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

Diplomarbeit Nr. 3230

Optimierte Datenstromverwaltung für globale Sensornetze

Stefan Wenz

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	Dipl.-Inf. Andreas Benzing
begonnen am:	1. September 2011
beendet am:	2. März 2012
CR-Klassifikation:	C.2.4, C.4

Abstract

Die stetig steigende Datenmenge stellt eine neue Herausforderung für bestehende Netzwerkstrukturen und Systeme dar. Globale Sensornetzwerke sind des Weiteren auf Grund des steigenden öffentlichen Interesses mit der zunehmenden Anzahl anfragender Clients konfrontiert. Um eine höhere Verfügbarkeit der Daten zu ermöglichen und zugleich die vorhandenen Ressourcen optimal zu nutzen, können verschiedene Maßnahmen ergriffen werden. Innerhalb einer brokerbasierten Netzwerkstruktur wird durch Replikation der Daten von hochfrequentierten Hotspots auf einen Broker mit ausreichend freien Ressourcen die Verfügbarkeit gesteigert. Problematisch ist die Auswahl des Replikationspunktes, die unter Berücksichtigung mehrerer Aspekte erfolgen kann. Ziel der Diplomarbeit ist eine latenzoptimierte Auswahl. Ausgangspunkt für die praktische Umsetzung der theoretisch erarbeiten Lösungen dieser Ausarbeitung ist das bestehende System *StreamReUse* , das von Dipl.-Inf. Andreas Benzing erstellt wurde. Die negativen Einflüsse auf die Latenz zwischen dem neuen Broker und dem anfragenden Client werden im Rahmen des aktuellen Auswahlverfahrens in *StreamReUse* vernachlässigt. Durch eine Modifikation des Auswahlalgorithmus kann die Wahl des Replikationspunktes geschickter erfolgen und somit wären diese negativen Einflüsse minimierbar. Das System könnte die Daten effektiver bereitstellen. Die allgemeine Lösung dieser Problematik ist die zentrale Aufgabe dieser Abschlussarbeit.
Neben der Optimierung der Replikationsstrategie wird des Weiteren auch das Routing der Clientanfragen analysiert und falls möglich optimiert. Zusätzlich können durch ein entsprechendes Anfragerouting Synergieeffekte erzielt werden, die in einer zusätzlichen Steigerung der Dienstgüte resultieren.

Abstract

The increasing ammount of Data Produces new challenges for existing network structures and systems. Additional Global Sensor Networks have to handle aN increasing number of clients which request data from the network. Various measures can increase the availability and optimize the usage of the resources. This systems enhances availability by the usage of datareplication from high-traffic hotspots on a non fully engaged broker with free resources. The difficulty is to select selection where to repicated. This selection can be done considering several aspects. The starting point for the practical implementation of the theories aquired by this paper is the existing system *StreamReUse* developed by Dipl.-Inf. Andreas Benzing. The effects on the latency between the repliacation and the client are currently neglected. These negativ effects can be minimized by a clever replicationpoint choice. Solving this problem in general is the goal part of this thesis.
In addition the routing of requests within global sensornetworks will analyzed and optimized. As fall-out, synergy effects could be realized to improve the performance of replication.

Danksagung

Die Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* entstand in enger Zusammenarbeit mit Hr. Dipl.-Inf. Andreas Benzing. Ihm gilt mein besonderer Dank für die vielseitige Unterstützung, seine Geduld und sein außerordentliches Engagement.

Des weiteren möchte ich Hr. Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel danken für die Ermöglichung der Arbeit.

Abschließend möchte ich bei allen Personen, die mich durch die Korrekturlesung und Inspiration unterstützt und somit diese Arbeit mit ermöglicht haben, bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation der Arbeit

Unsere Umwelt ist einer der stärksten und zugleich der unbeeinflussbarste Einflussfaktor auf unser tägliches Leben. Durch Wettervorhersagen versuchen wir diesen Einflussfaktor, den wir nicht beherrschen können, möglichst genau vorherzusagen und somit für uns effizient nutzbar zu machen. Wirtschaftliche Entscheidungen über die Platzierung von Produktionsstandorten oder eine mögliche Nutzung der lokalen Phänomene zur Energiegewinnung hängen stark von den erstellten Wetterprognosen ab. Durch die Erfassung der Wetterphänomene mittels einer stetig wachsenden Anzahl an Sensoren, versuchen wir möglichst viele Daten, die diesen entscheidenden Einflussfaktor betreffen, zu sammeln, um die Vorhersagen weiter zu verfeinern. Durch die Beobachtung und Dokumentation globaler Phänomene gelingt es neue Erkenntnisse und Einblicke in unsere Umwelt zu erhalten und die Entstehung der Phänomene besser zu verstehen.

Globale Sensornetzwerke sind bei der Datenerfassung und Verbreitung der nächste Schritt. Sie sollen es uns ermöglichen die Vielzahl der Daten der weltweit verteilten Sensoren in ein gemeinsames Netzwerk zu bündeln. Die Herausforderung für die Sensornetzwerktechnologie ist dabei die Bereitstellung von Daten in Echtzeit über einheitlich definierte Schnittstellen. Im Rahmen des Forschungsprojektes *Global sensor grid* der Forschungsgruppe *SimTech* an der *Universität Stuttgart* entsteht ein brokerbasiertes System zur Verwaltung all dieser Daten. Ein Teil der Grundlagen für diese Arbeit ist bereits mit den Veröffentlichungen von Dipl.-Inf. Andreas Benzing geschaffen worden. Diese behandeln zum einen die Indexierung und Wiederverwendbarkeit der übermittelten Daten[BKR11] sowie die mögliche Nutzung von Predictortechnologie zur effektiveren Verwendung der verfügbaren Ressourcen[BKVR10].

1.2. Zielsetzung der Arbeit

Die Diplomabschlussarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* basiert teilweise auf den zuvor genannten Forschungsergebnissen. Die Ergebnisse sind im Rahmen des Forschungsprojektes *Global sensor grid* als Java-Source-Codes mit dem Arbeitstitel *StreamReUse* umgesetzt. Innerhalb dieser wissenschaftlichen Ausarbeitung wird die Replikationsstrategie von globalen Sensornetzwerken analysiert und wenn möglich optimiert. Als Replikation wird bei diesem Netzwerksystem nicht das aus dem Bereich der Datenbanken bekannte, einmalige Übertragen von Daten auf ein Zielsystem bezeichnet, sondern die Etablierung eines Datenstroms, über den die ständig aktualisierten Daten direkt an den

Zielbroker der Replikation übertragen werden. Die mit den replizierten Daten verknüpften Anfragen werden ab dem Zeitpunkt der ersten Datenübertragung von dem Replikationspunkt bedient. Der Ursprungsbroker wird somit entlastet, da er die erforderlichen Daten lediglich einmal an das Replikationszielsystem übertragen muss und dennoch eine Vielzahl von ursprünglich an ihn gestellten Anfragen bedient werden können. Nachteilhaft wirkt sich eine Replikation für die anfragenden Clients aus.

Basierend auf der Annahme, dass innerhalb eines Netzwerkes das assoziierte Underlay stets den optimalen Routingpfad wählt, ist eine zusätzliche Verzögerung der Nachrichten als unvermeidbarer Begleitumstand anzunehmen. Bei der Auswahl des Zielsystems der Replikation kann durch geschickte Auswahl diese zusätzliche Verzögerung minimiert werden. In dem System *StreamReUse*, das als Grundlage für die spätere Umsetztung der theoretischen Lösungskonzepte dient, werden bei der Auswahl des Zielbrokers nur die direkt mit einem Broker assoziierten Nachbarknoten berücksichtigt. Die Auswahl unter den möglichen Replikationpunkten ist dabei lediglich von den verfügbaren Ressourcen und einer zufälligen Reihenfolge der gestellten Anfragen abhängig. Die Optimierung der Auswahl des Replikationszielbrokers, bezüglich der Minimierung der zusätzlichen Latenz, ist die zentrale Aufgabenstellung dieser Diplomarbeit.

Neben der zentralen Aufgabenstellung wird ebenfalls das Routing einer Anfrage analysiert und gegebenenfalls, basierend auf den neuen Anforderungen an die Replikationsstrategie, angepasst. Ziel ist die Nutzung der Query-Indizes für das Anfragerouting. Als Synergieeffekt resultiert aus dem alternativen Routing zusätzliche Replikationsmöglichkeiten, außer den direkten Nachbarknoten, die zur Auswahl angeboten werden.

1.3. Aufbau der Arbeit

Im Rahmen dieser Diplomarbeit wird, entsprechend der zentralen Aufgabenstellung, eine Replikationsstrategie für globale Sensornetzwerke entwickelt, die die Auswahl des Replikationspunktes bezüglich der Dienstgüte der Latenz optimiert. Als Grundlage für die theoretischen Lösungsansätze der Optimierungsmaßnahmen dienen dabei die in Kapitel 2 behandelten Arbeiten. Die dort erwähnten wissenschaftlichen Ausarbeitungen behandeln zum einen eine Auswahl verschiedener, bereits entwickelter, globaler Sensornetzwerke, zum anderen Optimierungsverfahren aus dem Bereich der Theoretischen Informatik, die die Ermittlung der kürzesten Wege innerhalb eines gewichteten Graphen betrachten. Basierend hierauf wird innerhalb des Kapitels 5 die Problemstellung der Optimierung der Datenströme theoretisch analysiert und geeignete Maßnahmen entwickelt.

Als erste praktische Maßnahme der Diplomabschlussarbeit wird nach der theoretischen Lösung der Problemstellung eine Evaluation des bestehenden Systems mittels eines Lasttests durchgeführt, um abschließend eine quantitative Aussage über den effektiven Nutzenzuwachs der Optimierung treffen zu können. Als erster Schritt wird hierfür ein Sensornetzwerk möglichst realitätsnah simuliert. Um das reale System möglichst gut abbilden zu können und die Komplexität dennoch einschränken zu können, müssen für die Simulation verschiedene Annahmen getroffen werden. Diese Annahmen und weitere Grundlagen, die

für das Verständnis der Arbeit notwendig sind, werden im Kapitel 4 beschrieben. Vor der abschließenden Evaluation in Kapitel 7 wird in dem Kapitel 6 die Umsetzung der theoretisch erarbeiteten Lösungskonzepte beschrieben. Das System *StreamReUse* dient hierfür als Grundlage. Einzelne Optimerungsmaßnahmen für einige Teilkomponenten des Systems werden an verschiedenen Stellen innerhalb der Ausarbeitung erläutert. Die in der abschließenden Evaluation des Systems *StreamReUse Plus* erfassten Kennzahlen werden in dem Kapitel 7 visualisiert und die Effekte der Modifikationen gegenüber dem Ursprungssystem anschaulich dargestellt. In dem abschließenden Kapitel 8 werden die aus der Evaluation erhaltenen Daten mit den Referenzdaten des Ursprungssystems *StreamReUse* verglichen und der mögliche Zusatznutzen aus den Optimierungsmaßnahmen der Replikationsstrategie des neuen Netzwerksystems erörtert. Zusätzlich wird in Kaptiel 8 ein Ausblick auf mögliche Folgearbeiten gegeben.

2. Verwandte Arbeiten

Die im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* entwickelten Konzepte thematisieren die Optimierung von Datenströmen in brokerbasierten globalen Sensornetzwerken. Als Beispielsystem wird dabei das Netzwerksysstem *StreamReUse* verwendent, das von Dipl.-Inf. Andreas Benzing bereitgestellt wird. *StreamReUse* realisiert dabei unter anderem Konzepte, die in der Ausarbeitung *Efficient support for multi-resolution queries in global sensor networks*[BKR11] beschrieben sind. In Anlehnung an den Arbeitstitel des aus dieser Ausarbeitung entstandenen Systems wurde der Projektname *StreamReUse Plus* als Arbeitstitel für die praktische Umsetzung der theoretisch erarbeiteten Lösungsansätze, die in dem Kapitel 5 erläutert werden, abgeleitet. Basis für die konzeptionelle Lösungsentwicklung sind die bereits in anderen globalen Sensornetzwerken realisierten theoretischen Lösungsansätze oder validen Lösungsansätze aus anderen Bereichen der Informatik, die eine ähnliche Problemstellung thematisieren. In dem Abschnitt 2.1 werden diese verwandten Systeme anaylsiert und geprüft, ob sie für die Lösung anwendbar sind. Des Weiteren werden in dem Kapitel 2.2 die Arbeiten aus dem Feld der Theoretischen Informatik erläutert, die für die Lösung der Problemstellung angewendet werden. Diese beschäftigen sich mit dem Finden von minimalen Pfaden innerhalb eines gewichteten Graphen.

2.1. Sensornetzwerke

Globale Sensornetzwerke werden aktuell vor allem im Bereich der Beobachtung von Wetterphänomenen eingesetzt. Durch die Nutzung von Smartphones und anderen Geräten als neue Datenquellen und die steigende Anzahl an außergewöhnlichen Wetterphänomenen nimmt sowohl die Datenmenge als auch das öffentliche Interesse rasant zu. Die Auswirkungen von extremen Wetterphänomenen sind global bemerkbar. Auf Grund der zentralen Bedeutung von Sensordaten für das Verständnis und die Verfeinerung der Prognosemöglichkeiten der Wetterphänomene steigt daher auch die Notwenigkeit der globalen Verfügbarkeit und der effektiven Echtzeitversorgung. Diese muss auch bei einer möglichen, zukünftig ansteigenden Datenmenge sichergestellt sein. Die hier vorgestellen verwandten Systeme setzen dabei unterschiedliche Schwerpunkte. Zunächst wird das System *Earth System Grid* analysiert, das vor allem die effektive Handhabung großer Datenmengen thematisiert. Die anschließenden untersuchten System *IrisNet*, *Hourglass* sowie *SplitStream* und *Scribe* sind neuere Systeme, deren Entwicklungsschwerpunkte auf der Echtzeitversorgung und der Anwendbarkeit in multiplen Bereichen liegen.

Earth System Grid ist in Zusammenarbeit mit dem amerikanischen Energieministerium entwickelt worden. Ziel der Entwicklung war die Schaffung eines effektiven Systems zur Verwaltung und Bereitstellung von archivierten Sensordaten. Das stetig ansteigende Datenvolumen, das aktuell bereits über 100 Terabyte umfasst[BBB+05] und ein zukünftig erwartetes Datenvolumen von mehreren Petabyte, war der limitierende Faktor für das Wachstum der bis dahin verfügbaren Systeme. Durch eine überarbeitete architektonische Struktur und die Umsetzung derer in *Earth System Grid* wird diese Problemstellung effizient gelöst, sodass auch bei dem zukünftig zu erwartendem Datenvolumen die Daten weiterhin effektiv bereitgestellt werden können. Der Zugriff erfolgt mittels eines Webinterfaces[ARU] oder über eine Schnittstelle im Internet. Das Interface ermöglicht die Suche nach den gewünschten Datensätzen und überträgt diese auf den anfragenden Client.

Im Gegensatz zu *StreamReUse Plus* ist bei *Earth System Grid* das Datenstreaming zwischen dem Broker, der die Daten bereitstellt und dem anfragenden Client als nicht zeitkritisch berücksichtigt, da hierbei lediglich auf bereits archivierte Daten zugegriffen werden kann und somit keine Echtzeitversorgung notwendig ist. Die Latenz zwischen Broker und Client ist somit nicht relevant und wurde daher auch nicht als Optimierungsfaktor bei der Entwicklung mit einbezogen. Die sichere Übertragung der angeforderten Datenpakete aus der Datenbank und eine effektive Verwaltungsmöglichkeit der zu erwarten Datenmengen waren die entscheidenden Kriterien.

Auch bei der Problemstellung, der Anpassung des Anfrageroutings, kann auf das System *Earth System Grid* nicht referenziert werden. Begründet durch den Einsatz einer Datenbank als Grundlagentechnologie auf die über ein Webfrontend oder ein Webinterface direkt zugegriffen wird, ist kein zusätzliches Routing außer dem automatisch optimierten Underlay-Routing innerhalb der Netzwerkestruktur notwendig. Da dieses Underlay-Routing auch innerhalb des Systems *StreamReUse* und *StreamReUse Plus* als automatisch optimiert angenommen wird, kann aus dem System *Earth System Grid* keine Lösung für die Routingproblematik abgeleitet werden.

Eine weitergehende Analyse im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* wird daher als nicht sinnvoll erachtet, da in *Earth System Grid* weder die Möglichkeit einer Replikation eines Datenstroms besteht, noch ein Routing der Anfragen außerhalb der Datenbank durchgeführt werden muss.

IrisNet ist als weiteres System bei den Recherchen analysiert worden. Der Name *IrisNet* ist dabei ein Akronym für **I**nternet-scale **R**esource-**I**ntensive **S**ensore **Net**work Service[CGN+05]. Entwickelt wurde das System an der Carnegie Mellon University in Zusammenarbeit mit Intel Research Pittsburgh.

Ziel des Systems *IrisNet* ist, wie das Akronym bereits andeutet, die Unterstützung des Datenstreamings aus neuen Quellen. Diese Datenquellen übermitteln teilweise Datenvolumina, die auf Grund der Größe von den bisherigen Systemen nicht effektiv verwaltet werden können. Basierend auf der Annahme, dass das Datenvolumen in naher Zukunft rasant ansteigt, mussten die existenten Systeme angepasst werden, um diese zusätzliche Belastung effektiv verarbeiten zu können. *IrisNet* stellt hierfür eine Softwareplattform bereit, die auf bereits

existierenden Netzwerkstrukturen aufbaut. Der Anwender hat die Möglichkeit die von ihm bereitgestellten Daten effizienter über das bestehende Netzwerk an andere Anwender zu übertragen.

Zur Effizienzsteigerung trägt vor allem die Möglichkeit der Datenvorverarbeitung bei. Durch die Verwendung von XML-Dokumenten zur strukturellen Beschreibung des Systems können die datenvorverarbeitenden Zusatzmodule einfach durch das Hinzufügen eines neuen Eintrags in das System migriert werden. Das Datenvolumen kann somit bereits im Voraus gesenkt und das Netzwerk zusätzlich entlastet werden.

