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

Diplomarbeit Nr. 2818

Realzeitverarbeitung von Datenströmen in globalen Sensornetzen

Marco Völz

Studiengang: Softwaretechnik

Prüfer: Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer: Dipl.-Inf. Andreas Benzing,
Dr. Boris Koldehofe

begonnen am: 23. Juni 2008
beendet am: 23. Dezember 2008

CR-Klassifikation: C.2.4, C.4, G.2.2, G.3, I.4.4

Kurzfassung

Im Bereich der Sensornetze zeichnet sich der Trend ab, immer größere Verbunde einzelner Netze, sogenannte globale Sensornetze, zu bilden. Gleichzeitig wächst auch die Menge der potentiellen Konsumenten der Datenströme, die von Sensornetzen bereitgestellt werden.

Durch den neuen globalen Kontext ergeben sich für die zur Verarbeitung der Sensordatenströme eingesetzten Systeme neue Herausforderungen. Gerade in den Anforderungen bezüglich Skalierbarkeit und Echtzeitverarbeitung der Datenströme weisen verfügbare Systeme erhebliche Defizite auf.

Das in dieser Arbeit konzipierte System zur Realzeitverarbeitung von Datenströmen in globalen Sensornetzen adressiert genau diese Defizite. Ein verteilter R-Baum wird verwendet, um proaktiv Datenströme gleichen Inhalts zu vermeiden und so die Produzenten zu entlasten. Zusätzlich wird mit dem Prädiktor ein Framework zur modellgestützten Vorhersage von Datenströmen integriert, so dass aktuelle Werte auch über Verbindungen mit hoher Latenz in Echtzeit verfügbar sind.

Abstract

The ongoing trend in the field of sensor networks is to interconnect single networks to so called global sensor networks. At the same time, the amount of potential consumers of the data provided by sensor networks is growing steadily.

In this global setting new challenges and requirements arise for the systems used to process the streams of sensordata. The new requirements, especially those concerning scalability and real-time processing, are difficult to meet for the available systems.

The system for real-time processing of data streams in global sensor networks presented in this thesis adresses the main drawbacks of available solutions. It employs a distributed R-Tree to proactively avoid transmitting data streams with identical content and alleviate the producers' workload. Furthermore, predictors, a framework for the model-driven forecast of data streams, are integrated into the system. This predictors can be used to guarantee the availability of real-time data for the consumer even over high-latency connections.

Inhaltsverzeichnis

1 Einleitung **9**

2 Grundlagen **13**
2.1 Globale Sensornetze . . . 13
2.2 Stream Processing Systems . . . 15
2.2.1 Distributed Stream Processing Systems . . . 19

3 Stream Processing im Kontext von globalen Sensornetzen **25**
3.1 Problemstellung . . . 26
3.2 Systemmodell . . . 29
3.3 Anwendungsszenario . . . 31

4 Verwandte Arbeiten **35**
4.1 Globale Sensornetze und Stream Processing Systems . . . 35
4.2 Verteilte Lösungen auf Basis von R-Bäumen . . . 38
4.3 Vorhersageverfahren . . . 41

5 Konzepte **45**
5.1 Anforderungen an den Algorithmus . . . 46
5.2 Architektur des Gesamtsystems . . . 50
5.3 Konzeption eines Distributed Stream Processing Systems . . . 52
5.3.1 Einfügen neuer Anfragen . . . 52
5.3.2 Auffinden geeigneter Datenquellen und Verarbeiten des Datenstroms . 55
5.3.3 Initialisierung der Baumstruktur . . . 56
5.3.4 Aufteilen eines übergelaufenen Knotens . . . 57
5.4 Der Algorithmus im Detail . . . 58
5.4.1 Einfügen neuer Anfragen . . . 59
5.4.2 Auffinden geeigneter Datenquellen . . . 62
5.4.3 Verarbeiten des Datenstroms . . . 65
5.4.4 Aufteilen eines übergelaufenen Knotens . . . 66
5.4.5 Inhärente Eigenschaften des Verfahrens . . . 69
5.5 Prädiktoren in einem Distributed Stream Processing System . . . 69
5.5.1 Vorhersagemodelle . . . 71

5.5.2 Integration von Prädiktoren 74
5.5.3 Auswirkungen der Prädiktoren auf die Architektur 77

6 Evaluation **81**
6.1 Ablauf der Experimente 81
6.1.1 Testszenario 1: 10 Anfragen 82
6.1.2 Testszenario 2: 100 Anfragen 83
6.1.3 Testszenario 3: 1000 Anfragen 83
6.2 Beschreibung der Experimente 83
6.2.1 Mehraufwand für Peers 84
6.2.2 Lokalität von Ereignissen 85
6.2.3 Entlastung der Gateways 86
6.2.4 Effekt von Prädiktoren 87
6.3 Bewertung der Simulationsergebnisse 90

7 Zusammenfassung und Ausblick **93**

Literaturverzeichnis **97**

Abbildungsverzeichnis

2.1 Aufbau eines Stream Processing Systems 15
2.2 Die „in-stream“ Verarbeitung von Nachrichten 16
2.3 Funktionsweise eines Fensteroperators 17
2.4 Kostenminimierung durch Neuplatzieren der Operatoren in SBON 19
2.5 Beispiel eines R-Baumes 21

3.1 Das Systemmodell 32
3.2 Kleinere Anfragen liefern Teilergebnisse für größere Anfragen 33

5.1 Vergleich der Datenverteilungsgraphen 49
5.2 Die mit Gateways initialisierte Baumstruktur 49
5.3 Die Gesamtarchitektur des konzipierten Systems 50
5.4 Der Zusammenhang von R-Baum und Datenverteilungsgraphen 51
5.5 Einfügen einer neuen Anfrage 53
5.6 Unterteilung von sich überschneidenden Anfragen 56
5.7 Aufteilung eines überlaufenden Knotens 58
5.8 Architektur eines Knotens 64
5.9 Anwenden eines Operators 66
5.10 Bestimmung der Silhouette 67
5.11 Die Integration von Prädiktoren in die Gesamtarchitektur 76
5.12 Kommunikation mit und ohne Prädiktor 77
5.13 Der Prädiktionskreislauf 77

6.1 Anordnung der Sensoren im Intel Research Lab 82
6.2 Das Verhältnis zwischen verwalteter und angefragter Fläche 84
6.3 Das durchschnittliche Einfügelevel 85
6.4 Der Anteil über die Wurzel eingefügter Anfragen 86
6.5 Der Anteil der ersetzten Gateways 87
6.6 Der Anteil der in Szenario 1 und 2 gesendeten Daten 88
6.7 Der Anteil der in Szenario 3a und 3b gesendeten Daten 88
6.8 Das Intervall zwischen Modellaktualisierungen in Szenario 1 und 2 89
6.9 Das Intervall zwischen Modellaktualisierungen in Szenario 3a und 3b 90

Tabellenverzeichnis

2.1 Vergleich von DBMS, Rule engine und SPS 18

5.1 Interpretationen der Silhouette 68
5.2 Vergleich des DR-Trees mit dem in dieser Arbeit konzipierten DSPS 70
5.3 Vergleich von Kalman-Filter, Konstantenmodell und Box-Jenkins-Modell . . . 74

6.1 Zusammenfassung der Evaluationsergebnisse 92

Verzeichnis der Algorithmen

5.1 Aufsteigen einer neuen Anfrage 60
5.2 Absinken einer neuen Anfrage 60
5.3 Auswahl des besten Kindknotens zum Absinken einer neuen Anfrage 60
5.4 Überprüfen einer Liste von Knoten beim Einfügen 61
5.5 Anfragen von Datenverbindungen auf Basis einer Ergebnisliste 62
5.6 Aufbau einer Datenverbindung 63
5.7 Aufteilung eines übergelaufenen Knotens 68
5.8 K-Means-Clustering 69
5.9 Kommunikation mit Prädiktor 75

Kapitel 1

Einleitung

Mit der zunehmenden Ausbreitung globaler Sensornetze werden Informationen über verschiedene Aspekte der Welt einer weltweiten Gruppe von Interessenten zur Verfügung gestellt. Dabei kann es sich z. B. um Wetterdaten, Finanzdaten oder auch beliebige andere Daten handeln. Die Zahl der an den Messungen beteiligten Sensoren geht mittlerweile in die Milliarden, und in den nächsten Jahren ist damit zu rechnen, dass sich die Zahl der Sensoren weiter erhöhen wird. Mit der Zahl der verfügbaren Sensordaten steigt gleichzeitig auch die Zahl der Interessenten und damit die Gesamtzahl der auf der Welt gestellten Anfragen an die einzelnen Sensornetze. Entsprechend ist auch der Anteil der Sensordaten am weltweiten Datenverkehr gestiegen.

Durch die Menge und Verfügbarkeit der Sensorinformationen entstand der Wunsch, Ereignisse der realen Welt als Informationsquelle für Anwendungen überall auf der Welt zu verwenden. Sei es für eine Simulation des aktuellen Wetters in Amerika, die wesentliche Daten für die Wettervorhersage in Europa liefert, oder für eine Simulation der Weltwirtschaftslage, um den Kursverlauf einzelner Leitindizes vorhersagen zu können.

