft als unstrukturierte Netze aufgebaut, deren Topologie sich, je nach Bedarf, frei anpassen lässt ohne dass hierbei auf besondere Randbedingungen geachtet werden muss.

In Broker basierten Publish/Subscribe-Systemen ist es gängige Praxis, die Overlayverbindungen zwischen den einzelnen Knoten möglichst effizient zu Nutzen, um unnöten Overhead, etwa durch ein Fluten der Netzwerke zu vermeiden. Einige Ansätze gehen darüber hinaus, in dem sie aktiv Einfluss auf die verwendete Overlaytopologie nehmen.

Kumar et al. reorganisieren Overleynetze entsprechend der von überliegenden Publish/Subscribe-Systemen erzeugten Nachrichtenströmen[KCC+05]. Hierzu nutzen sie eine SQL-ähnliche Beschreibungssprache, um die beinhalteten Nachrichtenströme zu beschreiben. Auf dieser Grundlage optimieren sie den Verlauf der Nachrichtenströme in Hinsicht auf deren Charakteristika und den zur Verfügung stehenden Verbindungen im Overlaynetz.

Ähnlich gehen Baldoni et al. vor, um die Struktur des Baumbasierten Publish/Subscribe-Systems SIENA auf einem bestehenden Overlay zu optimieren[BBQV07]. Hierzu führen die Broker einen verteilten, selbstorganisierten Optimierungsgalgorithmus ein, der in der Lage ist die verwendete Overlaytopologie schrittweise zu rekonfigurieren.

Die in den bisher genannten Ansätzen verwendeten Overlays sind jedoch stets unstrukturiert. Diese Ansätze gehen also nicht auf zusätzliche Randbedingungen ein, die etwa durch strukturierte P2P-Systeme an die aufgebauten Overlaynetze gestellt werden.

Neben den bereits vorgestellten Netzwerken existieren ebenfalls Overlaynetze mit variabler Struktur, die gegen definierte Strukturen konvergieren können.

Ein, die Transformation in den Mittelpunkt stellender Ansatz ist T-Man [JB04] von Jelasity et al. Hierbei wird ein Verfahren vorgeschlagen, wie die Topologie von Overlaynetzen,

entsprechend einer gegebenen Bewertungsfunktion in eine Zieltopologie transformiert werden kann. Die Funktion wird hierbei genutzt um die in Frage kommenden Nachbarn zu bewerten. Um Informationen über teilnehmende Knoten im Netzwerk zu verbreiten werden *Gossiping*-Algorithmen genutzt. So kann letztendlich die durch die Bewertungsfunktion vorgegebene Struktur des Overlaynetzes geformt werden. Hierzu werden die einzelnen Knoten mit immer neuen Nachbarkandidaten versorgt und wählen hieraus die, entsprechend der Bewertungsfunktion, optimalen Nachbarn aus. So bildet sich letztendlich die durch die Bewertungsfunktion vorgegebene, Netzwerktopologie heraus, indem die Knoten lokal sortiert werden.

Hierdurch ist nahezu jede Topologie, wie beispielsweise Linien, Ringe, Tori oder Bäume, erzeugbar. Jedoch bleibt es schwierig, korrekte und haltbare Aussagen über die Qualität der Topologie zu treffen, solange sich diese noch im Aufbau befindet.

Dieses Vorgehen eignet sich zwar prinzipiell für die Transformation, es muss aber zusätzlicher Aufwand betrieben werden, um das nicht weiter spezifizierte Verhalten während der Transformation vorhersagbar zu gestalten. Während dieser Zeit ist das Netzwerk also im eigentlichen Sinne nicht stukturiert. Ebenso beinhaltet dies kein Verfahren mit dem nach der Transformation endgültig auf eine einfache Implementierung des Zielprotokolls umgeschaltet werden kann.

Der umgekehrte Ansatz wird von Tariq et al. verfolgt [TKKR09]. Es wird gezeigt, wie Nachrichtenverteilsysteme auf Topologieänderungen in unterlagerten Schichten reagieren können, um eine gleichbleibende Servicequalität zu gewährleisten. Hierzu wird in den Abonierungsprozess eingegriffen, um Verteilbäume höherer Kapazität und geringerer Latenz zu bilden.

Grenzt man das Vorgehen auf die Nutzung strukturierter Overlaynetzte und Modifikationen an der Overlaystruktur ein, so muss eine bisher nicht betriebene Art der Tranformation realisiert werden, um die gesteckten Ziele zu erreichen.

Kapitel 4

Vorbedingungen für die Transformation

In diesem Kapitel werden weitere Grundlagen erarbeitet um die folgenden Transformationen zu ermöglichen. Diese umfassen auch die betrachtete Systemumgebung sowie deren Einschränkungen.

Dafür werden zunächst die Unterschiede zwischen den Overlaystrukturen aufgezeigt. Im Anschluss daran wird eine Architektur für den Umgang mit transformierbaren Netzwerken vorgeschlagen. In diese werden die nötigen Transformationskomponenten eingefügt und erläutert. Hierzu zählen Kriterien für die Transformation, die benötigte Adresstransformation. Die eigentlichen Transformationskomponenten und Vorgehen werden in Kapitel 5 behandelt.

4.1 Systemmodell

Für die Gültigkeit dieser Arbeit werden einige Voraussetzungen angenommen, die die Gültigkeit und den Bearbeitungsumfang einschränken sollen. Die erste und wichtigste Annahme ist, dass alle Knoten in den betrachteten Overlaynetzwerken paarweise über ein bestehendes Underlaynetzwerk kommunizieren. Dies gilt, solange keine Partitionierungen des Overlaynetzwerkes betrachtet werden. In Folge dessen werden die vorgeschlagenen Protokolle nur auf der Applikationsschicht betrachtet und somit auch nur auf dieser modelliert. Außerdem wird das Bootstrapping der Netzwerke nicht betrachtet. Es wird davon ausgegangen, dass die verwendeten Netze bereits aufgebaut sind und mindestens ein aktiver Netzwerkteilnehmer, beispielsweise durch Tracker oder andere out-of-band-Protokolle, bekannt ist. Für einige der betrachteten Verfahren wird zusätzlich davon ausgegangen, dass die Knoten, anders als in realen Netzwerken beziehungsweise dem Internet, über FIFO-Kommunikationskanäle kommunizieren.

Zusätzlich wird festgelegt, dass im Rahmen dieser Arbeit nur Transformationen zwischen Systemen durchgeführt werden, die auf Id-Räumen mit gleicher Bitlänge ihrer Bezeichner basieren. Die Zahl der verwalteten Ids muss also übereinstimmen.

Im Verlauf dieser Arbeit wird der Begriff der Verbindung relativ frei verwendet. Gemeint ist hiermit, keine tatsächlich bestehende Verbindung im Sinne einer TCP-Verbindung, sondern lediglich die Bekanntheit der Adressdaten des Zielknotens der Verbindung. Für diese Art von Verbindung ist es also hinreichend, wenn der Zielknoten bekannt ist und prinzipiell kontaktiert werden kann. Das Bestehen einer solchen Verbindung muss daher sporadisch durch Testnachrichten geprüft werden, falls sonst keine Kommunikation stattfindet.

4.2 Kriterium und Ablauf der Transformation

In diesem Abschnitt werden der generelle Ablauf der Transformation sowie die Detektion des Transformationsbedarfs eingehender betrachtet.

4.2.1 Kriterium für die Transformation

Im Wesentlichen gibt es zwei Möglichkeiten für ein auslösendes Kriterium für die Transformation. Zum einen kann die Notwendigkeit der Transformation außerhalb des Protokolls festgestellt werden. Dies kann sowohl aus der unterliegenden Komponente auf dem Protokollstack, also aus dem Bereich der Middleware, in der ggf. mehr Informationen über die native Netzwerkstruktur vorliegen, kommen. Ebenso kann der Transformationsbedarf in der überlagerten Applikation festgestellt werden, etwa um auf bevorstehende Änderungen in der Netzwerknutzung oder -auslastung zu reagieren. Zum anderen kann das Protokoll, sobald es im Transformationszustand betrieben wird, auf Basis der über seine direkten Nachbarn vorliegenden Informationen sein Zielprotokoll anpassen, um die vorliegende Netzwerksituation zu optimiren.

Im Folgenden wird davon ausgegangen, dass die Transformation der Netzwerktopologie von außen ausgelöst wird. Nach dem Start der Transformation auf einem beliebigen Knoten wird die Transformationsnachricht im gesamten Netzwerk geflutet.

4.2.2 Ablauf der Transformation

Das generelle Vorgehen um eine fest definierte Overlaystruktur in eine andere, ebenfalls fest definierte Overlaystruktur zu überführen umfasst die folgenden Schritte.

Als Grundlage der Transformation muss eine leistungsfähige Architektur zur Verfügung stehen, die den Austausch, beziehungsweise die Änderung, der sich auf dem Protokollstack befindenden Protokolle ermöglicht. Des Weiteren muss dafür Sorge getragen werden, dass die Zuständigkeitsbereiche der einzelnen Knoten, über den Zeitraum der gesamten Transformation, klar definiert sind. Damit dies gewährleistet werden kann, ist eine eindeutige,

bidirektionale Adresstransformation nötig. Diese kann abhängig von der, für die Transformation verwendete Protokollpaarung, unterschiedlich ausfallen. Um Einfluss auf den Verlauf der Transformation zu nehmen werden Mechanismen benötigt, um die Transformation einzuleiten, auszubreiten, gegebenenfalls zu steuern und schließlich zu beenden, falls diese nicht selbstterminierend ausgeführt ist.

Hierauf aufbauend werden geeignete Transformationsmechanismen benötigt, die nach Möglichkeit die in Abschnitt 3.2 gestellten Anforderungen erfüllen.

4.3 Vergleich von Chord und CAN

Da im Verlauf dieser Arbeit vor allem die Transformation der beiden, in dieser Arbeit relevanten P2P-Systeme Chord und CAN, betrachtet wird, ist es zweckmäßig zuerst einen Blick auf die relevanten Unterschiede und Gemeinsamkeiten der beiden Systeme zu werfen.

Generell ist hierbei anzumerken, dass es sich um zwei rivalisierende Ansätze handelt, strukturierte P2P-Systeme aufzubauen. Beide Systeme erzeugen, streng strukturierte Overlaynetze mit logarithmischem Netzwerkdurchmesser. Jedoch beruhen sie auf grundlegend verschiedenen Ansätzen. Dadurch sind die Systeme auf verschiedenen Koordinaten- oder Id-Räumen aufgebaut. Im Endeffekt bedeutet dies, dass eine Chord-Id lediglich über zwei benachbarte Ids verfügt, dies sind bei CAN, abhängig von der Anzahl der Dimensionen, $2 * D$ Nachbar-Ids. Betrachtet man jedoch den zum Netzwerkbetrieb benötigten lokalen Zustand, also im Wesentlichen die Routingtabellen, so benötigt CAN nur etwa $2 * D$ Routingeinträge, während Chord üblicherweise $\log N$ Einträge in der Routingtabelle hält. Zum Ausgleich des Dimensionsunterschieds der Id-Räume können raumfüllende Kurven verwendet werden. Dieses Vorgehen wird in Abschnitt 4.5 behandelt.

Der signifikante Unterschied zwischen den beiden Systemen ist jedoch die Bestimmung der Nachbarknoten. In Chord kann für jeden Nachbarknoten die einfache hinreichende Bedingung angegeben werden, dass der Knoten für eine der Finger-Ids zuständig sein muss. Dies gilt für alle Nachbarknoten mit Ausnahme der Successor-Liste, die aber sehr einfach zu ermitteln ist.

Wie schon erwähnt verfügt CAN im Gegensatz zu Chord üblicherweise über weniger Nachbarn. Diese Nachbarn sind jedoch in ihrer Gesamtheit wesentlich schwerer zu finden. Zwar können alle Ids, die von den Nachbarn verwaltet werden müssen, berechnet werden, dies ist aber eine sehr große Anzahl. Somit gilt zwar für die Nachbarn generell die gleiche hinreichende Bedingung wie für die Chord-Nachbarn, allerdings sind dies bedingt durch die Geometrie des Identifierraums extrem viele Ids, für die nur wenige Nachbarknoten zuständig sind. Bei einem regulären Beitritt zu einem CAN-Netzwerk wird dieses Problem umgangen, da dem Knoten, der einen Teil eines Bereichs abgibt, alle Nachbarn bekannt sind. Einige Methoden zum Umgang mit diesem Problem finden sich in Abschnitt 5.2.3.

Während die Einträge in der Routingtabelle von Chord jeweils als unidirektionale Verbindungen anzusehen sind, entsprechen die Einträge in den CAN-Routingtabellen bidirektionalen

Verbindungen. Es muss also zu jedem Eintrag in der Routingtabelle eines Knotens ein entsprechend komplementärer Eintrag in der Routingtabelle des Zielknotens existieren.

Im Hinblick auf eine Transformation wirft der Übergang zu Chord wesentlich weniger Probleme auf. Dies ist der Fall, da der hier genutzte direkt verkettete Ring eine sehr einfache Invariante darstellt. CAN hingegen benötigt Verbindungen zu allen seinen direkten Nachbarn.

4.4 Architektur

Um die Topologie des Overlaynetzwerkes zu ändern, muss entweder ein Protokoll verwendet werden, das von sich aus eine variable Topologie unterstützt oder es muss ein zusätzlicher Indirektionsmechanismus verwendet werden, um das verwendete Protokoll während des Betriebs zu wechseln. Abbildung 4.1 zeigt den Aufbau der vorgeschlagenen modularen Architektur des Protokollstacks. Hierbei sind beide erwähnten Möglichkeiten vorgesehen, durch die Einfluss auf die Topologie der durch die Protokolle gebildeten Netzwerke genommen werden kann.

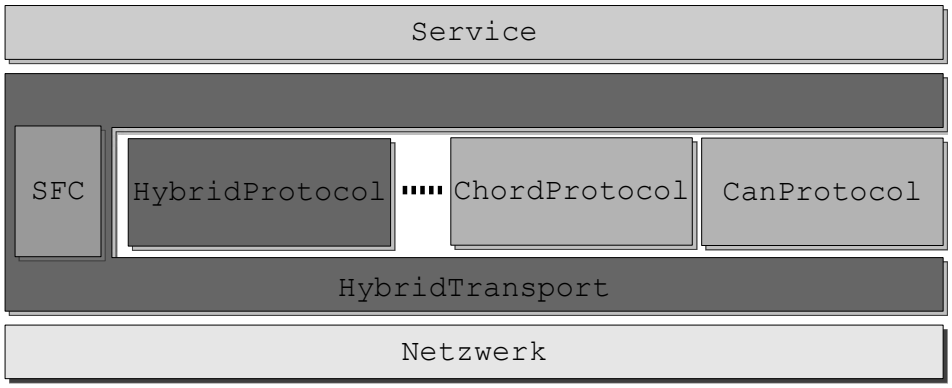

Abbildung 4.1: Protokollstack für die Transformation

Dieser Aufbau hat den Vorteil, dass nur minimale Modifikationen am Ausgangs- und Zielprotokoll vorgenommen werden müssen. Diese Änderungen beschränken sich auf einen Mechanismus, um die jeweils genutzten Routingtabellen auszulesen und zu füllen. Je nachdem, ob das Protokoll als Ausgangs- oder Zielprotokoll genutzt wird. Die Funktionsweise der Protokolle wird nicht beeinflusst.

Die zentralen, neu eingeführten Elemente auf dem Protokollstack sind der *HybridTransport* und das *HybridProtocol*. Der *HybridTransport* umschließt die zu nutzenden Protokolle, wird also jeweils zwischen überlagertem Service und Protokoll sowie zwischen Protokoll und unterliegenden Netzwerkschichten eingefügt. Für nicht-transformierende Protokolle agiert der *HybridTransport* transparent. Damit die Nachrichten in dem gekapselten Protokoll korrekt behandelt werden muss unter Umständen ein Adresstransformationen vorgenommen

werden. Zu diesem Zweck enthält der *HybridTransport* ein spezielles Modul, dass ihm dies ermöglicht. Dieses Modul ist als raumfüllende Kurve (*SFC*) in ihm eingebettet.

Aus Sicht eines normalen, nicht-transformierenden Protokolls müssen aus diesem Grund keine besonderen Gegebenheiten beachtet werden. Um dies zu ermöglichen, stellt der *HybridTransport* die entsprechenden APIs für die Protokolle zur Verfügung.

Das *HybridProtocol* ist der entscheidende Baustein, durch den die Transformation durchgeführt wird. Hierzu unterstützt es eine variable Topologie und reorganisiert diese entsprechend der beabsichtigten Transformation. Der *HybridProtocol*-Block steht hierbei stellvertretend für verschiedene unidirektional oder bidirektional ausgeführte Transformationsprotokolle. Typischerweise unterstützt ein *HybridProtocol* aber nur die Transformation zwischen zwei nicht-transformierenden Protokollen.

Um optimale Ergebnisse zu erzielen, muss für jede Protokollpaarung, zwischen denen die Topologien transformiert werden sollen, eine entsprechend angepasste Transformation vorgenommen werden. Hierdurch sind gegebenenfalls auch spezifische auslösende Kriterien für die Protokollpaarungen nötig.

Durch die Transparenz und integrierte Adresstransformation gegenüber nicht transformierenden Protokollen sind weitere Anwendungen der vorgestellten Architektur denkbar. So könnten Netzwerkknoten, die mit dieser Architektur ausgestattet sind druchaus mit Knoten einen Verbund bilden, die nur über den herkömmlichen Aufbau verfügen. Jedoch müssen hierbei gegebenenfalls weitere Randbedingungen beachtet werden. So muss der Aufbau der ausgetauschten Nachrichten entsprechend abgestimmt sein, oder zumindest transformiert werden können. Gegebenenfalls muss auf den Knoten, die die vorgeschlagene Architektur verwenden, ein speziell angepasstes Routing verwendet werden, um Zyklen zu vermeiden.

Unter dieser Voraussetzung können heterogene Netzwerke gebildet werden. Eine weitere wichtige Eigenschaft, die hierdurch erzeugt wird, ist die Eignung für ein inkrementelles Ausbringen der vorgeschlagenen Architektur. Hierdurch könnten Systeme im laufenden Betrieb, Knoten für Knoten umgestellt werden.

4.5 Adresstransformation

Um vor, nach und während der Transformation klare Zuständigkeiten der einzelnen Knoten für die einzelnen Ids zu erhalten und zu gewährleisten, ist eine jederzeit nachvollziehbare, bidirektionale und eindeutige Abbildung der verschiedenen Id-Repräsentationen untereinander notwendig. Besonderes Augenmerk erfordert hierbei die Mehrdimensionalität des CAN-Protokolls. Hierfür werden raumfüllende Kurven verwendet, welche in Abschnitt 4.5.1 genauer erläutert werden. Es ist wichtig, die bestehende Lokalität zu erhalten, um die Menge der zu transferierenden Ids für jeden Knoten zu minimieren und bestehende Nachbarschaftsbeziehungen weiter verwenden zu können. Die Lokalität ist ausschlaggebend da die in dieser Arbeit relevanten Protokolle die Gemeinsamkeit aufweisen, dass jedem Knoten mehr Knoten in seiner nahen Nachbarschaft bekannt sind, als weiter entfernte Knoten. Dies gilt auch für das Chord- und CAN-Protokoll.