IrisNet realisiert eine datenbankzentralisierte Architektur. Die Replikation von besonders oft angefragten Daten übernimmt somit die Datenbank und es muss auch in diesem System, ebenso wie in *Earth System Grid*, keine gesonderte Replikationsstrategie umgesetzt werden. Durch die datenbankzentralisierte Systemarchitektur und die an einigen Stellen seit 2003 bereits veraltete Technologie von *IrisNet* ist auch dieses System nicht weiter für die Erarbeitung der Lösung der Problemstellung dieser Diplomarbeit relevant.

Hourglass ist an der Harvard University[SPL+04] entstanden. *Hourglass* nutz bestehende Internetverbindungen, um die Daten von den erzeugenden Datenquellen an die anfragenden Clients zu übertragen. Ziel des Systems ist der Aufbau einer robusten, effektiven Struktur zur Verwaltung und Übermittelung von Sensordaten. *Hourglass* zielt nicht nur auf Wettersensordaten ab, sondern unterstützt bereits eine Vielzahl von Sensoren aus unterschiedlichen Bereichen. Basierend auf einer engmaschigen Netzwerkstruktur bietet *Hourglass* als zusätzlichen Vorteil die Migrationsmöglichkeit einer Fülle von Vorverarbeitungsmöglichkeiten der erhaltenen Sensordaten.

Durch die große Anzahl der vorhandenen Datenvorverarbeitungsmodule und die Möglichkeit zur Eigendefinition migrierbarer vorverarbeitenden Prozesse wird die Vielschichtigkeit der Einsetzbarkeit von *Hourglass* deutlich. Die durch diese Vorverarbeitung ermöglichte Lastreduktion erhöht die Leistungsfähigkeit des Gesamtsystems, da nicht benötigte Daten bereits vor der Übertragung gefiltert werden oder beispielsweise eine vorgezogene Aggregation eine vollumfängliche Datenübermittlung von einem Broker an den anfragenden Client überflüssig macht.

Außer der Möglichkeit Vorverarbeitungsprozesse in das Netzwerk einzubinden war die sichere Übertragung der Daten an den anfragenden Client ein weiteres wichtiges Entwicklungsziel. Zur Realisierung dieses Anspruches, trotz der Möglichkeit von Verbindungsabbrüchen der eingebundenen Datenquellen zu dem Netzwerk, besteht, neben der direkten Datenübertragung in das Netzwerk, eine weitere Möglichkeit der Datenversorgung durch die Quellsysteme. Datenquellen, die Teil von *Hourglass* sind, können die von ihnen erzeugten Daten lokal vorhalten und bei einem späteren Verbindungsaufbau nachträglich einbringen. *Hourglass* übernimmt somit, neben der Vorverarbeitung und Übertragung der Daten, eine weitere Funktionalität. Die verzögert eingebrachten Daten werden durch das System koordiniert und korrekt in den Datenfluss zu dem Client integriert.

Datenquellen, die nicht Teil des Netzwerksystems sind, haben dennoch die Möglichkeit die von *Hourglass* geschaffenen Strukturen mit zu nutzen. Mittels definierter Schnittstellen kann eine Datenquelle oder umfangreichere Quellsysteme in das bestehende Netzwerk eingebunden werden, ohne dass die Notwendigkeit besteht, dass diese Quellen selbst ein Teil des bestehenden *Hourglass* Netzwerkes sind. Ein verzögertes Einbringen der Daten ist von diesen, nur über das Inteface angebunden Systemen, nicht möglich. Die architektonische Gestaltung ist, im Gegensatz zu den beiden vorausgehenden Systemen *IrisNet* und *Earth System Grid*, nicht datenbankzentralisiert und eine Verwendung im Rahmen der Erarbeitung eines theoretischen Lösungsansatzes daher sinnig. Neben der architektonischen Struktur ist die Ähnlichkeit zu dem System *StreamReUse* ein weiteres Merkmal, dass *Hourglass* für eine tiefergehende Analyse qualifiziert.

Das System *Hourglass* löst die Problematik des Anfrageroutings durch die Etablierung eines Datenquellenverzeichnisses. Das Verzeichnis wird dabei verteilt angelegt, um Ausfallsicherheit und Erreichbarkeit zu erhöhen. In diesem Verzeichnis sind alle Datenquellsysteme der Netzwerkstruktur gelistet. Bei dem Einbringen von neuen Anfragen in das System werden die passenden Datenquellen aus dem Verzeichnis ermittelt, mit der Anfrage verknüpft. Ein Routing der Anfrage wird dabei nicht durchgeführt. Für die Erarbeitung der theoretischen Lösung für die Anfrageroutingproblematik ist *Hourglass* daher nicht refernzierbar, da *StreamReUse Plus* weiterhin dezentral organisiert bleiben sollte.

Splitstream und **Scribe** basieren auf dem populären Ansatz eines Peer-to-Peer Netzwerks. Dies ist ein häufig, zum Aufbau des Overlay einer Netzwerktopologie, genutzter Ansatz. Hierbei wächst das Overlay dynamisch durch die Integration der anfragenden Knoten in die bereits bestehende Struktur[TKK+10]. Die neuen Knoten nehmen in dem Netzwerk eine Doppelrolle ein. Sie werden dabei sowohl als Datenquelle als auch als anfragenden Knoten eingebunden. Mittels eines Anreizsystems, das dem Anwender beispielsweise die angefragten Daten schneller bereitstellt, werden diese angehalten, einen Teil der ihnen zur Verfügung stehenden Ressourcen mit dem Netzwerk zu teilen und die erhaltenen Daten an andere anfragende Teilnehmer weiterzuleiten. Durch diese bilaterale Nutzung ist hier eine besonders gute Skalierbarkeit gegeben, da die verfügbaren Ressourcen in Relation zu den Knoten innerhalb des Netzwerkes stehen. Ein weiterer Vorzug dieser Overlaytechnologie ist die sehr gute Realisierungsmöglichkeit von Application-Level Multicast.

Die globalen Sensornetzwerke *SplitStream*[CDK+03] als auch das System *Scribe*[CDKR02] basieren auf dieser Technologie und nutzen, im Gegensatz zu den beiden Systemen *Earth System Grid* und *IrisNet*, keine Datenbanktechnologie. Die Peer-to-Peer Infrastruktur wird als Basis für die Etablierung einer Struktur über das Internet genutzt. Beide Systeme wurden an der Rice University in Houston in Zusammenarbeit mit der Microsoft Research Gruppe in Cambrige entwickelt und behandeln die Problematik der Fairness und effektiven Nutzung der vorhandenen Ressourcen in Peer-to-Peer Netzwerken. Fairness beschreibt innerhalb eines Peer-to-Peer Netzwerk die Einhaltung festgelegter Reglementierungen[WS09], die eine Balance zwischen den Anwendern eines Peer-to-Peer Netzwerkes herstellen. Beispiele für solche Reglementierungen sind ressourcenabhängige Datenversorgung oder Gleichverteilung. Da beide Systeme von den gleichen Institutionen entwickelt wurden und die gleiche

Zielintention haben, ist es besonders wichtig an dieser Stelle die Unterschiede herauszuarbeiten.

Scribe war die erste Entwicklung dieser Kooperation und basiert auf dem System Pastry. Das Problem bei *Scribe* ist die hohe Last auf den inneren Knoten. Bedingt durch den architektonischen Aufbau wird die zur Verfügung stehende Ausgangsbandbreite entsprechend der Fanout aufgeteilt. Der Fanout beschreibt die Anzahl der notwendigen Datenvervielfältigung um alle Anfragen versorgen zu können. Dies führt zu einer sehr ungleichmäßigen Belastung. Wurzelknoten dürfen in dem System *Scribe* nur Daten empfangen, während innere Knoten den Datenstrom lediglich weiterleiten dürfen.

Diese Problematik greift das System *SplitStream* auf. Ein Datenstrom wird, entsprechend dem Systemnamen, in *SplitStream* in mehrere Teildatenströme aufgesplittet[BKC+01]. Für jeden dieser Teildatenströme wird im nächsten Schritt ein eigenständiger, optimaler Verteilbaum aufgebaut, der diesen Teildatenstrom koordiniert. *SplitStream* vermeidet durch diese Technik die ungleichmäßige Verteilung der Datenlast, da für jeden Verteilbaum neu festgelegt wird, wie die Knoten innerhalb des Netzwerkes agieren müssen. Jeder Teilnehmer wird somit sowohl zur Weiterleitung von Daten als auch als Datenempfänger genutzt und die Last effizienter verteilt.

Peer-to-Peer basierte Netzwerkstrukturen, wie auch in der wissenschaftliche Ausarbeitung *Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints*[TKK+10] beschrieben, werden als Grundlage für die Entwicklung eines Konzepts zum Anfragerouting innerhalb des Systems *StreamReUse Plus* verwendet. Beispielsweise wird das Konzept der One-Hop-Lookups, das in der Ausarbeitung *Effcient Routing for Peer-to-Peer Overlays*[GLR04] erläutert wird, als Ausgangspunkt für die Multi-Hop Routingstrategien verwendet. *SplitStream* und *Scribe* sind daher weiterhin als Referenzendokumente sinnvoll und vor allem als Basis für die Anpassung der Routingstrategie nutzbar.

StreamReUse ist die letzte Arbeit aus dem Bereich der globalen Sensornetzwerke die innerhalb des Kapitels 2 analysiert wird. Dieses, innerhalb der bisherigen Ausführung bereits mehrfach genannte System, ist von Dipl.-Inf. Andreas Benzing an der Universität Stuttgart am *Institut für Parallele und Verteilte Systeme* entwickelt worden. Der innerhalb dieser Diplomabschlussarbeit verwendete Name *StreamReUse* ist dabei lediglich ein Arbeitstitel. In das System sind beispielsweise Erkenntnisse aus der wissenschaftlichen Ausarbeitung *Efficient support for multi-resolution queries in global sensor networks*[BKR11] eingeflossen. Die Ausarbeitung betrachtet vor allem die Möglichkeiten zur effektiven Verwendung der von den Brokern an die anfragenden Clients übertragenen Daten.

Das Ziel der Entwicklung ist die Schaffung einer Möglichkeit zur Entlastung des Gesamtsystems. Dabei wird beachtet, dass hierfür die Migration von zusätzlichen Vorverarbeitungsprozessen in die Netzwerkstruktur wie in dem System *IrisNet* nicht notwendig ist. Diese optionalen Teile einer Netzwerkstruktur können die entstehende Datenlast besonders effektiv senken, jedoch ist der Nutzen stark von den zu verarbeitenden Daten abhängig und würde nur bei einer Spezialisieurung des Netzwerkes auf bestimmte Daten zu einer effizienten Datenreduktion führen. Der Ansatz, den Dipl.-Inf. Andreas Benzing entwickelt

und erforscht, behandelt die mehrfache Nutzung der bereits übermittelten Daten. Daten, die innerhalb eines Queries nochmals übermittelt werden müssten, da sich die angefragten Bereiche überschneiden, können innerhalb von *StreamReUse* aus den bereits übermittelten Datenpaketen extrahiert und somit für eine mögliche weitere Anfrage bereitgestellt werden, ohne dass eine zusätzliche Belastung für das Gesamtsystems erzeugt wird.

Die Arbeit bildet die Basis für *StreamReUse Plus*. Das System *StreamReUse Plus* setzt dabei die theoretisch entwickelten Ansätze zur Optimierung der Datenstromreplikation und dem Anfragerouting um. *Efficient support for multi-resolution queries in global sensor networks*[BKR11] sowie das globale Sensornetzwerksystem *StreamReUse* kann somit nicht als Grundlage für die Entwicklung neuer theoretischen Lösungsansätze der im Kapitel 3 erläuterten Problematik angewendet werden. *StreamReUse* ist allerdings das Grundlagensystem, das durch die innerhalb des Kapitels 5 gefundenen Lösungen modifiziert wird, um die Effizenz der Modifikationen zu validieren. Durch eine Evaluation der beiden Systeme *StreamReUse* und *StreamReUse Plus* wird in dem Kapitel 7 das Potential der entwickelten Lösungen verdeutlicht. Im Folgenden wird daher sowohl innerhalb der Problemstellung sowie bei der Beschreibung der theoretischen und praktischen Lösung Bezug auf *StreamReUse* und die damit verbundenen Arbeiten von Dipl.-Inf. Andreas Benzing genommen.

2.2. Optimierungsalgorithmen

Neben bekannten und bereits realisierten Sensornetzwerksystemen die in dem vorangegangenen Abschnitt beschrieben wurden, sind auch Algorithmen aus dem thematischen Bereich der Theoretischen Informatik für die Lösung der in Kapitel 3 erläuterten Problematik relevant. Die, der Problemstellung der Optimierung des Anfrageroutings zu Grunde liegende Problematik ist Findung von minimalen Pfaden innerhalb eines Graphen bei dem die Verbindungen zwischen den Knotenpunkten unterschiedliche Gewichtungen haben, wird in der Theoretischen Informatik in verschieden Algorithmen abgehandelt. Die Auswahl des richtigen Algorithmus, in Abhängig von den Eigenschaften des Graphen, ist dabei für ein korrektes Ergebnis entscheidend.

Der innerhalb dieser Arbeit zu behandelnde Graph ist dabei ein Abbild der Struktur eines realen Netzwerkes, das durch den Netzwerksimulator *PeerSim*[MJJV09] etabliert wird. Die Gewichtungen der Kanten entspricht den Verzögerungen, die eine Nachricht bei der Übertragung zwischen zwei Knoten erfährt. Dieser Verzögerungswert wird als Latenz bezeichnet und variiert in Abhängigkeit von mehreren Einflussfaktoren zwischen den Knoten eines Netzwerkes und zudem in der Zeit. Die zeitabhängige Varianz wurde abstrahiert, da auch bei einem realen System die Optimierung auf einer Momentaufnahme des Netzwerkes beruhen muss. Würde die Optimierung nicht auf Basis einer Momentaufnahme durchgeführt, müsste bei einem hochfluktuativen System eine stetige Reoptimierung durchgeführt werden, da in der für die Optimierung benötigten Zeit die zu optimierenden Parameter bereits Veränderungen erfahren haben könnten. Da es somit für die Durchführung des Optimierungsalgorithmus nicht relevant ist, ob die Daten, die als Grundlagen für diese

angenommen werden einer aktuellen Messung der Latenz oder als konstant angenommen Werte sind, ist diese Annahme valide.

Eine weiter Annahme ist, dass sämtliche Latenzen einen positiven Wert besitzen. Diese Annahme kann mittels eines indirekten Beweises belegt werden. Angenommen, eine negative Latenz wäre möglich, so würde dies bedeuten, dass der Empfänger die Nachricht bereits erhält bevor der Absender diese versendet hat.

Diese Möglichkeit würde zwangsläufig zu einem Paradoxon führen und ist daher auszuschließen. Eine Verzögerung von null ist ebenfalls nicht möglich. Auch hier kann dies mittels eines indirekten Beweis belegt werden. Die Geschwindkeit begründet sich auf der physikalischen Grundgleichung: $Geschwindigkeit = \frac{Strecke}{Zeit}$. Durch eine Umformung ergibt sich somit, dass bei einer Zeit von null die zurückgelegte Strecke ebenfalls null ist, was innerhalb unserer definierten Struktur nicht zulässig ist. Aus den beiden geführten Gegenbeweisen ist herzuleiten, dass die Zeit, die eine Nachricht für die Strecke zwischen zwei Knoten eines Netzwerkes benötigt, als ein Element der natürlichen Zahlen anzunehmen ist.

Aus den hergeleiteten Eigenschaften unseres netzwerkrepräsentierenden Graphen ist ableitbar, dass für die Lösung der Problematik der Findung des kürzesten Pfades der Algorithmus von Dijkstra effizient ist. Die Umsetzung und theoretische Grundlage des Algorithmus ist im Anhang A.4 beigefügt.

3. Problemstellung

Die Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* behandelt zwei thematische Schwerpunkte. Beide Aufgabenstellungen sind dabei gleichwertig.

Einerseits wird das bestehende System zur Datenstromverwaltung in globalen Sensornetzwerken für verschiedene Szenarien optimiert. Dabei wird die Effizienz der Nutzung der Ressourcen im System optimiert, wobei auch die Dienstgüte in Form von Latenz mit betrachtet wird. Bei einer Replikation auf einen anderen Knoten wird somit zukünftig nicht nur auf ausreichend freie Ressourcen geachtet werden um die problemverursachenden Anfragen an den Replikationspunkt zu übergeben, sondern als weiterer Aspekt bei der Auswahl des Replikationspunktes eine Optimierung der Auswahl des Zielbrokers bezüglich der Durchschnittslatenz zu dem anfragenden Clients durchgeführt. Die negativen Effekte, die mit einer Replikation verknüpft sind, werden somit minimiert, ohne die Gesamtleistungsfähigkeit des Systems negativ zu beeinflussen. Es wird sowohl bei jeder neu im Netzwerk angelegten Replikation als auch bei der Verschiebung zu einer existenten Replik stets analysiert, welcher der möglichen Replikationspunkte die geringste Druchschnittslatenz für alle Queries die zukünftig durch die Replik versorgt werden sollen aufweist. Die Möglichkeit des Systems, Datenströme in mehrere Teil aufzuspalten, wird aktiv genutzt, um eine ausgeglichene Lastverteilung zu erreichen.

Des Weiteren umfasst die Aufgabenstellung die Anpassung des Anfrageroutings. Hierfür werden nun die Query-Indizes verstärkt genutzt.