Ob die Nutzung von Echtzeitdaten in Simulationsanwendungen möglich ist, wird vor allem durch zwei Faktoren beeinflusst: Der erste ist das dadurch entstehende hohe Datenaufkommen. Wenn eine Simulation des Wetters in Nordamerika nun kontinuierlich von allen relevanten Sensoren gespeist werden soll, statt wie bisher mit stündlichen Momentaufnahmen, wird das resultierende Datenaufkommen wesentlich höher sein. Der zweite Faktor ist die inhärente Latenz eines Systems zur Verteilung von Sensordaten. Wenn große Datenströme quer über den Globus verschickt werden sollen, ist eine gewisse Verzögerung unvermeidlich. Sensordaten in Echtzeit verarbeiten zu können ist jedoch wünschenswert und – je nach Anwendungszweck – unter Umständen auch eine der zentralen Anforderungen einer Anwendung. Denkbar sind z. B. Anwendungen in der Wirtschaft, wo es im computergestützten Aktienhandel schon um Mikrosekunden in der Reaktionszeit geht. All das sind Anforderungen, mit denen sich Strukturen zur Organisation und Verteilung von Sensorinformationen beschäftigen müssen.

Im Bereich der Sensornetze hat sich Distributed Stream Processing (DSP) als ein wichtiges Kommunikationsparadigma herauskristallisiert. Dabei wird ein verteilter Graph aus

verschiedenen Operatoren aufgebaut, die eingehende Datenströme verarbeiten und in ausgehende Datenströme weiterleiten. Für niedrige Antwortzeiten in diesen Distributed Stream Processing Systems (DSPS) ist es unter anderem kritisch, die Netzlast so gering wie möglich zu halten und gleichzeitig für eine gerechte Verteilung der Lasten zu sorgen. Probleme, die direkte Auswirkungen auf die Last und deren Verteilung haben, wurden und werden in der Literatur bereits ausgiebig behandelt.

Einige der Fragen, die sich eher indirekt auf die Netzlast auswirken, werden in dieser Arbeit näher betrachtet werden. Darunter fallen z. B. Möglichkeiten, die räumlichen Ähnlichkeiten zwischen Anfragen – die in einem System der eingangs angesprochenen Größenordnung nahezu zwangsläufig auftreten werden – erkennen und ausnutzen zu können. Diese räumlichen Ähnlichkeiten können so ausgenutzt werden, dass die Netzlast gerechter verteilt wird. So kann vermieden werden, dass die gleiche Sensorinformation vom Produzenten an verschiedene Konsumenten geschickt werden muss. Stattdessen werden diejenigen, die bisher nur als Konsumenten von Informationen auftraten, auch für deren Weiterverteilung verantwortlich gemacht. Die Konsumenten müssen die Informationen, die sie von den Produzenten erhalten, auch untereinander teilen.

Ebenfalls wird der Frage nachgegangen, welche Ansätze verfolgt werden können, um trotz der Latenzen in einem Distributed Stream Processing System eine zeitnahe Verarbeitung der Sensordaten – bis hin zur Realzeitverarbeitung – zu ermöglichen. Um die Latenzen des Systems zu egalisieren, muss eine größtmögliche Unabhängigkeit von gesendeten Sensorinformationen erreicht werden. Auf das tatsächliche Senden von Sensordaten kann verzichtet werden, indem z. B. Modelle eingeführt werden, mit denen sich auf der Empfängerseite der jeweils nächste Sensorwert vorhersagen lässt. Somit arbeiten die Anwendungen, die Sensordaten anfordern, vollkommen autonom und können immer dann, wenn sie einen neuen Sensorwert benötigen, ihr Vorhersagemodell befragen. Weiterhin wird ein Mechanismus benötigt, der regelmäßig die Korrektheit des Modells überprüft. Inwieweit sich solche Ansätze in eine Verteilstruktur für Sensordaten integrieren lassen, welche Möglichkeiten zur Modellierung von Sensordaten bestehen und wie die Überprüfung solcher Modelle erfolgen kann, wird in dieser Arbeit ebenfalls genauer untersucht werden.

Die bisher verbreiteten Systeme zur Datenstromverarbeitung stoßen an ihre Grenzen, sobald der Kontext der verfügbaren Produzenten auf globale Sensornetze erweitert wird. Um den Defiziten dieser Systeme in den Bereichen Skalierbarkeit und Echtzeitverarbeitung zu begegnen, geht das in dieser Arbeit konzipierte System vor allem in zwei Punkten einen anderen Weg als bisher verfügbare Systeme: Durch die Verwendung eines verteilten R-Baumes zur Verwaltung der an das System gestellten Anfragen wird die Mehrfachübermittlung von identischen Datenströmen vermieden. Eine aufwändige Erkennung und Behebung einer solchen Mehrfachübermittlung wie in anderen System erübrigt sich somit. Der verteilte R-Baum wird auch dazu verwendet, um diejenigen Anwendungen, die Informationen vom System erhalten, dazu zu verpflichten, diese Informationen auch an andere Anwendungen weiterzuverteilen. Als zweiter wesentlicher Punkt werden in das hier vorgestellte System Prädiktoren zur modellgestützten Vorhersage von Datenverläufen integriert. Zwar wurden auch in anderen Systemen bereits Vorhersagemodelle eingesetzt, um die Menge der versendeten Pakete zu reduzieren, jedoch fand dies ausschließlich in sehr beschränkten

lokalen Umgebungen – vornehmlich einzelnen Sensornetzen – statt, um die Lebensdauer der Sensoren zu erhöhen. Im Gegensatz dazu werden die Prädiktoren in diesem System in der gesamten R-Baum-Struktur – genauer an jeder Stelle, an der Datenströme übertragen werden – eingesetzt, um das Datenaufkommen im Gesamtsystemzu reduzieren und die Verfügbarkeit von Datenströmen in Echtzeit gewährleisten zu können.

Der Rest der Arbeit gliedert sich wie folgt: Zunächst werden in Kapitel 2 die Grundlagen erläutert, auf deren Basis sich das konzipierte System in seinen Gesamtkontext einordnen lässt. In Kapitel 3 werden die generellen Probleme und Herausforderungen beim Entwurf eines Distributed Stream Processing Systems im Kontext der globalen Sensornetze dargelegt. Außerdem wird anhand der Defizite existierender Systeme dargelegt, warum es sich lohnt, ein neues Konzept für ein Distributed Stream Processing System zu erstellen. Auf Basis einiger vereinfachender Annahmen wird anschließend ein Systemmodell für das hier beschriebene Systems erstellt. Danach wird ein Anwendungsszenario für eine auf diesem Systemmodell basierende Anwendung beschrieben. Das darauffolgenden Kapitel 4 enthält die in der Literatur zu ähnlichen Problemen bereits behandelten Ansätze. Kapitel 5 stellt in zwei Teilen ein Konzept vor, mit dem die o. g. Herausforderungen angegangen werden. Dazu wird zunächst das Konzept für ein Distributed Stream Processing System entworfen. Dieses DSPS wird dann durch Vorhersagemodelle ergänzt, indem Prädiktoren in das vorgestellte DSPS integriert werden. Anschließend wird das beschriebene Gesamtkonzept in Kapitel 6 einer Evaluierung unterzogen und daraufhin bewertet, inwieweit es den vorher formulierten Anforderungen gerecht wird. Eine Zusammenfassung der Arbeit und ein Ausblick auf Herausforderungen, die bisher unbeantwortet bleiben mussten, erfolgt im abschließenden Kapitel 7.

Kapitel 2

Grundlagen

In diesem Kapitel wird die Basis für das in dieser Arbeit vorgestellte Konzept gelegt. Um das hier entworfene System zur Realzeitverarbeitung von Datenströmen in globalen Sensornetzen besser in seinen Kontext einordnen zu können, werden die betroffenen Teilgebiete im Detail betrachtet.

Zuerst erfolgt eine Einführung in den Aufbau, die Funktionsweise und die Aufgabengebiete von (globalen) Sensornetzen. Dann wird die dominierende Anwendungsklasse im Umgang mit Datenströmen, die Klasse der Stream Processing Systems, mitsamt ihren speziellen Anforderungen und Eigenschaften beschrieben. Da von Stream Processing Systems eine sehr hohe Skalierbarkeit gefordert wird, wurden diese um den Aspekt der Verteiltheit erweitert. Auf die spezielle Klasse der Distributed Stream Processing Systems, in die auch die in dieser Arbeit konzipierte Anwendung fällt, wird im darauffolgenden Abschnitt genauer eingegangen. Dabei werden zwei Eigenschaften, die von heutigen Systemen gefordert werden, einer gesonderten Betrachtung unterzogen: Die Unterstützung von Bereichsanfragen und die Realzeitverarbeitung von Datenströmen.

2.1 Globale Sensornetze

Ein *Sensornetz* (engl. wireless sensor network, WSN) bezeichnet eine Menge von Sensoren, die in einem lokalen Verbund – für gewöhnlich in einem drahtlosen Netzwerk – organisiert sind. Auf Grund der sinkenden Größe bei gleichzeitig immer preiswerteren Fertigungsmethoden steigt die Zahl der möglichen Einsatzmöglichkeiten für solche Sensoren.

Üblicherweise werden Sensornetze eingesetzt, um etwas oder jemanden zu untersuchen, zu überwachen oder zu kontrollieren. Das kann im zivilen Bereich geschehen, in dem z. B. Umweltbedingungen, die Tierwelt, eine Produktionsanlage oder ein Computersystem Gegenstand der Überwachung ist. Ebenso kann die Überwachung im militärischen Bereich erfolgen. Dort werden mit Hilfe von Sensornetzen z. B. Kriegsschauplätze und Krisengebiete überwacht oder Ziele für automatisierte Waffensysteme identifiziert.

In diesen Einsatzgebieten erfüllen Sensornetze ein großes Aufgabenspektrum. Dies verläuft von Meldungen über das Eintreffen oder Ausbleiben bestimmter Ereignisse über das Erkennen vorher definierter Zustände, eine Klassifikation erkannter Objekte, bis zur Verfolgung von Objekten im Einzugsbereich.