4.5.1 Raumfüllende Kurven

Raumfüllende Kurven (space-filling-curves) werden benutzt, um eine Abbildung zwischen multidimensionalen und eindimensionalen Räumen zu schaffen. Hierbei wird, entsprechend der jeweiligen Konstruktionsregel, eine kontinuierliche Kurve durch den multidimensionalen Raum gelegt. Die Position auf der Kurve entspricht somit der jeweiligen Position im Raum. Die verschiedenen bekannten raumfüllenden Kurven unterscheiden sich, bedingt durch ihren Verlauf, in ihrer Auswirkung auf die Lokalität der einzelnen Punkte auf der Kurve beziehungsweise im Raum. Die Lokalität beschreibt die Ähnlichkeit zwischen der Nachbarschaft der Punkte auf der Kurve und der Punkte im Raum. Liegen also Punkte im Raum nahe beieinander, folgt daraus, dass sie auch auf der Raumkurve nahe beieinander liegen. Detailliertere Betrachtungen einiger raumfüllender Kurven [MAK02] und rekursiver Eigenschaften von raumfüllender Kurven [ARR+97] können der Literatur entnommen werden.

Viele der Kurven sind rekursiv aufgebaut, lassen sich also wie Fraktale stufenweise stets weiter verfeinern. Einige Beispiele für diese Kurven sind die Hilbert-Kurve, Peano-Kurve, Gray-Kurve, E-Kurve und die Z-Kurve. Diese sind in Abbildung 4.2 dargestellt.

Für eine bessere Unterstützung verschiedener Transformationen wird auch in der Simulation und Evaluierung eine Austauschbarkeit der verwendeten Raumkurven vorgesehen.

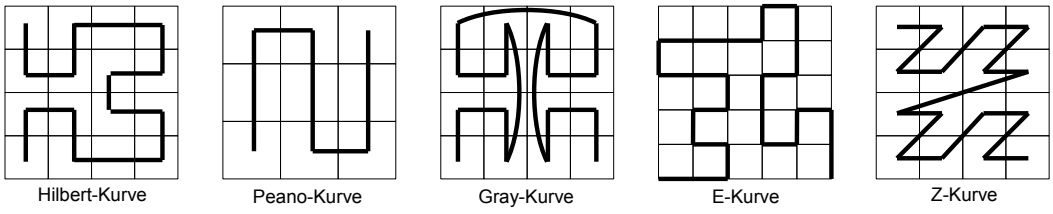

Abbildung 4.2: Beispiele für verschiedene Raumkurven

4.5.2 Sequenzielle Raumkurve

Die sequenzielle Raumkurve ist die einfachste Möglichkeit, eine raumfüllende Kurve zu erzeugen. Hierbei füllt die Kurve sequenziell eine Dimension nach der anderen. Hierdurch entsteht eine primitive Raumkurve, die zwar nicht frei von Sprüngen, jedoch sehr einfach zu implementieren ist. Um diese Kurve zu erzeugen, werden die Koordinaten der einzelnen Dimensionen zusammengefügt.

$$[112233445566778899]_{1D} \hat{=} [112233, 445566, 778899]_{3D}$$

Abbildung 4.3 und Abbildung 4.4 zeigen die entstehende Kurve beispielhaft.

Es sind zwar auch sequenzielle Raumkurven ohne Sprünge konstruierbar, aber deren programmgestützte Konstruktion ist nicht derart eindeutig und einfach, darum wird hier diese Raumkurve vorgestellt und verwendet.

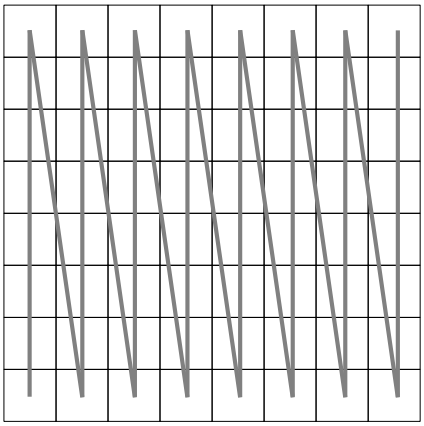

Abbildung 4.3: Sequenzielle 2D-Raumkurve

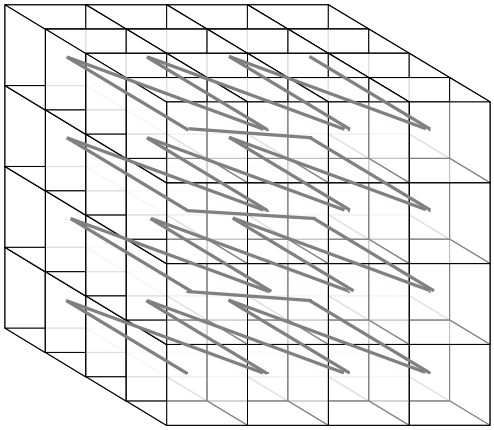

Abbildung 4.4: Sequenzielle 3D-Raumkurve

4.5.3 Hilbert-Kurve

Die Hilbert-Kurve ist eine rekursiv konstruierte raumfüllende Kurve, die schon im Jahre 1891 von David Hilbert entdeckt wurde. Sie lässt sich ebenfalls in die zeitgleich entdeckten Peano-Kurven einordnen. Im Gegensatz zur bereits vorgestellten sequenziellen Raumkurve ist die Hilbert-Kurve stetig, enthält also, abgesehen von den Übergängen an den Rändern des Id-Raumes, keine Sprünge. Außerdem zeichnet sie sich durch einen wesentlich besseren Erhalt der Lokalität, also der Nachbarschaft, der einzelnen Punkte im Raum bzw. auf der Kurve aus, was die beabsichtigte Id-Transformation erleichtert. Punkte, die auf der Kurve nahe beieinander liegen, liegen also auch im Raum nahe beieinander. Für ausführlichere Betrachtungen zu diesem Thema wird auf [MAK02] verwiesen.

Die Implementierung der Hilbert-Kurve ist allerdings um einiges aufwändiger. Einen Ansatz hierzu liefert [SLP09] und [But06].

Abbildung 4.5 und Abbildung 4.6 zeigen die entstehende Kurve beispielhaft.

Im Allgemeinen stellt die Hilbert-Kurve eine wesentlich elegantere Lösung der Adresstransformation als die sequenzielle Raumkurve dar.

4.5.4 Optimale Raumkurven

Neben den generischen Raumkurven, die geordnet und nach mathematischen Vorgaben den Raum, mehr oder weniger, gleichmäßig füllen, sind auch Raumkurven denkbar, die weniger auf das gleichmäßige Füllen des Raums sondern an der Optimierung der Nachbarschaftsbeziehungen ausgerichtet sind. Obwohl für einzelne Netzwerkstrukturen optimale Raumkurven existieren, die entsprechende Nachbarschaftsbeziehungen gewährleisten, dass

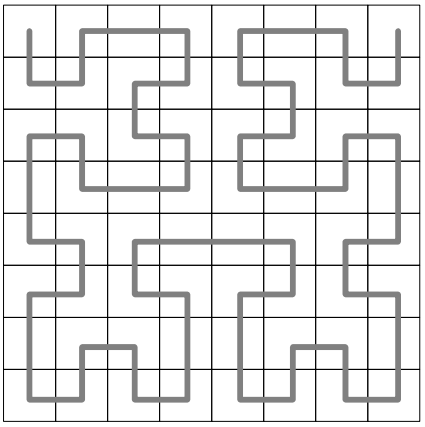

Abbildung 4.5: 2D-Hilbert-Kurve

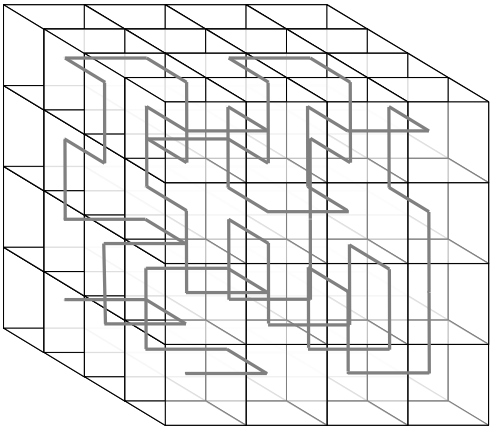

Abbildung 4.6: 3D-Hilbert-Kurve

keine oder nur wenige, zusätzliche Verbindungen aufgebaut werden müssen, sind diese nur für diese eine oder wenige Knotenverteilungen optimal.

Algorithmen zur Verwaltung und Steuerung der Netzwerke müssen mit verschiedenen Verteilungen der teilnehmenden Knoten zurechtkommen. Eine optimierte Raumkurve kann aber nicht für alle Knotenverteilungen optimal sein. Um für jede Transformation eine neue optimale Raumkurve zu nutzen, müsste zumindest die aktuelle Knotenverteilung bekannt sein, was wiederum eine meistens nicht gegebene globale Sicht voraussetzen würde. Solange dies der Fall ist, ist die Nutzung einer optimierten Raumkurve wenig sinnvoll.

4.5.5 Eignung der Raumkurven für die Transformation

Durch die Betrachtung der verschiedenen raumfüllenden Kurven stellt sich die Frage, welche Raumkurven sich besonders für die Übersetzung der verwendeten Identifier und somit für die geplante Transformation der Overlaynetze eignen. Hierzu können vor allem drei Kriterien betrachtet werden.

Die Kurve sollte einfach zu berechnen und zu handhaben sein. Außerdem ist es vorteilhaft für die Transformation, wenn Adressen, die auf der Kurve nahe beieinander liegen, auch im Raum nahe beieinander liegen. Hierdurch wird die Lokalität beziehungsweise die Nachbarschaft der Knoten weniger Veränderungen unterworfen. Von großer Bedeutung ist außerdem die Übertragbarkeit des Konstruktionsprinzips der Raumkurven auf mehrere Dimensionen. Ist dies nicht oder nur unter großem Aufwand der Fall, können Overlaynetze wie CAN, in denen die Relationen zwischen den Dimensionen eine große Rolle spielen, nur schwer unterstützt werden.

Die im weiteren Verlauf dieser Arbeit relevanten Raumkurven wurden bereits genauer beschrieben. Diese speziellen Kurven wurden auf Grund ihrer einfachen Berechenbarkeit im

Fall der sequenziellen Raumkurve, beziehungsweise ihrer guten Eignung für das Transformationsergebnis im Fall der Hilbert-Kurve ausgewählt. Aus den Ergebnissen von [MAK02] kann gefolgert werden, dass sich die Hilbert-Kurve im Hinblick auf Sprunghaftigkeit und Kontinuität am besten für die in dieser Arbeit beabsichtigte Transformation eignet.

Kapitel 5

Konzepte für die Transformation

In diesem Kapitel werden Möglichkeiten behandelt, um verschiedene Overlaystrukturen ineinander zu überführen. Diese werden in die bereits erläuterte Architektur eingefügt. Um einen Überblick über die Möglichkeiten zu geben, werden zunächst einige naive Transformationsmethoden erläutert. In Abbildung 5.2 wird schließlich ein Ansatz vertieft und im Anschluss zu den naiven Transformationsmethoden in Relation gesetzt.

Dabei wird das allgemeine Vorgehen für die Transformation am Beispiel der Hin- und Rücktransformation zwischen den Protokollen Chord und CAN beschrieben. Da die Überlegungen in dieser Arbeit auf einem modularen Aufbau der Transformation basieren sind andere Transformationsprotokolle mit abweichenden Ansätzen denkbar. Dies ist besonders dann der Fall, wenn dabei eine Transformation zwischen anderen Protokollen ermöglicht werden soll.

5.1 Naive Transformationsmethoden

Für die konkrete Durchführung der Transformation sind verschiedene Vorgehensweisen möglich. Hier wird zunächst ein generischer Ansatz beschrieben. Im Anschluss hieran wird ein optimierter Ansatz durch die Nutzung eines Hybridnetzwerkes erläutert. Diese beiden Ansätze werden anschließend in Abschnitt 6 bewertet und diskutiert.

Aus Gründen der Überschaubarkeit und Wiederverwendbarkeit bietet sich eine Aufteilung auf dedizierte und austauschbare Module für unidirektionale Transformationen an. Diese sollten sowohl das Transformationskriterium als auch die eigentlichen Transformationsfunktionalitäten enthalten.

Zunächst werden in diesem Abschnitt einige naive Ansätze für die Transformation erläutert. Hierbei werden ebenso die grundlegenden Annahmen sowie auch die sich dabei ergebenden Probleme weiter vertieft.

5.1.1 Direktes Überführen der Knoten

Der direkteste Lösungsansatz ist es, jeden einzelnen Knoten sequenziell aus dem Ausgangsnetzwerk in das Zielnetzwerk zu überführen. Der Knoten, der die Transformation auslöst, initialisiert hierbei das Zielnetzwerk. Entsprechend der Gestalt des Ausgangs- und Zielnetzwerkes muss hierbei gegebenenfalls die Id mit der in Abschnitt 4.5 schon behandelten Methode transformiert werden. Im weiteren Verlauf der Transformation verlässt jeder weitere Knoten das Ausgangsnetzwerk und tritt dem Zielnetzwerk bei. Die von den Knoten verwalteten Nutzdaten verbleiben zunächst auf den jeweiligen Knoten und werden erst nach Abschluss der Transformation auf ihre neuen Knoten transferiert.

Hierdurch ist es allerdings nicht ohne Weiteres möglich, während der Transformation Anfragen an das Netzwerk zu stellen. Wird dies dennoch versucht müssen insgesamt vier Lookups gestellt werden, jeweils zwei Anfragen an jedes der beiden Teilnetzwerke, von denen jeweils eine die originale und eine die transformierte Id nutzt. Hierbei steigen sowohl Netzwerklast durch die erhöhte Zahl der Anfragen als auch der Berechnungsaufwand auf den teilnehmenden Knoten, da es gegebenenfalls nicht ausreicht die Nutzdaten nur mit Hilfe der Id abzugleichen.

Zusätzlich kann dieser Ansatz erweitert werden, um eine partielle Transformation des Netzwerkes zu ermöglichen. In dieser könnten mehrere verschiedene Netzwerktopologien kooperativ betrieben werden.

Hierzu müsste allerdings global bekannt sein wo die Grenze für die Transformation verläuft, da ansonsten nicht sichergestellt werden kann dass der für die angefragte Id zuständige Knoten das ihm übergebene Adressformat behandeln kann. Wird dies vorausgesetzt, muss die Kommunikation dennoch über Gatewayknoten umgeleitet werden, denen die Adressabbildung möglich ist. Dies würde jedoch zu einer starken Mehrbelastung oder sogar Überlastung einzelner Netzwerkknoten führen. Diese beispielhafte Situation während der Transformation wird in Abbildung 5.1 gezeigt.

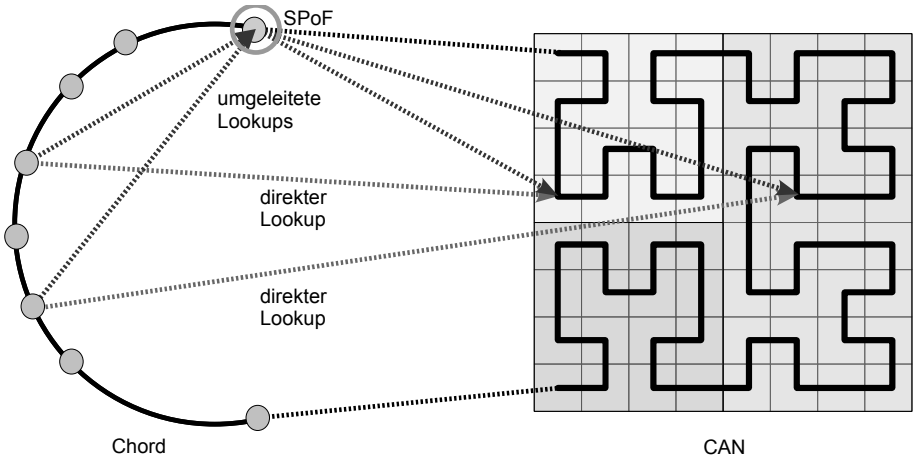

Abbildung 5.1: Koexistenz von Chord- und CAN-Netzwerken

Um diese Situation zu entschärfen, muss eine Hierarchie von Vertretern oder Gatewayknoten aufgebaut werden. Diese übernehmen beziehungsweise steuern die Kommunikation und Adressumwandlung zwischen den einzelnen Netzwerkteilen. So lässt sich die Last auf die individuellen Knoten reduzieren. In jedem Fall sind aber die an dieser Hierarchie beteiligten Knoten einer Mehrbelastung unterworfen, was die Fairness stark beeinträchtigt.

5.1.2 Teile und Herrsche

Ausgehend von dem Ansatz, das Problem in mehrere kleinere, leichter zu lösende Probleme aufzuteilen und so leichter beherrschbar zu machen, kann das Netz getreu dem Motto „Teile und Herrsche“ transformiert werden. Folgende vier Schritte sind dafür nötig.

Im ersten Schritt wird das Netzwerk in X kleinere Bereiche aufgeteilt. Dies kann durch begrenztes Fluten oder Coloring-Algorithmen erreicht werden. Alternativ kann auch der Initialknoten die Teilung explizit vornehmen und diese mit Hilfe einer Transformationsnachricht verbreiten.

Im Anschluss hieran muss in jedem Bereich die absolute Sicht hergestellt werden. Hierzu würden sich unter anderem Gossipingverfahren eignen, mit denen die benötigten Informationen in $O(\log n)$ Schritten verbreitet werden können.

Im nächsten Schritt kann aus jedem der X Bereiche eine kleine Instanz des Zielnetzwerkes gebildet werden. Der Aufbau dieser Instanz kann sehr gut optimiert werden, da jeder Knoten über die absolute Sicht in seinem Bereich verfügt. Im Falle eines CAN-Zielnetzwerkes kann jeder Knoten seine position im CAN-Partitionierungsbaum errechnen. Durch die Positionen in diesem kann jeder Knoten seine Ausdehnung und Posion bestimmen und seine individuelle Routingtabelle auf dieser Grundlage füllen. In Chord-Netzwerken reicht hierbei eine bloße Sortierung der Knoten aus um die hinreichende Invariante zu gewährleisten.

Im letzten Schritt werden die in den vorherigen Schritten erzeugten Bereiche wieder miteinander verschmolzen. Das Vorgehen muss hierzu an den Typ des jeweiligen Zielnetzwerkes angepasst werden. Das Verschmelzen der Bereiche kann optimiert werden, indem jeder Bereich zunächst ein Netzwerk auf einer Teilmenge des gesamten Id-Raums aufbaut. Dem entsprechend können optimierte Verfahren entwickelt werden um die Netzwerke effizienter zu verschmelzen. Auf diesem Gebiet wurden bereits verschiedene Anstrengungen zu verschiedenen Netzwerken unternommen, die als Ansatzpunkte hierfür dienen können. [Dat07, DA06, MJB05, SGH07] Abbildung 5.2 zeigt die drei Zustände der optimierten Transformation von einem CAN-Netzwerk in ein Chord-Netzwerk unter der Nutzung von fünf kleinen Chord-Netzwerken.