Beispielhaft wird dies am System *StreamReUse* umgesetzt. Der in dieser Diplomarbeit verwendete Name *StreamReUse* ist dabei lediglich ein Arbeitstitel für das durch Dipl.-Inf. Andreas Benzing bereitgestellte System. Entsprechend ist der Titel des Ergebnis der Diplomarbeit *StreamReUse Plus*. Das in dem System rudimentär implementierte Routing wird modifiziert. Als Synergieeffekt aus der Modifikation des Routingalgorithmus und der somit vermehrten Auswahl der möglichen Replikationspunkte kann die Platzierung von neuen Replikationen weiter optimiert werden. Nachdem die Anforderungen herausgearbeitet wurden, wird im Folgenden eine optimale Lösung für jede der Teilaufgaben zunächst theoretisch und anschließend praktisch erarbeitet. Die abschließende Evaluation verdeutlicht, wie gut die Anforderungen umgesetzt werden.

4. Grundlagen und Definitionen

Grundlage für diese Arbeit ist das bereits mehrfach erwähnte Netzwerksystem *StreamReUse*. Dieses System ist das Ergebnis der bereits zuvor referenzierten Arbeiten von Dipl.-Inf. Andreas Benzing. In der wissenschaftlichen Ausarbeitung *Efficient support for multi-resolution queries in global sensor networks* wird eine Indizierungsstruktur erarbeitet die beispielsweise auch innerhalb dieser Diplomarbeit Anwendung findet. Da diese Struktur eine wesentliche Komponente für die Anpassungen in dem Anfragerouting darstellt wird diese innerhalb des Abschnittes 4.2 weitergehend beschrieben. Als weiterer Fachbegriff wird in dem Abschnitt 4.3 der Terminus Replikation innerhalb von globalen Sensornetzwerken präzisiert. Zunächst werden jedoch einige der Abstraktionen und Annahmen in dem Abschnitt 4.1 erläutert und validiert.

4.1. Abstraktionen und Annahmen

Im folgenden Abschnitt werden die Annahmen sowie die notwendigen Abstraktionen für das Modell, die für die Lösung der Problemstellung getroffen wurden, erläutert und begründet.

4.1.1. Annahmen

- *Ein Netzwerk ist optimal vor dem Einfügen einer neuen Anfrage.*
  Da ein Netzwerk bei seiner Einrichtung ohne eine Anfrage ist, ist der Zustand hier optimal. Wird eine neue Anfrage eingefügt, wird danach eine Optimierung vorgenommen. Anschließend ist das Netzwerk definitionsgemäß wieder als optimal zu bezeichnen. Somit kann angenommen werden, dass das Netzwerk nach dem Einfügen jeder Anfrage ebenso optimal ist, wie ohne diese Anfrage. Das Netzwerk kann daher zu jedem Zeitpunkt und unabhängig von der Anzahl der darin vorhanden Anfragen als optimal angenommen werden.
- *Eine neue Anfrage wird immer von dem Broker mit der geringsten Latenz beantwortet.*
  Ein anfragender Client kann seine Anfrage an mehrere Broker, die Teil des Systems sind, übertragen. Der Broker mit der geringsten Latenz erhält zuerst die Nachricht, da die Nachrichtenlaufzeit hier am geringsten ist. Andere Broker, die die Nachricht erst verzögert empfangen, können entweder feststellen, dass diese Anfrage bereits in das Netzwerk eingebracht wurde und verwerfen diese oder leiten sie weiter. Würden die Broker die Nachricht nicht verwerfen, würde die Anfrage mehrfach beantwortet und der anfragende Client könnte als Reaktion diese Anfragen aus dem Netzwerk

entfernen. Die resultierende Endnetzlast würde somit gleich sein. Es könnte allerdings zu einem Overhead kommen, der das Gesamtresultat verfälschen könnte. Anschaulich wird dies in dem im Anhang A.1 eingefügten Abbildung dargestellten. Da das Ergebnis unter Vernachlässigung des möglichen Overheads gleich ist, kann diese Annahme als valide angenommen werden.

- *Ein Broker kann nur eine endliche Anzahl an Anfragen gleichzeitig verarbeiten.*
  Da die zur Verfügung stehenden Ressourcen eines Brokers endlich sind, kann durch diese auch nur eine endliche Anzahl an Anfragen gleichzeitig verarbeitet werden. Um weitere Anfragen innerhalb des gleichen Zeitraums verarbeiten zu können, müssen zusätzliche Ressourcen bereitgestellt werden. Diese kann entweder durch die bereitstellung zusätzliche Ressourcen, beispielsweise mehr Speicherplatz oder Arbeitsspeicher, direkt im Broker geschehen, oder durch die Nutzung von Ressourcen anderer Broker, also einer Replikation der Anfrage.
- *Die direkte Verbindung zwischen Broker und Client ist minimal.*
  Jede Verbindung innerhalb eines bekannten Netzwerkes wird durch das Unterlay automatisch optimal gewählt und weist die minimalste Latenz auf. Die Verbindungen innerhalb des Netzwerkes erfüllen dabei die bindende Dreiecksungleichung, die in der Abbildund 4.1 dargestellt wird. Jede direkte Verbindung innerhalb des Netzwerkes weist dort eine geringere Latenz auf als eine Verbindung über einen anderen Knoten des Netzwerkes.

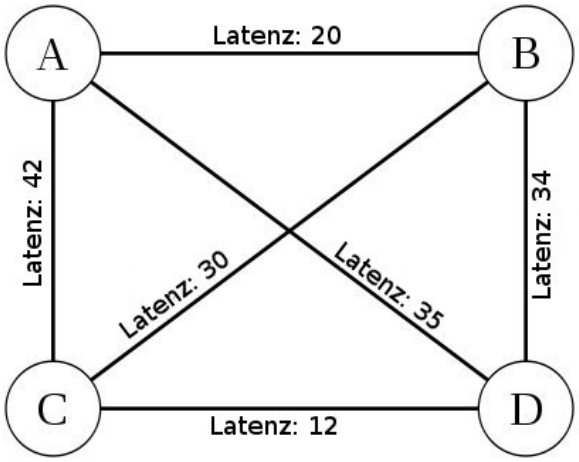

Abbildung 4.1.: Bindende Dreiecksungleichung für Netzwerke

4.1.2. Abstraktionen

- *Das bestehende Netzwerk ist ausfallsicher.*
  Die ausfallsichere Konstruktion der Hardware, sowie die Entwicklung von eventuell notwendigen Algorithmen, falls es doch zu einem Ausfall eines Teils oder des gesamten Netzwerkes kommen sollte, ist nicht Teil dieser Arbeit. Da diese Thematik getrennt behandelt werden kann und durch die Annahme der Ausfallsicherheit die Arbeit stark vereinfacht wird, ohne einen Einfluss auf das Ergebnis zu haben, kann diese hier getroffen werden.
- *Die Latenz zwischen den Brokern und zu allen anfragenden Clients ist konstant.*
  Die Latenz zwischen einem Broker und einem anderen Knoten ist durch das System nicht beeinflussbar. Eine Varianz ist in Realität als sicher anzunehmen, da die Latenz von anderen Tätigkeiten des Knoten im Netzwerk mit beeinflusst wird. Es kann jedoch angenommen werden, dass zum Replikationszeitpunkt eine Momentaufnahme des Netzwerkes mit den aktuellen Latenzwerten gemacht wird. Auf dieser Momentaufnahme, die ebenso nur konstante Latenzwerte enthält, basiert die Optimierung. Würde die Optimierung basierend auf den realen Werte durchgeführt, muss in einem hochfluktuativen Netzwerk eine stänidge Reoptimierung erfolgen, da sich die der Optimierung zu Grund liegenden Parameter verändert haben könnten. Die Annahme der konstanten Latenzwerte ist daher nur eine geringfügige Abstraktion, die lediglich die Funktionalität des Erzeugens der Momentaufnahme und Messung der aktuellen Latenzen nicht berücksichtigt.

4.2. Indexierung

Um ein effektives Routing der Anfragen zu ermöglichen, ist eine sinnvolle Verteilung der Broker und eine Aufteilung der von diesen abgedeckten Welt notwendig. Die Verteilung der Broker ist durch den Anwender nur bedingt beeinflussbar und wird daher auch im Rahmen dieser Arbeit als unveränderlich angenommen. Die Aufteilung der Welt und die Assoziierung der Teilgebiete mit dem jeweiligen Broker kann direkt beeinflusst werden. Das System *StreamReUse* unterteilt die simulierte Welt in Quadrate. Die Grenzen dieser Quadrate sind, unter Berücksichtigung des späteren Anwendungsgebietes, durch Längen- und Breitengrade definiert. Eine Übertragung auf die reale Welt ist somit einfach möglich, da keine weiteren Anpassungen notwendig sind. Ausgangspunkt für die Unterteilung ist die gesamte Welt bzw. das gesamte zu erfassenden Gebiet, welches mit Epsilon bezeichnet wird (vergleiche hierzu Abbildung 4.2 Schritt 1). Die anschließend beschriebenen Unterteilungsschritte werden jeweils wechselweise durchgeführt. Die erste Unterteilung ist eine meridiane Teilung (Abbildung 4.2 Schritt 2). Dies bedeutet, dass eine gedachte Trennlinie eingezogen wird, die das zu erfassende Gebiet in eine östliches und ein westliches Teilgebiet unterteilt. Alle Broker, die mit einem Bereich innerhalb des östlichen Gebietes assoziiert sind, werden mit *Eins* bezeichnet und entsprechend die konträren Broker mit *Null*. Der nächste Schritt bei der Reduktion der Größe des zu erfassenden Bereichs ist eine äquatoriale Teilung

(Abbildung 4.2 Schritt 3). Als Konsequenz dieser Teilung ist die bekannte Welt nun in vier Areale unterteilt. Zwei der Gebiete sind aktuell mit *Null* bezeichnet und zwei weitere mit *Eins*. Zusätzlich zu der bereits vorhandenen Bezeichnung müssen, als Resultat des aktuellen Teilungsschrittes, den Brokern nördlich der aktuellen Trennlinie eine *Eins* und den Brokern südlich eine *Null* angehängt werden. Bei jeder weiteren Unterteilung müssen die weiteren Trennungen simultan in allen Bereichen erfolgen (Abbildung 4.2 Schritt 4).

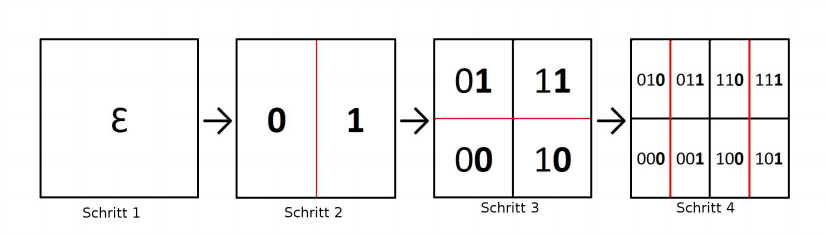

Abbildung 4.2.: Schematische Darstellung des Indexierungsalgorihtmus

Der aus diesem Indexierungsverfahren resultierende Ausdruck zur eindeutigen Beschreibung eines Brokers wird als *DZ-Ausdruck* bezeichnet. Weitere Dimensionen können durch zusätzliche Stellen des resultierenden *DZ-Ausdrucks* dargestellt werden. Die resultierende Indexstruktur ist eine der Grundlagen für die Anpassungen des Routings einer Anfrage innerhalb des Netzwerkes. Ein mögliches Anfragerouting kann somit auf Basis des *DZ-Ausdrucks* oder auf Grundlage der mit den Brokern assoziierten Längen- und Breitengraden erfolgen.

Soll das Routing mittels des *DZ-Ausdruck* durchgeführt werden, ist jedoch zu beachten, dass eine Verschiebung der Anfrage abhängig von dem Ausgangspunkt ist. Anschaulich wird dies in der Abbildung 4.3 dargestellt. Dort ist exemplarisch die jeweilige durchzuführenden Verschiebung eingetragen. Ziel der Anfrage ist der Broker 17 (DZ-Ausdruck = 010001). Der erste Unterschied ist an der zweiten Stelle des *DZ-Ausdrucks*. Ausgehend von dem Broker 0 (DZ-Ausdruck = 000000) muss eine Verschiebung nach unten zu Broker 21 (DZ-Ausdruck = 010101) erfolgen, um den kürzesten Pfad zu ermitteln. Ist der Ausgangsbroker 5 (DZ-Ausdruck = 000101) ist der nächste Routingschritt 16 (DZ-Ausdruck = 010000) und somit eine Verschiebung zu dem oberen Nachbarknoten. Diese und weitere Varianzen erhöhen zusätzlich die Komplexität eines *DZ-Ausdruck* basiertes Routing.

Alternativ kann das Routing basierend auf den mit dem Knoten assozierten Längen- und Breitengrad erfolgen. Hierfür müssen die jeweiligen Distanzen zwischen den verfügbaren nächsten Routingpunkten und dem Zielknoten errechnet werden. Aus der verfügbaren Menge wird anschließend der Knoten mit der geringsten Distanz als nächster Schritt gewählt. Bei diesem Routingverfahren müssen ebenfalls die umschließenden Grenzen beachtet werden, jedoch keine inneren Grenze, wie es bei einem *DZ-Ausdruck* basierten Routing notwendig ist.

Auf Grund der geringeren Komplexität ist daher im Rahmen dieser Diplomarbeit das Routing, basierend auf dem mit den Broker verknüpften Längen- und Breitengraden, implementiert.

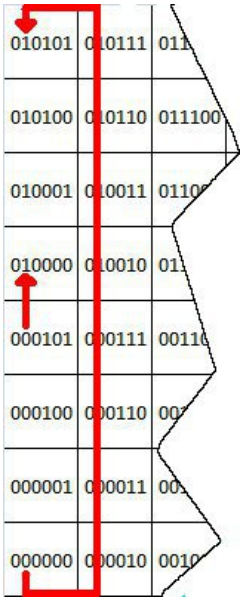

Abbildung 4.3.: Darstellung der Routingbewegungen

4.3. Replikation

Allgemein ist mit dem Begriff der Replikation die Vervielfältigung und Übertragung von Daten verknüpft[PGVA08]. Vor allem mit dem Bereich der Datenbanken[Adl11] ist der Ausdruck der Replikation in der Informationstechnologie eng verbunden. Datenreplikationen werden dort zur Steigerung der Erreichbarkeit und Ausfallsicherheit proaktiv durch das Datenbankensystem selbständig angelegt. Bei einer Replikation in diesen Systemen werden die Daten einmalig von dem replizierenden Datenbankknoten auf ein ausgewähltes Zielsystem übertragen. Eine Veränderung der Daten auf dem ursprünglichen Knoten bewirkt ein Update der replizierten Daten. Die Verwendung des Terminus Replikation, der innerhalb von brokerbasierten globalen Sensornetzwerken angewendet wird, unterscheidet sich von dieser Definition des Begriffes Replikation nicht grundlegend, weist jedoch einige systemspezifische Besonderheiten auf.

Ein erster Unterschied ist der Replikationszeitpunt. Eine Replikation wird innerhalb eines globalen Sensornetzwerkes nicht proaktiv, sondern rein reaktiv angelegt. Dies bedeutet, dass bei einer Überlastung einer Datenquelle durch eine Anfrage, deren Datenvolumen die verfügbare Bandbreite übersteigt, diese Anfrage auf einen Knoten innerhalb des Netzwerkes weitergeleitet wird, der über ausreichend freie Ressourcen verfügt und zukünfitg von ihm die Daten erhält. Die würde aber zu keiner Entlastung führen, da weiterhin das gleiche Datenvolumen von dem ausgelasteten Broker übertragen werden muss. Vor einer Replikation wird von dem überlasteten Knoten daher geprüft, welche weiteren Anfragen von ihm derzeit noch bedient werden und ob deren angefragte Daten gleich oder ähnlich der zu replizierenden sind. Ähnlich sind Anfragen die das gleiche Gebiet oder ein Teilgebiet

dessen anfragen und eventuell in der zeitlichen oder räumlichen Dimension variieren. Diese Anfragen werden als inkludierte Anfragen bezeichnet, da die von ihnen angeforderten Daten vollständig aus dem zur Replikation anstehendem Datenpaket ableitbar sind. Die Daten müssen, bedingt durch die Inklusion, von dem überlasteten Knoten nur einmalig an das Replikationsziel übertragen werden. Das Replikationsziel, welches die Daten von dem replizierenden Netzwerkknoten erhält, vervielfältigt den Datenstrom und überträgt die Daten an die anfragenden Clientknoten. Die Anzahl der anfragenden Knoten wird dabei als Fanout bezeichnet und definiert die Quantität, die der empfangene Datenstrom durch das Replikationszielsystem vervielfältigt werden muss. Der replizierende Knoten wird somit entlastet, da er mehrere Datenströme bündel kann und lediglich einmal die Bandbreite zur Übertragung der angefragten Informationen benötigt. Der zuvor überlastete Knoten wird somit im Rahmen der Möglichkeiten entlastet.

Indirekt wurde bei den vorangegangenen Ausführungen bereits die zweite Besonderheit erwähnt. Während innerhalb von Datenbanken und ähnlichen Systemen die Daten nach einer erfolgreichen Replikation selten geändert werden und hierfür eine Updatefunktion ausreichend ist, muss innerhalb von globalen Sensornetzwerken ein beständiger Datenstrom etabliert werden, um die hohe Datenfluktuation abbilden zu können. Dies ist vergleichbar mit einem kontinuierlichen Aufruf der Updatefunktion.

Das Replikationsziel stellt dauerhaft die zum Zeitpunkt der Replikationsanfrage freien Ressourcen bereit und versorgt die umgeleiteten Anfragen des Replikationsquellknotens. Daher kann es auch von weniger frequentierten Brokern zu Replikationsanfragen kommen.

Als Ansatz für mögliche Optimierungsmaßnahmen wurde im Rahmen der Erstanalyse nicht die Etablierung des Datenstroms lokalisiert, sondern die Auswahl des Zielknotens der Datenreplikation sowie die Auswahl der zu replizierenden Anfragen. In einem ersten Optimierungsschritt wird zunächst die Auswahl des Replikationszielknotens für die Platzierung neuer Repliken angepasst. Die optimierte Auswahl der zu replizierenden Anfragen bezüglich der latenzoptimalen Gruppierung überstieg leider den Umfang dieser Diplomarbeit. Die Auswahlproblematik und deren Optimierung bezüglich des Replikationszielkonten wird in dem Kapitel 5 zunächst theoretisch und anschließend, wie in dem Kapitel 6 beschrieben, praktisch gelöst.