Die Sensoren bieten also einen bestimmten Dienst oder eine gewünschte Information an. Ein Interessent, der Zugang zu einer solchen Auskunft benötigt, stellt eine Anfrage an ein Sensornetz. Dies geschieht für gewöhnlich über einen spezielles System, das den Zugang zum Sensornetz bereitstellt und Verwaltungsaufgaben innerhalb des Sensorenverbundes übernimmt. Dieser Zugang wird als *Gateway* bezeichnet. Dort wird die komplexe Anfrage des Interessenten in kleinere Teile aufgebrochen und auf einzelne Sensoren verteilt. Hat der Gateway alle nötigen Daten von den durch ihn verwalteten Sensoren erhalten, sendet er eine Antwort an den Interessenten. Die Kommunikation innerhalb eines Sensornetzes verläuft oftmals so, dass die Sensoren proaktiv ihre Messungen an einen Zwischenspeicher, die sogenannte *Senke*, übermitteln. Wird eine Anfrage an das Netz gestellt, muss der Gateway – der z. B. auch gleichzeitig die Senke eines Sensornetzes d – nicht erst die Antworten aller Sensoren abwarten, sondern kann direkt die aktuellen Werte aus dem Zwischenspeicher übermitteln.

Die Ergebnisse, die Sensornetze ihren jeweiligen Interessenten liefern, sind ebenso vielfältig wie ihre Einsatzgebiete und Aufgaben. Es kann sich um einen konstanten Datenstrom handeln, falls ein Interessent die dauerhafte Überwachung eines Objektes oder einer Umgebung angefordert hat. Soll ein Zustand überwacht werden, besteht die Antwort eines Sensornetzes meist aus der Meldung „interessanter“ Einzelereignisse. Weiterhin denkbar sind Antworten, die eine bestimmte Menge von Einzelereignissen zu einem komplexen Ereignis kombinieren und zurückliefern. Schon anhand dieses keineswegs vollständigen Auszugs sollte die große Bandbreite der möglichen Ergebnisarten deutlich werden.

Trotz ihrer Unterschiedlichkeiten bezüglich Einsatzgebiet, Anwendungszweck und geliefertem Ergebnis weisen verschiedene Sensornetze einige gemeinsame Charakteristika auf. Bei einem Sensornetz handelt es sich im Regelfall um einen Verbund einer sehr großen Zahl von Sensoren, die über sehr stark begrenzte Ressourcen verfügen. Die Sensoren kommunizieren untereinander und mit ihrem Gateway für gewöhnlich über eine drahtlose Verbindung. Viele ihrer Einsatzgebiete befinden sich in Bereichen, die für den Menschen nicht oder nur schwer zugänglich sind. Ob es sich um eine Messung in einem Krisengebiet oder eine Unterwassermessung handelt – klar ist, dass die Sensorenhardware extremen Außenbedingungen ausgesetzt ist. Der Ausfall einzelner Sensoren ist somit wahrscheinlich und im Hinblick auf ihre sehr begrenzten Energievorräte schließlich auch unvermeidlich.

Im Bereich der Sensornetze ist die momentane Entwicklung eindeutig: Immer mehr Sensoren sollen einer stetig wachsenden Menge von Interessenten immer detailliertere und umfassendere Informationen bereitstellen. Die ursprünglich nur im lokalen Kontext verwendeten Sensornetze werden zu globalen Verbünden zusammengeschlossen. Man spricht dabei von *globalen Sensornetzen* (engl. global sensor networks, GSN). Mit globalen Sensornetzen können, zum Beispiel auf Grund der wesentlich größeren Flächenabdeckung und der Kombination von Daten unterschiedlicher Sensortypen, Anfragen einer neuen Qualität beantwortet werden. Während heute Sensornetze[ARE+05] mit einer Größenordnung von 1000 Sensoren

auf einer Fläche von 1300 × 300 m zu den größten der Welt zählen[1], ist die Vision klar: Es wird darauf hingearbeitet, ein sowohl in der Fläche als auch in Bezug auf die Vielfalt darin verfügbarer Sensordaten umfassendes globales Sensornetz erstellen zu können.

Um auf die Herausforderungen reagieren zu können, die sich zum einen aus den erwähnten Charakteristika der einzelnen Sensornetze und zum anderen aus deren Zusammenschluss zu globalen Sensornetzen ergeben, wurde eine neue Klasse von Systemen entworfen. Diese Klasse von Systemen zur Verarbeitung von Datenströmen (engl. stream-processing system, SPS) wird im nachfolgenden Abschnitt genauer erläutert.

2.2 Stream Processing Systems

Die Aufgabe eines Stream Processing Systems kann abstrakt formuliert werden: Einer Masse von Konsumenten – den Interessenten oder Anfragestellern – eine Menge an Informationen bereitstellen bzw. zugänglich machen. Die Informationen stammen von Produzenten – z. B. den Sensoren bzw. den Gateways zu Sensornetzen. Das bedeutet also, dass Interessenten ihre Anfragen an ein SPS richten, statt direkt die betreffenden Produzenten anzusprechen. Das Stream Processing System übernimmt dann die Aufgabe, die Anfragen an die einzelnen Produzenten weiterzuleiten und schließlich die gewünschte Information zu übermitteln.

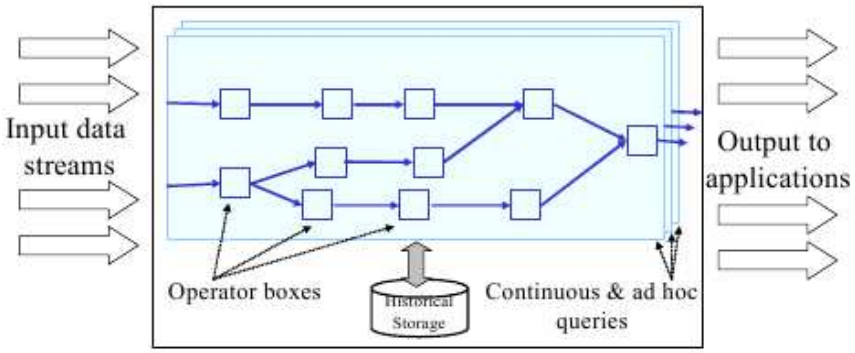

Abbildung 2.1: Aufbau eines Stream Processing Systems: Die Datenströme der Produzenten (links) werden durch Operatoren bearbeitet und in Beziehung zueinander gesetzt, bevor sie an die Konsumenten (rechts) weitergeleitet werden [CBB+03]

Die Abbildung 2.1 zeigt den Aufbau eines Stream Processing Systems als Vermittler von Informationen zwischen Produzenten und Konsumenten.

Die Notwendigkeit, eine neue Klasse von Anwendungen zu entwerfen, ergab sich aus den Anforderungen an Systeme, die uzr Verarbeitung von Datenströmen eingesetzt werden

[1] ExScal: http://cast.cse.ohio-state.edu/exscal/

sollten. Die Autoren von [ScZ05] bestimmen acht Anforderungen, die eine Anwendung erfüllen muss, die Datenströme verarbeitet.

Anforderung 1: Die Anwendung muss Nachrichten in Echtzeit während des Flusses (engl. in-stream) verarbeiten können, ohne diese vorher zwischenspeichern zu müssen. Der Verarbeitungsprozess wird wesentlich verlängert, wenn übermittelte Nachrichten erst in einer Datenbank abgelegt werden müssen. Die Abbildung 2.2 zeigt den Ablauf der geforderten Verarbeitungsweise.

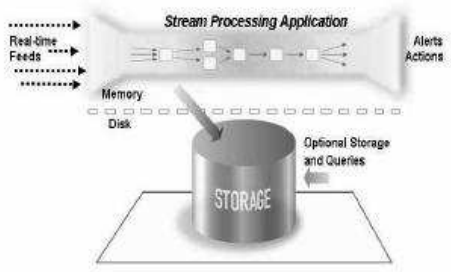

Abbildung 2.2: Anforderung 1: Die „in-stream“ Verarbeitung von Nachrichten mit *optionalem* Zwischenspeicher [ScZ05]

Anforderung 2: Die Anwendung muss eine Anfragesprache einsetzen, die Unterstützung für datenstromorientierte Operatoren bietet. Die Autoren nennen diese Sprache „StreamSQL“; ein um Fensteroperatoren und Merge erweiterter SQL-Dialekt. Die Abbildung 2.3 zeigt einen Fensteroperator (engl. sliding window operator). Dieser erhält mehrere Sensorwerte als Eingabe, die er z. B. zu einem Mittelwert aggregiert. Zwei Parameter bestimmen das Verhalten eines Fensteroperators: Die Fenstergröße und der Versatz. Mit Fenstergröße wird die Zahl der Eingabewerte für den Operator bezeichnet. Nachdem der Operator ausgeführt wurde, wird das Fenster um einen bestimmten Wert weiterverschoben, bevor er erneut ausgeführt wird. Die Länge dieser Verschiebung bestimmt der Versatz. Aus der Abbildung 2.3 wird auch deutlich, dass sich die Fenster der einzelnen Ausführungen überlappen können.

Anforderung 3: Die Anwendung muss Mechanismen bieten, die verminderte Zuverlässigkeit abzufangen, die sich aus der Echtzeitverarbeitung von Datenströmen ergibt. Ein Operator, der Daten aus mehreren Quellen benötigt, muss den Fall berücksichtigen, dass ihn die Daten aus einer seiner Quellen verspätet oder auch gar nicht erreichen. Weiterhin ist es möglich, dass die Einzelereignisse des Datenstroms in einer anderen Reihenfolge eintreffen, als sie erzeugt wurden. Auch darauf muss die Anwendung reagieren können, ohne die gesamte Datenstromverarbeitung zu blockieren.