Die folgenden Probleme mit diesem Lösungsansatz haben zu dessen Ausschluss geführt: Ein Problem an diesem Ansatz ist die Größe und Anzahl der Bereiche optimal festzulegen. Bei der Nutzung zu vieler Bereiche steigt der Aufwand zur Verschmelzung der einzelnen Bereiche stark an. Werden zu wenige Bereiche genutzt, steigt der Aufwand zur Herstellung der absoluten Sichten in den Bereichen ebenso stark an. Ein weiteres hiermit zusammenhängendes Problem ist die Wahl der Grenzen zwischen den Bereichen. Für den Fall, dass diese

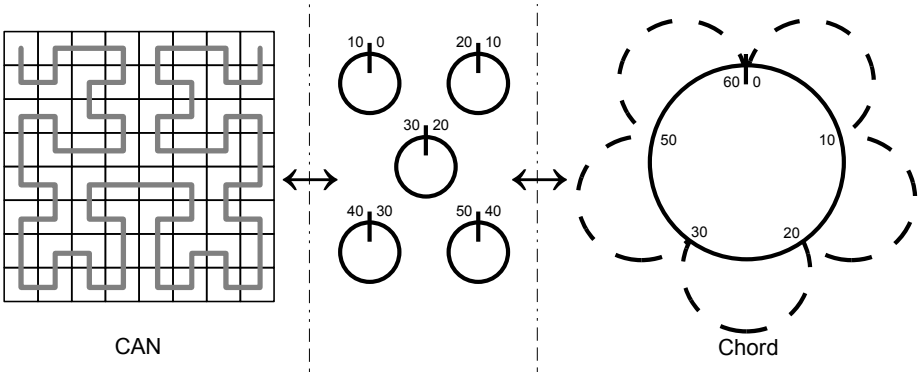

Abbildung 5.2: Transformation durch Teilen und Zusammenführen

statisch vorgegeben sind, ist die ganze Transformation sehr unflexibel. Werden die Bereiche dynamisch gehandhabt steigt der Koordinations- und damit Kommunikationsaufwand. Das dritte Hauptproblem umfasst die Problemstellung des Verschmelzens verschiedener Bereiche in bestimmten Netzwerken. Im konkreten Fall betrifft dies das Verschmelzen von mehreren CAN-Netzwerken. Entsprechend treten Probleme auf, wenn CAN-Knoten für mehrere Bereiche verantwortlich sind oder sich ein Bereich nicht auf einen eindimensionalen Bereich auf dem Chord-Ring reduzieren lässt.

Weiterhin kann es bei dieser Lösung zu Problemen führen, die globale Sicht in einem Bereich herzustellen, wenn Knoten das Netzwerk verlassen oder betreten, sofern die Transformation noch nicht abgeschlossen ist.

5.1.3 Nebenläufige Neukonstruktion

Bei dieser Transformationsmethode wird das Zielnetzwerk nebenläufig zum Ausgangsnetzwerk aufgebaut und betrieben. Während dieser Zeit werden alle Anfragen weiter an das Ausgangsnetzwerk gestellt. Das neue Netzwerk übernimmt die Funktion erst, wenn alle Informationen komplett transferiert wurden, d. h. wenn alle Schlüssel/Wert-Paare im Ausgangs- und im Zielnetzwerk vorhanden sind. Obwohl alle Änderungen an Inhalten sowie beitretende und verlassende Knoten in beiden Netzwerken behandelt werden, müssen alle Anfragen vom Ausgangsnetzwerk bearbeitet werden.

Im Einzelnen folgt dieser Vorgang den folgenden Schritten: Sobald ein Knoten die Transformation beginnt, initialisiert dieser eine Instanz des Zielnetzwerkes. In diesem Netzwerk ist der Initialknoten für den gesamten Identifierraum zuständig. Daraufhin wird das Netzwerk mit Transformationsnachrichten geflutet.

Zur Ausbreitung der Transformationswelle wird der Echo-Algorithmus [Teloo] verwendet. Dieser erreicht mit einer Hinwelle alle Knoten und initialisiert hierbei die Transformation auf allen Knoten. Jeder Knoten den die Transformationsnachricht erreicht tritt dem Zielnetzwerk bei. Als Kontaktknoten wird hierfür der Vorgänger aus der Transformationswelle verwendet, außerdem ist dieser durch die Nutzung des Echo Algorithmus den einzelnen

Knoten schon bekannt. So wird verhindert, dass einzelne Knoten dadurch überlastet werden. Zusätzlich wird eine Rückwelle ausgelöst die genau dann über die Knoten zurück läuft, wenn der Beitrittsvorgang im Zielnetzwerk abgeschlossen ist. Als Ergebnis hiervon kommt die Rückwelle beim Initialknoten an, wenn alle Knoten dem Zielnetzwerk beigetreten sind. Zu diesem Zeitpunkt existieren Ausgang- und Zielnetzwerk simultan auf allen Knoten in ihrer finalen Topologie. Jedoch enthält das Zielnetzwerk bisher keine Nutzdaten, also Schlüssel/Wert-Paare. Um diese zu übertragen, wird eine weitere Transformationsnachricht über den, mit dem Echo-Algorithmus implementierten, Weiterleitungsbaum gesendet. Jeder Knoten, den diese Transformationsnachricht erreicht, fügt alle seine Schlüssel/Wert-Paare dem Zielnetzwerk hinzu.

Bei diesem Vorgang werden für alle Interaktionen mit dem Zielnetzwerk und mit der in Abschnitt 4.5 behandelten Adresstransformation neu erzeugte Ids verwendet. Die Zuständigkeit für einen Großteil der Inhalte wird so auf andere Knoten verschoben. Folglich müssen auch die entsprechenden Schlüssel/Wert-Paare nicht nur lokal sondern über das Netzwerk auf andere Knoten übertragen werden. Der Abschluss der Nutzdatenübertragung löst wiederum eine Rückwelle aus, mit deren Eintreffen das Ausgangsnetzwerk verlassen werden kann.

Abschnitt 5.3 zeigt die beiden nebenläufig betriebenen Netzwerke und die Protokollinstanzen, die auf den gleichen Knoten ausgeführt werden.

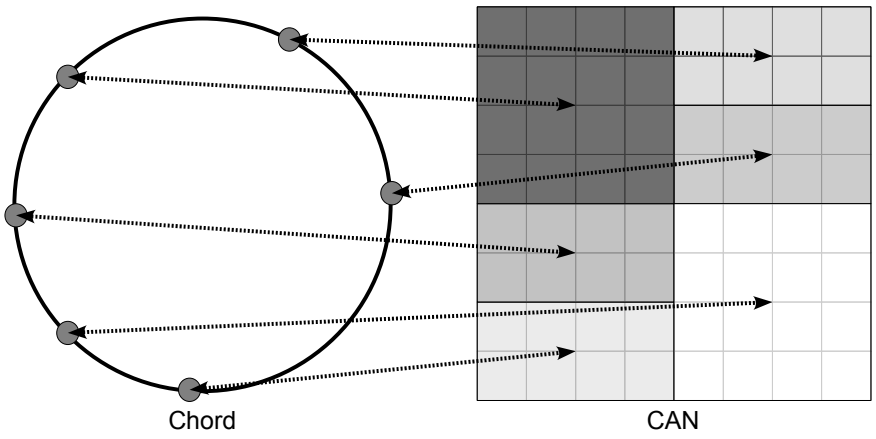

Abbildung 5.3: Nebenläufig betriebene Chord- und CAN-Netzwerke

Obwohl dieser Ansatz bereits viele der gestellten Anforderungen erfüllt, kann er negative Auswirkungen auf die Leistung überlagerter Dienste haben. Wird beispielsweise ein Publish/Subscribe-System betrieben das auf Verteilbäumen basiert, müssen die Verteilbäume vollkommen neu aufgebaut werden.

Aufwandsabschätzung Chord $\leftrightarrow$ CAN

Da dieser Ansatz im Folgenden als Referenz dienen soll, wird der hierfür benötigte Aufwand in Hinsicht auf die Nachrichtenkomplexität nochmals genauer abgeschätzt. Der Aufwand für diese Transformationsmethode setzt sich aus den folgenden Bestandteilen zusammen: Ausschlaggebend ist hierbei dass bis zum Abschluss des Informationstransfers beide Netzwerke simultan betrieben werden müssen. Zusätzlich müssen für die Dauer der Transformation beide Netzwerke aufrecht erhalten werden, hierdurch fällt der entsprechende Kommunikationsaufwand für beide Netzwerke an.

Es sei n die Anzahl der Knoten und e die Anzahl der Kanten beziehungsweise Nachbarschaftsbeziehungen.

Um die Transformation einzuleiten wird die Hinwelle des Echo-Algorithmus genutzt. Zur Durchführung des Algorithmus senden alle Knoten eine Nachricht über ihre inzidenten Kanten. Dies Entspricht $2e$ Nachrichten für die Hinwelle. Ausgenommen ist hiervon jeweils die Aktivierungskante. Dadurch werden $-n$ Nachrichten weniger versendet. Wiederum die Ausnahme hiervon bildet der initiale Knoten, der Nachrichten über alle seiner inzidenten Kanten sendet, also $+1$. Insgesamt entspricht dies $2e - n + 1$ Nachrichten für die Anwendung des Echo-Algorithmus, dies entspricht einer Hin- und einer Rückwelle. Hiervon werden im Verlauf der Transformation zwei Wellen, also $4e - 2n + 2$ Nachrichten benötigt.

Für den Beitritt in das neue Netzwerk werden im Mittel $\sum_{i=1}^{n-1} 2 + \log i$ Nachrichten benötigt, um den geeigneten Knoten für den Beitritt im neuen Netzwerk zu finden. Dies entspricht einer Nachricht für den Beginn des Beitritts, einer für die Übertragung der Beitrittsdaten und der mittleren Routingdistanz über die bereits beigetretenen Knoten.

Anschließend werden abhängig vom Protokoll des Zielnetzwerkes weitere Nachrichten für den Aufbau und die Validierung der Routingtabelle des Zielnetzwerkes benötigt. Für Chord als Zielprotokoll sind dies im Mittel $\log^2 n$ Nachrichten pro Knoten für den direkten Lookup der Nachbarn. Für CAN als Zielprotokoll sind dies mindestens $d/2$ Nachrichten pro Knoten für das Informieren der Nachbarn des neu beigetretenen Knotens. Hieraus ergeben sich $n * \log^2 n$ beziehungsweise $n * d/2$ Nachrichten für die gesamte Transformation.

Um die Transformation zu komplettieren müssen noch die Schlüssel/Wert-Paare auf die im neuen Netzwerk zuständigen Knoten übertragen werden. Hierzu muss zunächst der Partnerknoten durch einen zusätzlichen Lookup ermittelt werden. Hierzu werden $n * \log n$ Nachrichten benötigt. Und noch einmal n Nachrichten für die Übertragung selbst.

Beim Vergleich der Hin- und Rücktransformation unterscheidet sich lediglich der Aufwand für den Aufbau und die Validierung der Routingtabelle. Im Fall von CAN findet der Aufbau der Routingtabelle auf dem beitretenden Knoten in einem Schritt statt. Zusätzlich fallen Nachrichten für das Informieren der Hälfte der Nachbarknoten an. Die anderen Aufwände unterscheiden sich nicht signifikant.

Weitere Betrachtungen zur Komplexität finden sich in Abschnitt 5.3.1.

5.2 Transformation mit Hilfe eines Hybridnetzwerkes

Dieser optimierte Ansatz beruht auf dem Grundsatz möglichst viele Informationen aus dem Ausgangsnetzwerk wiederzuverwenden und auch auf dem bereits verworfenen Ansatz der direkten Überführung der Knoten aus Abschnitt 5.1.1. Besonders die bereits bestehenden Routingtabellen sollen wiederverwendet werden, da hiervon viele Einträge auch im Zielnetzwerk benötigt werden.

5.2.1 Ansatz

Um den lokalen Zustand möglichst klein zu gestalten werden die Protokolle nicht nebenläufig betrieben, sondern werden möglichst nahtlos ineinander überführt. Damit hierbei keine Informationen verloren gehen werden diese in das jeweils folgende Protokoll übernommen.

Zunächst werden für jeden teilnehmenden Knoten drei Zustände eingeführt, in denen jeweils nur ein Protokoll zurzeit auf dem jeweiligen Knoten betrieben wird. Diese beschreiben den Betrieb des Ausgangsprotokolls vor der Transformation (*pre*), den Betrieb des Transformationsprotokolls während der Transformation (*trans*) und den Betrieb des Zielprotokolls nach der Transformation (*post*).

Auch unter Nutzung dieses Ansatzes ist eine Id-Transformation mit Hilfe der Hilbert-Kurve vorgesehen, jedoch ist diese nur nötig, wenn sich der Kommunikationspartner in einem anderen Transformationszustand befindet. Hierdurch und durch die Nutzung der in Abschnitt 4.4 vorgestellten Architektur, wird der unabhängige Betrieb der Protokolle in den Zuständen *pre* und *post* ermöglicht. Der Unterschied zu den anderen vorgestellten Verfahren liegt darin, dass die Id-Transformation erst mit dem Wechsel des genutzten Protokolls stattfindet.

Innerhalb des hybriden Transformationsprotokolls müssen die Zuständigkeit und der Id-Verlauf definiert werden. Hierzu werden einige Vorgaben aus den Chord- und CAN-Protokollen verwendet. Die Zuständigkeit für bestimmte Id-Bereiche wird von Chord abgeleitet, jedoch werden hieraus einzelne Bereiche gebildet, wie sie in CAN üblich sind. Als Konsequenz hieraus entstehen auf jedem Knoten mehrere Bereiche die sich dadurch auszeichnen, dass sie von einem Teil der verwendeten Raumkurve stetig durchlaufen werden. Abbildung 5.4 zeigt einige Knoten auf einem, entsprechend der Hilbert-Kurve verlaufenden Chord-Ring und deren Zuständigkeit für die einzelnen Ids. Das Routing basiert entsprechend der Bereichsdefinition ebenfalls auf dem CAN-Protokoll und wird in Abschnitt 5.2.4 weiter erläutert. Abbildung 5.5 zeigt die potentiellen Verbindungen eines Bereichs im hybriden Protokoll.

5.2.2 Ablauf der Tansformation

Um die optimierte Transformation durchzuführen, müssen folgende Aktionen auf jedem Knoten ausgeführt werden. Zuerst muss die Transformation initialisiert werden. Hierzu geht

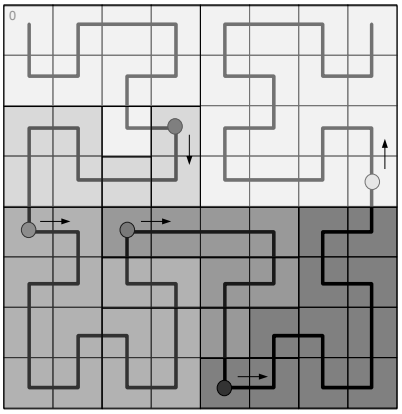

Abbildung 5.4: Zuständigkeit im Hybridnetzwerk

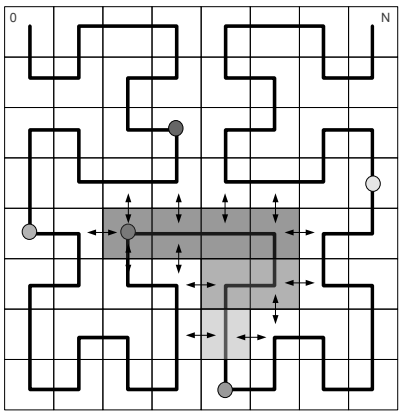

Abbildung 5.5: Verbindungen im Hybridnetzwerk

der Initialknoten in den *trans* Zustand über und sendet eine Transformationsnachricht an seine Nachbarn.

Die Transformationsnachricht enthält neben dem Vorgänger im Transformationsbaum auch das Zielprotokoll sowie das verwendete Transformationsprotokoll. Das Transformationsprotokoll wiederum ist abhängig vom Zielprotokoll und sammelt alle, für den Betrieb des Zielprotokolls benötigten, Routingtabelleneinträge. Der Übergang von *pre* zu *trans* findet, wie auch bei der nebenläufigen Transformation, durch den Echo-Algorithmus[Teloo] statt. Konkret breitet sich die Transformationswelle hierbei epidemisch aus. Zum Zustandsübergang zählt ebenfalls die Initialisierung des Transformationsprotokolls auf dem jeweiligen Knoten. Hierzu müssen sowohl die Id als auch der Zuständigkeitsbereich ermittelt werden. Der Aufwand hierfür ist vor allem von der für die Adresstransformation verwendeten Raumkurve abhängig. Um den Zuständigkeitsbereich zu beschreiben werden mehrere virtuelle Protokollknoten (*VProtocol*) innerhalb des Protokolls verwaltet. Diese beschreiben rechteckige Bereiche und überspannen einen kontinuierlichen Bereich auf der Raumkurve. In Abbildung 5.6 sind ein fehlerhafter und ein korrekt aufgebauter Bereich dargestellt. Zusätzlich werden alle Routinginformationen, über die der Anfangsprotokollknoten verfügt, in die jeweiligen *VProtocol*e übernommen.

Im Fall der Transformation vom CAN- zum Chord-Protokoll ist zusätzlich das Vorhalten einer Chord-Routingtabelle vorgesehen. Dies ist nur dann nötig wenn eine möglichst reibungslose Transformation ins Chord-Netzwerk ermöglicht werden soll. Dies wird später weiter erläutert.

Nach der Initialisierung des Transformationsprotokolls wird mit der Suche der im Zielprotokoll benötigten Nachbarn begonnen. Die verschiedenen Vorgehensweisen hierzu werden in Abschnitt 5.2.3 erläutert. Sobald alle benötigten Nachbarn gefunden sind kann die Rückwelle des Echo-Algorithmus gestartet beziehungsweise fortgesetzt werden. Die Blätter des

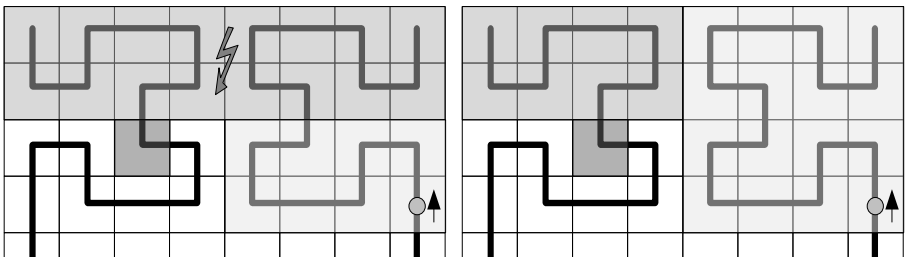

Abbildung 5.6: Fehlerhaft (links) und korrekt (rechts) konstruierte Bereiche im Transformationsprotokoll

Transformationsbaums senden hierbei in regelmäßigen Abständen den Fortschritt ihrer Transformation und Optimierung über den Transformationsbaum zum Initialknoten.

In diesem Fall wird mit der Optimierung des lokalen Zustands fortgefahren. Der lokale Zustand kann bei Verwendung von regelmäßigen Transformationsnachrichten schon optimiert werden bevor alle Nachbarn bekannt sind. Falls es hierbei zur Migration von *VProtocol*en kommt muss der neue Knoten die entsprechenden Nachbarn suchen. Die Optimierung des lokalen Zustands wird in Abschnitt 5.2.5 eingehender beschrieben.

Der Fortschritt und die Terminierung der Transformation wird durch die regelmäßigen Statusnachrichten vom Initialknoten erkannt. Hierbei ist kein spezielles Verfahren zur Wahrung der Kausalität der Beobachtung (Doppelzählverfahren) nötig, da das Terminierungsattribut als stabil angenommen wird. Auch der Grad der Optimierung wird auf Grund des angewandten Verfahrens als stabil angenommen. Der Grad der Optimierung, der hierbei durch das Aggregat der Rankingwerte definiert ist, kann also nicht sinken. Der Initiator kann die Transformation beenden, sobald alle Knoten über die hierfür erforderlichen Nachbarschaftsinformationen verfügen und lediglich noch Optimierungen vorgenommen werden.