5. Theoretisches Lösungskonzept

Bevor die Problemstellung durch Modifikation des System *StreamReUse* praktisch realisiert wird, wird im folgenden Kapitel eine theoretische Lösung erarbeitet und begründet. Die Aufgabenstellung der Diplomarbeit umfasst zwei Problemstellungen. Zum einen die Optimierung der Routingstrategie, unter der Nutzung des Query-Indizes und zum anderen die Erarbeitung einer Routingstrategie, die auf dem bestehenden System aufbauen soll und zukünftig aber die Dienstgüte in Form von Latenz mit berücksichtigt.

Nach einer ersten Voranalyse wird zunächst das Anfragerouting optimiert, da durch die anschließende größere Anzahl an möglichen Replikationspunkten Synergieeffekte für die Optimierung der Replikationsstrategie realisiert werden. Im Abschnitt 5.1 werden daher zunächst drei verschiedene theoretische Konzepte zum möglichen Routing einer Anfrage erläutert. Der abschließende Vergleich und die Begründung der Entscheidung werden dann im darauf folgenden Abschnitt 5.1.2 erfolgen.

Der Abschnitt 4.3 erläutert die verschiedenen theoretischen Überlegungen, die für die Lösung der Anfragereplikationsproblematik notwendig sind. In dem Abschnitt 5.1.2 wird die abschließende Realisierungsentscheidung für die umzusetzende Anfragereplikation begründet.

Für ein besseres Verständnis der nachfolgenden Ausführungen und zur Präzisierung von verwendeten Fachbegriffen wurden bereits in dem Kapitel 4 einige der Grundlagentechnologien beschrieben sowie Fachbegriffe genauer spezifiziert, die sowohl im Rahmen der theoretischen als auch praktischen Lösungserarbeitung angewendet werden.

5.1. Anfragerouting

Grundlage für eine Anpassung im Anfragerouting ist, neben der bereits erläuterten Indexstruktur, vor allem eine Analyse der verwandten Systeme und den dort umgesetzten Anfrageroutingstrategien. Des Weiteren wird eine Analyse des bestehenden Routings in *StreamReUse* durchgeführt. *StreamReUse* wird ebenfalls als Grundlage untersucht, um mögliche Ansatzpunkte für eine Optimierung des Verfahrens zu finden.

Als erster Ansatzpunkt wird die Initialisierungsphase von *StreamReUse* untersucht, um die Struktur des Netzwerkes tiefergehend zu verstehen und den neuen Routingalgorithmus möglichst optimal an die etablierte Struktur anzupassen. Bei der Analyse wurde, wie bereits bei der Voranalyse vermutet, festgestellt, dass das mittels des verwendeten Netzwerksimulationswerkezeuges *PeerSim*[MJJV09] erzeugte Netzwerk eine lediglich eingeschränkt nutzbare

Struktur etabliert. In dem bestehenden System *StreamReUse* wird eine Netzwerkstruktur initialisiert, die lediglich die Datenquellsysteme beinhaltet. Diese dienen sowohl als möglicher Replikationspunkt und auch zur Verwaltung der Daten der ihnen zugeordneten Sensoren. Als zusätzliche Einschränkung wurde, neben dem nicht Initialisieren der Clientknoten, bei der Voranalyse festgestellt, dass keine Latenzen zwischen den Knoten abgebildet werden. Die Clientknoten sind eine essentielle Grundlage für die realitätsnahe Umsetzung eines Anfrageroutings.

Innerhalb des derzeit umgesetzten Systems wird die Routingproblematik auf Grund der fehlenden Voraussetzungen pragmatisch gelöst. Wird eine neue Anfrage in *StreamReUse* erzeugt, berechnet der Generator mittels der Grenzen des angefragten Gebiets, welcher Broker in dem Netzwerk die erforderlichen Daten bereitstellt. Im nächsten Schritt wird die Anfrage direkt an den errechneten Broker übertragen, der unmittelbar mit der Datenübermittlung an das Zielsystem beginnt.

Diese Strategie wird als *Direct Routing* bezeichnet, da das Zielsystem direkt angesprochen wird. Im Abschnitt 5.1.1 wird diese Strategie tiefergehend erläutert. Stehen dem Zielknoten nicht ausreichende Ressourcen für eine Bearbeitung zur Verfügung, kann er die Anfrage weiterleiten. Dieser, mit *Replikation* bezeichneter Vorgang, wird in dem Kapitel 4.3 theoretisch abgehandelt.

Das aktuelle in *StreamReUse* umgesetzte Routing bietet somit keine Grundlage für die theoretische Problemlösung des Anfrageroutings. Für die geplante Verbesserung werden daher alternative Ansätze in verwandten Systemen gesucht. Als weitere Analysemöglichkeiten werden außer den verwandten Systemen andere Arbeiten aus dem Bereich der Netzwerktechnologie untersucht.

5.1.1. Routingstrategien

Aus den erarbeiteten Grundlagen lassen sich verschiedene Routingstrategien ableiten. Drei der Strategien werden für die Modifikation von *StreamReUse* berücksichtigt. Diese Varianten werden im Anschluss näherer erläutert und deren Vor- sowie Nachteile dargelegt. Die abschließende Realisierungsentscheidung wird in dem Abschnitt 5.1.2 beschrieben und begründet. Analysiert werden die beiden Extrema *Single Hop Routing* und *Direct Routing* sowie eine abgeschwächte Variante, die als *Multi Hop Routing* bezeichnet wird. Die Bezeichnungen der Routingstrategien sind frei gewählt und eventuelle Überschneidungen mit bereits vorhandenen Technologien nicht beabsichtigt.

Alle Routingstrategien nutzen die eindeutige Positionierung jedes Brokers innerhalb der simulierten Welt mittels eines Längen- und eines Breitengrades, ähnlich der eindeutigen Bestimmbarkeit der Position eines Punktes innerhalb eines zweidimensionalen Raumes mittels des kartesischen Koordinatensystems. Alternativ kann für die Bestimmung auch der DZ-Ausdruck analysiert werden. Hierbei sind jedoch Besonderheiten zu berücksichtigen, die in dem Abschnitt 4.2 beschrieben werden. Mittels der eindeutigen Positionierung ist es jedem Broker möglich die Distanz zwischen dem aktuellen Knoten des Routingpfades sowie den möglichen weiteren Routingpunkten und dem Zielsystem zu bestimmen. Welche der

Knoten bei der Berechnung berücksichtigt werden und wie eine mögliche Einschränkung der Auswahl erfolgt, ist von der gewählten Strategie abhängig.

Single Hop Routing

Die erste Routingstrategie wird mit *single hop routing* bezeichnet. Bei dieser Strategie wird einen Anfrage stets um exakt einen Schritt näher an den definierten Zielknoten innerhalb eines Zyklus verschoben.

Diese Strategie kann zwei Ausprägungen annehmen. Ist die Distanz um die eine Anfrage näher an den Zielknoten gebracht werden kann fix und auch zu in einer späteren Phase des Netzwerkes nicht variierbar wird eine *strikte single hop routing* Strategie realisiert. Wird nach einem festgesetzten Zeitpunkt, einer festgelegten Anzahl an Routingvorgängen, einer Mindestanzahl von bekannten Knoten innerhalb des weiterleitenden Knotens oder bedingt durch andere Kriterien die Möglichkeit eines *direct routings* eingeräumt, ist eine *loose singel hop routing* Strategie umgesetzt. Die gewählten Bezeichnungen der Strategien finden ebenfalls in dieser Form lediglich im Rahmen dieser Diplomarbeit Anwendung.

Der nächste Routingpunkt ist bei dieser Strategie bedingt durch die Schrittweite immer ein Element aus der Menge der Nachbarknoten. Die Auswahl aus dieser Menge basiert, wie in der Überleitung bereits angemerkt, auf der errechneten Distanz jedes Knotens zu dem Zielsystem der Anfrage. Basierend auf dieser Distanz kann bereits eine Vorauswahl getroffen und die Auswahlmöglichkeiten auf zwei Broker reduziert werden. Diese Reduktion ist möglich, da jeweils immer zwei der Nachbarn konträr sind. Eine Verschiebung zu einem der ausgeschlossenen Broker würde in einer Erhöhung der Distanz zu dem definierten Zielsystem resultieren und wäre somit kontraproduktiv. Auf Grund der bekannten Indexstruktur und des daraus resultierenden Overlay kann die Anzahl somit auf zwei potentielle Routingpunkte festgesetzt werden.

Der gewählte Routingpfad kann, basierend auf diesen Ausführungen, daher stets als vermutlich optimal angenommen werden, da jede Bewegung den Abstand zu dem Zielsystem verkleinert. Nicht eindeutig ist die Wahl, welcher der beiden Punkte tatsächlich der nächste Schritt auf dem Routingpfad ist.

Für diese komplexere Auswahl kann mittels eines theoretischen Hilfskonstrukts, das zunächst den Längen- und Breitengrad des Ziels der Anfrage mit dem des aktuellen Brokers vergleicht, eine weitere eingrenzende Vorauswahl getroffen werden. Kann bei einem der beiden Vergleiche eine Übereinstimmung festgestellt werden, ist eine eindeutige Entscheidung möglich.

Ist der Breitengrad des Zielknotens mit dem des aktuellen Brokers identisch, muss die Anfrage in vertikaler Richtung weitergeleitet werden. Eine weitere horizontale Verschiebung ist nicht mehr nötig, da das angefragte Gebiet, bedingt durch diese Übereinstimmung, ober- oder unterhalb des aktuellen Brokers sein muss. Ähnlich verhält es sich bei einer Übereinstimmung des Längengrades. Der nächste Routingpunkt bei einer solchen Übereinstimmung

ist dabei horizontal zu dem aktuellen Knoten angeordnet. Ein Routing kann hier folglich nur zu dem linken oder rechten Nachbarknoten erfolgen.

Ist keine Übereinstimmung feststellbar, ist eine eindeutige Auswahl erschwert, da keine ausreichenden Kenntnisse über die Gesamtstruktur des Overlaynetzwerkes vorhanden sind. Diese Uneindeutigkeit der Auswahlmöglichkeit ermöglicht es, dass mehrere Pfade zwischen zwei Punkten gewählt werden können. Basierend auf dieser theoretischen Vorarbeit wird der nächste Routingschritt in dem beschriebenen uneindeutigen Fall über einen Zufallswert bestimmt.

Die Anwendbarkeit eines randomisierten Algorithmuses wird durch zwei Thesen gestützt. Zum einen durch die bereits nachweislich effiziente Funktionsweise dieser Algorithmenart in anderen Bereichen der Informatik[Hro], wobei diese Algorithmen oftmals nachweislich weniger Ressourcenintensiv sind und dennoch das gewünschte Ergebnis hervorbringen. Zum Anderen durch die nicht eindeutige Bestimmbarkeit des optimalen Pfades auf Grund der fehlenden Kenntnisse über die nachfolgenden Knoten.

Basierend auf dieser Unkenntnis kann angenommen werden, dass jeder der gewählten Pfade der vermutlich minimalste Pfad zwischen Ausgangsbroker und Zielknoten ist.

Ein weiterer Vorzug, der sich auf die geplante Replikationsstrategie auswirkt, ist die Anzahl an möglichen Replikationspunkten, die jedem Broker zur Verfügung stehen. Durch eine Varianz des gewählten Routingpfades durch die zufallsbasierte Auswahl im uneindeutigen Fall, wird die Anzahl der Replikationspunkte vergrößert und somit zusätzliche Replikationsmöglichkeiten für den Broker geschaffen.

Eine *single hop routing* impliziert neben den genannten Vorzügen aber auch einige Nachteile. Durch die geringe Flexibilität der strikten Umsetzung werden, vor allem zu einem späteren Zeitpunkt, unnötige Daten erfasst und erzeugen eine nicht notwendige Belastung der Ressourcen des Netzwerkes. Zu diesem Zeitpunkt ist es wahrscheinlich, dass bereits alle Informationen über das Netzwerk vorhanden sind.

Um diese Nachteile zu minimieren kann die Umsetzung einer *loose single hop routing* Strategie erfolgen. Der dabei zu definierende Zeitpunkt, der den Wechsel zu einer *direct routing* Strategie erlaubt, ist jedoch nur schwer zu ermitteln und kann nur unzureichend mittels einer mathematischen Funktion bestimmt werden. Werden die Restriktionen zu früh aufgehoben, geht der positive Effekt der Strategie verloren. Bei einem zu späten Aufheben entsteht weiterhin ein signifikanter Overhead.

Direct Routing

Als entgegengesetztes Extrema wird das *direct routing* im Rahmen der theoretischen Lösungserarbeitung analysiert. Diese Strategie setzt voraus, dass jedem Teilnehmer des Netzwerkes der Anfragen oder Pakete innerhalb des Netzwerkes weiterleitet die komplette Overlaytopologie bekannt ist. Netzwerkstrukturen mit einer hohen Fluktuation sind daher eine ungeeignete Basis für die effektive Nutzung dieser Routing Strategie. Da wir bei einem globalen Sensornetzwerk aber gemäß der Annahmen aus Kapitel 4.1 von einem Netzwerk

ausgehenden, dessen Brokerstruktur nur wenige Veränderungen erfährt, wäre diese notwendige Voraussetzung ausreichend erfüllt.

Kritisch ist der aus der notwendigen Datenvorhaltung entstehende Overhead zu betrachten.

So steigt der Datenumfang zum Vorhalten der notwendigen Brokerinformationen im Verhältnis zu den in dem Netzwerk vorhanden Brokern. Folglich nimmt somit der Anteil der effektiv nutzbaren Ressourcen ab.

Diese Annahme lässt sich mittels der Endlichkeit der vorhandenen Ressourcen stützen. Durch das Hinzufügen eines neuen Brokers müssen dessen neue Daten lokal auf jedem bereits vorhandenen Broker zusätzlich vorgehalten werden. Die für die Datenvorhaltung genutzte Speicherkapazität steigt somit stetig an und folglich nehmen die für die Anwendung verfügbaren Ressourcen jedes Brokers innerhalb des Netzwerkes ebenso beständig ab.

Als weiterer Kritikpunkt für das *direct routing* ist der ebenfalls mit der Anzahl der Broker steigende Bandbreitenbedarf anzuführen. Um die lokal vorgehaltenen Daten stets aktuell und somit überhaupt für die Anwendung effektiv nutzbar zu halten, müssen diese innerhalb eines zu definierenden Zeitintervalls aktualisiert werden. Die Aktualisierung erfordert das Übertragen von mindestens zwei Nachrichten zu jedem der innerhalb des Netzwerkes bekannten Broker. Die Anzahl ergibt sich aus der Notwendigkeit einer Datenanfrage und der damit verbundenen Rückantwort des entsprechenden Netzwerkknotens und ist folglich nich beeinflussbar.

Wird ein neuer Broker hinzugefügt, steigt entsprechend auch die Anzahl der notwendigen Nachrichten. Die Anzahl der notwendigen Nachrichten ist mittels der pronischen Zahlenfolge[1] bestimmbar, die ein kubisches Wachstum besitzt. Da die Anzahl der notwendigen Nachrichten nicht beeinflussbar ist, kann der Ressourcenbedarf nur mittels des festzulegenden Zeitintervalls innerhalb dessen die Daten für die Optimierung durch das System als anwendbar gelten beeinflusst werden. Die Problematik ist daher die effektive Bestimmung des Zeitintervalls.

Ein zu kurzes Zeitintervall kann die vollständige Auslastung der Systemressourcen beschleunigen, da wesentlich mehr Nachrichten für die Aktualisierung der vorgehaltenen Brokerinformationen versendet werden. Ein zu lang gewähltes Intervall führt häufiger zu veralteten Daten, die für eine Optimierung nicht mehr nutzbar sind.

Multi Hop Routing

Multi hop routing realisiert eine Zwischenlösung der beiden zuvor behandelten Extrema. Es ist keine vollständige Kenntnis der Netzwerkstruktur notwendig, wodurch der entstehende Overhead, im Gegensatz zu dem erläuterten *direct routing*, reduziert wird. Die Overlaystruktur wird durch jeden Broker innerhalb des Netzwerkes eigenständig, schrittweise erarbeitet und kann anschließend für ein *direct routing* genutzt werden.

[1] $Anzahldernotwendigen Nachrichten = \text{Anzahl der Broker} * (\text{Anzahl der Broker} - 1)$

Wird eine neue Anfrage in das Netzwerk eingefügt, erfolgt das Routing zunächst nach dem Prinzip des *single hop routings* wie in Abschnitt 5.1.1 erläutert. An die weitergeleitete Anfrage angehängt ist eine Liste, in die alle Knoten, die bei dem Routing passiert werden, aufgenommen werden. Der Zielbroker und optional auch die passierten Broker lesen diese Liste aus und erlernen somit schrittweise die Overlaystruktur des Netzwerkes. Sind dem Broker Netzwerkknoten bekannt, so vergleicht er diese mit dem Ziel der Anfrage und kann bei einer Übereinstimmung die Anfrage direkt an das Zielsystem übertragen. Alternativ kann, falls keine Übereinstimmung vorhanden ist, in der Menge der bekannte Broker nach einem möglichen Knoten gesucht werden, dessen Distanz zum Zielsystem kleiner ist als die der bisher für das Routing anwendbaren Nachbarknoten. Dieser wird folglich als nächster Routingpunkt ausgewählt.

Vorteilhaft ist hierbei die möglicherweise verkürzte Laufzeit einzelner Nachrichten. Vor allem hochfrequentierte Knoten sind rasch innerhalb der etablierten Struktur bekannt und erhalten die mit ihnen verknüpften Nachrichten schneller.

Durch den beständigen Nachrichtenfluss zwischen den Knoten kann die Aktualisierung der Daten ohne das Versenden zusätzlicher Nachrichten erfolgen.