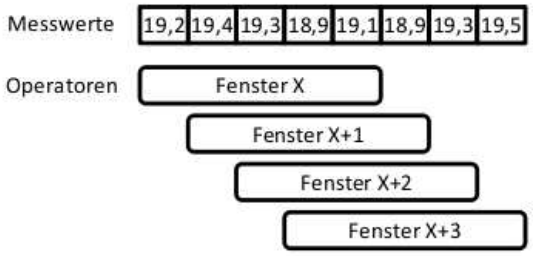

Abbildung 2.3: Funktionsweise eines Fensteroperators mit der Breite 5 und dem Versatz 1

Anforderung 4: Die Anwendung muss berechenbare und wiederholbare Ergebnisse produzieren. Gerade im Hinblick auf Replikation und Fehlertoleranz ist es wichtig, dass die Operatoren der Anwendung deterministisch arbeiten und reproduzierbare Ergebnisse liefern.

Anforderung 5: Die Anwendung muss in der Lage sein, Datenströme mit in einer Datenbank gespeicherten Zuständen zu kombinieren. Anwendungen, die z. B. auf das Eintreten eines bestimmten Zustandes reagieren sollen, müssen in der Lage sein, diesen Zustand anhand bestimmter Muster im Datenstrom zu erkennen. Außerdem muss die Anwendung sowohl einen Strom aufgezeichneter Daten als auch einen „Live"-Datenstrom verarbeiten können und nahtlos zwischen diesen Eingabearten wechseln können.

Anforderung 6: Die Anwendung muss spezielle Mechanismen bieten, die eine hohe Verfügbarkeit gewährleisten. Da in den meisten Einsatzgebieten von Stream Processing Systems der Faktor Zeit eine tragende Rolle spielt, kann im Fehlerfall nicht darauf gewartet werden, bis die Anwendung, bzw. Teile der Anwendung, neu gestartet und z. B. aus einer Logdatei heraus mit der letzten bekannten Konfiguration initialisiert wurden. Stattdessen sind Lösungen gefragt, die durch Replikation und regelmäßige Statusaktualisierungen gewährleisten können, dass fehlerhafte Anwendungsteile innerhalb kürzester Zeit erkannt und ersetzt werden.

Anforderung 7: Die Anwendung muss die Möglichkeit bieten, die Datenstromverarbeitung auf mehrere Prozessoren und weitere Systeme zu verteilen, um eine hohe Skalierbarkeit zu gewährleisten. Diese Verteilung sollte vollkommen automatisch und transparent für den Benutzer der Anwendung erfolgen. Diese Anforderung ist von enormer Wichtigkeit für die Tragfähigkeit einer Anwendung.

	DBMS	Rule engines	SPS
1.: Datenverarbeitung „in-stream“	Nein	Ja	Ja
2.: Unterstützung von „StreamSQL“	Nein	Nein	Ja
3.: Beachtung verminderter Datenzuverlässigkeit	Schwierig	Möglich	Möglich
4.: Berechenbare und reproduzierbare Ergebnisse	Schwierig	Möglich	Möglich
5.: Kombinieren von gespeicherten Daten und Datenströmen	Nein	Nein	Ja
6.: Hohe Verfügbarkeit	Möglich	Möglich	Möglich
7.: Verteilbarkeit und Skalierbarkeit	Möglich	Möglich	Möglich
8.: „Echtzeitfähigkeit“	Möglich	Möglich	Möglich

Tabelle 2.1: Vergleich von DBMS, Rule engine und SPS

Anforderung 8: Die Anwendung darf nicht selbst der Flaschenhals bei der Echtzeitverarbeitung eines Datenstromes sein. Daher müssen alle Komponenten der Anwendung auf die Verarbeitung von Datenströmen ausgerichtet werden; die Anwendung muss also selbst „echtzeitfähig“ sein. Die Reduzierung des Zusatzaufwandes für die Datenverarbeitung auf ein Minimum muss dafür sorgen, dass das Stream Processing System auch in der Lage ist, mit großen Datenströmen „Schritt zu halten“ und diese in Echtzeit an die Konsumenten übermitteln zu können.

Vor der Einführung der Stream Processing Systems standen vor allem zwei Anwendungsklassen für die Verarbeitung von Datenströmen zur Verfügung: Traditionelle Datenbanksysteme (engl. database management systems, DBMS) und regelbasierte Anwendungen (engl. rule engines). Unter Berücksichtigung der o. g. Anforderungen an Systeme zur Verarbeitung von Datenströmen wird schnell klar, warum eine neue Klasse von Anwendungen erarbeitet wurde. Die Tabelle 2.1 verschafft einen Überblick über die drei Anwendungsklassen und ihre Konformität mit den Anforderungen.

Die ersten Vertreter der Stream Processing Systems [CcC+02, ACc+03] waren noch zentralistisch organisiert. Das bedeutet, dass die Operatoren auf einem zentralen System zusammen verwaltet wurden und diese Systeme die Anforderung 7 nur bedingt erfüllen konnten. Schon bei den direkten Nachfolgern [CBB+03, AAB+05, HCZ07] konnte sich jedoch Distributed Stream Processing als dominierendes Entwurfsparadigma etablieren. Worin genau der Unterschied zu den Stream Processing Systems der ersten Generation besteht und welche neuen Herausforderungen sich aus der Verteiltheit der jüngeren Systeme ergeben, wird im nächsten Abschnitt näher erläutert.

2.2.1 Distributed Stream Processing Systems

Distributed Stream Processing Systems sind Systeme zur Datenstromverarbeitung, deren Operatoren sich nicht auf einem zentralen System befinden. Stattdessen werden die komplexen Operationen, die die Konsumenten auf die einzelnen Datenströme anwenden wollen, in Einzeloperatoren aufgespalten. Diese Operatoren werden anschließend nach verschiedenen Kriterien, wie zur Verfügung stehende Bandbreite und Rechenleistung, im Netzwerk zwischen Produzenten und Konsumenten der Informationen verteilt. Dabei wird ein verteilter Graph von Operatoren aufgebaut, der nicht nur die Skalierbarkeit des entstehenden Systems wesentlich verbessert, sondern auch für eine Wiederverwendbarkeit einzelner Operatoren sorgt. Die Verteilung der Operatoren trägt auch zur Verfügbarkeit und Ausfallsicherheit des Gesamtsystems bei, da so z. B. bestimmte Teile des Operatorengraphen auf anderen Systemen repliziert und bei Bedarf aktiviert werden können. Außerdem wird so eine ungleiche Verfügbarkeit von Ressourcen auf den beteiligten Systemen berücksichtigt.

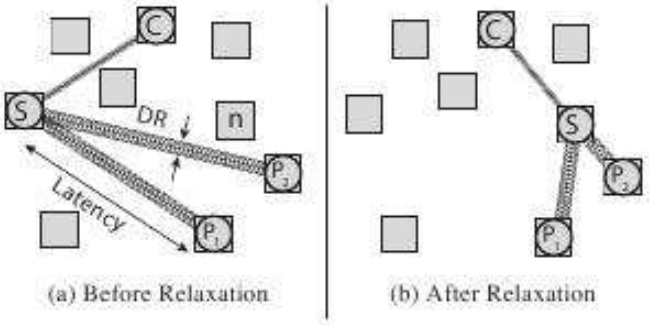

Abbildung 2.4: Der Relaxation-Algorithmus von SBON minimiert die Gesamtkosten im System [PLS+06]

Die Abbildung 2.4 zeigt, wie in SBON [PLS+06], einer Erweiterung für DSPS, die Optimierung der Operatorenplatzierung vorgenommen wird. Dafür wird vom sogenannten Relaxation-Algorithmus die aus der Physik bekannte Formel zur Berechnung der Federkraft $\vec{F} = \frac{1}{2}k\vec{s}$ für die Berechnung der Gesamtkosten im System benutzt. Die Latenz für eine Verbindung l wird dabei in die Formel als Federdehnung $s_l = LAT(l)$ und die Bandbreite (engl. data rate, DR) als Federkonstante $k_l = DR(l)$ eingesetzt. Diese Gesamtkosten werden in regelmäßigen Abständen von SBON minimiert, indem die Platzierung einzelner Operatoren geändert wird.

Durch die Verteiltheit des Stream Processing Systems ergeben sich auch neue Fragen und Forschungsgebiete: Gibt es eine optimale Platzierung der Operatoren im Netzwerk und wie kann diese erreicht werden? Wie kann eine hohe Verfügbarkeit erreicht werden, ohne den Operatorengraphen vollständig zu replizieren? Was für Möglichkeiten gibt es, die

Kommunikation zwischen den Produzenten und den Konsumenten der Sensorinformationen auf ein Minimum zu reduzieren?

Fragestellungen in Bezug auf Operatoren selbst werden in dieser Arbeit jedoch auf Grund der Zeit- und Platzbeschränkungen ausgeklammert. Stattdessen wird in dieser Arbeit auf zwei der generellen Anforderungen an Distributed Stream Processing Systems eine andere Antwort präsentiert, als dies in bisher verbreiteten DSPS üblich ist. Daher werden diese zwei Anforderungen in den nächsten Abschnitten im Detail betrachtet: Die Unterstützung von Bereichsanfragen und die Gewährleistung der Echtzeitverarbeitung von Datenströmen.

Unterstützung von Bereichsanfragen

Durch den Zusammenschluss zu globalen Sensornetzen wird anfragenden Interessenten ein sehr großer geographischer Bereich zugänglich gemacht. Angesichts der angestrebten Größe von GSNs ist davon auszugehen, dass es auch Konsumenten geben wird, die sich nur für Informationen über einen Ausschnitt des gesamten verfügbaren Bereichs interessieren. Das Distributed Stream Processing System muss als Vermittler zwischen Konsument und Produzent von Informationen also die Möglichkeit bieten, den Interessenten den Bereich spezifizieren zu lassen, zu dem er Sensorinformationen erhalten will. Somit muss ein DSPS Anfragen, die bestimmte geographische Regionen beinhalten – sogenannte *Bereichsanfragen* –, verarbeiten können.