Zu diesem Zeitpunkt oder nach Überschreiten einer Optimierungs- oder Zeitschwelle kann der Initialknoten das Ende des Vorgangs einleiten, indem er eine weitere Transformationsnachricht mit Hilfe des Echo-Algorithmus aussendet. Durch die entsprechende Rückwelle kann die Terminierung des gesamten Vorgangs festgestellt werden.

Dieser Vorgang ist am Beispiel einer Transformation von Chord nach CAN in Abbildung 5.7 dargestellt. Hierbei ist zu beachten, dass Knoten im CAN für mehrere Bereiche verantwortlich sein können. Die Transformation findet von Chord (links) über verschiedene Transformationsstufen bis hin zum Endprodukt der Transformation (rechts) zu CAN statt. Dies wird durch den Schritt der Optimierung, der in Abschnitt 5.2.5 beschrieben wird, verbessert.

Transformationsalgorithmus

Abbildung 5.1 und Abbildung 5.2 zeigen die allgemeine algorithmische Vorgehensweise um die Transformation aus Sicht eines einzelnen Knotens durchzuführen. Aus Gründen der Verständlichkeit wurden hierbei einige Vereinfachungen vorgenommen.

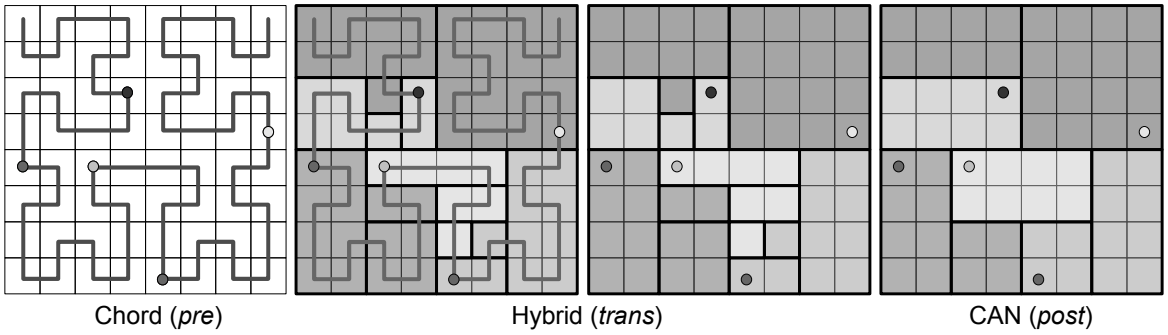

Abbildung 5.7: Transformation mit Hilfe eines Hybridnetzwerkes

Die dem Knoten neu zugestellten Nachrichten laufen hierbei in der Methode *processEvent* auf. Hier wird eine erste Unterscheidung der vorzunehmenden Aufgaben getroffen. Während in *processTransformation* alle die Transformation betreffenden Aktionen durchgeführt werden, finden die sonstigen, üblicherweise bei Eintreffen einer Nachricht, ausgeführten Aktionen in *processEventLocal* statt. Im Weiteren werden nur die Transformationsspezifischen Aktionen weiter behandelt.

```
processEvent( message )
  // do transformation specific event processing
  processTransformation( message )
  // do node specific event processing
  processEventLocal( message )

processTransformation( message )
  // if message is a transformation message
  if ( message IS TransformationMessage ) {
    // prevent cycles - only start same transformation once
    if ( message.transformationStatus IS NOT HybridTransport.transformationStatus ) {
      if ( message.TransformationStatus IS trans ) {
        // initilize transformation
        startTransformation( message )
      } else if ( message.TransformationStatus IS post ) {
        // stop transformation
        finishTransformation( message )
      }
    }
  }
  // in every case check if any useable contained information and optimize local state
  filterNeededIds( message )
  optimizeLocalState( )
```

Listing 5.1: Algorithmus für Transformation mit Hilfe eines Hybridnetzwerkes

Für die Transformation ist generell jede empfangene Nachricht von Interesse, da jede Nachricht Informationen über bisher unbekannte Knoten enthalten kann. Nach dem Empfang von Nachrichten wird außerdem der lokale Zustand optimiert. Dies wird am Ende der

processTransformation Methode für jede Nachricht geprüft. Abgesehen von dieser Ausnahme sind für die Transformation im Weiteren nur die Transformationsnachrichten von Interesse. Wird eine Transformationsnachricht mit einem vom lokalen Zustand abweichenden Transformationsstatus empfangen, wird die Transformation lokal gestartet beziehungsweise beendet.

Im letzten Schritt, also nach dem Starten oder Stoppen der Transformation, werden neue Transformationsnachrichten erstellt und an alle Nachbarn, ausgenommen des Senders der auslösenden Nachricht, weitergeleitet. Hierdurch wird der Transformationsbaum, entsprechend des Echo-Algorithmus aufgebaut.

```
startTransformation( message )
  // set target protocol id
  id = sfc.translate( preProtocol.id )
  // copy neighbor information
  getNeighborInformation( preProtocol )

  // start transformation on transport layer
  transport.startTransformation( message )

  // forward transformation message
  forwardMessage( message )

  // prepare id search
  setUpNeededIds( )
  filterNeededIds( neighbors )

  // start timers - only once every CHALLANGE_TIME do
  startActiveSearchTimer( 0, CHALLANGE_TIME )
  startMutualOptimisationTimer( 0, CHALLANGE_TIME )

finishTransformation( message )
  // stop timers
  stopActiveSearchTimer( )
  stopMutualOptimisationTimer( )

  // set target protocol id
  toProtocol.id = id
  // copy neighbor information
  setNeighborInformation( pastProtocol )

  // forward transformation message
  forwardMessage( message )

  // stop transformation on transport layer
  transport.finishTransformation( message )

forwardMessage( message )
  // forward to all neighbors except transformation parent and save parent
  transformationParent = message.sender
  foreach ( Node n IN neighbors ) {
    if ( n IS NOT transformationParent ) {
      TransformationMessage msg = new TransformationMessage( node, n.protocol.id )
```

```
    msg.tStatus = message.tStatus
    transport.send( n, msg )
  }
}
```

Listing 5.2: Algorithmen zum Start und Stop der Transformation

Zum Starten und Ausbreiten der Transformation wird die Methode *startTransformation* genutzt. Entsprechend des vorgestellten Vorgehens wird nun die Id des *HybridProtokoll*s gesetzt. Hierbei muss die bereits behandelte Id-Transformation angewendet werden. Außerdem werden die Routinginformationen sowie die verwalteten Inhalte aus dem Ausgangsprotokoll in das Transformationsprotokoll übertragen. Im Anschluss hieran kann das *HybridProtokoll* die weitere Kommunikation übernehmen. Hierzu wird die Transformation im *HybridTransport* (*this.ht*) gestartet, woraufhin alle zukünftig eintreffenden Pakete an das *HybridProtokoll* weitergeleitet werden. Zuvor wurden nur die Transformationsnachrichten an das *HybridProtokoll* weitergeleitet.

Zusätzlich wird die aktive Suche nach bisher nicht bekannten, aber benötigten Knoten initialisiert. Die Suche wird hierbei durch einen Timer gesteuert, der erst am Ende der Transformation gestoppt wird und bis zu diesem Zeitpunkt alle *CHALLANGE_TIME* neu aufgerufen wird. Der erste Aufruf erfolgt allerdings sofort. Entsprechend wird die Nachbarsuche und Optimierung zum Beenden der Transformation gestoppt. Mit welchen Mechanismen die Nachbarn gefunden werden können wird in Abschnitt 5.2.3 beschrieben. Ähnlich wie die Suche nach Nachbarn wird hier auch die gegenseitige Optimierung des Zustands der Knoten gestartet. Die zugehörigen Mechanismen werden in Abschnitt 5.2.5 beschrieben.

Analog hierzu ist der Ablauf in der Methode *finishTransformation*. In dieser wird das Zielprotokoll vorbereitet, in dem die Id und notwendigen Routingeinträge und damit die Nachbarschaftsbeziehungen transferiert werden. Zuvor wird jedoch die Nachbarsuche sowie die gegenseitige Optimierung ausgehend von dem aktuellen Knoten gestoppt.

5.2.3 Nachbarsuche

Eines der zentralen Probleme dieses Ansatzes ist die Ermittlung der Nachbarn im Zielnetzwerk. Bei anderen Ansätzen wird dieses Problem umgangen, indem der jeweilige Beitrittsmechanismus des Netzwerkes verwendet wird. In jedem Fall muss zunächst festgestellt werden, welche Ids von den Nachbarn verwaltet werden.

Dabei sind verschiedene Vorgehensweisen denkbar. Der direkteste Ansatz hierzu ist der Lookup der in Frage kommenden Ids. Hierzu muss eine überschaubare Zahl von potentiellen Nachbar-Ids vorliegen. Ist dies nicht der Fall können Gossiping-Algorithmen eingesetzt werden, mit deren Hilfe die Knoten des Transformationsnetzwerks Informationen über ihre Nachbarknoten austauschen können. Hierdurch kann das Problem der Nachbarsuche mit logarithmischem Aufwand gelöst werden. Alternativ zu den beiden erwähnten Möglichkeiten können spezifisch auf das Zielnetzwerk und deren Besonderheiten abgestimmte Verfahren genutzt werden.

Chord Nachbarsuche

Die Suche der Nachbarn in einem Chord-Netzwerk folgt strengen Regeln und kann genau vorausbestimmt werden. Hierdurch kann die Nützlichkeit von direkten Lookups bewertet werden.

Abbildung 5.8 zeigt die Nachbarn eines Chord-Knotens. Diese können ausgehend von der Id X des jeweiligen Knotens berechnet werden. Hierbei sind die Nachbarn in der Finger-Liste als die Knoten definiert, die für die Ids $x + 2^i$ mit $i = 0..\log N$ verantwortlich sind. Zusätzlich werden die $S = SuccessorListSize$ nächsten Knoten auf dem Ring benötigt.

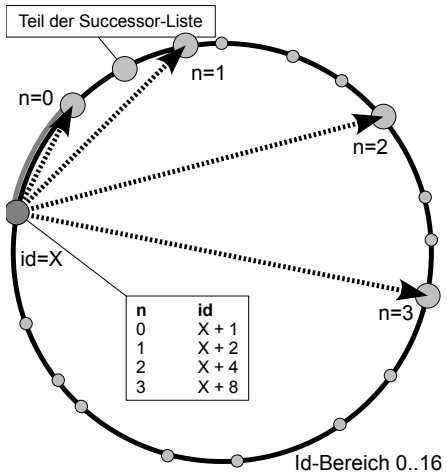

Abbildung 5.8: Nachbarn eines Chord-Knotens

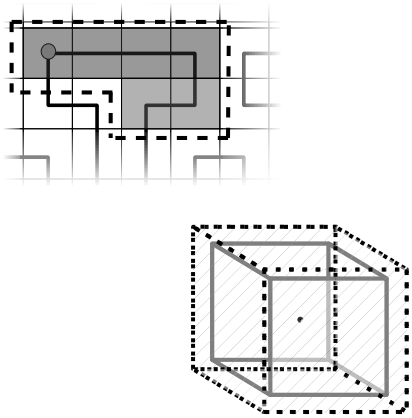

Abbildung 5.9: Nachbarn eines CAN-Bereichs

Hiermit ergeben sich $\log N + S$ Ids, für die Lookups durchgeführt werden müssen. Dies lässt sich ein wenig durch die Nutzung der Überschneidung zwischen Finger-Liste und Successor-Liste reduzieren. Für Werte aus realen Netzwerken lassen sich folgende Werte annehmen. Für die Größe des Identifierraums $N = 2^{128}$ und $S = 8$ für die Länge der Successor-Liste ergibt sich die Zahl der Lookups zu $\log 2^{128} + 8 = 136$. Für ein Netz mit 1.000 Knoten würden also 136.000 Lookups benötigt werden, damit alle Knoten ihre Routingtabellen füllen können.

Jedoch benötigt das Chord-Protokoll zum Beginn des Betriebs nur die hinreichende Invariante, welche besagt dass der nächste Knoten auf dem Ring bekannt sein muss. Um dennoch eine gewisse Ausfallsicherheit zu gewährleisten sollte zumindest die Successor-Liste gefüllt werden. Hierdurch würden für ein Netz mit 1.000 Knoten 8.000 Lookups benötigt werden. Die weiteren Lookups dienen nurnoch der Optimierung des Routings nach der Transformation und müssen nicht der Berechnungsvorschrift entsprechen. Also kann die Finger-Liste mit den sortierten Einträgen aus der Routingtabelle des Ausgangsprotokolls gefüllt werden. Diese können dann im regulären Betrieb des Chord-Protokolls aktualisiert und berechnet werden.

Sofern mehr Lookups als nötig im Laufe der Transformation durchgeführt werden sollen, um die Qualität des Zielnetzwerkes zu verbessern, müssen sowohl eine Chord-Finger-Tabelle, sowie eine Chord-Successor-Tabelle im Transformationsprotokoll vorgehalten werden. Ist dies nicht der Fall ist eine Chord-Successor-Tabelle ausreichend. Die Successor-Tabelle kann beim Zustandswechsel mit Hilfe der zur Verfügung stehenden Daten befüllt werden.

CAN Nachbarsuche

Die Komplexität des CAN-Identifier-Raums verursacht bei der Nachbarsuche einige Schwierigkeiten. Hierdurch ist es nicht praktikabel eine Liste von benötigten Ids zu erstellen und diese den bereits bekannten Nachbarn zuzuordnen, wie es bei Chord ohne weiteres möglich ist.

Abbildung 5.9 zeigt die potentiellen Nachbarn eines CAN-Bereiches in zwei und drei Dimensionen. Um alle benötigten Ids zu errechnen müsste die vollständige Hülle um alle Id-Bereiche des Knotens berechnet und aufgelistet werden. Genauer berechnet sich die Zahl der Ids aus dem Produkt der Ausdehnung des Id-Bereichs in jeder Dimension plus zwei. Dies entspricht $NumId = 2 * \prod_{d}^{D}(B_d + 2)$ und steigt mit zunehmender Zahl der verwendeten Dimensionen exponentiell an. Dies steht jedoch in einem sehr großen Verhältnis zur tatsächlichen Zahl der Nachbarn, die bei $2 * D$ liegt. Also ist es nicht erfolgsversprechend für jede Kandidaten-Id einen Lookup durchzuführen. Die entstehende Liste kann durch die bereits bekannten Nachbarn zwar stark eingeschränkt werden, jedoch bleibt typischerweise eine große Anzahl der Ids unbelegt.

Eine Option hiermit umzugehen wäre sequenziell nur für wenige der Ids Lookups durchzuführen, die Ergebnisse in die Nachbarliste einzufügen und die benötigten Ids hiernach neu zu berechnen. Beim sequenziellen Aufbau eines CAN-Netzwerkes tritt dies nicht auf, da die Tatsache ausgenutzt wird, dass dem aufgeteilten Knoten alle benötigten Nachbarn bereits bekannt sind. Es müssen also lediglich die Nachbarn des neuen Knotens benachrichtigt werden, damit diese den neuen Knoten in ihre Routingtabelle aufnehmen.

Auch das zuvor bei der Untersuchung des Chord-Protokolls vorgeschlagene Verzichten auf eine komplett gefüllte Routingtabelle und ist unter Nutzung des CAN-Protokolls nicht möglich, da ein Fehlen von Nachbarn nicht ohne weiteres toleriert werden kann.

Eine speziell auf die vorliegende CAN-Topologie abgestimmte Methode, die Nachbarn eines Bereiches zu finden, begründet sich auf dem im CAN-Protokoll vorgesehenen Verfahren um das Netzwerk nach Knotenausfällen wiederherzustellen. Hierfür können die alternativen Routingpfade traversiert werden, die normalerweise genutzt werden, um das Netzwerk von ausgefallenen Knoten zu heilen.

In einem CAN-Netzwerk kann hierfür der Partitionierungsbaum genutzt werden. Dieser steht während der Transformation nicht zur Verfügung, kann jedoch zumindest näherungsweise aus der Größe und Position des Id-Bereiches ermittelt werden. Im Rahmen dieser Arbeit wird dieser nicht verwendet und nicht konstruiert.

Ein anderer Weg alternative Routingpfade zu finden, um sie zu traversieren, besteht darin umgedrehte Lookups durchzuführen. Hierzu werden Nachrichten mit dem eigentlichen Quellknoten als Ziel an einen bekannten, entfernten Knoten zum normalen Routing übergeben. Durch die Aufzeichnung und Auswertung des Routingpfades, auf dem die Nachricht zurück zum Quellknoten geroutet wird, können neue Knoten und somit neue Nachbarkandidaten ermittelt werden.

5.2.4 Routing

Die vorgeschlagene Vorgehensweise für die Transformation baut in jedem Fall auf der Wiederverwendung und Weiternutzung der bisherigen Nachbarn auf. Da diese bereits am Anfang der Transformation zur Verfügung stehen und im Rahmen des Greedy-Routings genutzt werden können, sollte sich eine Routingperformance im Bereich des Ausgansnetzes ergeben. Diese liegt sowohl im Fall von Chord, wie auch im Fall von CAN bei $log_2 n$.

Die Adresstransformation kann sich jedoch negativ auf das Routing auswirken. Denn durch diese werden die Positionen aller Knoten im Id-Raum geändert. Da geeignete Adresstransformationen jedoch die Identifier und somit auch die entsprechenden Knoten einigermaßen gleichmäßig über den Id-Raum verteilen, sollte dies das Routing nicht zu sehr beeinträchtigen.

Durch Greedy-Forwarding der Nachrichten kann hierbei bereits ein einigermaßen stabiles Routing erreicht werden, ohne zusätzliche Maßnahmen zu ergreifen. Hierbei wird jede Nachricht an den Nachbarn weitergeleitet, der dem Ziel der Nachricht geometrisch am nächsten liegt. Im Zustand *trans* muss jedoch zusätzlich ein angepasstes Routing verwendet werden, da durch einen stark unterschiedlichen Transformationszustand der beteiligten Knoten, Zyklen in den Routingpfaden auftreten können. Abbildung 5.10 zeigt eine beispielhafte Situation in der dies der Fall ist. Hierbei ist der Zielknoten Z markiert. Dass sich alle Kommunikationspartner eines Knotens, der sich im Zustand *trans* befindet, ebenfalls im Zustand *trans* befinden lässt sich einfach mit Hilfe der hier vorausgesetzten FIFO-Kommunikationskanälen sicherstellen. Abbildung 5.3 zeigt den angewandten Routingmechanismus.

```
route( message )
  foreach( n IN neighbors )
    if ( distance( n, message.target ) < distance( nextHop, message.target ) )
      nextNode = n
  forward( message, message.target, nextHop )
```

Listing 5.3: Routingalgorithmus

Wenn das Ziel der weiterzuleitenden Nachricht aus Sicht des Startknotens, der nur über Chord-Verbindungen verfügt, vor dem Startknoten auf dem virtuellen Chord Ring liegt, kann dieser nicht direkt erreicht werden. Darum wird die Nachricht zunächst an den dem Ziel am nächsten liegenden Knoten weitergeleitet. Dies ist der in der Abbildung der untere rechte Knoten, da dieser dem Ziel am nächsten liegt. Dieser leitet die Nachricht aber aufgrund seiner

Nachbarschaftsbeziehungen wieder in Richtung des Startknotens. Sobald der Startknoten erreicht ist, ist der Zyklus geschlossen. Von hier aus würde die Nachricht weiter im Kreis geleitet werden, wenn keine weiteren Vorkehrungen getroffen werden.