Die lokal vorgehaltenen Daten müssen hierfür als weitere Information einen Zeitstempel beinhalten, der bei einer Informationsabfrage konsultiert wird und prüft, ob die Daten noch für eine eventuell notwendige Berechnung ausreichend aktuell sind. Sollten die Daten von dem System als veraltet angesehen werden, werden sie mittels einer Statusnachricht aktualisiert.

Das Versenden von Statusnachrichten wirkt sich dabei nachteilhaft auf die Gesamtperformance des Netzwerksystems aus. Entscheidend ist hierbei, wie bereits im Rahmen des *direct routing* erläutert, das zu definierende Zeitfenster in dem die Daten durch das System noch anwendbar sind. Für die Definition des Zeitfensters gelten die zuvor erläuterten Prinzipien.

Als weiterer Nachteil ist anzuführen, dass die Brokeranzahl, die ein Knoten erlernt, nicht quantifiziert werden kann. Erhält ein Broker stets über den gleichen Knoten Anfragen, erlernt er nur diesen und den einmaligen Routingpfad. Der Nutzen wäre somit gering.

5.1.2. Auswahl der Routingstrategie

Ziel der Anpassung der Routingstrategie ist die Nutzung der vorhandenen Query-Indizes sowie die mögliche Realisierung von Synergieeffekten für die Optimierung der Replikationsstrategie. Im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* wird daher eine *striktes single hop routing* realisiert.

Die Vorzüge dieses Routingalgorithmus sind die Vielzahl an Broker, die erlernt werden. Dass die Anzahl der Broker bei dieser Routingstrategie am höchsten ist, ist aus der Abbildung A.4 im beigefügten Anhang erkennbar. Somit können bei einer Optimierung der Replikationsstrategie hohe Synergieeffekte realisiert werden, wobei ein hoher Nachrichten Overhead, der bei einem *direct routing* für die Aktualisierung der Daten notwendig ist, vermieden werden kann.

Nachteilhaft ist die verzögerte Erstversorgung des anfragenden Knotens. Diese wird durch das Anfragerouting bedingt, da, im Gegensatz zu der bisherigen Routingsstrategie, die eine Anfrage direkt an das Zielsystem übertagen hat, Zeit für das Routing benötigt wird. Da jedoch nicht die Erstversorgung der anfragenden Knoten mit Daten in einem globalen Sensornetzwerk entscheidend ist, sondern der Schwerpunkt auf der möglichst dauerhaften echtzeitnahen Datenversorgung im Anschluss liegt, ist die längere Laufzeit einer Anfrage durch die Realisierung eines *strikten single hop routings* weniger gewichtig wie die spätere Möglichkeit zur optimalen Positionierung der notwendigen Replikationen und der damit verbundenen Minimierung der negativen Latenzeffekte.

Ein weiterer entscheidender Vorteil ist die mögliche Nutzung der Query-Indizes. Basierend auf den in den Abschnitten 4.2 und 5.1 erläuterten möglichen Routingalgorithmen wird dies möglichst gut realisiert.

Von einem Wechsel zu einem *direct routing*, wie es bei der Umsetzung einer *loose singel hop routing* Strategie möglich ist, wird abgesehen, da zu keinem festen Zeitpunkt von einer ausreichenden Anzahl bekannter Knoten ausgegangen werden kann. Das Kriterium ist somit nicht ausreichend genau spezifizierbar und stellt einen zu großen Unsicherheitsfaktor dar.

5.2. Replikation

Bei einer Überlastung eines Knotens durch eine Anfrage, die seine noch verfügbaren Ressourcen übersteigt, kann dieser versuchen die Daten auf einen andern Knoten innerhalb des Netzwerkes zu übertragen und die dort freien Ressourcen für die Datenversorgung zu nutzen. Dieser Vorgang wird als Datenreplikation bezeichnet.

Würde bei diesem Vorgang lediglich die überlastende Anfrage repliziert, würde die resultierende Last für den replizierenden Knoten nicht reduziert. Ziel der Replikation ist es daher Anfragen, die gleiche oder ähnliche Daten erhalten, gemeinsam zu bündeln und auf das Replikationsziel auszulagern. Ähnlich sind Datenanfragen in globalen Sensornetzwerken, die das gleiche Gebiet oder ein Teilgebiet dessen umfassen, mit einer zeitlichen und räumlich gleichen oder geringeren Auflösung. Um den Vorgang der Datenreplikation zu optimieren sind zwei Ansätze möglich.

Zum Einen kann die Auswahl der zu replizierenden Anfragen optimiert werden, sodass Anfragen, die bezüglich des Optimierungsparameters am Besten zusammenpassen, gemeinsam repliziert werden. Zum Anderen kann die in dieser Ausarbeitung behandelte Auswahl des Replikationsziels optimiert werden. Die optimierte Zusammenfassung der zu replizierenden Anfragen wird im Rahmen dieser Ausarbeitung nicht behandelt, da diese zusätzliche Aufgabenstellung den Umfang übersteigen würde. Sie ist aber eine mögliche Aufgabenstellung für eine anknüpfende wissenschaftliche Ausarbeitung.

Für die Auswahl des optimalen Replikationszielknotens können ebenso mehrere Parameter berücksichtigt werden. Im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* wird eine Optimierung bezüglich der Latenz durchgeführt. Die latenzoptimierte Auswahl basiert auf dem direkten Parametervergleich. Eine zu große Auswahl an

Replikationsmöglichkeiten verlängert folglich den Auswahlvorgang. Die Optimierung muss daher schrittweise erfolgen.

Zunächst werden Broker, die, basierend auf den getroffenen Annahmen nicht optimal sein können, schrittweise aus dem weiteren Optimierungsvorgang ausgeschlossen. In dem hierauf folgenden Optimierungsvorgang erfolgt dann die endgültige Auswahl mittels des bereits erwähnten direkten Parametervergleichs.

5.2.1. Bestimmung der möglichen Replikationspunktmenge

Da wir von einem nicht vollvermaschten Netzwerk ausgehen, ist die Anzahl der möglichen Replikationspuntke zunächst auf die assoziierten Nachbarknoten eingeschränkt. Basierend auf den in dem Kapitel 5.1 beschriebenen Anpassungen im Anfragerouting kann diese Menge erweitertet werden.

Jeder Broker innerhalb des Netzwerkes erlernt somit über die Zeit die vollständige Overlaystruktur. Als mögliche Replikationspunkte für eine Anfrage werden nun auch die im Zuge des Routings der anderen zu replizierenden Anfrage passierten Netzwerkknoten mit berücksichtigt.

Diese Knoten können als mögliche Optima für die Replikation der Anfrage angenommen werden, da jeder der Punkte auf dem Routingpfad auf dem vermutlich minimalsten Pfad zwischen Anfragen annehmenden Broker und Zielbroker liegt. Die Latenz des anfrageannehmenden Brokers ist, basierend auf den Annahmen aus Kapitel 4.1, die geringste Latenz zwischen einem Broker des Netzwerkes und dem anfragenden Clientknoten. Da zwar von einer geographischen Unabhängigkeit der Latenz zwischen zwei Puntke ausgegangen wird, nicht aber von einer logischen Entkopplung, wird als Folgerung angommen, dass jeder Broker des Routingpfades ebenfalls die geringeste Latenz aus der Menge der Nachbarknoten zu dem anfragenden Client bestizt.

Andere, dem replizierenden Broker bekannte Knoten innerhalb des Netzwerkes, werden bei der Auswahl nicht berücksichtigt, da diese gemäß der Annahmen kein Optimum sein können.

5.2.2. Optimierte Auswahl des Replikationspunktes

Nach der Bestimmung der möglichen optimalen Replikationspunkte muss nun aus dieser Menge das globale Optimum bestimmt werden.

Die Menge kann zunächst mittels einer Prüfung der vorhandenen freien Ressourcen weiter reduziert werden. Sind nicht genügend Ressourcen vorhanden, kann dementsprechend auch keine Replikation erfolgen.

Innerhalb der resultierenden Menge kann nun mittels eines Parametervergleichs das Optimum betimmt werden. Beachtet werden muss hierbei die Verhältnismäßgkeit. Die negativen Effekte, auf die aus der Replikation resultierende Latenz einer datenumfangreichen Anfrage

sind höher zu bewerten als die negativen Effekte auf eine Anfrage, deren Datenmenge gering ist.

Als Resultat lässt sich eine Gleichung für die Bestimmung der Optimierungskennzahl herleiten.

$$\textbf{Optimierungskennzahl} = \sum \text{Lastanteil} * \text{negative Latenzentwicklung} + \text{Basislatenz}.$$

Die dabei verwendeten Parameter lassen sich mittels der nachfolgenden Gleichungen bestimmen.

$$\text{Lastanteil} = \frac{\sum \mathbf{DV\ Q}}{\mathbf{DV\ Q}_i}$$

$$\text{negative Latenzentwicklung} = (\mathbf{L}_{(\mathbf{C},\ \mathbf{RZK})} - \mathbf{L}_{(\mathbf{C},\ \mathbf{RK})})$$

$$\text{Basislatenz} = \mathbf{A} * \mathbf{L}_{(\mathbf{RZK},\ \mathbf{RK})}$$

Verwendete Abkürzungen:

- **L** = Latenz
- **C** = Anfragender Client
- **RK** = Replizierender Knoten
- **RZK** = Replikationszielknoten
- **A** = Anzahl der zu replizierenden Anfragen
- **DV** = Datenvolumen
- **Q** = Query

Mittels dieser Gleichung ist für jeden der möglichen Replikationspunkte eine individuelle Optimierungskennzahl errechenbar. Der Broker mit der minimalsten Optimierungskennzahl weist dabei die geringste negative Latenzentwicklung pro Lasteinheit auf und ist somit der optimale Replikationspunkt.

5.2.3. Anpassungsfähigkeit der Auswahl

Durch Anwendung eines Vorfaktors kann bei Bedarf eine individuelle Gewichtung zwischen Last und Latenz erfolgen und das System somit, je nach Umgebungsanforderung, angepasst werden. Die Gleichung zur Bestimmung der Optimierungskennzahl bleibt hierfür unverändert, lediglich die Berechung der Parameter wird angepasst.

Angepasste Parametergleichungen:

$$\text{Lastanteil} = \textbf{Lastfaktor} * \frac{\sum \mathbf{DV\ Q}}{\mathbf{DV\ Q}_i}$$

$$\text{negative Latenzentwicklung} = \textbf{Latenzfaktor} * (\mathbf{L}_{(\mathbf{C},\ \mathbf{RZK})} - \mathbf{L}_{(\mathbf{C},\ \mathbf{RK})})$$

$$\text{Basislatenz} = \textbf{Latenzfaktor} * \mathbf{A} * \mathbf{L}_{(\mathbf{RZK},\ \mathbf{RK})}$$

6. Praktische Problemlösung

Das folgenden Kapitel der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* erläutert die praktische Realisierung der theoretisch erarbeitet Lösungsansätze aus Kapitel 5 und die damit verbundenen Herausforderungen. Als Grundlage für die Implementierung wurde das System *StreamReUse* verwendet. Die durchgeführten Anpassungen an *StreamReUse* können über die in der Konfigurationsdatei definierten Parameter *ROUTING_PLUS* und *REPLICATION_PLUS* aktiviert oder deaktiviert werden.

6.1. Grundlage für die Entwicklung

StreamReUse ist ein brokerbasiertes globales Sensornetzwerksystem das von Dipl.-Inf. Andreas Benzing entwickelt wurde. Das System nutzt das Netzwerksimulationswerkzeug *PeerSim* um die notwendigen Basisnetzwerkstruktur zu simulieren. Bei der ersten Analyse des bereitgestellten Java-Source-Codes wurde festgestellt, dass hierdurch zwar eine Overlaystruktur etabliert wird, jedoch kein Underlay hiermit verknüpft ist. Da die Underlaystruktur eine zwingende Voraussetzung für die Möglichkeit einer Latenzoptimierung der Replikationen innerhalb von *StreamReUse Plus* darstellt, muss dies zunächst nachgebessert werden.

PeerSim kann, mittels der beinhalteten Klasse *E2ENetwork*, eine sollte Struktur erzeugen. Bei dem Initialisieren der leeren Basisstruktur mit *E2ENetwork* wird bereits festgelegt, ob das zu simulierende Netzwerk symmetrisch oder asymmetrisch ist. Ein Netzwerke wird als asymmetrisch bezeichnet, wenn der beschreibende Netzwerkgraph gerichtet ist. Verbindungen können in diesem Fall nicht immer bidirektional genutzt werden.

StreamReUse Plus basiert gemäß unserer Annahmen auf einer symmetrischen Netzwerkstruktur. Der beschreibende Netzwerkgraph ist folglich ungerichtet. Dies bedeutet das alle in *StreamReUse Plus* vorhandenen Verbindungen bidirektional genutzt werden können und unabhängig von der gewählten Richtung stets die gleiche Latenz aufweisen.

Die zu definierenden Latenzen werden durch die Methoden von *E2ENetwork* automatisch für die Verbindung zwischen zwei Knoten wechselseitig festgelegt. Durch diese Vorgaben ergibt sich als Format für die Definition der Latenzwerte eine strikte obere Dreiecksmatrix. Bei einer strikte obere Dreiecksmatrix sind alle Einträge unterhalb sowie auf der Diagonalen der Matrix null. Diese Matrix kann als Binärdatei übergeben werden, falls für die Underlaykonstruktion die bereits umgesetzten Hilfsklasse *TriangularMatrixParser* von *PeerSim* verwendet wird. Alternativ können eigene Methoden zur Underlaygenerierung basierend auf der *E2ENetwork*-Klasse implementiert werden.

Komplex ist die korrekte Berechnung der Latenzwerte die die in Kaptiel 4.1 beschrieben Dreiecksungleichung erfüllen müssen und dennoch eine Varianz aufweisen sollen. Für die Realisierung in *StreamReUse Plus* wird die Latenz daher auf Basis einer Grundlatenz, in Kombination mit einer zufallszahlabhängigen Varianz, die sich ebenfalls aus der Grundlatenz ableitet errechnet. Das die so erstellte Underlaystruktur die Dreiecksungleichung erfüllt wird im Anhang A.5 nachgewiesen.

Die Latenzberechnungen für die nicht durch die Overlaystruktur verknüpften Knoten basiert auf der, mittels des Overlaygenerators erstellten, Overlaystruktur. Diese Struktur wird in einen gewichteten Graphen überführt, der die Grundlage für die spätere Berechnung der minimalsten Pfad ist. Dieser Graph wird benötigt um im nächsten Schritt realitätsnahe Latenzen für alle noch nicht verknüpften Knoten zu erstellen, die weiterhin die Dreiecksungleichung erfüllen.

Ausgehende von den minimalen Pfaden zwischen dem Ausgangsknoten und jedem anderen Netzwerkknoten, die mit einem Dijkstr Algorithmus bestimmt werden, kann die Latenzsumme des minimalen Pfades errechnet werden. Aus der Latenzsumme ist die Latenz zwischen den beiden nicht verknüpften Knoten mittels der Gleichung „$Latenz = Latenzsumme - (0.1 * Grundlatenz)$“ ableitbar.

Zur Umsetzung in ein simuliertes Underlay wird nicht die bereitgestellte Klasse genutzt, sondern, die bereits zuvor erwähnte Möglichkeit einer selbst implementiert Methode. Begründet wird diese Entscheidung durch die Eliminierung der sonst notwendigen Umformung des generierten Underlay in eine Binärdatei.

Als weiter Voraussetzung war die Einbindung von Clientknoten notwendig. Im Unterschied zu den vorhanden Netzwerkknoten sollten diese Komponenten lediglich neue Anfragen in das Netzwerk einfügen, selbst aber keine Sensordaten erfassten oder verwalten. Des Weiteren werden diese Punkte nicht als mögliche Repliaktiosnmöglichkeiten berücksichtigt.

Die Anzahl der vorhanden Clientknoten kann über den Parameter *CLIENTNODES* in der Konfigurationsdatei von *StreamReUse Plus* flexibel an die Umgebung angepasst werden. Eine entsprechende Verknüpfung mit einem Broker innerhalb des Underlay wird automatisch erzeugt.

Nachdem durch die Schaffung eines erweiterten Overlays und eines damit verknüpften Underlay die notwendigen Voraussetzungen vorhanden sind, wird im Abschnitt 6.2 die Umsetzung des modifizierten Routings erläutert.

Für die Erzeugung des Overlay sowie des damit assoziierten Underlay werden die folgenden Klassen benötigt:

- OverlayGenerator
  Erzeugt eine vermaschtes Netzwerk das für die Etablierung einer Netzwerkstruktur in *PeerSim* verwendet wird.
- Varianz
  Erzeugt Latenzwerte die, basierend auf der gegebene Grundlatenz variieren und denoch die Drecksungleichung weiterhin erfüllen.

- OverlaySetup
  Liest das erzeugte Overlay ein und verknüpft alle Knoten mit deren Nachbarn.
  Formatierung der einzulesenden *topologyOverlay.cfg*:
  Falls die aktuelle Zeile einen Broker definiert:

$$< AktuellerBroker >< rechterNachbar >< unterer Nachbar >< linkerNachbar >< obererNachbar >$$

  oder falls der betreffende Zeile einen Client definiert:

$$< AktuellerClient >< assoziierterBroker >$$

- WeigthedGraph
  Erzeugt einen gewichten Graphen der unter Anwendung eines Dijkstra Algorithmusdie Latenzen für die noch nicht verknüpften Knoten erstellt.

- UnderlaySetup
  Liest die erzeugte strikte obere Dreiecksmatrix ein und erzeugt die so definerten Verbindungen zwischen den Knoten mit den entsprechenden Latenzen.
  Formatierung der einzulesenden Matrix aus *topologyUnderlay.cfg*:
  <Gesamtanzahl der Knoten im Netzwerk> <...Latenzen der Knoten zu Knoten ....>
  <0> <0> <...Latenzen Knoten 1 und dem durch die Spalte definierten Knoten....>
  <0> <0> <0> <...Latenzen Knoten 2 und dem durch die Spalte definierten Knoten....>

  Weiter Zeilen können entsprechend dem Schema ergänz werden.