Auch in anderen Gebieten der Informatik werden Anwendungen benötigt, zu deren Anforderung es zählt, Anfragen zu bestimmten geographischen Regionen beantworten zu können. Auf dem Gebiet der Geoinformationssysteme stellt die Bereichsanfrage ebenfalls die dominierende Anfrageart dar. Die meisten der in diesem Gebiet eingesetzten Systeme verwenden zur effizienten Verwaltung der anfragbaren Datenobjekte sogenannte R-Bäume.

In der Literatur finden sich zur Beantwortung von Bereichsanfragen auch andere Vorgehensweisen, wie z. B. der Ansatz, eine beliebige verteilte Hashtabelle (engl. distributed hashtable, DHT) mit einer raumfüllenden Kurve (engl. space-filling curve, SFC) auszustatten und so die Zahl der Dimensionen auf eine zu reduzieren. Dieser Ansatz weist jedoch erhebliche Nachteile auf. So ist die Wahl der Hashfunktion mit erheblichem Aufwand verbunden, da diese lokalitätsbewahrend sein muss, um auf der SFC die ursprünglichen Nachbarschaftsverhältnisse zu erhalten. Außerdem ist eine durch eine SFC ergänzte DHT nicht ohne enormen weiteren Aufwand in der Lage, Überschneidungsbereiche zweidimensionaler Objekte zu erkennen.

Folglich wird in dem in dieser Arbeit vorgestellten Distributed Stream Processing System eine Verwaltungsstruktur auf Basis eines R-Baumes verwendet. Daher folgt im nächsten Abschnitt eine Einführung in die Funktionsweise des R-Baumes. Außerdem finden einige der bekannten R-Baum-Varianten Erwähnung.

R-Baum Der R-Baum [Gut84] kann als eine Erweiterung des B-Baums [BM02, Bay72] auf mehrdimensionale Einträge bezeichnet werden. Sein Zweck ist es, mehrdimensionale Objekte in einer hierarchischen Struktur zu organisieren, und so die Zeit, die zur Beantwortung einer Anfrage benötigt wird, stark zu reduzieren. Dafür wird im zweidimensionalen Fall jedes Datenobjekt durch sein *minimum bounding rectangle* (MBR) repräsentiert. Dieses *MBR* stellt das kleinste Rechteck dar, das groß genug ist, eine gegebene Menge an ihm zugeordneten Objekten zu umschließen. Innerhalb des Baumes werden einzelne Objekte zu größeren MBRs zusammengefasst, die wiederum von größeren MBRs umschlossen werden, und so weiter. So entsteht eine Hierarchie, die in einem Baum dargestellt werden kann. In der Abbildung 2.5

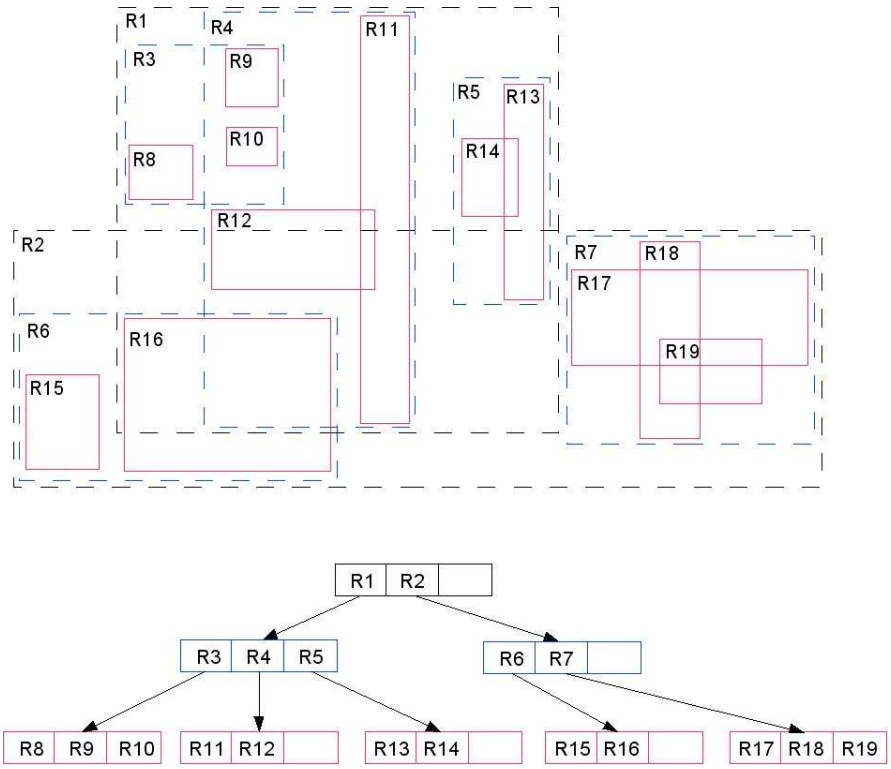

Abbildung 2.5: Beispiel eines R-Baumes für $M_{max} = 3$
(Quelle: Wikipedia, http://en.wikipedia.org/wiki/Image:R-tree.jpg)

werden diese MBRs durch die Bereiche $R1$ bis $R7$ dargestellt. In einem R-Baum werden einzelne Objekte also zu Organisationseinheiten zusammengefasst. Diese Einheiten werden wiederum zu größeren Einheiten zusammengefasst, so dass die bereits erwähnte Hierarchie entsteht. Der Bereich $R1$ enthält zum Beispiel die kleineren Bereiche $R3$, $R4$ und $R5$. In R-Bäumen gibt es zwei verschiedene Arten von Knoten: Die erste Art dient lediglich zur Verwaltung der Struktur. In einem Baum mit n Levels sind diese Knoten auf den Levels L_0 bis L_{n-1} anzutreffen. Die Blätter des Baumes sind die Daten selbst, in der Abbildung durch $R8$ bis $R19$ repräsentiert. Sie können sowohl durch Punkte als auch durch Rechtecke repräsentiert werden. Die *Zahl der Kinder* ist für die Knoten des Baumes mit M_{min} und M_{max} beschränkt,

im vorliegenden Beispiel ist $M_{max} = 3$ und $M_{min} = 2$. Sollte ein Knoten *überlaufen*, also mehr als M_{max} Kinder aufweisen, nachdem er ein neues Datenobjekt eingefügt hat, wird eine *Knotenaufteilung* eingeleitet. Das MBR des Knotens wird in zwei möglichst gleichgroße Stücke geteilt, so dass dann statt des bisher vorhandenen Knotens nun zwei Knoten zur Verwaltung der Datenobjekte eingesetzt werden. Die neuen Knoten müssen ihrerseits natürlich auch die o. g. Ober- und Untergrenzen respektieren. Verwaltet auch das darüber liegende MBR bereits M_{max} Kinder, so propagiert sich die Knotenaufteilung weiter nach oben. Das *Einfügen neuer Objekte* in einem R-Baum erfolgt so, dass mit dem MBR des neuen Objektes der Baum von oben nach unten traversiert wird. Angefangen bei den beiden Bereichen $R1$ und $R2$ wird bestimmt, ob und um wie viel sie sich mit dem neuen Bereich überschneiden. Dann wird derjenige Knoten ausgewählt, der entweder den größten Überschneidungsbereich mit dem neuen vorweisen kann, oder aber derjenige, der am wenigsten in der Fläche vergrößert werden muss, um das neue Datenobjekt aufzunehmen. Dann wird genauso bei den Kindern des ausgewählten Knotens fortgefahren, bis das vorletzte Level erreicht ist. Dort wird das neue Datenobjekt dann als neues Kind eingefügt und die MBRs der übergeordneten Knoten werden ggf. vergrößert.

R*-Baum Der R*-Baum [BKSS90] stellt eine Weiterentwicklung des R-Baumes dar. Die grundlegende Struktur wurde beibehalten, als tiefgreifendste Veränderung wurden die Algorithmen zum Einfügen neuer Knoten und zum Aufteilen übergelaufener Knoten komplett überarbeitet. Das Ziel des R*-Baumes ist es, zusätzlich zur Minimierung der Fläche der einzelnen Bereiche, die Überschneidungen zwischen den MBRs so gering wie möglich zu halten. Dazu wurde das *Prinzip des Wiedereinfügens* (engl. re-insertion) erarbeitet, das dafür sorgt, dass bei einem Knotenüberlauf zunächst ein gewisser Teil der Daten gelöscht und wieder neu in den Baum eingefügt wird. Die dahinter stehende Absicht ist, das rechenintensive Aufteilen von Knoten umgehen zu können, da es laut der Studien der Autoren sehr wahrscheinlich sei, dass nicht alle Daten wieder in den gleichen Knoten eingefügt würden. Der R*-Baum bietet im Vergleich zum R-Baum eine deutlich schnellere Antwortzeit auf Anfragen. Dies wird jedoch auch damit erkauft, dass Zeit, die benötigt wird, um neue Objekte einzufügen, um ein Vielfaches höher ist.

Weitere R-Baum-Varianten Es existieren weitere R-Baum-Varianten, die sich vorwiegend mit speziellen Problemen auseinandersetzen. Der R^+-Baum [SRF87] fügt Objekte in mehrere Teilbereiche ein, wenn sich ein neues Element mit mehreren bereits bestehenden MBRs überschneidet. Dies steigert vor allem die Antwortgeschwindigkeit bei der Suche nach Punkten oder sehr kleinen Bereichen, bei denen die Wahrscheinlichkeit sehr groß ist, dass für die Suche mehrere Äste eines normalen R-Baumes traversiert werden müssten. Der Hilbert-R-Tree [KF94] benutzt Fraktale, um die Platzverschwendung durch normale MBRs bei sehr dünn gesäten Punktdaten auf ein Minimum zu reduzieren. Da diese Varianten keine nennenswerten Verbesserungen für das hier betrachtete Problem liefern oder sogar kontraproduktive Veränderungen vornehmen (z. B. die Mehrfachmitgliedschaft von Datensätzen), werden diese Varianten hier nicht weitergehend beschrieben und auch in der Folge nicht berücksichtigt.