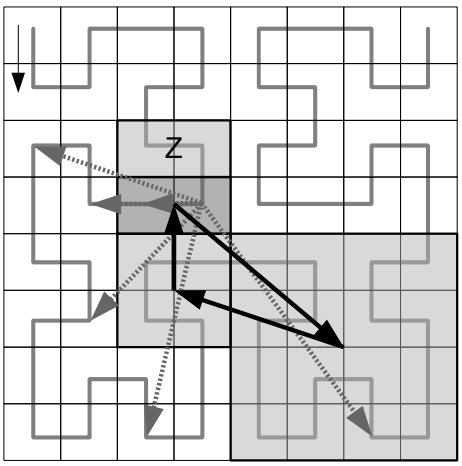

Abbildung 5.10: Zyklus im Routingpfad des Transformationsprotokolls

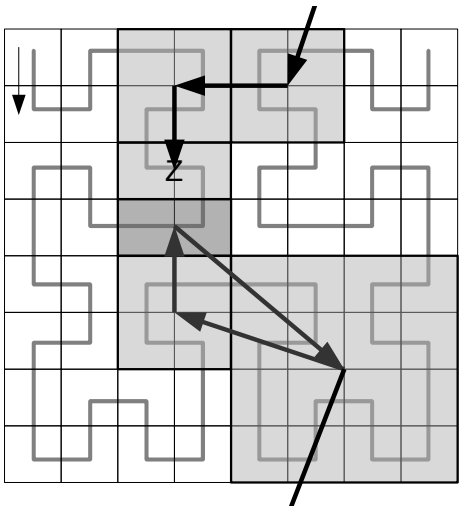

Abbildung 5.11: Verbessertes Routing im Transformationsprotokoll

Verbesserter Routingalgorithmus

Um mit allen auftretenden Situationen umgehen zu können wird der folgende Routingmechanismus vorgeschlagen.

Da erwartet wird, dass die meisten Nachrichten ihr Ziel nach wie vor auf kurzen Pfaden erreichen, wird der genutzte Algorithmus im Allgemeinen beibehalten und nur um ein Ausnahmebehandlung ergänzt. Hierzu werden zwei Aspekte benötigt. Zum einen wird ein Mechanismus zum Erkennen, der gegebenenfalls entstehenden Zyklen, benötigt. Zum anderen muss ein weiterer Mechanismus vorgesehen werden, um nach Erkennung von Zyklen, aus diesen auszubrechen.

Der einfachste Weg einen Zyklus im Routingpfad einer Nachricht zu erkennen, ist es den Pfad der Nachricht aufzuzeichnen. Hierdurch würden jedoch alle, nicht nur die in Zyklen geleiteten Nachrichten gesondert bearbeitet werden müssen. Einfacher ist es, wenn alle Knoten die einmaligen Ids der in letzter Zeit weitergeleiteten Nachrichten speichert. In diesem Fall kann ein Zyklus erkannt werden, in dem ein doppeltes Weiterleiten der gleichen Nachricht detektiert und darauf reagiert wird.

Um das Ausbrechen aus dem Zyklus zu gewährleisten kommen verschiedene Möglichkeiten in Betracht. Hierbei kann zwischen allgemein gültigen und speziell auf die bestehende Topologie angepassten Methoden unterschieden werden. Spezielle Methoden erreichen hier durch

das Nutzen von Besonderheiten der jeweiligen Topologie oft bessere Ergebnisse, müssen jedoch dementsprechend angepasst werden. Darum wird hier ein allgemeingültiger Ansatz bevorzugt, da die tatsächlichen topologischen Gegebenheiten sich während der Transformation im Wandel befinden. Zusätzlich müssen so keine Änderungen am Routingmechanismus vorgenommen werden, um weitere Protokolle in der Transformation zu unterstützen.

Sobald ein Zyklus auf dem Routingpfad erkannt wird, wird die weitere Weiterleitung verhindert. Zunächst wird für alle weiteren Weiterleitungen dieser Nachricht die Aufzeichnung der Route, also der passierten Knoten aktiviert. Ist dies vorgesehen, ist es ausreichend ein entsprechendes Flag in der Nachricht zu setzen, wodurch im Folgenden der Routingpfad aufgezeichnet wird und das angepasste Routing verwendet wird.

Der angepasste Routingmechanismus ist wie der ursprüngliche Routingmechanismus einfach gehalten. Hierzu wird ein Ausschluss der bereits beschrittenen Routingpfade vorgenommen, indem ausgehend von allen dem jeweiligen Knoten bekannten Nachbarn, zunächst alle in dem bisherigen Routingpfad der Nachricht vorkommenden Knoten entfernt werden. Im Anschluss wird mit Hilfe der verbleibenden Nachbarn wieder das bereits erläutere Greedy Routing ausgeführt. Sind hierbei keine Nachbarn mehr übrig, wurde die Nachricht also bereits von allen Nachbarn weitergeleitet, wird diese an den Vorgänger auf dem Routingpfad zurückgegeben. Somit beschreibt dieser Algorithmus eine Tiefensuche auf dem Overlaynetzwerk.

Abbildung 5.4 zeigt die Detektion von Zyklen, sowie den angewandten Routingalgorithmus nach der Erkennung eines Zyklus auf dem Routingpfad.

```
duplicateMessageDetected( message )
  if ( message.uid IN this.routedMessages )
    message.alternativeRouting = true;

alternativeRoute( message )
  notUsedNeighbors = ( neighbors - message.route )
  foreach( n IN notUsedNeighbors )
    if ( distance( n, message.target ) < distance( nextHop, message.target ) )
      nextHop = n
  if ( nextNode IS NOT DEFINED )
    nextNode = message.route.last

  message.route.add( this )
  forward( message, message.target, nextHop )
```

Listing 5.4: Modifizierter Routingalgorithmus

Abbildung 5.11 zeigt die Anwendung des verbesserten Routingalgorithmus in der gleichen Situation. Hierbei wird zuerst, der durch das normale Greedy-Forwarding entstehende Routingpfad beschritten. Sobald der Zyklus erkannt wird, wird dieser durchbrochen und das Routing auf dem alternativen Routingpfad zu Ende geführt.

Durch den hier vorgestellten Algorithmus lassen sich die Routingprobleme relativ einfach für eine weite Bandbreite von Anfangs- und Zielprotokollen lösen. Hierbei steigt die Länge der Routingpfade, im Vergleich zum nicht modifizierten Routingmechanismus, voraussichtlich

leicht an. Jedoch werden hierbei keine der Knoten außerordentlich mehr belastet. Das heißt, die Belastung des Netzes durch die Nachrichtenweiterleitung steigt an, wird aber gleichmäßig verteilt.

5.2.5 Optimierung des Zustandes in der Transformation

Der lokale Zustand des Transformationsprotokolls umfasst alle zum jeweiligen Knoten gehörenden *VProtocol* sowie gegebenenfalls die zusätzlichen Routingtabellen für das Zielnetzwerk. Die zusätzlichen Routingtabellen für das Zielnetzwerk eignen sich nicht für die Optimierung. Somit bleiben zwei Möglichkeiten, den lokalen Zustand in der Transformation zu optimieren, bestehen.

Zum einen kann versucht werden die Zahl der *VProtocol*e zu verringern, indem auf dem Knoten befindliche *VProtocol*e miteinander vereint werden. Zum anderen können *VProtocol*e auf andere oder von anderen Knoten migriert werden, sofern eine Verbesserung des Gesamtzustandes erreicht wird.

Um dies zu erreichen wird zuerst eine Rankingfunktion für die *VProtocol*e eingeführt, die die Bewertung dieser erlaubt. Abbildung 5.5 zeigt die Rankingfunktion.

```
rankingValue( space )
  value = 0
  value += G0 * space.content.length
  value += G1 * space.length
  value += G2 * space.length.bitCount
  value += G3 * space.getProportionValue( space )
  return Round( value )

getProportionValue( space )
  long avg = 0
  double value = 0.0
  foreach ( d IN dimensions )
    avg += space.length( d )
  avg = avg / dimensions
  foreach ( d IN dimensions )
    value += Math.abs(space.length( d ) - avg);
  value = value / avg
  return value
```

Listing 5.5: Rankingfunktion für *VProtocol*e

Die Rankingfunktion beinhaltet die Zahl der verwalteten Inhalte, von dem *VProtocol* gehaltenen Ids, sowie Größenordnung des Id-Bereiches (entspricht dem Logarithmus der Anzahl der gehaltenen Ids). Zusätzlich wird ein Maß für die Geometrie des Id-Bereichs erhoben. Dieses bevorzugt möglichst gleiche Ausmaße in jeder Dimension. Im Endeffekt wird ein möglichst großer und gleichmäßig geformter Id-Bereich den höchsten Rankingwert erhalten.

Die ermittelten Werte werden gewichtet und auf einen Ganzzahlwert gerundet, um die Vergleichbarkeit zu vereinfachen.

Mit Hilfe der Rankingfunktion kann der lokale Zustand mit quadratischem Vergleichsaufwand minimiert werden. Hierzu werden die Knoten im Folgenden paarweise betrachtet. Optimiert werden soll der Zustand auf den Knoten A und B. Jeder Knoten verfügt über *VProtocol*e $A1, A2$ beziehungsweise $B1$. So werden die *VProtocol*e zu Testzwecken paarweise verschmolzen, sofern dies möglich ist. Was nur der Fall ist, wenn die *VProtocol*e direkt aneinander grenzen. Im Folgenden sei es für $A2$ und $B1$ möglich. Wenn nun gilt $R_{A2} + R_{B1} <= R_{A2+B1}$ so bleiben $A2$ und $B1$ verschmolzen und werden auf den Knoten B migriert. Faktisch muss hierfür nur $A2$ migriert werden.

Bei dieser Optimierung wird darauf geachtet, dass jeder Knoten nach der Optimierung über einen möglichst ähnlichen Rankingwert, jedoch mindestens über einen Id-Bereich verfügt.

Zusätzlich kann die Rankingfunktion um den Grad der Wiederverwendung der bekannten Verbindungen erweitert werden. Hiermit würde sich der Rankingwert jedoch abhängig vom Knoten, auf den das *VProtocol* migriert wird, berechnen. Dies erhöht den Berechnungsaufwand für die Optimierung um den Faktor zwei, begünstigt aber auch eine bessere Optimierung.

Neben der Anpassung und Erweiterung der Rankingfunktion kann die Optimierung um ein mehrdimensionales Zusammenfügen der einzelnen Bereiche erweitert werden. Dies betrifft ein Auflösen des paarweisen Verschmelzens beziehungsweise Optimierens zu Gunsten höherwertiger Zusammenführungen. Dies ist vor allem mit steigender Komplexität der Id-Bereiche von Vorteil. Die Komplexität der Id-Bereiche ist wiederum von der Zahl, der für den Id-Raum verwendeten Dimensionen, abhängig.

Besonderheitden der CAN→Chord-Transformation

Bei der Transformation vom CAN- zum Chord-Protokoll fällt die Berechnung des Rankingwertes für die einzelnen *VProtocol*e besonders einfach aus.

Hierbei muss lediglich die Größe der Id-Bereiche und gegebenenfalls die Anzahl der Inhalte beachtet werden. Somit spielen die Gewichtungsparameter $G0$ und $G1$ hierbei die übergeordnete Rolle.

Besonderheitden der Chord→CAN-Transformation

Bei der Transformation von Chord zu CAN finden alle Parameter der Rankingfuntion Anwendung.

5.2.6 Aufwandsabschätzung

Ähnlich der Aufwandsabschätzung für die Transformation durch nebenläufige Neukonstruktion in Abschnitt 5.1.3 wird hier der Aufwand für die geplante Transformation in Hinsicht auf die Nachrichtenkomplexität abgeschätzt.

Auch hier sei n die Anzahl der Knoten und e die Anzahl der Kanten beziehungsweise Nachbarschaftsbeziehungen.

Für die Einleitung der Transformation und den Aufbau des Transformationsbaums sind unter Zuhilfenahme des Echo Algorithmus $2e - n + 1$ Nachrichten nötig. Zusätzlich muss der Aufwand für die Suche der benötigten Nachbarn ermittelt werden. Bei der Transformation in ein Chord-Netzwerk sind dies $s + \log n$ Nachrichten pro Knoten, wobei s für die Länge der Successor Liste steht. Hierbei ist eine Wiederverwendung der bestehenden Nachbarn jedoch noch nicht berücksichtigt. Hierdurch lässt sich die Zahl der tatsächlich benötigten Nachrichten weiter reduzieren. Im Folgenden wird davon ausgegangen, dass hierbei mindestens 2 Nachbarn wiederverwendet werden können. Dies gilt falls die Hilbert-Kurve oder eine andere sprungfreie raumfüllende Kurve, für die Transformation verwendet wird. Soll durch die Transformation ein CAN-Netzwerk erzeugt werden, ist die Abschätzung weitaus schwieriger. Im Folgenden wird davon ausgegangen, dass die Nachbarsuche mit durch $\log n$ Nachrichten pro Knoten abgeschlossen werden kann.

Diese sind abhängig von der Zahl der Nachbarn, die während der Transformation kontaktiert werden können. Je nach verwendeten Protokollen und dem Fortschritt der Transformation bewegt sich die Zahl der Nachbarn zwischen $d/2$ und $\log n + d/2$. Dies ergibt sich aus den zu Beginn der Transformation zur Verfügung stehenden, sowie den im Verlauf dazugewonnen Nachbarn.

Um die Optimierung abzuschließen wird davon ausgegangen, dass alle Nachbarn kontaktiert werden. Dies ist gegebenenfalls mehrfach nötig, jedoch wird davon ausgegangen, dass die Optimierung nach $\log n$ Optimierungsrunden abgeschlossen ist. Da ein Knoten maximal über $\log n + d/2$ Nachbarn verfügt, ergibt sich der Optimierungsaufwand zu $\log n * (\log n + d/2)$.

Für die Optimierung selbst werden pro Optimierungsschritt 2 Nachrichten benötigt um die zur Optimierung benötigten Informationen auszutauschen, sowie k Nachrichten um die Inhalte zu transferieren. Für das Gesamtnetz ergibt sich hieraus ein Aufwand von $n * (2 + k) * \log n$ Nachrichten.

Für die Überwachung der Transformation und somit für die Feststellung der Terminierung fällt ein Mehrfaches der Größe des Transformationsbaumes, also $x * (n - 1)$ Nachrichten an. Hierbei fällt jeweils nur der Aufwand für die Rückwelle des Echo Algorithmus an, da ausgehend von den Blättern des Transformationsbaum regelmäßige Fortschrittsnachrichten gesendet werden. Zum Beenden der Transformation fällt nochmals der Aufwand für die volle Ausführung des Echo Algorithmus an, also $2e - n + 1$. Somit wird sichergestellt, dass jeder Knoten die Transformation beendet hat.

Weitere Betrachtungen zur Komplexität finden sich in Abschnitt 5.3.1.

5.2.7 Auswirkung auf Publish/Subscribe-Systeme

Die Transformation mit Hilfe eines Hybridnetzes wirkt sich während der Transformation relativ wenig auf überlagerte Services aus. Dies ist darin begründet, dass bis zum Abschluss des

Transformationsvorgangs keine, bereits bestehenden Verbindungen verworfen werden. Dies findet erst mit der finalen Umstellung, der gesamten Kommunikation auf das Zielprotokoll, statt. Hierdurch können während der Transformation fast alle bestehenden Verteilbäume oder ähnliche von überlagerten Services aufgebaute Strukturen erhalten bleiben. Allerdings nur, wenn die Adressen innerhalb der, an das P2P-System gestellten, Anfragen entsprechend transformiert wurden. Dies kann für die Services transparent im *HybridTransport* geschehen.

Die Ausnahme von dieser Regel wird erst, durch die Optimierung der Netzwerkstruktur, geschaffen. Hierbei werden erstmals Id-Bereiche zwischen den einzelnen Knoten migriert und somit auch überlagerte Systeme beeinflusst. Durch diese Migrationen betroffene Verteilstrukturen müssten in diesen Fällen neu aufgebaut werden. Bei Wahl einer geeigneten Adressabbildung kann hierbei vermieden werden, dass alle Verteilstrukturen neu aufgebaut werden müssen, wie dies zum Teil bei anderen Ansätzen der Fall ist.

Durch die angewandte Praktik, die Verbindungen während der Transformation nur zu vermehren, steigt gezwungenermaßen das Clustering, also der Grad der Bekanntheit in der Nachbarschaft im Netzwerk an. Jedoch kann dieses zuvor durch die Anwendung der Adresstransformation und die damit einhergehende Änderung der Beschaffenheit des Id-Raumes, vermindert werden.

5.2.8 Dauerhafter Betrieb im Transformationszustand

Neben dem Betrieb des Transformationsprotokolls für die Dauer der Transformation ist auch ein dauerhaftes Verbleiben im Transformationsprotokoll denkbar. Die hierbei erreichte Leistung hängt von der konkreten Ausprägung des Transformationsprotokolls ab. Also vom verwendeten Anfangs- und Zielprotokoll.

Geht man nun noch von einer weiteren Variabilisierung des Transferprotokolls, sowie einigen zusätzlichen Modifikationen aus, lässt sich ein stabiles Netz erschaffen, das von verschiedenen heterogenen Bereichen zusammengesetzt ist. Um dies umzusetzen werden die Folgenden bisher nicht vorgesehenen Bestandteile benötigt.

Wichtigster Bestandteil ist ein Mechanismus, um die verschiedenen Bereiche voneinander abzutrennen. Dafür ist es ausreichend, die jeweiligen Id-Bereiche, in denen ein bestimmtes Protokoll aufrechterhalten wird, absolut bekannt zu machen. Dies kann initial beim Wechsel in den Transformationsmodus geschehen und später inkrementell angepasst werden.

Da das Transformationsprotokoll nicht nur für eine begrenzte Zeitspanne betrieben werden soll, muss die Performance des Systems verbessert werden. Hierzu müssen vor allem geeignete und optimierte Routingverfahren angewandt werden, um die Belastung des Systems zu vermindern.

5.3 Vergleich der vorgestellten Transformationen

In diesem Abschnitt werden die Transformation durch nebenläufige Neukonstruktion aus Abschnitt 5.1.3 und die Transformation mit Hilfe eines Hybridnetzwerkes aus Abschnitt 5.2 miteinander verglichen.

Der relevanteste Unterschied zwischen den beiden Ansätzen, ist die Fähigkeit der Transformation mit Hilfe eines Hybridnetzes, die Lokalität einzelner Knoten zu erhalten. Hierdurch werden Informationen, die im Ausgangsnetz vorhanden sind und im Zielnetz benötigt werden nicht verworfen und neu ermittelt, sondern direkt weiterverwendet.

Ein weiterer Aspekt, ist der Grad der möglichen Variationsmöglichkeiten und des Modifikationspotentials. Während der nebenläufige Neuaufbau über den gesamten Ablauf, kaum Einflussmöglichkeiten bietet, sind diese unter dem Einfluss eines Hybridnetzes zahlreich. Hierbei können verschiedene Aspekte variiert werden. Hierzu zählen beispielsweise das während der Transformation angewandte Routingprotokoll und die konkrete Ausprägung der verwendeten Optimierung der Id-Bereiche. Beide Ansätze lassen sich beeinflussen, in dem die verwendete Adresstransformation variiert wird.

5.3.1 Vergleich der prognostizierten Aufwände

Abbildung 5.2 zeigt nochmals die erwarteten Aufwände für die verschiedenen Transformationen.