6.2. Entwicklung des Anfragerouting

Für die Umsetzung des in Abschnitt 5.1 beschriebenen theoretischen Lösungsansatzes zur möglichen Optimierung des Anfrageroutings ist zunächst das aktuelle Routing analysiert worden.

Ziel war die Findung eines möglichen Einstiegspunktes für die Modifikation. Da die Analyse hervorbrachte, dass in *StreamReUse* kein Routing umgesetzt wird sondern neue Anfragen direkt an den Zielbroker übermittelt werden, war keine Modifikation möglich sondern eine komplette Neuimplementierung notwendig.

Als erste Schritt wird jede neue Anfrage mit einem der initialisierten Clientknoten verknüpft um einen definierten Ausgangspunkt für das Routing festlegen zu können. Gemäß der Annahme aus Kapitel 4.1 ist somit auch der erste Knoten des Routingpfades vorgegeben.

Jeder nachfolgende Knoten vergleicht zunächst den *DZ-Ausdruck* der das Zielgebiet des Query definiert mit dem ihm zugeordneten. Stimmen beide Ausrücke überein ist der Zielknoten erreicht und es erfolgt keine weiteres Routing.

Ist der *DZ-Ausdruck* des angefragten Teilgebiets länger als der dem Broker zugeordnete, werden nun entsprechend der der Länge des dem Broker zugeordneten *DZ-Ausdrucks* die Stellen verglichen. Da beide Ausdrücke in *StreamReUse Plus* als BitSet gespeichert werden kann hierzu auf vordefinierte Methoden zurückgegriffen werden. Lediglich die Länge der Ausdrücke wird für den Vergleich angepasst.

Stimmen die Ausdrücke nicht überein muss der Broker entscheiden welcher der Nachbarknoten der nächste Schritt auf dem Routingpfad darstellt.

Auch hierzu wird zunächst der *DZ-Ausdruck* referenziert. Mittels eines Zufallswertes wird entschieden ob im ersten Durchlauf die geraden oder ungeraden Stellen analysiert werden.

Abweichungen zwischen dem Zielausdruck und dem mit dem Broker verknüpften Ausdruck an einer geraden Stelle schließen die vertikal assoziierten Nachbarknoten als nächste Routingpunkte aus. Entsprechend konträr ist das Verhalten bei einer Abweichung an einer ungeraden Stelle des *DZ-Ausdruck*.

Abschließend wird unter den beiden noch möglichen nächsten Knoten innerhalb des Routingpfades entschieden.

Auf Grund der bereits zuvor erläuterten Komplexität des *DZ-Ausdruck* basierten Routings wird diese Entscheidung basierend auf den mit den Knoten verknüpften Längen- und Breitengraden gefällt. Die somit errechenbare Distanz zwischen jedem der möglichen Nachbarknoten und dem Zielknoten ermöglicht eine eindeutige Entscheidung.

Zu Beachten ist, dass falls die errechnete Distanz größer als die Hälfte der Distanz die die gesamtverfügbaren Welt beschreibt ist, ein Routing über die umfassenden Grenzen hinweg möglich ist. Um diese Möglichkeit korrekt in der Berechnung zu berücksichtigen muss die errechnete Distanz um die Hälfte der maximalen Distanz reduziert werden. Der dafür benötigte Maximalwert lässt sich aus der Konfigurationsdatei ableiten.

Jeder der Routingschritte wird in einer der Anfrage zugeordneten Liste festgehalten und von dem Zielknoten ausgelesen.

Alle für die Bestimmung des nächsten Knoten des Routingpfades notwendigen Berechnungen sind in die Klasse *RoutingUtilities* ausgelagert um eventuelle Anpassungen an diesen zu ermöglichen.

Abschließend nochmals die Routingschritte übersichtlich und kurz dargestellt. Ist nach einem der Schritte bereits eine eindeutiger Knoten als nächster Routingpunkt festzustellen bricht der Algorithmus ab.

- Längenanpassung des zu vergleichenden *DZ-Ausdruck* der Anfrage.
- Vergleich des Broker *DZ-Ausdruck* mit dem des angefragten Teilgebiets.
- Erzeugen der Zufallszahl um festzulegen ob zunächst die ungerade oder geraden Stellen des *DZ-Ausdrucks* betrachtet werden sollen.
- Vergleich der Stellen entsprechend der Zufallszahl.

- Wird keine Abweichung festgestellt werden die noch nicht analysierten Stellen betrachtet.
- Einschränkung der möglichen Routingpunkte abhänig von der parität der abweichenden Stelle.
- Distanzberechnung der eingeschränkten Knotenmenge zu dem Zielsystem.

6.3. Entwicklung der latenzoptimierten Replikation

Als weiter Aufgabenstellung wird in Abschnitt 5.2 die Replikation von Anfragen in *StreamReUse Plus* theoretische gelöst. Bedingt durch die zuvor beschriebene Umsetzung des Anfrageroutings und der Protokollierung aller Routingschritte der Anfrage ist die Anzahl der möglichen Replikationspunkte gewachsen. Unter dieser nun umfangreicheren Menge muss eine latenzoptimierte Auswahl getroffen werden.

Die Ermittlung des optimalen Replikationsnetzwerkknotenpunktes basiert dabei auf der in dem Abschnitt 5.2 beschriebenen Gleichung.

Um die Vorauswahl möglichst zu vereinfachen werden die möglichen Replikationspunkte einer Anfrage direkt aus der der Anfrage angehängten Routingtabelle entnommen.

Gemäß der Annahmen und deren Folgerungen im Kapitel 5.2.1 sind nur diese Punkte ein mögliches Optimum zur Replikation. Da eine Replikation nur bei einer gleichzeitigen Verschiebung mehrerer Anfragen lastreduzierend für den replizierenden Broker ist, werden die jeder Anfrage angehängten Routingptabellen im ersten Schritt auf mögliche Übereinstimmungen geprüft.

Gibt es nur eine Übereinstimmung ist die Entscheidung eindeutig und die Optimierungslogik bricht ab.

Ist mehr als eine oder keine Übereinstimmung in den verschieden Anfrageroutingtabellen festzustellen ist die korrekte Wahl des optimalen Replikationspunktes komplexer und weiter Schritte müssen durchgeführt werden.

Ist keine Übereinstimmung festzustellen müssen alle Netzwerkknoten die von allen zu replizierenden Anfragen passiert wurden verglichen werden. Hierzu muss mittels der in Kapitel 5.2.2 beschriebenen Gleichung für jeden möglichen Replikationsknoten der Optimierungsparameter bestimmt werden. Der Vergleich wird dadurch aufwendiger da die Menge der zu vergleichenden Knoten größer ist, jedoch ist eine weitere Optimierung über die bereits erläuterte Optimierungslogik hinaus nicht notwendig.

Wird mehr als eine Übereinstimmung gefunden und sind die errechneten Optimierungsparameter für mehrere Knoten identisch wird im letzten Schritt der Optimierungslogik die verfügbaren Ressourcen der möglichen Replikationspunkte geprüft. Sollte nach der Durchführung dieser Schritte die Wahl noch immer nicht eindeutig sein, wird der zuerst gefunden Punkt aus der Menge der verbleibenden Optima als Replikationsziel ausgewählt.

Der Bestimmung des optimalen Replikationspunkts ist bei der praktischen Umsetzung somit stufenweise realisiert worden. Ist nach einer Stufe die Menge der zur Auswahl stehenden Knoten bereits eindeutig auf einen Knoten reduzierbar wird nach diesem abgebrochen.

Die Stufen sind im Nachfolgenden nochmals übersichtlich dargestellt.

- Prüfung der an die Anfragen angehängten Routingpfade auf Übereinstimmungen.
- Errechnung des Optimierungsparameter für die Menge der zu Auswahl stehenden Punkte.
- Vergleich der noch verfügbaren Ressourcen auf jedem der Broker.
- Auswahl des ersten Knoten aus der Menge der noch zur Auswahl stehenden Knoten.

Die Auswahllogik ist in die Klasse *ReplicationUtilities* ausgelagert worden um eventuell notwendige Anpassungen einfacher durchführen zu können.

6.4. Probleme bei der praktischen Umsetzung

Neben den bereits zu erwartenden Komplikationen die sich aus der Nutzung eines bis dahin unbekannten Frameworks ergeben war die unzureichenden Dokumentation von *PeerSim* eine besondere Herausforderung. Da auch Fragen an die Entwickler und weiteren Nutzer über das mit *PeerSim* verknüpfte Forum[1] unbeantwortet blieben musste an einigen Stellen mittels *Reverse Engineering* die Funktionalität von *PeerSim* nachvollzogen werden. Die oftmals sehr trickreiche Implementierung erschwerte diese Arbeit.

Das bereitgestellte System *StreamReUse* war hingegen leicht zu modifizieren. Einige der Variablenbenennungen waren hier irreführende gewählt. Diese Probleme waren aber dank der guten Betreuung durch Dipl.-Inf. Andreas Benzing leicht zu überwinden.

Weitere Probleme die sich im Laufe der praktischen Umsetzung ergaben waren zum Teil den noch nicht so gut ausgereiften Programmierfähigkeiten zuzuschreiben.

[1] http://sourceforge.net/projects/peersim/forums

7. Evaluation

Um die Effizienz der Optimierung nachzuvollziehen wird das modifizierte System mit dem nicht modifizierten System *StreamReUse* verglichen. Dieser Vergleich erfasst Kennzahlen die sowohl eine Analyse des Anfrageroutings als auch der Replikation ermöglichen. Da innerhalb des ursprünglich bereitgestellen Java-Source-Codes kein Underlay verknüpft wurde und auch die anfragenden Clients noch nicht initialisiert sind, muss die Evaluation bereits auf einer angepassten Variante des ursprünglich bereitgestellten System *StreamReUse* durchgeführt werden.

Die erfassten Kennzahlen bezüglich der notwendigen Routingschritte können nicht mit dem ursprünglichen System *StreamReUse* verglichen werden, da das ursprünglich rudimentär implementierte Routing in *StreamReUse* mit keinem Aufwand verbunden war.

Der Querygenerator berechnete in *StreamReUse* das Zielsystem bei der Erzeugung des Query und leitete die Anfrage in dem erzeugenden Cycle direkt an diesen Knoten weiter. Da auch keinerlei Latenzen oder Nachrichtenverzögerung berücksichtigt ist, war der Zeitraum zwischen der Anfrageerstellung und dem etablieren des Datenstroms null. Diese Zeitfenster kann unter realen Bedingungen nie realisiert werden. Da hier die Modelannahme des modifizierte System *StreamReUse Plus* realitätsgetreuer sind, ist der notwendige Zeitraum zwangsläufig größer null.

Die der Evaluation zu Grunde liegende Systemkonfiguration umfasst *64 Broker* sowie *10 Clientknoten*. Die notwendige *Grundlatenz ist mit 10* vorgegeben, sodass die maximale Latenz zwischen zwei Knoten in *StreamReUse Plus* 19 sein kann. Weiter Konfigurationsdetails können der im Anhang A.6 beigefügten Konfigurationsdatei entnommen werden.

Um die notwendigen Protokolldateien zu erzeugen wurde im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* das Werkzeug *Log4j*[Fou11] eingesetzt. Die erfassten Daten sind nicht der Diplomarbeit beigefügt, da diese zu Umfangreich sind. Eine zusammengestellte Übersicht ist aber in Anhang A.7 eingefügt.

Der Abschnitt 7.1 zeigt den Aufwand der für das weiterleiten eine Anfrage notwendig ist und interpretiert die gezeigten Resultate. Der darauf folgenden Abschnitt 7.2 stellt die Effizienz der angewendeten theoretischen Lösungskonzepte zur optimierten Platzierung der Replikationen innerhalb des bekannten Netzwerkes dar.

7.1. Evaluation Anfragerouting

Das Routing einer Anfrage in *StreamReUse Plus* passiert, in Abhängigkeit von dem einfügenden Client und dem diesem zugeordneten Broker sowie dem Zielknoten der Anfrage, eine abweichende Anzahl an Knoten. Die Anzahl der zu passierenden Knoten korreliert nicht mit der für das Routing notwendigen Anzahl von Vergleichen. Diese Unabhängigkeit lässt sich aus dem Vergleich der beiden Diagramme 7.1 und 7.2 deutlich erkennen und verdeutlicht, dass die Vorselektierung bedingt durch einen Vergleich von Längen- und Breitengrad effektiv zu einem zügigeren Routing der Anfrage beiträgt.

Die in dem Diagramm 7.1 dargestellten Daten zeigen die Anzahl der durchschnittlich passierten Knoten einer Anfrage. Deutlich zu erkennen ist, dass die Anzahl der Punkte trotz der Anpassungen der Umgebung stets innerhalb eines definierbaren Intervalls sind.

Dieses Intervall ist aus den Eigenschaften eines Quadtree[1] ableitbar und wird mit bedingt durch die in der Konfiguration festgelegte Anzahl Brokern innerhalb des Netzwerkes.

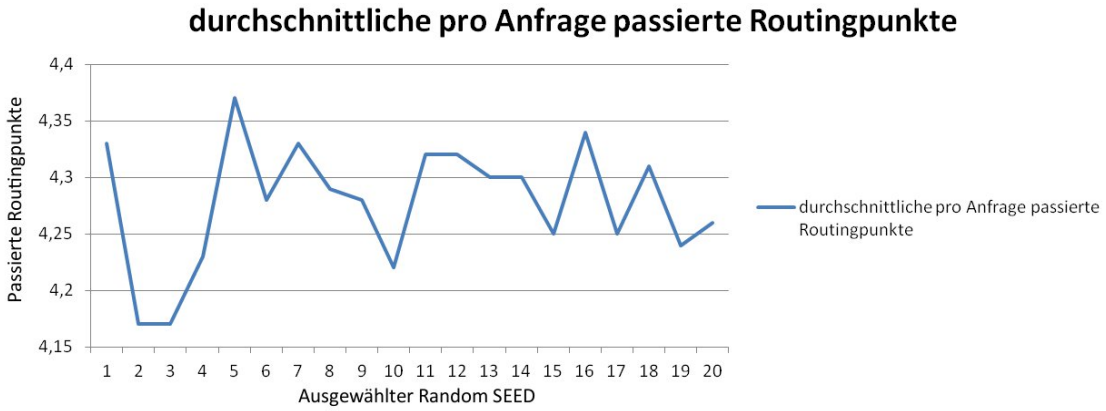

Abbildung 7.1.: Anzahl der durchschnittlich passierten Routingpunkte einer Anfrage

Als weiter Eigenschaft des Routings ist die ebenfalls sehr konstante Anzahl der notwendige Vergleiche die in dem Diagramm 7.2 zu erkennen ist anzuführen. Auch diese Konstanz lässt sich wiederum mit den Eigenschaften des angewendeten Quadtree begründen.

Durch eine Weiterreichung der bereits geprüften Ausschlusskriterien wäre die Anzahl der Vergleiche eventuell noch weiter reduzierbar, müsste aber mit dem für die Weiterleitung notwendigen Datenvolumen verglichen werden. Eine erste Optimierung des angewendeten Algorithmus wurde bereits im Rahmen dieser Abschlussarbeit durchgeführt. Der frühzeitige Vergleich des Längen- und Breitengrades des Zielknotenpunktes mit dem des weiterleitenden Knoten, erspart eventuell unnötige Vergleiche mit den Nachbarknoten da durch eine

[1] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Quadtree, 29.02.2012

Übereinstimmung die Richtung des nächsten Routingschritts frühzeitig festgelegt werden kann. Die notwendige Anzahl an Vergleichen wird durch diesen vorausgehenden Vergleich signifikant reduziert.

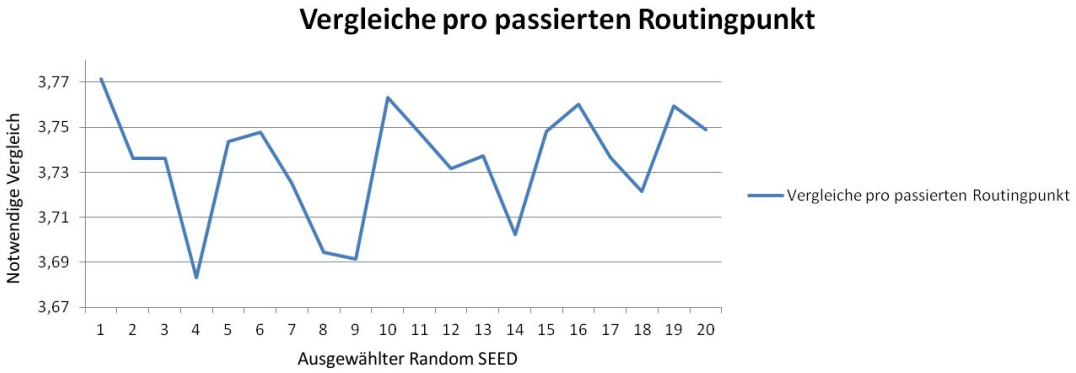

Abbildung 7.2.: Anzahl der durchschnittlich notwendigen Vergleiche

Abschließend lässt sich feststellen, dass das Routing einer Anfrage bereits effektiv umgesetzt wurde.

Um weiteres Optimierungspotential zu nutzen müssten mehr Daten lokal vorgehalten werden. Es ist daher kritisch abzuwägen ob, der realisierbare Zusatznutzen im Verhältnis zu dem dafür notwendigen Aufwand steht.

7.2. Evaluation Anfragereplikation

Die Replikation der Anfragen in *StreamReUse Plus* kann direkt mit der Anfragereplikation von *StreamReUse* verglichen werden. In der Abbildung 7.3 ist die durchschnittliche Latenz der Anfragen nach einer erfolgten Replikation dargestellt. Aus dem Diagramm ist deutlich die Reduktion der durchschnittlichen Latenz für jede Anfrage erkennbar.