Echtzeitverarbeitung von Datenströmen

Wie schon aus den in Kapitel 2.2 formulierten Anforderungen deutlich wurde, ist die Echtzeitverarbeitung von Datenströmen einer der zentralen Aspekte für Distributed Stream Processing Systems. Etliche Anwendungen – von Systemen für den computergesteuerten Aktienhandel bis hin zu Katastrophenwarnsystemen – sind für eine effektive Funktionsweise auf eine besonders schnelle Übermittlung und Verarbeitung der von ihnen angefragten Sensordatenströme angewiesen.

Während die Echtzeitverarbeitung innerhalb von drahtlosen Sensornetzwerken – also in einer lokal sehr beschränkten Umgebung – noch mit verhältnismäßig wenig Aufwand zu realisieren war, entstehen durch die Erweiterung der Produzentenmenge auf globale Sensornetze ganz neue Probleme. Durch die möglicherweise große Entfernung zwischen Produzenten und Konsumenten muss in GSNs noch die zusätzliche Latenz für diese Verbindung beachtet werden. Die Latenz für die Verbindung zwischen den beiden Parteien wird somit in einem globalen Sensornetz größer sein, als dies noch im Kontext eines einzelnen Sensornetzes der Fall war. Eine Möglichkeit, diese potentiell sehr hohe Latenz mit den Echtzeitanforderungen der anfragenden Anwendungen in Einklang zu bringen, stellt die Einführung eines Vorhersageverfahrens dar. In dem in dieser Arbeit konzipierten DSPS werden dafür sogenannte Prädiktoren eingesetzt, die auf Seite der Konsumenten den Datenstrom des Produzenten vorhersagen und so auch hohe Latenzen zwischen diesen beiden Seiten egalisieren können.

Kapitel 3

Stream Processing im Kontext von globalen Sensornetzen

Die Aufgabe von Distributed Stream Processing Systems im Kontext von globalen Sensornetzen wurde bereits im vorhergehenden Kapitel abstrakt zusammengefasst: Das Zugänglichmachen und die anschließende Vermittlung von Informationsströmen zwischen Mengen von Produzenten und Konsumenten.

Eine als Konsument auftretende Anwendung übermittelt ihr Interesse an bestimmten Daten und eventuell auf diesen Daten auszuführende komplexe Operationen an das Stream Processing System. Dieses übernimmt dann in einer für den Konsumenten transparenten Weise die Aufgabe, die Anfrage an die richtigen Produzenten weiterzuleiten, die komplexen Operationen in Einzeloperatoren aufzubrechen und diese auf die von den Produzenten angeforderten Datenströme anzuwenden. Schließlich werden die Ergebnisse an den Konsumenten übermittelt.

Als Produzent wird ein globales Sensornetz aus vielen einzelnen Sensornetzen erstellt. Aus der flächenmäßig großen Ausbreitung des Netzes und der Vielzahl an einzelnen verfügbaren Sensornetzen resultieren Anforderungen an die zur Datenstromverarbeitung eingesetzten Systeme. Unter anderem wird wegen der Größenordnung der verfügbaren Produzenten eine gute Skalierbarkeit gefordert, die nur gewährleistet werden kann, wenn ein Stream Processing System verteilt realisiert wird. Die Anforderungen an die daraus resultierenden Distributed Stream Processing Systems werden im folgenden Abschnitt genauer behandelt. Außerdem werden die Herausforderungen, die sich bei der Konzipierung eines DSPS ergeben, im Detail analysiert. Anhand der Fragen, die von den bisher existierenden Stream Processing Systems nur unzureichend behandelt werden, wird die Notwendigkeit für ein weiteres DSPS aufgezeigt.

Diese große Zahl an wichtigen und interessanten Herausforderungen und Themengebieten wird für das im daran anschließenden Abschnitt beschriebene Systemmodell auf einen Umfang beschränkt, der dem zeitlichen Rahmen dieser Arbeit Rechnung trägt. Dazu werden Annahmen getroffen und Voraussetzungen definiert, auf denen diese Arbeit basiert.

Danach wird ein reales Anwendungsszenario für ein auf diesem Systemmodell basierendes Distributed Stream Processing System beschrieben, wie es in dieser Arbeit konzipiert wird.

3.1 Problemstellung

Heutige Distributed Stream Processing Systems müssen sich daran messen lassen, ob sie in der Lage sind, ihre Vermittlerrolle zwischen Produzenten und Konsumenten auch noch im Kontext der globalen Sensornetze und den daraus resultierenden Größenordnungen für beide Parteien erfüllen zu können.

Die üblicherweise zur Bewertung von DSPS herangezogenen Kriterien müssen also erneut auf ihre Gültigkeit überprüft werden. Außerdem wird die Liste der zu erfüllenden Kriterien durch die Steigerung der Größenordnung um weitere Punkte ergänzt.

Als Bewertungskriterien, um Aussagen über die effektive Arbeitsweise eines DSPS treffen zu können, werden unter anderem herangezogen:

- Effizienz
- Vermeidung redundanter Datenströme
- Skalierbarkeit
- Dienstqualität (engl. quality of service, QoS)
- Datenqualität (engl. quality of data, QoD)

Diese Bewertungskriterien werden nun genauer beschrieben und anschließend daraufhin untersucht, inwieweit die in der Literatur beschriebenen Stream Processing Systems diese Kriterien erfüllen.

Effizienz Die Effizienz eines DSPS bezieht sich vor allem auf dessen Fähigkeit, die Gesamtkosten zur Datenstromübertragung zu minimieren. Durch die Verteiltheit der Operatoren innerhalb des Systems und die teils großen Entfernungen zwischen Produzenten und Konsumenten kann durch eine Neuplatzierung einzelner Operatoren in Bezug auf die Gesamtkosten innerhalb des Distributed Stream Processing Systems ein wesentlicher Beitrag zur Effizienzsteigerung erreicht werden.

Vermeidung redundanter Datenströme Neben der Kostenminimierung für den Transport der angeforderten Datenströme gibt es weitere Kriterien, die die Datenübertragung thematisieren. Ein Kriterium ist die Frage, wie gut es dem System gelingt, das Übermitteln redundanter Datenströme zu vermeiden. Das bedeutet, dass das System Situationen erkennen kann, in denen die gleiche Information von unterschiedlichen Konsumenten angefordert wird. Als Reaktion darauf sollte das DSPS seine internen Datenströme so organisieren, dass Informationsströme mehrfach genutzt werden, statt sie mehrfach zu übertragen.

Skalierbarkeit Bei der Neuplatzierung der Operatoren zur Effizienzsteigerung und der Vermeidung redundanter Datenströme kommen Algorithmen zum Einsatz, die sich auch auf die Skalierbarkeit eines DSPS auswirken. Diese Algorithmen dürfen im Gegenzug nicht etwa das Entstehen von Flaschenhälsen während der Verarbeitung der Sensordatenströme fördern, sondern müssen stets auch berücksichtigen, dass das System einer großen Masse von Interessenten eine gewaltige Menge von Informationen übermitteln muss. Neben den beiden bereits genannten Kriterien Effizienz und Redundanzvermeidung wirkt sich auch die Belastung der Gateways zu den Sensornetzen auf die Skalierbarkeit eines DSPS aus. Ein Gateway kann nur eine begrenzte Anzahl von ausgehenden Verbindungen bedienen. Steigt die Zahl der ausgehenden Verbindungen für die Gateways mit der Zahl der an das DSPS gestellten Anfragen, stellen die Gateways den Flaschenhals der Anwendung dar und begrenzen dessen Skalierbarkeit.

Dienstqualität Für DSPS stellt sich in diesem Bereich die Frage, ob überhaupt und wenn ja welche Garantien bezüglich der Dienstqualität gegeben werden können. In den Bereich der Dienstqualität fallen beispielsweise Konzepte zur Ausfallsicherheit bzw. einer schnellen Wiederverfügbarkeit der vom System angebotenen Dienste. Solche Garantien können durch Vorgehensweisen wie z. B. die Replikation kritischer Teildienste erkauft werden. Ein weiterer Punkt, der die Dienstqualität eines DSPS betrifft, ist dessen Fähigkeit, Echtzeitdatenströme zur Verfügung stellen zu können. Diese Anforderung könnte auch als garantiertes Maximalalter eines Datensatzes in die Datenqualität aufgenommen werden.

Datenqualität Zur Datenqualität gehören alle Anforderungen, die die Beschaffenheit der vom Konsumenten empfangenen Daten betreffen. Dazu gehört z. B. auch die Ungenauigkeit, mit die Datensätze, die beim Konsumenten eintreffen, behaftet sind. Diese Ungenauigkeit kann verschiedene Ursachen haben: Zum einen weisen die Sensoren selbst eine inhärente Ungenauigkeit auf, zum anderen gibt es etliche Ansätze, die das Datenaufkommen in einem DSPS reduzieren, indem sie die Genauigkeit der übertragenen Daten einschränken. Neben der Frage, ob ein Stream Processing System Garantien bezüglich der Qualität der übermittelten Daten macht, ist also auch interessant, ob es Möglichkeiten gibt, Datenverkehr auf Kosten der Datengenauigkeit einsparen zu können.