Um die Aussage dieser Abschätzung weiter zu konkretisieren, werden diese aggregiert. Hierbei entspricht e dem mittleren Knotengrad, also $e = \log n$ bei der Transformation vom Chord- ins CAN-Netzwerk. Und $e = d/2$ bei der Transformation vom CAN- ins Chord-Netzwerk. Hierbei ergibt sich für jeden Knoten während der nebenläufigen Neukonstruktion ein Aufwand von etwa $\log n + e + d/2 + 3$. Für die Transformation mit Hilfe eines Hybridnetzes werden $\log n + \log^2 n + e + 3$ Nachrichten benötigt. Bei diesen Betrachtungen sind die Aufwände für die Übertragung der gespeicherten Inhalte nicht eingerechnet, da diese je nach Nutzungsweise des Systems variieren.

	Nebenläufige Neukonstruktion	**Transformation durch Hybridnetz**
Chord→CAN	$2\log n + d/2 + 3$	$2\log n + \log^2 n + 3$
CAN→Chord	$\log n + d + 3$	$\log n + \log^2 n + d/2 + 3$

Tabelle 5.1: Gesamtaufwände der verschiedenen Transformationsmethoden (ohne Inhaltstransfer) pro Knoten

Auf Basis der Gesamtaufwände pro Knoten kann die Zahl der benötigten Nachrichten berechnet werden. Die Berechnung der Gesmtaufwände ist in Abbildung 5.1 nochmals zusammengefasst. Für ein Chord-Netz mit 1.000 Knoten ergeben sich je nach gewählter

Transformationsmethode etwa 13.000 beziehungsweise 18.000 Nachrichten bei der Nutzung von 8 Dimensionen. Entsprechend würden für eine Rücktransformation 14.000 beziehungsweise 19.000 Nachrichten anfallen. Analog berechnen sich die Werte für große Netze, hier mit 100.000 Teilnehmern zu 1.700.000, 3.800.000, 1.600.000 und 3.700.000 Nachrichten.

Nach den hier ermittelten Nachrichtenzahlen benötigt der bloße Transformationsvorgang durch nebenläufige Neukonstruktion etwa ein Viertel beziehungsweise in großen Netzen die Hälfte der Nachrichten. Jedoch müssen nach der Konstruktion des Netzes die gesamten Inhalte in das neue Netz migriert werden. Um zu entscheiden, welche Transformationsmethode vorzuziehen ist sind somit weitere Informationen nötig. Generell lässt sich aber festhalten, dass größere Netze besser durch nebenläufige Neukonstuktion errichtet werden können, während starkt genutzte, also mit vielen Daten belegte Netze besser durch ein Hybridnetz transformiert werden können.

5.3.2 Eignung der verschiedenen Ansätze

Aus den bisherigen Betrachtungen lassen sich bestimmte Randbedingungen festlegen, wobei die Stärken und Schwächen der beiden behandelten Verfahren ausschlaggebend für deren Eignung sind. Hierzu wird ein Szenario angenommen, in dem der Bedarf zur Transformation von einem überlagerten Publish/Subscribe-System festgestellt wird.

Ausgehend von diesem Szenario, eignet sich die nebenläufige Neukonstruktion vor allem für Fälle, in denen die Zahl der unterschiedlichen Inhalte, also Themen und somit Verteilstrukturen, relativ klein ist, während die Zahl der Abonnenten jedes Themas relativ hoch ist. Wenn das Publish/Subscribe-System also vor allem auf einigen großen Verteilstrukturen entlang des Overlaynetzes beruht. Dies ist der Fall, da in einer solchen Situation in beiden Systemen, mit einer sehr hohen Wahrscheinlichkeit, die entsprechenden Verteilbäume zumindest teilweise neu aufgebaut werden müssen, da die zuvor im Overlay genutzten Verbindungen im Zielnetz nicht vorhanden sind. Somit wird der Hauptnachteil der nebenläufigen Neukonstruktion ausgeglichen.

Entsprechend eignet sich eine Belastung des Publish/Subscribe-Systems mit Themen, mit niedriger Popularität, also kleinen Verteilbäumen, besonders für die Transformation mit Hilfe eines Hybridnetzes. Hierbei kommt die Stärke des Ansatzes am besten zum Tragen, da der Anteil der nicht zu modifizierenden Verteilbäume hierbei am höchsten ist.

		Nebenläufige Neukonstruktion		Transformation durch Hybridnetz	
		pro Knoten	$\sum$	pro Knoten	$\sum$
Hin- und Rückwelle		$e+1$	$2e-n+1$	$e+1$	$2e-n+1$
Nachbarsuche	Chord→CAN	-	-	$\log n$	$n*\log n$
	CAN→Chord	-	-	$s+\log n-2$	$n*(s+\log n-2)$
Beitritt zum Netz		$2+\log n$	$\sum_{i=1}^{n-1} 2+\log i$	-	-
Aufbau, Validierung der Routingtabelle	Chord→CAN	$d/2$	$n*d/2$	-	-
	CAN→Chord	$\log^2 n$	$n*\log^2 n$	-	-
Übertragen der Schlüssel/Werte Paare bzw. Optimierung		$\log n+k$	$n*(\log n+k)$	$\log n*(\log n+d/2)$	$n*(\log n*(\log n+d/2))$
notwendige Wellen		2		2	

Tabelle 5.2: Aufwände der verschiedenen Transformationsmethoden

Kapitel 6

Bewertung

In diesem Kapitel wird die Leistungsfähigkeit des entwickelten Algorithmus durch die Beschreibung einiger durchgeführter Experimente gezeigt.

6.1 PeerSim

Um die Experimente durchzuführen, mit denen die Leistungsfähigkeit der entwickelten Protokolle bewertet werden kann, wird der Simulator Peersim [JMJV] verwendet. Peersim wurde an der University of Bologna entwickelt und bietet sowohl einen zyklus- wie auch einen eventgetriebenen Betriebsmodus.

Peersim besteht aus einem Verbund verschiedener Netzwerkkomponenten, die per Konfigurationsdateien zu einer kompletten Simulation zusammengefügt werden können. Hierbei stehen für die einzelnen Komponenten Java Interfaces zur Verfügung. Somit können eigene Protokolle relativ einfach implementiert und evaluiert werden.

Wichtige Komponenten werden hierbei durch die folgenden Interfaces realisiert.

Node

Die einzelnen Knoten des simulierten Netzwerkes. Die Knoten beinhalten jeweils den gleichen Protokollstack, der in der Konfiguration festgelgegt wird.

Protocol

Die in den Knoten enthaltenen Protokolle können je nach Simulationstyp *CDProtocol* oder *EDProtocol* sein. *CDProtocol* enthält Anweisungen, die in jedem Zyklus ausgeführt werden sollen. *EDProtocol* enthält Anweisungen für die Reaktion auf Events.

Linkable

Diese Komponente definiert ein Overlaynetz, in dem es Zugriff auf die Topologie gewährt. Hiermit ist es eines der wichtigen Interfaces, die zur Erstellung eines Protokolls implementiert werden müssen.

Control

Um während der Simulation Beobachtungen anzustellen oder Einfluss auf das simulierte Netz zu nehmen, werden die *Control*-Komponenten verwendet. Diese werden, unabhängig vom Netz, zyklisch Ausgeführt, haben aber zugriff auf dieses. So ist es möglich aktuelle Netzparameter und -werte zu erfassen, aggregieren und für die Auswertung abzuspeichern, und so den aktuellen Simulationszustand zu erfassen. Andererseits ist es auch Möglich, in den Betrieb des Netzes einzugreifen und beispielsweise Nachrichten an einzelne Netzkonten zu senden. Diese Möglichkeit wird im Folgenden genutzt, um Netzwerklast zu erzeugen und die Transformation zu steuern.

6.2 Aufbau der Simulation

Der Aufbau der Simulation orientiert sich an der in Abschnitt 4.4 vorgestellten Architektur. Die Simulation ist ereignisgesteuert implementiert. Um dies innerhalb des Peersim-Simulators zu ermöglichen, müssen die Protokolle das Interface *peersim.edsim.EDProtocol* implementieren. Abschnitt 6.1 zeigt den in der Simulation verwendeten Protokollstack. Dieser orientiert sich stark an der in Abschnitt 4.4 vorgeschlagenen Architektur (Abbildung 4.1).

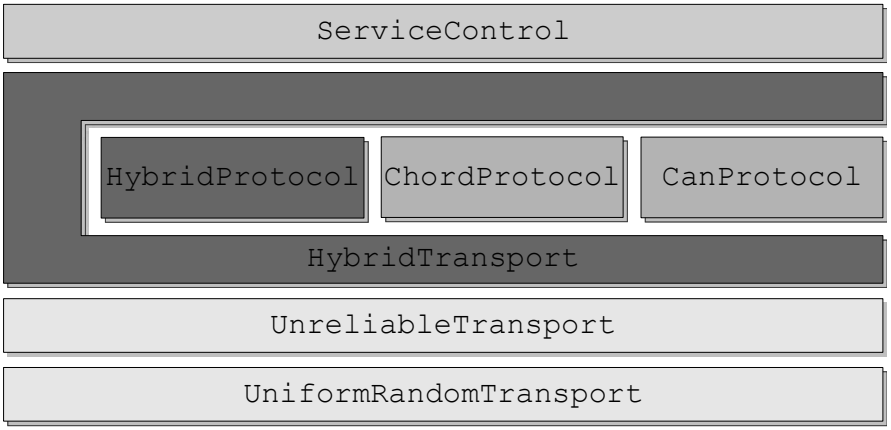

Abbildung 6.1: Protokollstack in der Simulation

Der *HybridTransport* implementiert die Schnittstellen *Protocol* und *Linkable*. Das *HybridProtocol* implementiert die Schnittstelle *Protocol* und ist somit im Großen und Ganzen wie die Ausgangs- und Endprotokolle aufgebaut. Abweichend hiervon ist die Kommunikation mit

dem *HybridTransport* vorgesehen und fester Bestandteil des geplanten Ablaufs. Für nicht-transformierende Protokolle ist der *HybridTransport* transparent und muss nicht weiter beachtet werden.

6.2.1 Bestandteile der Simulation

Die Simulation ist aus den hier im Folgenden näher beschriebenen Bausteinen aufgebaut:

HybridTransport

Der *HybridTransport* dient im Wesentlichen dem Zweck, die eingehenden und ausgehenden Nachrichten der Knoten abhänging vom Transformationsstatus des Knotens auf eines der Protokolle umzuleiten. Hierfür werden die entsprechenden Interfaces implementiert. Diese sind *peersim.transport.Transport* für die Kommunikation mit tieferen Schichten sowie *peersim.core.Linkable* und *peersim.edsim.EDProtocol* für die Kommunikation mit höheren Schichten. Hierzu wird der Transformationsstatus des Knotens gehalten und kann auch von anderen Komponenten, die diesen benötigen, abgefragt werden.

Eine weitere Aufgabe dieses Protokolls ist es, die Zieladressen der Nachrichten, wie in Abschnitt 4.5 beschrieben, mit Hilfe einer raumfüllenden Kurve in das benötigte Format zu konvertieren. Dies ist notwendig, damit das auf dem Knoten aktuell aktive Protokoll eingehende Nachrichten auch ohne Kenntnis der Transformation verarbeiten kann.

HybridProtocol

Das *HybridProtocol* ist der zentrale Baustein, durch den die Transformation ermöglicht wird. Das Protokoll nutzt die Daten aus der Routingtabelle des Ausgangsprotokolls und komplettiert diese, um am Ende der Transformation die Routingtabelle des Zielprotokolls zu füllen.

Zusätzlich muss ein angepasstes Routing verwendet werden, um Zyklen in den Routingpfaden während der Transformation zu vermeiden.

ChordProtocol und *CanProtocol*

Das *ChordProtocol* und das *CanProtocol* sind einfache Implementierungen des Chord- beziehungsweise des CAN-Protokolls. Die beiden Protokolle wurden bereits in Abschnitt 2.2.1 und Abschnitt 2.2.2 vorgestellt und in Abschnitt 4.3 miteinander verglichen. Als Besonderheit bieten beide Protokolle die Möglichkeit, die genutzte Id sowie die Nachbarn auszulesen, beziehungsweise neu zu setzen. Beim *ChordProtocol* betrifft dies sowohl die Knoten in der Successor-Liste sowie die in der Fingertabelle. Im Falle des *CanProtocol*s muss beachtet werden, dass ein Knoten bzw. eine Protokollinstanz für mehrere Id-Bereiche verantwortlich

sein kann, wodurch auch zusätzliche Nachbarschaftsbeziehungen übernommen werden müssen.

Hiermit sind die Anforderungen erfüllt, um das *ChordProtocol* sowohl als Anfangs- als auch als Endprotokoll zu nutzen.

UnreliableTransport

Der *UnreliableTransport* implementiert die Schnittstelle *peersim.transport.Transport* und bietet die Möglichkeit, eine Droprate für die ausgetauschten Nachrichten zu definieren. So kann der in realen Netzwerken eintretende Nachrichtenverlust auf einfache Weise simuliert werden.

Nachrichtenverluste müssen in der Simulation beachtet werden, da diese durch die Verwendung des Internetprotokolls (IP) üblich sind. Hierbei werden im Rahmen der Random Early Detection (RED) zufällig einzelne Pakete verworfen. Der Anteil der zu verwerfenden Pakete ist abhängig von der Auslastung des Knotens, der passiert werden soll. Hierdurch wird bei Nutzung des Transmission Control Protocols (TCP) die Paketrate der einzelnen Verbindungen reguliert.

Das Verhalten des *UnreliableTransport*s stellt zwar keine absolut realistische Verhaltensweise von echten Netzwerken dar, genügt aber, um grundlegende Aussagen über die Störanfälligkeit der untersuchten Verfahren und Protokolle zu treffen. Um eine vollständigere Simulation zu ermöglichen, müsste die Simulation die unterliegende Topologie sowie deren Auslastung beinhalten, was außerhalb des Betrachtungsrahmens dieser Arbeit liegt.

UniformRandomTransport

Der *UniformRandomTransport* bietet die Möglichkeit, zufällige Verzögerungen auf einem festgelegten Intervall für die Weiterleitung der Nachrichten zu simulieren. Hierdurch werden einfache Betrachtungen des Zeitaufwands für die Transformationsvorgänge ermöglicht. Ähnlich dem Verhalten bei der Simulation des Nachrichtenverlustes ist auch diese eine starke Vereinfachung. Auch hier müsste für eine weiterreichende Simulation die unterliegende Topologie in die Simulation aufgenommen werden, was ebenfalls außerhalb des Betrachtungsrahmens dieser Arbeit liegt.

6.2.2 Implementierung der Simulation

Für die Simulation müssen einige allgemeine Werte festgelegt werden. Die hier festgelegten Werte gelten soweit als nicht anders beschrieben. Hierbei handelt es sich vor allem um die Zahl, der in der Simulation verwendeten Knoten. Diese wird für die Simulation mit 200 Knoten pro Netzwerk festgelegt. Des Weiteren wird die Länge der verwendeten Identifier auf 16-Bit festgelegt. Dies entspricht zwar nicht den in der Realität genutzten Werten, schränkt aber den Berechnungs- und Speicheraufwand stark ein, ohne hierbei zu viel Aussagekraft der

ermittelten Ergebnisse zu verlieren. Dies ist nötig, da sich die Optimierung der Id-Bereiche vorläufig nur durch eine sehr rechenlastige Implementierung lösen ließ.

Zusätzlich sind einige protokollspezifische Werte notwendig. Beim CAN-Protokoll wird der Identifier auf 4 Dimensionen aufgeteilt, die jeweils eine Identifierlänge von 4 Bit nutzen. Die Chord-Successor-Liste, die die Redundanz des Protokolls beeinflusst, wird auf 8 festgelegt.

Einige Aspekte wurden zu Simulationszwecken vereinfacht oder auf die Simulationsumgebung angepasst. Einer dieser Aspekte ist der lokale Zustand. Hierbei wird, anders als bisher vorgesehen, nicht nur ein Protokoll auf dem Protokollstack betrieben, sondern alle im Verlauf der Simulation benötigten Protokolle werden während der ganzen Simulation im Speicher gehalten. Hiervon ist aber zu jedem Zeitpunkt nur eines der betreffenden Protokolle aktiv (gilt nur für die eigentlichen Protokolle, nicht die zusätzlichen Transportschichten und Services). Dies wird durch den bereits erläuterten *HybridTransport* gesteuert.

Die Auslösung der Transformation erfolgt mit Hilfe eines speziellen *Control*s. Dieser *TransformationStarter* sendet zum konfigurierbaren Startzeitpunkt der Transformation eine einzelne Transformationsnachricht an einen zufällig ausgewählten Knoten des Netzwerkes. Von hier an übernehmen, die innerhalb der Protokolle für die Ausbreitung der Transformationsinformationen zuständigen Mechanismen das weitere Vorgehen.

Um die Netzwerklast während der Simulation zu erzeugen gibt es im Peersim-Simulator generell zwei Möglichkeiten. Zum Einen kann ein Service auf dem Protokollstack ausgeführt werden der seinerseits wiederum andere Protokolle als Underlay nutzt indem die APIs dieser Protokolle genutzt werden. Um Aktionen des Services auszulösen wird ein *Control* verwendet. Zum Anderen kann direkt ein *Control* verwendet werden, um Nachrichten ins Netzwerk einzugeben, von wo aus diese an ihr Ziel weitergeleitet werden. Durch die Wahl zufälliger Start- und Endpunkte der Nachrichten sowie eine ausreichende Anzahl von Nachrichten wird eine gleichmäßige Nachrichtenbelastung im Netzwerk erzeugt.

Für die Ermittlung der im Folgenden vorgestellten Werte wird ebenfalls ein *Control* genutzt. In diesem Fall werden allerdings keine Nachrichten an die Knoten versendet, sondern es wird auf die Methoden der Protokolle zugegriffen, um die jeweiligen Werte zu ermitteln.

6.3 Ergebnisse der Simulation

In diesem Abschnitt werden die in der Simulation ermittelten Ergebnisse in Auszügen dargestellt und erläutert. Um Aussagen über die Leistungsfähigkeit der entwickelten Verfahren zu machen, wird im Folgenden die Transformation durch ein Hybridnetzwerk aus Abschnitt 5.2 evaluiert. Hierzu wird auch die erbrachte Leistung mit der nebenläufigen Neukonstruktion aus Abschnitt 5.1.3 verglichen.

Des Weiteren wird diese Evaluation auf die interessantere der beiden möglichen Transformationsrichtungen eingeschränkt. Es wird also nur die Transformation von einem Chord- in ein CAN-Netzwerk betrachtet.

Da im Rahmen dieser Arbeit nicht alle Aspekte hinreichend simuliert werden konnten, aber stark von den ermittelten Simulationsergebnissen abhängig sind, werden zusätzlich weitergehende Schlussfolgerungen aus den Simulationsergebnissen gezogen.

6.3.1 Komplexität

Das wesentliche Attribut der Transformation stellen die verschiedenen Arten der Komplexität dar. Hierbei ist es möglich, die Nachrichtenkomplexität, also die Zahl der für die Transformation benötigten Nachrichten zu betrachten. Dies geschieht in Abschnitt 6.3.1.

Alternativ kann für einige Anwendungen der Zeitaufwand der Transformation ermittelt werden. Dies geschieht in Abschnitt 6.3.1.

Nachrichtenkomplexität

Bei der Evaluation des Nachrichtenaufwands werden verschiedene Bereiche der Transformation betrachtet. Zum Einen kann die Zahl der für die Transformation selbst benötigten Nachrichten, wie etwa die Transformations-, Beitritts- und Routingnachrichten gezählt werden. Zum Anderen kann die Zahl, der im Zuge der Transformation zu migrierenden Schlüssel/Wert-Paare, also die Zahl der Inhalte, die nicht weiter auf dem gleichen Knoten gespeichert werden können, gezählt werden. Diese zweite Betrachtung findet im Rahmen der Fairnessbetrachtungen in Abschnitt 6.3.3 statt.