Gemessen wurde hier die Gesamtlatenz der Anfrage, ausgehend von dem Datenquellsystem über den ausgewählten Replikationspunkt zu dem anfragenden Client. Die dargestellten negativen Effekte konnten durch die vorgenommene Optimierung bereits signifikant reduziert werden.

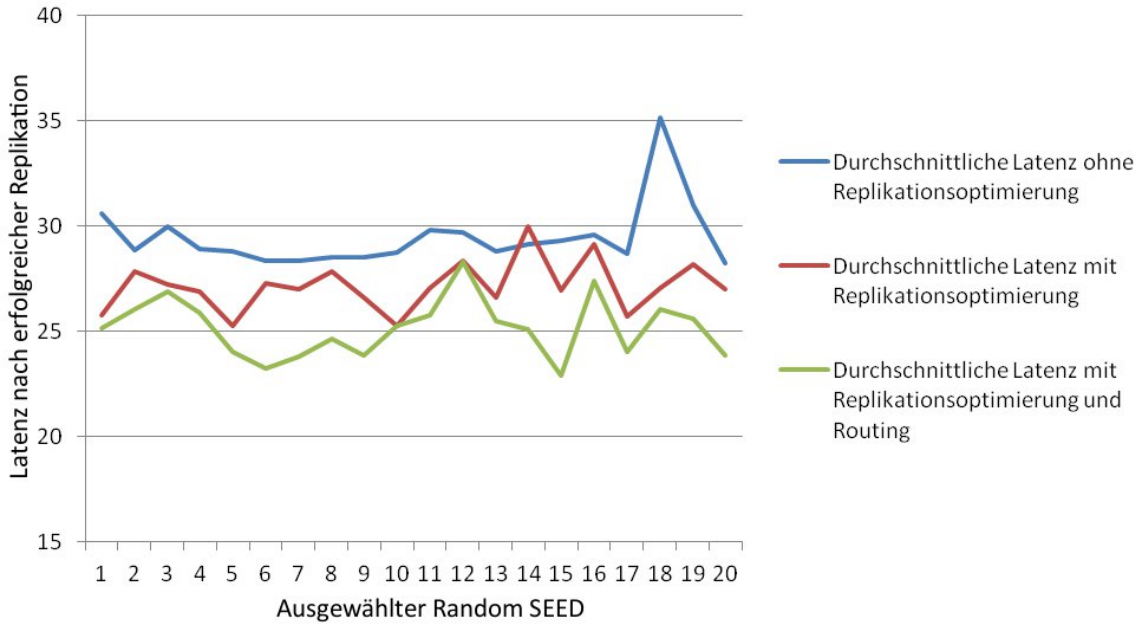

Abbildung 7.3.: Durchschnittle Latenz einer Anfrage nach der Replikation

Durch die zweigeteilte Aufgabenstellung konnte, wie in Abschnitt 5.2.1 bereits beschrieben, Synergieeffekte realisiert werden. Dies sind ebenfalls aus dem Diagramm 7.3 abzulesen.

Eine zusätzliche Erweiterung der möglichen Replikationspunktmenge würde in einem erhöhten Aufwand für die Datenhaltung resultieren. Es wäre abzuwägen ob eine solche Erweiterung sinnvoll ist da keine größeren Latenzreduktionen zu einem späteren Zeitpunkt des Netzwerkes erkennbar sind obwohl hier dem replizierenden Knoten mehr Replikationsmöglichkeiten bekannt sind. Somit ist der zusätzlich realisierbare Nutzen aus der erweiterten Datenhaltung wahrscheinlich geringer als der dafür notwendige Aufwand.

7.3. Zusammenfassende Bewertung der Evaluation

Die im Rahmen der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* erfassten Daten zeigen, dass die, zunächst rein theoretisch entwickelten Lösungen, für die Problemstellung der Optimierung von Datenstromreplikationen in globalen Sensornetzwerken sowie dem effizienten Anfragerouting in *StreamReUse Plus* gut umgesetzt wurden und die erhofften Synergieeffekte realisiert werden konnten. Die im Anhang A.7 eingefügten Daten sind eine Konsolidierung der gesamten für die Evaluation erfassten Daten. Insgesamt wurden 20 von einander verschiedene Konfigurationen analysiert.

Es wird durch die hohe Varianz der Umgebungen aufgezeigt, dass der erzielten Zusatznutzen nicht von der Umgebung abhängig ist, sondern durch die effektiven Vorüberlegungen und die geschickte Implementierung realisiert wird.

8. Zusammenfassung und Ausblick

Globale Wetterphänomene, die einen großen Einfluss auf unser tägliches Leben haben, zu erfassen und die Daten über eine einheitlich definierte Schnittstelle bereitzustellen, ist eine der Aufgaben von globalen Sensornetzwerken. Das Forschungsprojekt *Global sensor grid* der Forschungsgruppe *SimTech* an der *Universität Stuttgart* entwickelt unter der Leitung von *Prof. Dr. rer. nat. Kurt Rothermel* ein brokerbasiertes globales Sensornetzwerksystem.

Die Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* setzt sich mit zwei Problemstellungen aus dem Umfeld der globalen Sensornetzwerke auseinander, wobei es keine Stufenfolge zwischen den beiden Aufgabenstellungen gibt. Die beiden Aufgaben ergänzen sich gegenseitig, sodass Synergieeffekte realisiert werden.

Die erste Aufgabenstellung behandelt die Replikationsstrategie von globalen Sensornetzwerken und wie die zwangsläufig entstehenden negativen Effekte einer Replikation durch eine geschickte Auswahl des Replikationspunktes minimiert werden können.
Die zweite Aufgabenstellung behandelt die Analyse und Anpassung des Anfrageroutings innerhalb von globalen Sensornetzwerken. Die Anpassungen sind dabei auf die umgesetzte Routingstrategie abgestimmt und nutzen Query-Indizies als Entscheidungsbasis für die effektive Gestaltung des Routingpfads.

Die aktuell verfügbaren Systeme, die in dem Kapitel 2 behandelt werden, basieren entweder auf anderen Technologien oder müssen, bedingt durch die gewählte Architektur, das Routing sowie die Datenstromreplikation in anderer Form umsetzen. Ähnlichkeiten mit anderen globalen Sensornetzwerken sind daher nicht herleitbar.

Beide Problemstellungen sind zunächst theoretisch in dem Kapitel 5 gelöst und anschließend, wie im Kapitel 6 beschrieben, basierend auf dem System *StreamReUse*, praktisch realisiert worden. Dass die erarbeiteten Lösungen die Problemstellungen effektiv lösen, wird durch die Evaluationsergebnisse aus Kapitel 7 nachgewiesen.

Die Evaluation zeigt jedoch auch, dass noch weiteres Optimierungspotential bei der Datenstromreplikation sowie dem Anfragerouting innerhalb von globalen Sensornetzwerksystemen vorhanden ist. Wie bereits in Abschnitt 5.2 angesprochen, ist vor allem bei der Auswahl der zu replizierenden Anfragen eine weitere Optimierung möglich. Auch eine geschicktere Vorauswahl der Routingpunkte könnte zusätzliche Performance für das Gesamtsystem verfügbar machen.

Neben der aktuellen Untersuchung der durch den Broker bedienten Anfragen auf Inklusionen, wäre eine weitere Analyse bezüglich der gemeinsam genutzten Routingpunkte sinnvoll. Aktuell werden die gebündelten Anfragen auf das für sie errechnete globale Optimum repliziert. Dies muss aber nicht zwangsläufig für alle Anfragen tatsächlich ein lokales Optimum

darstellen. Die möglichen negativen Effekte, die sich für die Latenz einer Anfrage nach der Replikation dieser auf einen Knoten, der nicht ihr Optimum ist, ergeben, werden in diesem Fall durch die positiven Effekte der optimiert platzierten Anfragen überdeckt.

Ein erster Ansatz für eine optimierte Zusammenfassung könnten die an die Anfrage angehängten Routingtabellen darstellen, aus der die Menge der gemeinsamen möglichen Optima ableitbar sind. Die Entfernung der nicht optimal plazierbaren Anfragen aus der Menge der zu replizierenden Anfragen ist kein valider Ansatz, da dies im Extremfall zu einer Replikation einer einzelnen Anfrage führen könnte und somit keine Entlastung des replizierenden Brokers realisiert werden würde.

Optimierungen beim Routing der Anfragen wären, neben der geschickten Vorauswahl der möglichen Routingknoten, durch die Anpassung der umgesetzten *strikten single hop routing* Strategie möglich. Hierzu müsste der Zeitpunkt zu dem von einer ausreichenden Anzahl an möglichen Replikationspunkten ausgegangen werden kann genauer spezifiziert werden. Alternativ könnte auch ein anderes Kriterium für den Wechseln zu dem *direct routing* entwickelt werden. Dies würde, nach einer ausreichend genauen Bestimmung des Kriteriums, die Umsetsetzung *loose single hop routing* Strategie ermöglichen.

Die Realisierung des aufgezeigten weiteren Optimierungspotentials ist problemlos auf Basis des im Rahmen der Diplomarbeit entwickelten Systems *StreamReUse Plus* möglich.

Abschließend kann somit festgestellt werden, dass sowohl eine Optimierung der Replikation als auch eine Anpassung des Anfrageroutings gemäß der Aufgabenstellung der Diplomarbeit *Optimierte Datenstromverwaltung für globale Sensornetze* umgesetzt wurde, aber noch weiteres Optimierungspotential vorhanden ist.

A. Anhang

A.1. Begründete Verwerfen gleicher Anfragen

Bedingt durch das übereinstimmenden Endresultat kann als Abstraktion angenommen werden das gleiche Anfrage die von dem gleichen Client in das Netzwerk eingefügt werden nur von einem Broker innerhalb des Netzwerkes angenommen und verarbeitet werden. Die Übereinstimmung des Resultates unabhängig von der Anfrage wird in der angefügten Abbilung schematisch nachgewiesen.

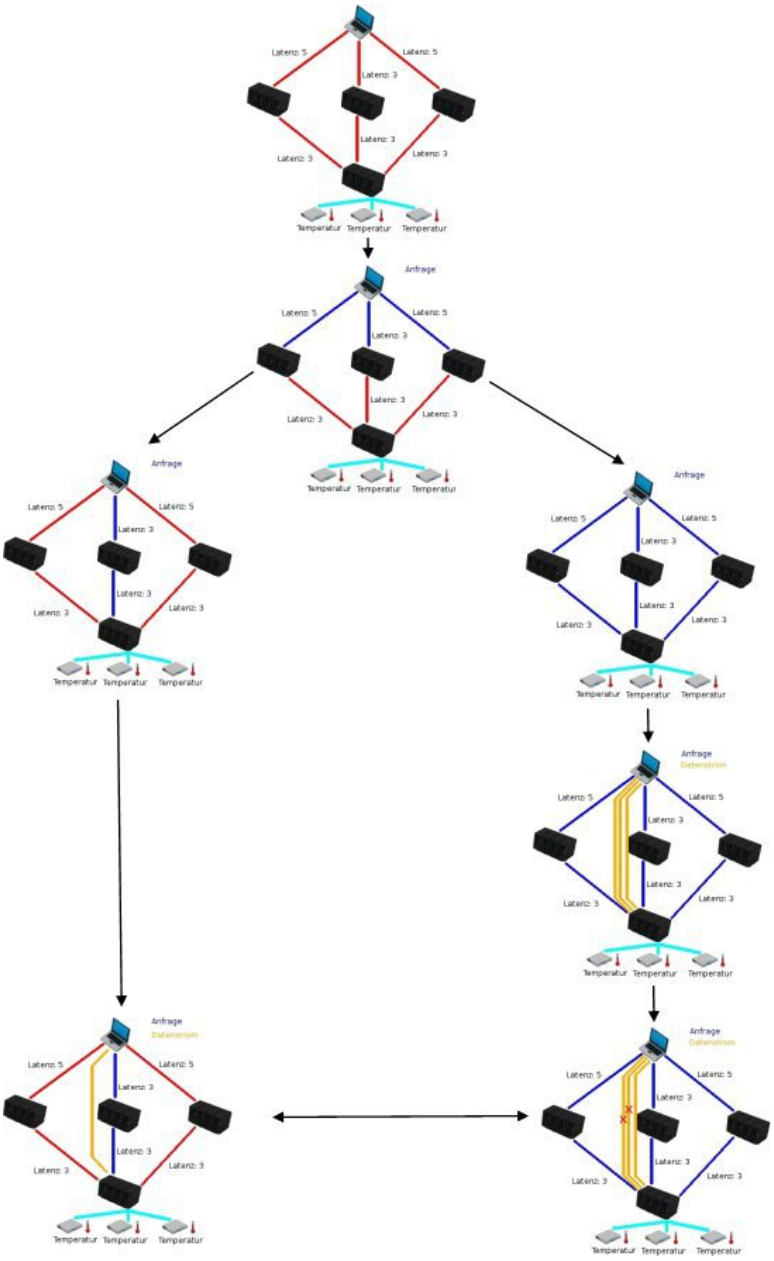

Abbildung A.1.: Schematische Darstellung des Indexierungsalgorihtmus

A.2. Netzwerkstruktur *StreamReUse Plus*

A.2.1. Netzwerkstruktur mit DZ-Ausruck

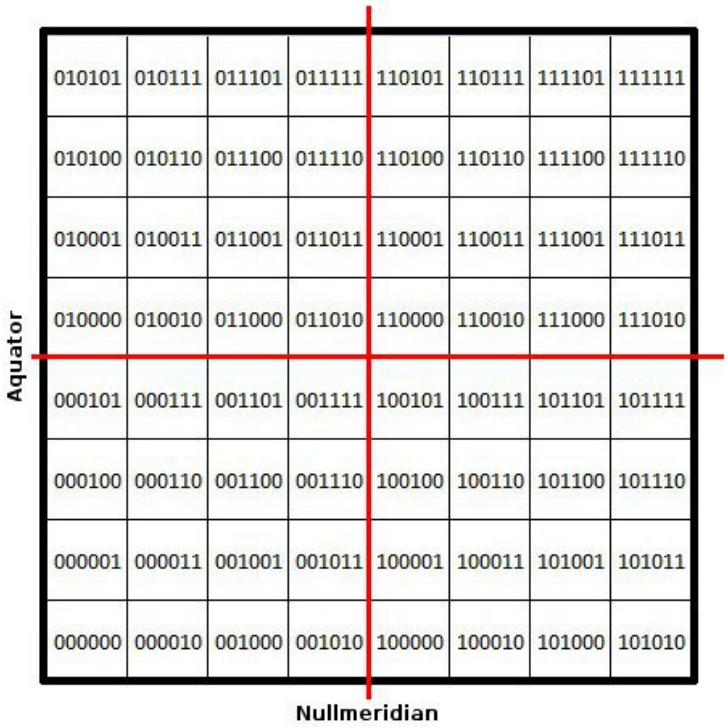

Abbildung A.2.: Schematische Darstellung des Indexierungsalgorihtmus

A.2.2. Netzwerkstruktur mit dezimaler Brokernummerierung

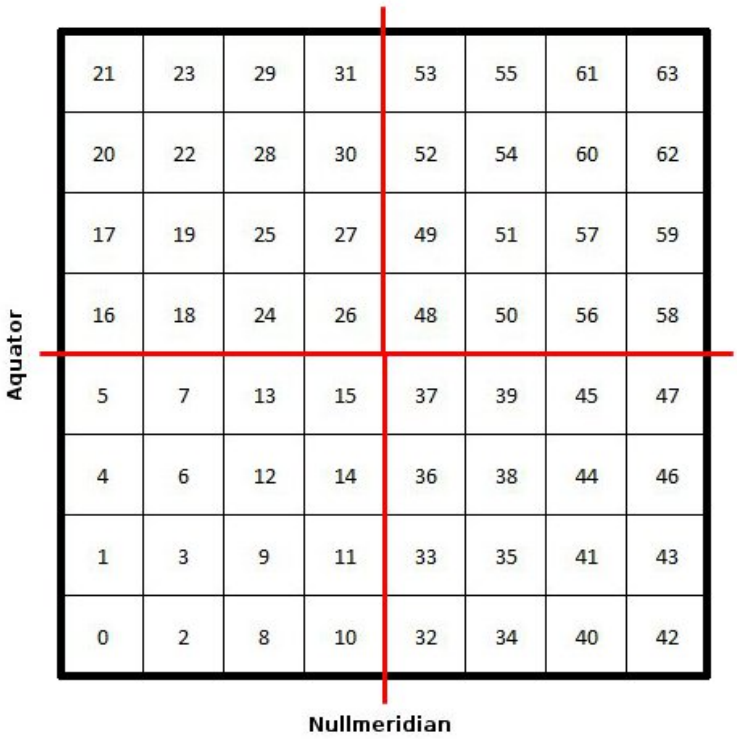

Abbildung A.3.: Schematische Darstellung des Indexierungsalgorihtmus

A.3. Vergleichsschema möglicher Routingpfade

Das heir dargestellte Schema zeigt die möglichen Routingpfade, abhänig von der realisierten Strategie.

Rot ist eine *multi hop routing* Pfad abgebildet. Türkis ist das Extrma *single hop routing* und blau das konträre Extrema *direct routing*.

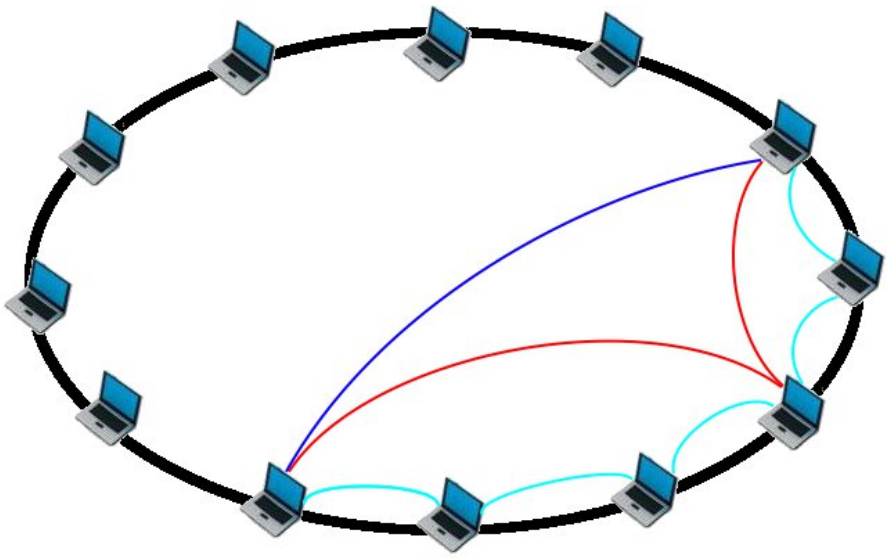

Abbildung A.4.: Schematische Darstellung der möglichen Routingpfade

A.4. Dijkstra Algorithmus

Beschreibung der Funktionsweise des Dijkstra Algorithmus[Gru10].