Jedes dieser Kriterien stellt einen eigenen Forschungsbereich dar – in manchen Fällen sogar mehrere. Durch die Komplexität der einzelnen Bereiche und dadurch, dass sich die DSPS teilweise auf bestimmte Anwendungsbereiche fokussieren, ist die Situation entstanden, dass kein DSPS existiert, das alle der oben definierten Leistungskriterien zur Zufriedenheit erfüllt. Gleichzeitig gibt es aber Kriterien, für deren Optimierung Lösungen für viele Systeme verfügbar sind. Mit SBON [PLS+06] wurde ein Overlay-Netzwerk für beliebige DSPS vorgeschlagen, das unter anderem eine Optimierung der Operatorenplatzierung und damit eine Minimierung der Gesamtkosten im System vornehmen kann. Auch die Forderung nach einer hohen Verfügbarkeit hat in viele Systeme Eingang gefunden [HBR+05, HBR+03, AAB+05].

Zur Vermeidung von redundanten Datenströmen bietet SBON ebenfalls eine Lösung. Dies geschieht allerdings reaktiv durch das periodische Verschmelzen von Operatoren; es han-

delt sich hierbei also eher um ein nachträgliches Erkennen und Beheben von redundanten Datenströmen statt einer proaktiven Vermeidung. Auch die Frage, wie auf Kosten der Datengenauigkeit eine Reduktion des Datenverkehrs im System erreicht werden kann, ist im Bereich des Stream Processing nur ungenügend beantwortet worden. Existierende Lösungen [JCW04, San06, LGS06, TM06] arbeiten mit Vorhersagemodellen, beziehen sich jedoch nur auf den sehr lokalen Kontext eines einzelnen Sensornetzes. Für die Kommunikation innerhalb eines gesamten DSPS wurden derartige Vorhersageverfahren bisher nicht eingesetzt. Die Echtzeitverfügbarkeit von Datenströmen auch über Verbindungen mit hoher Latenz garantieren zu können, wird bisher in keinem der verfügbaren Systeme zur Datenstromverarbeitung thematisiert.

An den beiden Anforderungen zur Skalierbarkeit und Dienstqualität lassen sich exemplarisch die Defizite der bisher existierenden Stream Processing Systems aufzeigen. In dieser Arbeit wird das Konzept für ein neues Distributed Stream Processing System entworfen, um auf diese Defizite zu reagieren. Da es auf Grund von Zeit- und Platzbeschränkungen nicht möglich ist, alle der oben genannten Leistungskriterien mit der gleichen Intensität zu bearbeiten, werden zwei Probleme ausgewählt, auf denen der Schwerpunkt der Arbeit liegt: Die Vermeidung von redundanten Datenströmen mit daraus resultierender Entlastung der Produzenten und die Integration eines Vorhersagekonzeptes mittels Prädiktoren in das gesamte DSPS, um die Verfügbarkeit von Datenströmen in Echtzeit unabhängig von der Latenz zwischen Produzent und Konsument gewährleisten zu können.

Bei der Konstruktion des Systems werden also zwei Schwerpunkte gesetzt. Der erste beschäftigt sich mit Möglichkeiten, durch den Einsatz eines verteilten R-Baumes zur Anfragenverwaltung bereits das Entstehen redundanter Datenströme zu vermeiden. Stattdessen sollen die Gemeinsamkeiten zwischen den verschiedenen Anfragen ausgenutzt werden und die Konsumenten von Informationen dazu verpflichtet werden, die ihnen übermittelte Information auch anderen Konsumenten zur Verfügung zu stellen. Dadurch werden auch die Sensornetze und ihre Gateways nicht mehr als unbedingt nötig mit Anfragen belastet und somit die Skalierbarkeit des Systems erhöht. Der zweite Schwerpunkt versucht, durch einen Verzicht auf Genauigkeit in den gelieferten Sensordaten eine Reduzierung das Datenaufkommen im Gesamtsystem zu erreichen. Dies geschieht durch den Einsatz von Prädiktoren, die den Verlauf der übertragenen Datenströme vorhersagen. Da der Konsument direkt auf den nächsten Vorhersagewert zugreifen kann, wenn er ihn benötigt, statt wie bisher auf das nächste Paket mit Sensorinformationen warten zu müssen, wird durch Prädiktoren die Echtzeitverarbeitung von Sensordaten auch trotz Verbindungen mit hoher Latenz möglich.

Die konkreten Herausforderungen beim Entwurf dieses Systems orientieren sich an diesen beiden Schwerpunkten.

Alle am Gesamtsystem beteiligten Systeme – also sowohl die Anfragensteller als auch die Gateways, die diese Anfragen beantworten – werden in eine verteilte R-Baum Struktur eingebettet, die es ermöglicht, Überschneidungen zwischen den einzelnen Anfragen zu erkennen.

Wenn eine Überschneidung zwischen zwei Anfragen erkannt wird, muss das System entsprechend reagieren. Eine der beiden Anfragen soll Teile der von ihr verarbeiteten Information

für eine andere Anfrage zur Verfügung stellen, woraus folgt, dass die Anfrage in mehrere Stücke unterteilt werden muss. Dabei müssen wichtige Detailfragen geklärt werden: Wie genau wird die Anfrage unterteilt und was passiert in dem Fall, wenn sich eine neue Anfrage mit mehreren bereits bestehenden Anfragen überschneidet?

Eine Aufteilung der Anfragen wirft weitere Fragen auf: Wie kann man die Auswirkungen für den Anfragensteller transparent halten? Gibt es Probleme, die unterstützten Operatoren mit der Aufteilung der Anfragen zu vereinbaren? Es sind außerdem zusätzliche Mechanismen notwendig, um mit einer Gewichtung jedes Teilstücks der unterschiedlichen Größe der Anfrageteile Rechnung zu tragen. Diese Gewichtung muss relativ zu dem Anteil sein, den das Stück an der Gesamtanfrage hat.

Durch die Verwendung von Prädiktoren ergeben sich daraus weitere Herausforderungen. Ganz zentral ist die Frage, wie sich das Prinzip der Vorhersage in die Verteilstruktur für Sensordaten integrieren lässt. Außerdem muss das Konzept mit den anderen bereits erwähnten Mechanismen harmonieren. Das bedeutet konkret: Da Prädiktoren an Datenverbindungen gebunden sind, also überall dort eingesetzt werden sollen, wo Sensordaten übermittelt werden, müssen sie an die dynamische Unterteilung der Anfragen angepasst werden. Wird also eine Anfrage in mehrere Teile unterteilt, muss jedem dieser Anfrageteile ein eigener Prädiktor zugeordnet werden.

Dadurch, dass Prädiktoren darauf abzielen, die Kommunikation innerhalb des Systems stark zu reduzieren, ergeben sich weitere Herausforderungen. Durch die verringerte Anzahl an Paketen wird es schwieriger, Knotenausfälle zu erkennen. Es muss ein Verfahren entwickelt werden, das bei minimalem Mehraufwand in der Lage ist, Knotenausfälle zuverlässig zu erkennen und außerdem Reparaturmaßnahmen einleiten kann.

Für die nun auf zwei Hauptkriterien eingeschränkte Problemstellung wird im nächsten Abschnitt ein Systemmodell beschrieben. Auf Grund zeitlicher Restriktionen für diese Arbeit werden weitere Annahmen getroffen, die die Komplexität des hier konzipierten Systems einschränken.

3.2 Systemmodell

Für das in dieser Arbeit konzipierte Distributed Stream Processing System wurden in den vorherigen Abschnitten bereits zwei Schwerpunkte gesetzt: Die Vermeidung von redundanten Datenströmen mit daraus resultierender Entlastung der Produzenten und die Integration eines Vorhersagekonzeptes mittels Prädiktoren in das gesamte DSPS. Neben der Konzentration auf diese Schwerpunkte werden noch andere *Einschränkungen* vorgenommen: Die Art der verfügbaren Sensorinformationen wird auf *Temperaturdaten* eingeschränkt. Als Möglichkeit zur Anfrage wird die *Bereichsanfrage* benutzt. Der Zweck des Systems ist es also, Temperaturdaten zu einem definierten geographischen Anfragebereich zu übermitteln. Auf die Datenströme kann der *Aggregationsoperator* in den Ausprägungen Minimum, Maximum und Durchschnitt angewendet werden.

Das in dieser Arbeit beschriebene Gesamtsystem ist aus einer Menge von Einzelsystemen aufgebaut. Diese Einzelsysteme nehmen bestimmte *Rollen* innerhalb des Gesamtsystems ein. Es existieren *Produzenten* und *Konsumenten* von Informationen. Außerdem existiert eine Menge von Systemen, die bisher nicht in das Gesamtsystem integriert sind, aber ein Informationsbedürfnis besitzen. Dies sind die *Interessenten*. Ein Interessent tritt mit einer *Anfrage* dem Gesamtsystem bei und wird somit zum Konsumenten. Damit dies möglich ist, existieren Systeme, die *Anfragen entgegennehmen* und weiterverarbeiten. Innerhalb eines Stream Processing Systems gibt es auch immer Systeme, die die eigentliche *Vermittlung der Datenströme* vornehmen.

Die Rolle der Produzenten wird in diesem System von den *Gateways* eingenommen. Sie kapseln die hinter ihnen liegenden Sensornetze nach außen. Gateways sind also dafür zuständig, eingehende Anfragen zu dem von ihnen verwalteten Sensornetz anzunehmen, zu verarbeiten und schließlich den resultierenden Datenstrom zu erzeugen. Für jedes Sensornetz existiert dabei genau ein Gateway.