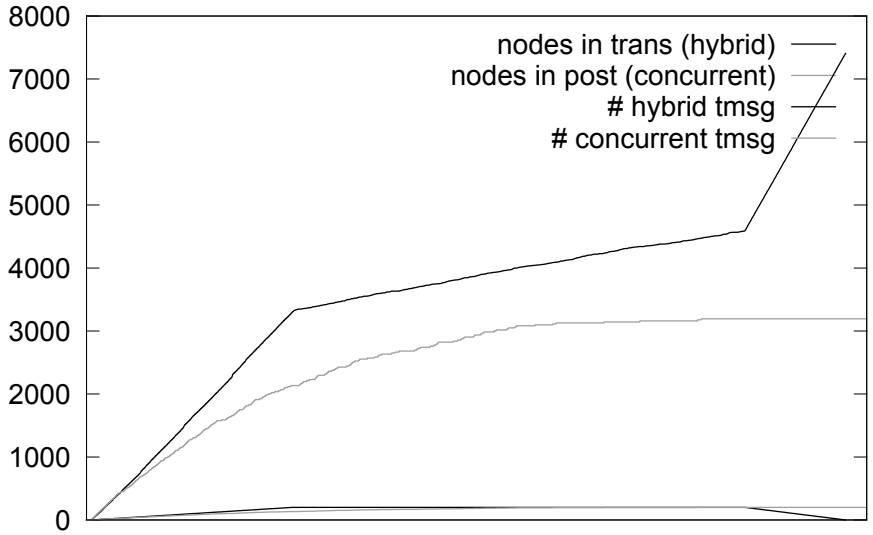

Abbildung 6.2: Zahl der Transformationsnachrichten für die Transformation

Abbildung 6.2 zeigt die Zahl der während der Transformation durch ein Hybridnetz versendeten Transformationsnachrichten, sowie die bei der nebenläufigen Neukonstruktion

versendeten Transformationsnachrichten. Die Zeitachse ist hierbei auf ein gleichzeitiges Ende der beiden Transformationsvoränge skaliert.

Hierbei sind die drei Phasen der Transformation mit Hilfe eines Hybridnetzes sehr gut zu erkennen. In diesen Phasen wird die Transformation ausgebreitet, optimiert und schließlich beendet. Die Übergänge zwischen den Phasen finden nach dem Versand von 3.300, 4.600 und 7.400 Transformationsnachrichten statt. Unterlegt ist dies mit der Transformationswelle des nebenläufigen Neuaufbaus. Diese ist zwar über den gesamten Bereich flacher, jedoch enthält diese nicht die Nachrichten für die Migration, beziehungsweise den Neuaufbau, der im Netz gespeicherten Daten. Die hier simulierte Transformation konvergiert nach dem versenden von 3.200 Transformationsnachrichten. Auf Grund der in weiten Teilen unveränderten Zuständigkeitsbereichen, weist das durch die Transformation durch das Hybridnetz entstandene Netz, deutlich weniger Migrationsbedarf auf.

Der Grund, dass die Transformation durch das Hybridnetz mehrere hundert Transformationsnachrichten mehr benötigt, um alle Knoten in den Transformationszustand zu überführen, findet sich in der zu diesem Zeitpunkt bereits begonnen gegenseitigen Optimierung. Dies ist bereits möglich, da von FIFO-Kanälen ausgegangen wird, wodurch kein Knoten in direkten Kontakt mit einem Knoten treten kann, der nicht sich nicht im Transformationszutand befindet. Um dies zu gewährleisten wird, nach dem Empfang einer Transformationsnachricht, unverzüglich die Transformation lokal begonnen und weiter ausgebreitet.

Vergleich mit dem prognostizierten Aufwand

Abbildung 6.1 zeigt die anhand der in Abbildung 5.3.1 vorgestellten Abschätzungen ermittelten Erwartungswerte für die simulierte Transformation.

	Nebenläufige Neukonstruktion	**Transformation durch Hybridnetz**
Chord→CAN	1.920	2.580
CAN→Chord	1.860	2.520

Tabelle 6.1: Prognostizierte Aufwände für die Transformation von 200 Knoten

Den abgeschätzen Nachrichtenzahlen stehen etwa 3.200 beziehungsweise 7.400 Nachrichten gegenüber. Hiermit liegt der Aufwand für die nebenläufige Neukonstruktion durchaus in der abgeschätzten Größenordnung, wenn auch etwas zu niedrig angesetzt. Der Aufwand für die Transformation durch ein Hybridnetz benötigt in der Simulation jedoch etwa die dreifache Zahl der prognostizierten Transformationsnachrichten.

Für dieses stark erhöhte Nachrichtenaufkommen sind zwei Aspekte verantwortlich. Zum einen findet hierbei bereits die gegenseitige Optimierung der Bereiche statt. Hierbei werden Bereiche auf andere Knoten migriert, wofür ebenfalls Transformationsnachrichten verwendet werden. Dieser zusätzliche Aufwand wird mit der Formel $n * (\log n * (\log n + d/2))$ zu 1.980

Nachrichten für 200 Teilnehmer abgeschätzt. Wird dies berücksichtigt, liegt auch dieser Transformationsaufwand wesentlich näher am prognostizierten Bereich. Dieses kann durch den zweiten Aspekt erklärt werden. Ausschlaggebend ist hierfür der unzureichende Grad der Optimierung. Hierdurch entstehen ungünstig geformte Id-Bereiche, die mehr Nachbarn als nötig aufweisen. Somit steigt der Koordinierungs- und Kommunikationsaufwand.

Zeitaufwand

Die Betrachtung des Zeitaufwands ist abgesehen von einer grundsätzlichen Relevanz für die Verwendbarkeit nur in recht wenigen Szenarien von Wichtigkeit. Dies ist vor allem immer dann der Fall, wenn das verbindende Netzwerk nicht ausgelastet ist und die vermittelten Nachrichtenpakete nicht bezahlt werden müssen. Dies entspricht im Allgemeinen dem Betrieb innerhalb von Firmennetzwerken.

Darum wird hier nur eine begrenzte theoretische Betrachtung des Zeitaufwandes vorgenommen. Ausschlaggebend sind hierbei die Ergebnisse in Abschnitt 6.3.1 sowie der Grad der Parallelisierung des Transformationsvorgangs. Zusätzlich wird im Folgenden eine mittlere Latenz der für die Nachrichtenweiterleitung von T angenommen.

Um die Parallelisierbarkeit zu ermitteln wird an die Ergebnisse der Aufwandsabschätzung angeknüpft. Hierbei sind sowohl die Wellen des Transformationsstarts und -stopps wie auch die für die Optimierung benötigten Durchgänge ausschlaggebend. Für die Transformationswellen wird eine Ausbreitungsdauer von jeweils $T * \log n$ angenommen. Dies folgt aus dem mittleren Netzwerkdurchmesser, der sequenziell durchschritten werden muss um alle Knoten zu erreichen. Mit der jeweiligen Antwortwelle wird also $2T * \log n$ pro Ausführung des Echo-Algorithmus benötigt.

Für die Optimierung wurden im Rahmen der Aufwandsabschätzung $\log n$ Runden angenommen. Diese können in sich parallel ausgeführt werden. Die einzelnen Optimierungsrunden wiederum müssen sequenziell durchgeführt werden. Für eine Optimierungsrunde werden $2T$ benötigt.

Aus den angestellten Betrachtungen folgt ein Zeitaufwand für die gesamte Transformation unter Nutzung eines Hybridnetzes von $6T * \log n$.

6.3.2 Routingperformance

In diesem Abschnitt werden die in der Simulation ermittelten Werte für die Routingperformance betrachtet. Hierbei wurde sowohl die Leistungsfähigkeit ohne spezielle Vorkehrungen sowie mit einem an die Probleme angepassten Routing ermittelt.

Abbildung 6.3 zeigt die erzielte Routingqualität unter Verwendung des Greedy-Forwardings. Hierzu wurde das Netzwerk vom Chord-Protokoll in das Transformationsprotokoll überführt und ohne weitere Optimierung betrieben. Hierbei erreichen etwa 95% der Nachrichten ihren Zielknoten innerhalb von 20 Hops. Diese TTL wurde eingeführt, um das Netz nicht unnötig

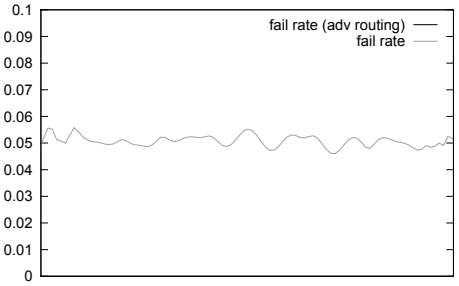

Abbildung 6.3: Nicht zugestellte Nachrichten der verwendeten Algorithmen

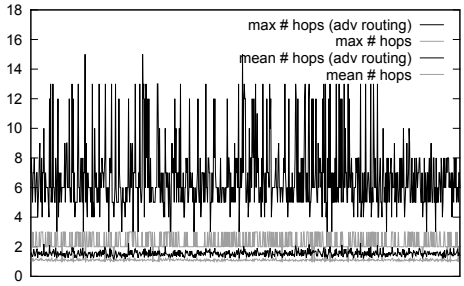

Abbildung 6.4: Routingperformance der verwendeten Algorithmen

duch nicht zustellbare Nachrichten zu belasten. Unter Verwendung des in Abschnitt 5.2.4 vorgestellten verbesserten Algorithmus treten keine Nachrichtenverluste mehr auf.

Abbildung 6.4 zeigt die hierbei erzielte Routingperformance in Form der benötigten Pfadlängen. Erwartungsgemäß steigt hierbei die durchschnittliche und maximale Länge der Routingpfade, unter Verwendung des verbesserten Routungalgorithmus, an.

6.3.3 Lastbalancierung

In Abschnitt 2.1.2 wurden bereits verschiedene Arten der Lastbalancierung erläutert. Diese werden hier unter Zuhilfenahme der Ergebnisse der Simulation nochmals betrachtet.

Belastung durch Suche

Die Belastung durch die Suche ähnelt sehr der Betrachtung des Routings in Abschnitt 6.3.2. Darum wurde hierfür keine erneute Simulation durchgeführt.

Aus den Betrachtungen des Routings lässt sich unter Zuhilfenahme der Annahme, dass die Lookups gleichmäßig über den Id-Raum verteilt werden, die Belastung durch die Suche schlussfolgern. Hierbei wird klar, dass bedingt durch den verwendeten Routingalgorithmus, verlängerte Routingpfade entstehen. Hiermit steigt auch die Gesamtbelastung des Netzes durch die Suche proportional zur Verlängerung der Routingpfade an.

Verteilung der Inhalte

Ausgehend von einer Gleichverteilung der Inhalte ergibt sich die Verteilung dieser direkt aus der Größe der von den einzelnen Knoten vor, während und nach der Transformation

verwalteten Id-Bereiche. Aus den entsprechenden Differenzen lässt sich der benötigte Migrationsaufwand für die Schlüssel/Wert-Paare herleiten. Zusätzlich kann die Effektivität der Optimierung sehr gut an diesen Werten abgelesen werden.

Abbildung 6.5 zeigt hierbei die Zahl der auf den Knoten gehaltenen Id-Bereiche sowie Ids unter Variation der Optimierungsparameter. Die Erste Variante (1) gewichtet die Ausprägung, also die Geometrie, des jeweiligen Id-Bereiches besonders hoch, während in der zweiten Variante (2) die Zahl der Ids höher bewertet wird. Die oberen beiden Kurven zeigen die mittlere Zahl der von den Knoten gehaltenen Ids, während die unteren beiden Kurven die durchschnittliche Zahl der gehaltenen Id-Bereiche zeigen.

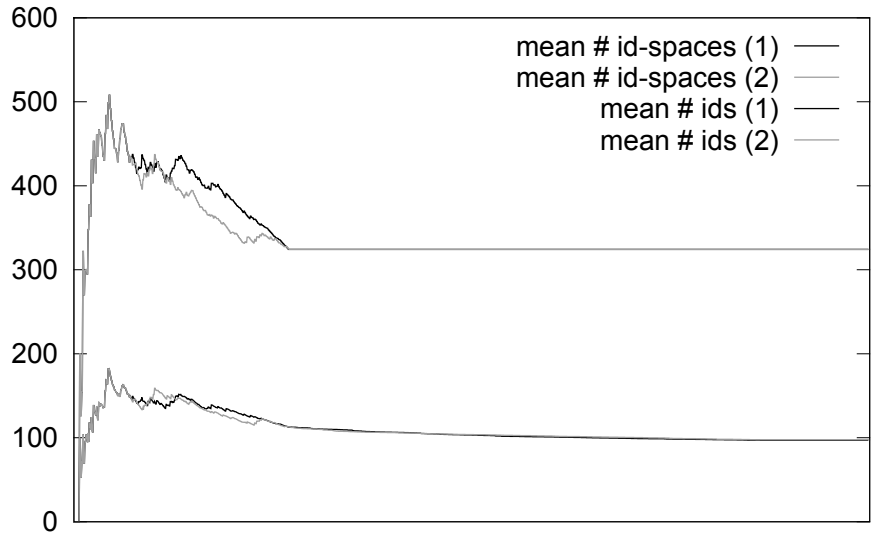

Abbildung 6.5: Variation der Optimierungsparameter

Durch die Variation der Optimierungsparameter ergeben sich einige Abweichungen im Verlauf der Transformation. Diese betreffen jedoch nur den Bereich der Startphase der Transformation, in der die einzelnen Konten in den Transformationszustand überführt werden. Da die gegenseitige Optimierung zu diesem Zeitpunkt bereits aktiv ist, kann ein Ansteigen und anschließendes leichtes Abfallen der durchschnittlichen Größe der Id-Bereiche beobachtet werden. Mit fortschreitendem Verlauf nähern sich die betrachteten Durchschnittswerte einander an. In beiden Fällen ist das Optimierungspotential jedoch sehr schnell erschöpft. Der Grund hierfür wird in Abschnitt 6.3.5 eingehender behandelt.

Abbildung 6.6 zeigt den Einfluss der verwendeten CAN-Dimensionen auf die Transformation. Auch hier zeigen die oberen beiden Kurven die mittlere Zahl der von den Knoten gehaltenen Ids, während die unteren beiden Kurven die durchschnittliche Zahl der gehaltenen Id-Bereiche zeigen.

Abweichend von dem zuvor beschriebenen Abläufen kann hier beobachtet werden, wie die höhere Dimensionalität des Id-Raums, in den die Knoten eingebettet werden dazu genutzt wird, um die Optimierung zu unterstützen. Dies begründet sich in der größeren Auswahl an

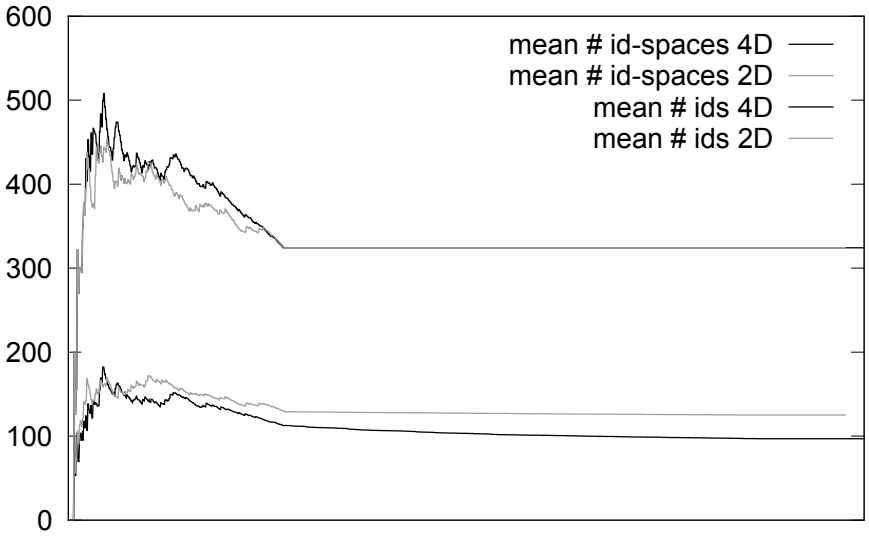

Abbildung 6.6: Variation der CAN-Dimensionen

Nachbarn mit ähnlichen Id-Verantwortlichkeiten und somit mehr geeigneten Kandidaten für die gegenseitige Optimierung.

6.3.4 Einfluss der raumfüllenden Kurve

Der Einfluss der für die Id-Transformation verwendeten raumfüllenden Kurven begründet sich durch die Veränderungen in der Erhaltung der Lokalität.

Untersucht wurde der Unterschied zwischen der Nutzung der sequenziellen Raumkurve sowie der Hilbert-Kurve. Beide wurden in Abschnitt 4.5.1 eingehend erläutert.

Hierbei überrascht es nicht, dass unter Nutzung der Hilbert-Kurve bessere Ergebnisse in der Optimierung erzielt werden, als unter Nutzung der sequenziellen Raumkurve. Auch hier ist der Grund, ähnlich dem Sachverhalt der gesteigerten Dimensionalität auf die höhere Lokalität zurückzuführen. Ähnliche Ids, die auf der eindimensionalen Kurve nahe beieinander liegen, liegen also auch im Raum nahe beieinander. Hieraus folgt der Aufbau, für die Optimierung, erfolgversprechender Nachbarschaftsbeziehungen.

Abbildung 6.8 zeigt die entsprechenden Knotengrade und versendeten Transformationsnachrichten. Die oberen Kurven zeigen den mittleren Knotengrad, während die steigenden Kurven die Zahl der versendeten Transformationsnachrichten beschreibt. Durch die Verknüpfung der Informationen aus beiden Betrachtungen wird deutlich, dass die Nutzung der Hilbert-Kurve sich auch auf den entstehenden Knotengrad auswirkt. Da dieser während der Transformation bedingt durch das verwendete Transformationskonzept nur, durch das hinzufügen von im Zielnetz benötigten Nachbarn, steigt und mehr neue Nachbarn benötigt werden, erhöht sich, als Folge hieraus der durchschnittliche Knotengrad. Zusätzlich müssen

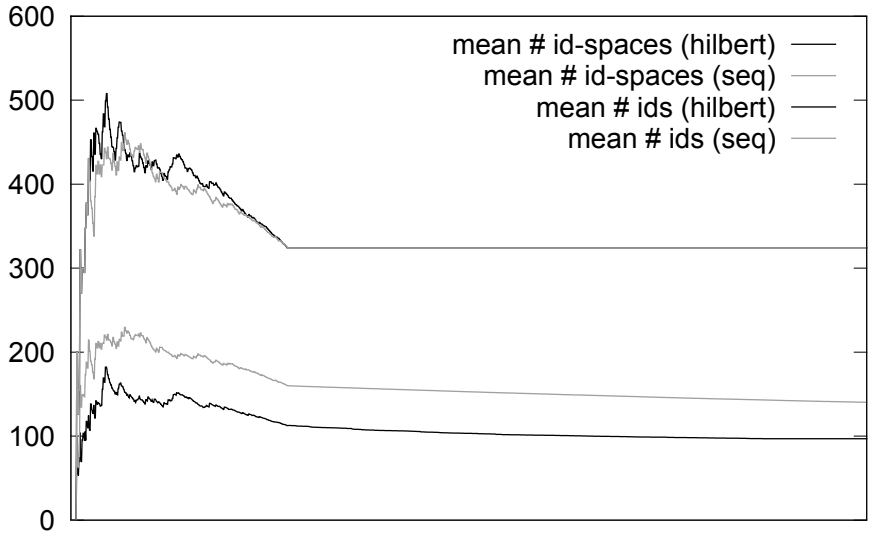

Abbildung 6.7: Zahl der Ids und Id-Bereiche während der Transformation unter Nutzung verschiedener Raumfüllender Kurven

mehr Transformationsnachrichten versendet werden, um die benötigten Nachbarn zu finden. Ebenso müssen während der ganzen Transformation mehr Transformationsnachrichten versendet werden, um diese zu steuern.