1. Initialisierung aller Knoten mit der Distanz unendlich
2. Initialisiere die Distanz des Startknotens mit 0
3. Markierung der Distanz des Startknotens als permanent
4. Markiere alle anderen Distanzen als temporär.
5. Setze Startknoten als aktiv.
6. Berechne die temporären Distanzen aller Nachbarknoten des aktiven Knotens durch Addition von dessen Distanz mit den Kantengewichten.
7. Wenn eine solche berechnete Distanz für einen Knoten kleiner ist als die aktuelle aktualisiere die Distanz und setze den aktuellen Knoten als Vorgänger.
   Dieser Schritt wird auch als Update bezeichnet und ist die zentrale Idee von Dijkstra.
8. Wähle einen Knoten mit minimaler temporärer Distanz als aktiv. Markiere seine Distanz als permanent.

Wiederhole 4-7, bis es keinen Knoten mit permanenter Distanz gibt, dessen Nachbarn noch temporäre Distanzen haben.

Umsetzung als Pseudocode:

```
function Dijkstra(Graph, source):
for each vertex v in Graph  // Initialization
  dist[v] := infinity  // Abstand: Ausgangspunkt nach v = Inphinity
  previous[v] := undefined  // Vorausgehender Knoten im optimalen Pfad
  dist[source] := 0  // Abstand der Ausgangspunkts zu sich selbst
  Q := Set mit allen Knoten des Graphes // alle Knoten im Graph sind nicht optimiert,
                                        also Element von Q
  while Q is not empty:
    u := Element aus Q mit kleinster dist[ ]
    remove u from Q
    für jeden Nachbarknoten v aus u:
      alt := dist[u] + dist_between(u, v)
      if alt < dist[v]
        dist[v] := alt
        previous[v] := u
       return previous[ ]
```

A.5. Latenzberechnung

Zur Bestimmung der Latenz zwischen den Knoten des simulierten Netzwerkes wird zunächst folgenden Algorithmik angewendet.

1. Festelegung der Latenzen zu den durch das Overlay assoziierten Nachbarknoten.
   Formel zur Berechnung der Latenz:
   $Gesamtlatenz = Grundlatenz + Grundlatenz * [0.1, 0.9]$
2. Ermittlung der kürzesten Pfad zwischen zwei Knoten
   Anwendung des zuvor in Anhang A.4 erläuterten Dijkstra Algorithmus.
3. Aufsummierung aller Latenzen des Pfades.
4. Subtraktion eines Grundlatenz abhängigen Wertes.

Durch die Varainz jedes Latenzwertes mittels einer Zufallszahl die auf dem Intervall zwischen 0.1 und 0.9 zu bestimmen ist, kann folgendet allgemeingültiger Beweise geführt werden.

Sei die Latenz zwischen einem beliebigen Knoten a und eine beliebigen Knoten b mit AB definert. Des weiteren ist die Latenz zwischen dem zuvor bestimmten Knoten b und einem weiteren beliebigen Knoten c mit BC definiert. So lässt sich hieraus ableiten das für die Verbindung zwischen Knoten a und Knoten c gelten muss:

- $AB = Grundlatenz + Grundlatenz * v_{ab}$
- $BC = Grundlatenz + Grundlatenz * v_{bc}$
- $AC = 2 * Grundlatenz + Grundlatenz * v_{ab} + Grundlatenz * v_{bc} - Grundlatenz * 0.1$

Aus diesen verbindlichen Kriterien lässt sich festlegen das gelten muss: $AC < AB + BC$, da:

$2 * Grundlatenz + Grundlatenz * v_{ab} + Grundlatenz * v_{bc} - Grundlatenz * 0.1 <$
$2 * Grundlatenz + Grundlatenz * v_{ab} + Grundlatenz * v_{bc}$

Somit ergibt sich für alle Knotenn innerhalb des Netzwerkes die in Abbildung A.5 dargestellte Relation.

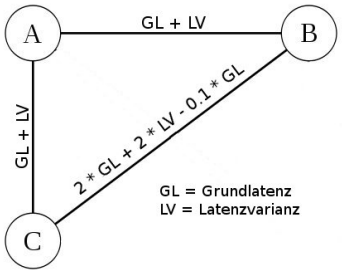

Abbildung A.5.: Bindende Dreiecksungleichung für Netzwerke

A.6. Konfiguration der Evaluationsumgebung

```
#random seed - Alle verwedente SEED zur Dokumentation
#SEED 4214354122955550635
#SEED 4214354122955550633
#SEED 4214354122955550630
#SEED 4214354122955550626
#SEED 4214354122755650621
#SEED 4214354122955550521
#SEED 4214354122955520521
#SEED 4214354122955541621
#SEED 4214354122918551671
#SEED 4214374122955550621
#SEED 4214354122755550621
#SEED 4214354122345550621
#SEED 4214354122955550621
#SEED 4214354122957540625
#SEED 4214354222955550538
#SEED 4214254222955550621
#SEED 4274554122955550621
#SEED 4284359122955550621
#SEED 5214359122955550621
#SEED 5214654122955550627
#SEED 4214354122955550656

NUMGENTYPE 0
ZIPFSKEW 3.0
MAXLENGTH 10

BROKERBITS 3
INTERMEDIATEBITS 2
RESOLUTIONBITS (BROKERBITS+INTERMEDIATEBITS)

MAXLOAD 1.0

IDX_DIMS 2
DIMENSIONS 4

#DATABASE jdbc:postgresql://nimbus.informatik.uni-stuttgart.de/gsg
DATABASE

# Number of Clientnodes which will included in network
# Optimum: Every Query that will appears will be served by on Client
CLIENTNODES 10
```

```
# Minimum Latency-value
LATENCY 10

# Files to link
UNDERLAY topologyUnderlay.cfg
OVERLAY topologyOverlay.cfg

# Balance between latency and load reduction
# Total amount of coefficients should be 10
# Only integer values are valid
LATENCY_COEFFICENT 5
LOAD_COEFFICENT 5

# Boolean to activated or deactivated plus Routing
#PLUS_REPLICATION true
PLUS_REPLICATION false

# Boolean to activated or deactivated plus Replication
#PLUS_ROUTING true
PLUS_ROUTING false

# Value needed for esablishing underlay
# The ratio between the time units used in the configuration
#file and the time units used in the Peersim simulator.
RATIO 1.0

# network size
NETSIZE 2^(BROKERBITS*2) + CLIENTNODES

# data size
DATASIZE_0 2^(RESOLUTIONBITS)
DATASIZE_1 DATASIZE_0

# data size in x, y and t resolution
DATASIZE_2 DATASIZE_0
DATASIZE_3 DATASIZE_0
DATASIZE_4 DATASIZE_0


MAX_QUERYSIZE_0 DATASIZE_0
MAX_QUERYSIZE_1 DATASIZE_1
MAX_QUERYSIZE_2 DATASIZE_2
MAX_QUERYSIZE_3 DATASIZE_3
MAX_QUERYSIZE_4 DATASIZE_4
```

```
MAX_IN_BANDWIDTH 25*(DATASIZE_0^DIMENSIONS)/NETSIZE
MAX_OUT_BANDWIDTH 25*(DATASIZE_0^DIMENSIONS)/NETSIZE

# parameters of periodic execution
CYCLES 7500000
CYCLE NETSIZE*100
QUERYCYCLE 5000

# parameters of message transfer
MINDELAY 0
MAXDELAY 0
DROP 0

GENERATE_TOPOLOGY true

random.seed SEED

network.size NETSIZE
network.node GeneralNode

simulation.endtime CYCLE*CYCLES
simulation.logtime CYCLE

simulation.experiments 1

################### protocols ===========================

protocol.link peersim.core.IdleProtocol

protocol.sru StreamReuse
{
  linkable link
  step 2
  from 1
  transport ut
}

protocol.urt UniformRandomTransport
{
  mindelay (CYCLE*MINDELAY)/100
  maxdelay (CYCLE*MAXDELAY)/100
}

protocol.ut UnreliableTransport
```

```
{
  transport urt
  drop DROP
}

################### initialization =====================

init.topology TopologyGenerator

init.baselink WireFromFile
{
  protocol link
  file topologyOverlay.cfg
}

init.sch CDScheduler
{
  protocol sru
}

init.initializer Initializer
{
  protocol sru
  log4jconfig log4j.cfg
}

include.init topology baselink sch initializer

################ controls ==============================

control.0 QueryGenerator
{
  protocol sru
  step CYCLE
  type NUMGENTYPE
}

control.1 BandwidthObserver
{
  protocol sru
  step CYCLE
  from CYCLE-1
}

#End of config file
```

A.7. Zusammengefasste Übersichtstabellen

A.7.1. Übersichtstabelle der durchschnittlichen Latenzen

SEED	Durchschnittliche Latenz ohne Replikationsoptimierung	Durchschnittliche Latenz mit Replikationsoptimierung	Durchschnittliche Latenz mit Replikationsoptimierung und Routing
1	30,59	25,75	25,17
2	28,84	27,83	26,04
3	30	27,2	26,9
4	28,88	26,9	25,86
5	28,81	25,25	24
6	28,33	27,26	23,21
7	28,34	26,97	23,8
8	28,53	27,85	24,65
9	28,52	26,61	23,83
10	28,75	25,26	25,26
11	29,83	27,08	25,77
12	29,72	28,33	28,31
13	28,79	26,6	25,48
14	29,16	29,95	25,08
15	29,32	26,93	22,89
16	29,6	29,14	27,41
17	28,69	25,69	24,03
18	35,15	27,03	26,03
19	31,01	28,19	25,58
20	28,21	27	23,83

Tabelle A.1.: Durchschnittlichen Latenzen einer Anfrage nach der Replikation

A.7.2. Übersichtstabelle des Anfrageroutings

SEED	durchschnittliche pro Anfrage passierte Routingpunkte	Vergleich	Vergleiche pro passierten Routingpunkt	passierte Routingpunkte
1	4,33	16,33	3,77	11663
2	4,17	15,58	3,74	11663
3	4,17	15,58	3,74	11663
4	4,23	15,58	3,68	11663
5	4,37	16,36	3,74	11243
6	4,28	16,04	3,75	10784
7	4,33	16,13	3,73	10835
8	4,29	15,85	3,69	11822
9	4,28	15,8	3,69	11579
10	4,22	15,88	3,76	60
11	4,32	16,19	3,75	12094
12	4,32	16,12	3,73	11378
13	4,3	16,07	3,74	11154
14	4,3	15,92	3,70	12811
15	4,25	15,93	3,75	10871
16	4,34	16,32	3,76	11438
17	4,25	15,88	3,74	11389
18	4,31	16,04	3,72	10558
19	4,24	15,94	3,76	9712
20	4,26	15,97	3,75	10462

Tabelle A.2.: Durchschnittlicher Aufwand des Anfrageroutings

Literaturverzeichnis

[Adl11] Y. Adler. Replikation in Datenbanken. In *Vortrag: Datenbanken II*. Hochschule für Technik, Wirtschaft und Kultur Leipzig Fachbereich Informatik, Mathematik und Naturwissenschaften, 2011. (Zitiert auf Seite 27)

[AHS06] K. Aberer, M. Hauswirth, A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pp. 1199–1202. VLDB Endowment, 2006. URL http://dl.acm.org/citation.cfm?id=1182635.1164243.

[ARU] U. C. for Atmospheric Research (UCAR). Earth System Grid. URL http://www.earthsystemgrid.org/. (Zitiert auf Seite 14)

[BBB+05] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, D. Williams. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. *Proceedings of the IEEE*, 93(3):485 –495, 2005. doi:10.1109/JPROC.2004.842745. (Zitiert auf Seite 14)

[BKC+01] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Comput.*, 27:1457–1478, 2001. doi:10.1016/S0167-8191(01)00099-0. URL http://dl.acm.org/citation.cfm?id=543586.543590. (Zitiert auf Seite 17)

[BKR11] A. Benzing, B. Koldehofe, K. Rothermel. Efficient support for multi-resolution queries in global sensor networks. In *Proceedings of the 5th International Conference on Communication System Software and Middleware*, COMSWARE '11, pp. 11:1–11:12. ACM, New York, NY, USA, 2011. doi:http://doi.acm.org/10.1145/2016551.2016562. URL http://doi.acm.org/10.1145/2016551.2016562. (Zitiert auf den Seiten 9, 13, 17 und 18)

[BKVR10] A. Benzing, B. Koldehofe, M. Volz, K. Rothermel. Multilevel Predictions for the Aggregation of Data in Global Sensor Networks. In *Proceedings of the 2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '10, pp. 169–178. IEEE Computer Society, Washington, DC, USA, 2010. doi:http://dx.doi.org/10.1109/DS-RT.2010.26. URL http://dx.doi.org/10.1109/DS-RT.2010.26. (Zitiert auf Seite 9)

[CDK+03] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37:298–313, 2003. doi:http://doi.acm.org/10.1145/1165389.945474. URL http://doi.acm.org/10.1145/1165389.945474. (Zitiert auf Seite 16)

[CDKR02] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *Selected Areas in Communications, IEEE Journal on*, 20(8):1489 – 1499, 2002. doi:10.1109/JSAC.2002.803069. (Zitiert auf Seite 16)

[CGN+05] J. Campbell, P. B. Gibbons, S. Nath, P. Pillai, S. Seshan, R. Sukthankar. IrisNet: an internet-scale architecture for multimedia sensors. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pp. 81–88. ACM, New York, NY, USA, 2005. doi:http://doi.acm.org/10.1145/1101149.1101162. URL http://doi.acm.org/10.1145/1101149.1101162. (Zitiert auf Seite 14)

[EJE05] K. Eriksson, C. Johnson, D. Estep. Optimierung. In *Angewandte Mathematik: Body and Soul*, pp. 909–923. Springer Berlin Heidelberg, 2005. URL http://dx.doi.org/10.1007/3-540-27531-2_7.

[Fou11] T. A. S. Foundation. Log4j Homepage, 2011. URL http://logging.apache.org/log4j/. (Zitiert auf Seite 45)

[GLR04] A. Gupta, B. Liskov, R. Rodrigues. Efficient Routing for Peer-to-Peer Overlays, 2004. (Zitiert auf Seite 17)

[Gru10] B. Gruppe. *Graphsuchalgorithmus: Dijkstra-Algorithmus, Algorithmus Von Kruskal, Algorithmus Von Prim, Breitensuche, A*-Algorithmus, D*-Algorithmus*. General Books LLC, 2010. URL http://books.google.de/books?id=SJWkcQAACAAJ. (Zitiert auf Seite 58)

[Hro] P. D. J. Hromkovic. Zufall und zufallsgesteuerte Algorithmen. Lehrstuhl für Informatik I RWTH Aachen 52056 Aachen. URL http://ddi.cs.uni-potsdam.de/HyFISCH/Spitzenforschung/Hromkovic.pdf. (Zitiert auf Seite 32)

[MJJV09] A. Montresor, M. Jelasity, G. P. Jesi, S. Voulgaris. PeerSim: A Scalable P2P Simulator. In *P2P'09 - Proceedings of the 9th International Conference on Peer-to-Peer*, pp. 99–100. Seattle, WA, 2009. doi:http://dx.doi.org/10.1109/P2P.2009.5284506. URL http://peersim.sourceforge.net/. (Zitiert auf den Seiten 18 und 29)

[PGVA08] P. Padmanabhan, L. Gruenwald, A. Vallur, M. Atiquzzaman. A survey of data replication techniques for mobile ad hoc network databases. *The VLDB Journal*, 17:1143–1164, 2008. doi:http://dx.doi.org/10.1007/s00778-007-0055-0. URL http://dx.doi.org/10.1007/s00778-007-0055-0. (Zitiert auf Seite 27)

[SPL+04] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, M. Welsh. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical report, Harvard University, 2004. (Zitiert auf Seite 15)

[TKK+10] M. A. Tariq, G. G. Koch, B. Koldehofe, I. Khan, K. Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *Proceedings of the 16th international Euro-Par conference on Parallel processing: Part I*, EuroPar'10, pp. 458–470. Springer-Verlag, Berlin, Heidelberg, 2010. URL http://dl.acm.org/citation.cfm?id=1887695.1887746. (Zitiert auf den Seiten 16 und 17)

[TKKR09] M. A. Tariq, B. Koldehofe, G. G. Koch, K. Rothermel. Providing Probabilistic Latency Bounds for Dynamic Publish/Subscribe Systems. In *Kommunikation in Verteilten Systemen (KiVS)*, Informatik aktuell, pp. 155–166. Springer Berlin Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-540-92666-5_13.

[TM03] M. Tubaishat, S. Madria. Sensor networks: an overview. *Potentials, IEEE*, 22(2):20 – 23, 2003. doi:10.1109/MP.2003.1197877.

[WS09] M. Wählisch, T. C. Schmidt. An a priori estimator for the delay distribution in global hybrid multicast. In *Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*, Co-Next Student Workshop '09, pp. 19–20. ACM, New York, NY, USA, 2009. doi:http://doi.acm.org/10.1145/1658997.1659008. URL http://doi.acm.org/10.1145/1658997.1659008. (Zitiert auf Seite 16)

Erklärung

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

(Stefan Wenz)