6.3.5 Problem der paarweisen Optimierung

Aus den angestellten Untersuchungen geht hervor, dass die Transformation mit Hilfe eines Hybridnetzes funktionsfähig ist. Jedoch werden auch die noch vorhandenen Probleme in der Umsetzung deutlich. Das hierbei massivste Problem ist die mangelnde Vereinigung der auf den Knoten befindlichen Id-Bereiche. Diese sind zwar zusammenhängend, jedoch gelingt es durch den verwendeten Algorithmus nicht diese zusammenzufügen. Der Hauptgrund hierfür ist das verwendete paarweise Verbinden der Bereiche. Dies ist bisher sowohl lokal, als auch während der gegenseitigen Optimierung, mit anderen Knoten, der Fall.

Durch diese Praxis bilden sich jedoch Mengen von Bereichen, auf den Knoten, die nicht weiter paarweise verbunden werden können. Abbildung 6.9 zeigt einige Situation beispielhaft, in denen drei Id-Bereiche abgebildet sind, die zwar zu oder zu viert, aber nicht paarweise kombiniert werden können.

Hierbei ist jeder der drei abgebildeten Komplexe offensichtlich kombinierbar. Ebenso offensichtlich kann dies ein paarweise arbeitender Algorithmus, wie der in dieser Evaluation verwendete, aber nicht leisten. Für zukünftige Umsetzungen muss also besonders Augenmerk auf die Verbesserung der Optimierungsmechanismen gelegt werden, um diesen auch höherwertigere Kombinationen zu ermöglichen.

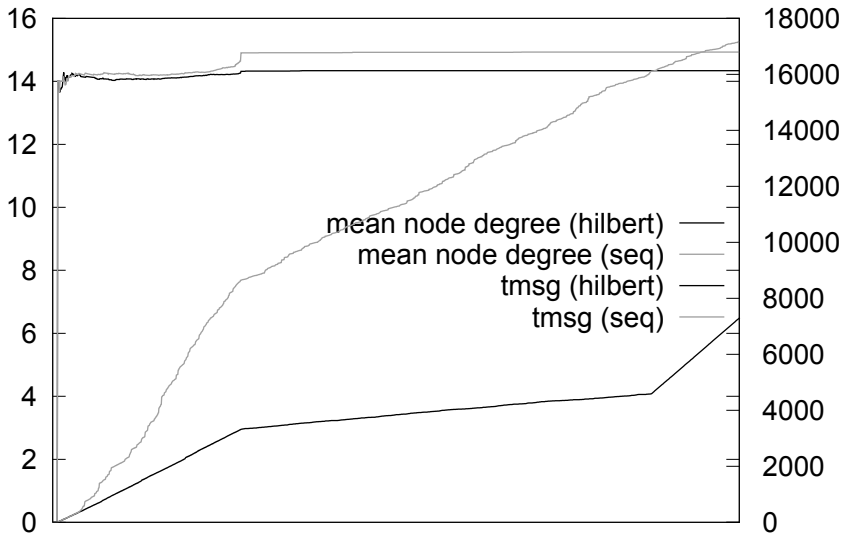

Abbildung 6.8: Knotengrade und Zahl der Transformationsnachrichten während der Transformation unter Nutzung verschiedener Raumfüllender Kurven

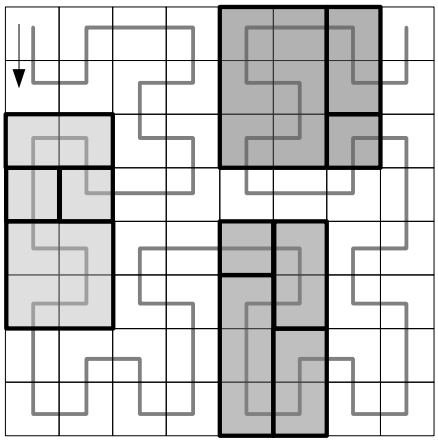

Abbildung 6.9: Kombinationsproblem der Id-Bereiche

Kapitel 7

Fazit

Overlaybasierte Publish/Subscribe-Systeme bauen die von ihnen genutzten Verteilstrukturen entlang der Verbindungen im unterlagerten Overlay auf. Nach signifikanten Änderungen in der Belastung dieser Systeme kann die Leistungsfähigkeit dieser durch die Rekonfiguration des Overlays erhöht werden.

Hierfür wurden im Rahmen dieser Arbeit, ausgehend von dem Ansatzpunkt, die Topologie strukturierter Overlaynetze zu rekonfigurieren um diese so besser an die Bedürfnisse der auf ihnen aufgesetzen Services anzupassen, verschiedene Transformationsverfahren entwickelt. Als Ausgangs- und Endpunkte der Transformation wurden die Protokolle Chord und CAN gewählt, da diese sehr unterschiedliche Topologien aufweisen. Dieser strukturelle Unterschied macht die Transformation besonders reizvoll.

Um diese auf die Overlaynetze anzuwenden wurde eine modularisierte Systemarchitektur vorschlagen. Mit Hilfe dieser wurden die erforderlichen Module ermittelt, konzeptioniert und umgesetzt.

Diese sind im Einzelnen: Ein Indirektionsmechanismus, der eine höhere Flexibilität im Umgang mit mehreren simultan oder abwechselnd verwendeten Protokollen erhöht. Eine bidirektionale Adressabbildung, zur Kompensation verschiedenartiger Id-Räume. In diesem Zusammenhang wurden die Eignung und die Komplexität verschiedener raumfüllender Kurven, in Hinsicht auf ihre Nutzung und Auswirkungen in der Transformation, betrachtet. Hierbei wurden eine sequenzielle Raumkurve, sowie die Hilbert-Kurve für die weitere Verwendung ausgewählt. Im Kernteil wurden Transformationsprotokolle, die in Kombination oder im Austausch mit den verwendeten Anfangs- und Zielprotokollen verwendet werden, entwickelt. Hierzu wurden zunächst verschiedene Ansätze im Groben diskutiert. Zwei konkurrierende Ansätze wurden weiter vertieft. Hierbei handelt es sich um die nebenläufige Neukonstruktion eines Netzwerkes, sowie die Transformation mit Hilfe eines Hybridnetzes. Für diese wurden in diesem Rahmen ein eigener Optimierungs- und ein angepasster Routingalgorithmus entwickelt.

Nach der Weiterentwicklung und Abschätzung, der für die Verfahren benötigten Aufwände, wurde eine simulationsgestützte Evaluation durchgeführt. In dieser konnten viele der prognostizierten Eigenschaften nachgewiesen werden. Jedoch wurden hierbei auch einige

Probleme des Optimierungsalgorithmus festgestellt und erläutert. Da dieser verschiedene Id-Bereiche nur paarweise zusammenfügen kann, konvergiert die Optimierung der Knotenzustände während der Transformation unerwartet früh. Trotz seiner eingeschränkten Leistungsfähigkeit hat sich dieser außerdem als ausgesprochen rechenlastig herausgestellt.

Hieraus muss geschlussfolgert werden, dass die vorgestellten und evaluierten Verfahren, obwohl beide die Transformation in einer kontrollierten Umgebung leisten können, noch nicht praxistauglich sind.

Für die Eignung der jeweiligen Verfahren, für verschiedene Situationen, können einige Grundaussagen festgehalten werden. Die Transformation durch eine nebenläufige Neukonstruktion des Zielnetzwerkes eignet sich besonders für große Netze, mit vielen Teilnehmern. Jedoch leidet dieser Ansatz sehr unter der im Netzwerk vorhandenen Datenlast. Sind also viele Daten in dem betreffenden Netzwerk gespeichert oder sind viele Verteilbäume über dem Netzwerk aufgespannt, lässt die Leistungsfähigkeit stark nach.

Die Transformation mit Hilfe eines Hybridnetzes leidet hingegen eher unter der Zahl der Teilnehmer im Netzwerk. Dies ist dem Umstand geschuldet, dass hierbei keine optimierten Beitrittsalgorithmen oder Ähnliches genutzt werden können um die Komplexität zu reduzieren. Jedoch ist die Zahl der zu migrierenden Inhalte wesentlich geringer, da dies nur im Rahmen der gegenseitigen Optimierung stattfindet.

Ein weiteres Plus für die Transformation mit Hilfe eines Hybridnetzes sind die bisher nicht ausgeschöpften Optimierungspotentiale. Die Optimierungspotentiale betreffen sowohl den Routingalgorithmus, besonders jedoch den gegenseitigen Optimierungsalgorithmus. Dieser muss dringend in Hinsicht auf auf komplexere Kombinationsmöglichkeiten verschiedener Id-Bereiche erweitert werden. Zusätzlich muss der insgesamt benötigte Rechenaufwand durch einen geschickteren Umgang mit den vorliegenden Daten verbessert werden.

Durch diese Verbesserungen kann, mit Hilfe der Transformation durch ein Hybridnetz, ein Zielnetz erzeugt und genutzt werden, dass dem Ausgangsnetz relativ ähnlich ist, jedoch andere Charakterisika aufweist, die sich wiederum auf die hierauf aufbauenden Services auswirken.

In zukünftigen overlaybasierten Systemen spielt die Topologie des Overlays und deren Rekonfigurationsmöglichkeiten eine entscheidende Rolle für die Effizienz und die erbrachte Leistung der Systeme. Mit entsprechenden Möglichkeiten ausgestattete Systeme können wesentlich effizienter auf Veränderungen reagieren, als Systeme, die keine Möglichkeit haben ihr verwendetes Overlay anzupassen.

Um die in dieser Arbeit angestellten Betrachtungen und Annahmen weiter zu validieren, ist es sinnvoll die bereits in der Evaluationsphase angestellten Bemühungen fortzuführen. Die hierbei ermittelten Ergebnisse können in die Weiterentwicklung der bisherigen Vorgehensweisen einfließen. Besonderes Augenmerk muss hierbei auf Aussagen über die Robustheit der verwendeten Verfahren gelegt werden. Hierzu zählen das Verhalten unter Churn, gezielten und ungezielten Angriffen und schließlich auch kompletten Partitionierungen des zu transformierenden Netzwerkes. Zusätzlich können das verwendete Routingprotokoll und die Optimierung der Zuständigkeit der Knoten weiteren Untersuchungen unterzogen werden.

Auch die Erweiterung des Ansatzes um zusätzliche Overlaystrukturen und P2P-Systeme kann ein Weiterentwicklungspunkt sein.

Ebenso interessant sind die tatsächlichen Auswirkungen auf die, auf den betrachteten Overlaynetzwerken aufgesetzten, Publish/Subscribe-Systeme. Hierüber konnten bis zu diesem Zeitpunkt keine Evaluierungen durchgeführt werden. Es kann untersucht werden, ob sich die erwarteten Eigenschaften der erzeugten Netzwerke wie prognostiziert ausnutzen lassen, um die Leistungsfähigkeit zu erhöhen.

In dieser Arbeit wurden Grundlagen geschaffen, um ansonsten recht starre strukturierte Overlays in ihrer Gestalt zu beeinflussen. Dies kann genutzt werden um die Bandbreite der Reaktionsmechanismen zu erhöhen, die Services, die auf strukturierten Overlays basieren, zur Verfügung stehen. Aus diesen Reaktionsmechanismen können Strategien entwickelt werden, mit deren Hilfe in Zukunft auf die Leistung ansonsten einschränkenden Belastungsänderungen der Systeme zu reagieren.

Literaturverzeichnis

[ADHS05] Aberer, K. ; Datta, A. ; Hauswirth, M. ; Schmidt, R.: Indexing data-oriented overlay networks. In: *Proceedings of the 31st international conference on Very large data bases* VLDB Endowment, 2005, S. 696 (Zitiert auf Seite 24)

[ARR+97] Asano, T. ; Ranjan, D. ; Roos, T. ; Welzl, E. ; Widmayer, P.: Space-filling curves and their use in the design of geometric data structures. In: *Theoretical Computer Science* 181 (1997), Nr. 1, S. 3–15 (Zitiert auf Seite 32)

[BA99] Barabási, A.L. ; Albert, R.: Emergence of scaling in random networks. In: *Science* 286 (1999), Nr. 5439, S. 509 (Zitiert auf Seite 6)

[BBQV07] Baldoni, R. ; Beraldi, R. ; Querzoni, L. ; Virgillito, A.: Efficient publish/-subscribe through a self-organizing broker overlay and its application to SIENA. In: *The Computer Journal* 50 (2007), Nr. 4, S. 444. – ISSN 0010–4620 (Zitiert auf Seite 24)

[BMVV05] Baldoni, R. ; Marchetti, C. ; Virgillito, A. ; Vitenberg, R.: Content-based publish-subscribe over structured overlay networks. In: *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on* IEEE, 2005. – ISBN 0769523315, S. 437–446 (Zitiert auf Seite 18)

[But06] Butz, AR: Alternative algorithm for Hilbert's space-filling curve. In: *Computers, IEEE Transactions on* 100 (2006), Nr. 4, S. 424–426. – ISSN 0018–9340 (Zitiert auf Seite 33)

[DA06] Datta, A. ; Aberer, K.: The challenges of merging two similar structured overlays: A tale of two networks. In: *Self-Organizing Systems* (2006), S. 7–22 (Zitiert auf Seite 39)

[Dat07] Datta, A.: Merging intra-planetary index structures: decentralized bootstrapping of overlays. In: *First International Conference on Self-Adaptive and Self-Organizing Systems, 2007. SASO'07*, 2007, S. 109–118 (Zitiert auf Seite 39)

[DZD+03] Dabek, F. ; Zhao, B. ; Druschel, P. ; Kubiatowicz, J. ; Stoica, I.: Towards a common API for structured peer-to-peer overlays. In: *Peer-to-Peer Systems II* (2003), S. 33–44 (Zitiert auf Seite 13)

[GHW07] Ghodsi, A. ; Haridi, S. ; Weatherspoon, H.: Exploiting the synergy between gossiping and structured overlays. In: *ACM SIGOPS Operating Systems Review* 41 (2007), Nr. 5, S. 66 (Zitiert auf Seite 24)

[JB04] Jelasity, M. ; Babaoglu, O.: T-Man: Fast gossip-based construction of large-scale overlay topologies. In: *University of Bologna, Department of Computer Science, UBLCS-2004-7, Bologna, Italy* (2004) (Zitiert auf Seite 24)

[JMJV] Jelasity, Márk ; Montresor, Alberto ; Jesi, Gian P. ; Voulgaris, Spyros: *The Peersim Simulator*. – `http://peersim.sf.net` (Zitiert auf Seite 61)

[JMW+08] Jaeger, M.A. ; Muhl, G. ; Werner, M. ; Parzyjegla, H. ; Heiss, H.U.: Algorithms for Reconfiguring Self-Stabilizing Publish/Subscribe Systems. In: *Autonomous Systems-Self-Organization, Management, and Control: Proceedings of the 8th International Workshop Held at Shanghai Jiao Tong University, Shanghai, China, October 6-7, 2008* Springer Verlag, 2008. – ISBN 1402088884, S. 135 (Zitiert auf Seite 16)

[KCC+05] Kumar, V. ; Cooper, B.F. ; Cai, Z. ; Eisenhauer, G. ; Schwan, K.: Resource-aware distributed stream management using dynamic overlays. In: *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on* IEEE, 2005. – ISBN 0769523315, S. 783–792 (Zitiert auf Seite 24)

[MAK02] Mokbel, M.F. ; Aref, W.G. ; Kamel, I.: Performance of multi-dimensional space-filling curves. In: *Proceedings of the 10th ACM international symposium on Advances in geographic information systems* ACM, 2002, S. 149–154 (Zitiert auf den Seiten 32, 33 und 35)

[MJB05] Montresor, A. ; Jelasity, M. ; Babaoglu, O.: Chord on demand. (2005) (Zitiert auf Seite 39)

[PEFK09] Popescu, A. ; Erman, D. ; Fiedler, M. ; Kouvatsos, D.: Routing in Content Addressable Networks: Algorithms and Performance. In: *20th ITC-Specialist Seminar* IEEE, 2009 (Zitiert auf Seite 13)

[RD01] Rowstron, A. ; Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: *Middleware 2001* Springer, 2001, S. 329–350 (Zitiert auf den Seiten 2, 9 und 17)

[RFH+01] Ratnasamy, S. ; Francis, P. ; Handley, M. ; Karp, R. ; Schenker, S.: A scalable content-addressable network. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* ACM, 2001, S. 172 (Zitiert auf den Seiten 2, 9 und 11)

[RHKS01] Ratnasamy, S. ; Handley, M. ; Karp, R. ; Shenker, S.: Application-level multicast using content-addressable networks. In: *Networked Group Communication* (2001), S. 14–29 (Zitiert auf Seite 17)

[RKCD01] Rowstron, A. ; Kermarrec, A.M. ; Castro, M. ; Druschel, P.: SCRIBE: The design of a large-scale event notification infrastructure. In: *Networked Group Communication* (2001), S. 30–43 (Zitiert auf Seite 17)

[SGH07] Shafaat, T.M. ; Ghodsi, A. ; Haridi, S.: Handling network partitions and mergers in structured overlay networks. (2007) (Zitiert auf den Seiten 24 und 39)

[SLP09] Stevens, RJ ; Lehar, AF ; Preston, FH: Manipulation and presentation of multidimensional image data using the Peano scan. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2009), Nr. 5, S. 520–526. – ISSN 0162–8828 (Zitiert auf Seite 33)

[SMK+01] Stoica, I. ; Morris, R. ; Karger, D. ; Kaashoek, M.F. ; Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* ACM, 2001, S. 149–160 (Zitiert auf den Seiten 2 und 9)

[Tel00] Tel, G.: *Introduction to distributed algorithms*. Cambridge University Press, 2000. – 194–196 S. – ISBN 0521794838 (Zitiert auf den Seiten 40 und 44)

[TKKR09] Tariq, M.A. ; Koldehofe, B. ; Koch, G.G. ; Rothermel, K.: Providing probabilistic latency bounds for dynamic publish/subscribe systems. In: *Kommunikation in Verteilten Systemen (KiVS)* Springer, 2009, S. 155–166 (Zitiert auf Seite 25)

[WBH+08] Waldhorst, Oliver ; Blankenhorn, Christian ; Haage, Dirk ; Holz, Ralph ; Koch, Gerald ; Koldehofe, Boris ; Lampi, Fleming ; Mayer, Christoph ; Mies, Sebastian: Spontaneous Virtual Networks: On the Road towards the Internet's Next Generation. In: *it- Information Technology Special Issue on Next Generation Internet* 50 (2008), Dezember, Nr. 6. `http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=ART-2008-18&engl=0` (Zitiert auf Seite 9)

[ZKJ01] Zhao, B.Y. ; Kubiatowicz, J. ; Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. In: *Computer* 74 (2001), S. 11–20 (Zitiert auf den Seiten 2 und 9)

Alle URLs zuletzt am 14.01.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Christoph Schlameuß)