Die von dem in dieser Arbeit konzipierten DSPS betrachtete *Welt* ist ein Rechteck begrenzten Ausmaßes. Innerhalb dieser Welt sind ebenfalls rechteckige Sensornetze angeordnet. Die Sensornetze selbst weisen keine Überschneidungen auf, so dass jedem geographischen Bereich der betrachteten Welt genau ein Sensornetz – und damit genau ein Gateway – zugeordnet ist.

Das hier vorgestellte Verfahren vernachlässigt die Unterschiede zwischen einzelnen Sensornetzen bezüglich ihrer Sensorendichte und -verteilung und den eventuell daraus resultierenden Ungenauigkeiten. Die Mechanismen, die sich mit Problemen dieser Art beschäftigen, laufen innerhalb der Sensornetze selbst ab; die Kommunikation in den einzelnen Netzen wird hier jedoch durch den jeweiligen Gateway gekapselt und dadurch nicht weiter betrachtet werden. Die *interne Kommunikation* zwischen Gateway und Sensornetz ist damit nicht Teil dieser Arbeit.

Einer Menge von *Peers* kommt die Rolle der Interessenten und Konsumenten zu. Interessenten kontaktieren ein beliebiges in das Gesamtsystem integrierte Einzelsystem mit einer Anfrage und treten somit dem Gesamtsystem als Konsumenten bei.

Die *Anfragen*, die Interessenten an das Gesamtsystem stellen, sind – wie auch schon die Welt und die einzelnen Sensornetze – als Rechtecke definiert. Die Welt wird durch die Sensornetze gerastert, was bedeutet, dass die kleinste Schrittweite für die Seitenlängen einer Anfrage die Länge eines Sensornetzes darstellt.

Eine *Anfrage A* wird definiert als

$A = (\gamma, \omega, \epsilon, \tau)$
mit dem geographischen Bereich $\gamma \in \Gamma = \{((x_1|y_1), (x_2|y_2))|x_i, y_i \in [0, \lambda] \subset \mathbb{N}\}$,
dem Operator $\omega \in \Omega = \{\text{AGG_AVG}, \text{AGG_MAX}, \text{AGG_MIN}\}$,
der vom Interessenten tolerierten Ungenauigkeit $\epsilon \in \mathbb{R}$ in $^\circ$C
und der gewünschten Aktualisierungsrate $\tau \in \mathbb{N}$ in Sekunden

Die Begrenzung der betrachteten Welt wird hier mit λ bezeichnet. Ein Interessent muss in seiner Anfrage die maximale von ihm tolerierte Ungenauigkeit für die Datenströme, die ihm übermittelt werden, mit ϵ spezifizieren. Diese Werte werden später dazu benutzt, auf Kosten der Datengenauigkeit das Datenaufkommen im Gesamtsystem zu reduzieren. Der Parameter τ sorgt dafür, unnötige Aktualisierungen zu verhindern. Kann z. B. die Anwendung, die die Sensordaten benötigt, nur alle y Sekunden einen neuen Datensatz verarbeiten, so wären die in der Zwischenzeit gesendete Daten wertlos. Die Anwendung würde sie nicht nutzen können und daher verwerfen.

Generell wird zwischen zwei *Arten von Anfragen* unterschieden, deren Charakter vor allem in der Art der Antwort deutlich wird: Die Antwort auf eine *kontinuierliche Anfrage* ist ein Datenstrom von Messwerten, der die angeforderte Sensorinformation so lange stets aktualisiert bereitstellt, bis die Anfrage explizit beendet wird. Im Gegensatz dazu stellt die Antwort auf eine *spontane Anfrage* einen einzelnen Wert dar, der zum Anfragezeitpunkt vom Gateway ermittelt wird. Eine kontinuierliche Anfrage ist somit vergleichbar mit einem Abonnement auf bestimmte Sensordaten.

Für die interne Organisation werden alle Teilnehmer des Gesamtsystems, die kontinuierliche Anfragen stellen, zusammen in einer verteilten R-Baum Struktur verwaltet. Sowohl Peers als auch Gateways stellen also *Knoten* innerhalb dieser Baumstruktur dar. Alle Knoten des Baumes können neue *Anfragen entgegennehmen* und weiterverarbeiten.

Zur *Vermittlung der Datenströme* werden die Peers herangezogen. Jeder Interessent, der dem Gesamtsystem betritt und von diesem Informationen erhält, muss diese anderen Interessenten bei Bedarf zur Verfügung stellen.

Über die Interessenten, die Anfragen an das in dieser Arbeit entworfene System stellen werden, können keine verbindlichen Aussagen getroffen werden. Die Funktionstüchtigkeit und Skalierbarkeit des Systems hängt wie auch bei den verbreiteten Peer-to-Peer-Systemen davon ab, dass ein Teilnehmer im Gesamtsystem über die nötigen Ressourcen verfügt, die Datenmenge, die er anfordert, auch an andere Teilnehmer weiterleiten zu können.

Die Anordnung und Organisation der Sensornetze und ihrer Gateways ist für die Konsumenten vollkommen transparent. Anfragen können ungeachtet der physischen Grenzen der einzelnen Sensornetzwerke an einen beliebigen rechteckigen Ausschnitt der Welt gerichtet werden. In Abbildung 3.1 ist zu sehen, wie die Konsumenten K_1 bis K_3 jeweils Anfragen zu einem bestimmten geographischen Bereich stellen. Diese Anfragen werden an die dafür verantwortlichen Gateways, die in der Abbildung mit P als Produzenten gekennzeichnet sind, weitergeleitet. Daraufhin stellen die Gateways den Konsumenten die angefragte Information zur Verfügung.

3.3 Anwendungsszenario

In diesem Abschnitt wird ein Anwendungsszenario für ein Distributed Stream Processing System geliefert, das auf dem oben formulierten Systemmodell basiert. Dies zeigt, dass sich die vorgenommenen Einschränkungen nicht auf die Einsetzbarkeit des Systems auswirken.

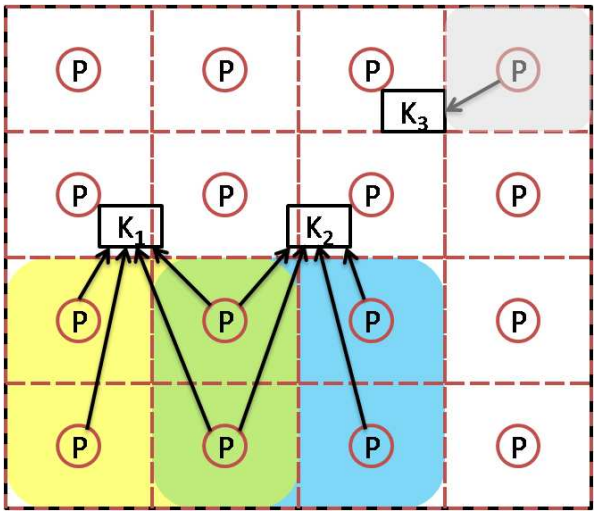

Abbildung 3.1: Das Systemmodell mit sich überschneidenden Anfragen

Der Anwendungszweck des Systems wurde bereits auf das Übertragen von Temperaturdaten zu bestimmten geographischen Bereichen festgelegt. Dies soll im Kontext einer Vernetzung einer Vielzahl von Sensoren zu einem globalen Sensornetz geschehen. Um ein Szenario globaler Größenordnung darstellen zu können, werden Konsumenten benötigt, die neben lokalen Interessen auch globale Ziele verfolgen.

Die Rolle der Konsumenten wird in diesem Anwendungsszenario von Nichtregierungsorganisationen (engl. non-governmental organizations, NGOs) eingenommen, die sich sowohl für sehr lokale, als auch für globale Belange einsetzen. Die Produzenten sind die Gateways zu einzelnen Sensornetzen, die zu einem globalen Sensornetz verbunden werden. Als betrachtete Welt wird eine als Rechteck approximierte Version unserer Erde gewählt. Obwohl dies eine sehr ungenaue Approximierung darstellt, ist sie dennoch für den hier vorgesehenen Zweck ausreichend.

Auf der Welt und auch speziell in Deutschland besteht eine große Zahl von Nichtregierungsorganisationen. Im Jahre 2004 waren über 7300 Organisationen bei der Union of International Associations (UIA) registriert, auf die die streng interpretierte Bezeichnung NGO zutraf. Erweitert man diese Definition auf alle privaten Akteure und Interessengruppen, ist von mehr als 20 000 Organisationen die Rede [IA06].

Diese Organisationen beschäftigen sich unter anderem mit der Entwicklung der Umweltbedingungen in verschiedensten Gebieten der Erde. Der Fokus einer NGO kann global oder auch sehr lokal sein, je nach der Einbettung der Organisation in ihren politischen Kontext. Beispiele sind die Betrachtung des Weltklimas in Bezug auf die globale Erwärmung durch Organisationen wie Greenpeace[1], die Überwachung von Lebensräumen bedrohter

[1] http://www.greenpeace.org

Tierarten durch den Bund für Umwelt- und Naturschutz Deutschland (BUND)[2] oder den Naturschutzbund Deutschland (NABU)[3].

Für das erste Beispiel ist es notwendig, regelmäßige Aktualisierungen für die aktuelle Durchschnittstemperatur zu erhalten. Eine Organisation wird ihre Anfragen abhängig von ihrem geographischen Fokus platzieren. Es ist also denkbar, dass es verschiedene Organisationen gibt, die an den gemittelten Temperaturdaten von jeweils unterschiedlichen geographischen Bereichen interessiert sind, angefangen von der Entwicklung innerhalb einer Stadt oder eines Naturschutzgebietes bis hin zur Entwicklung der weltweiten Situation. Diese Anfragebereiche beinhalten sich gegenseitig, was bedeutet, dass die Ergebnisdaten der kleineren Anfragen als Teilergebnisse für die größeren Anfragen dienen können.

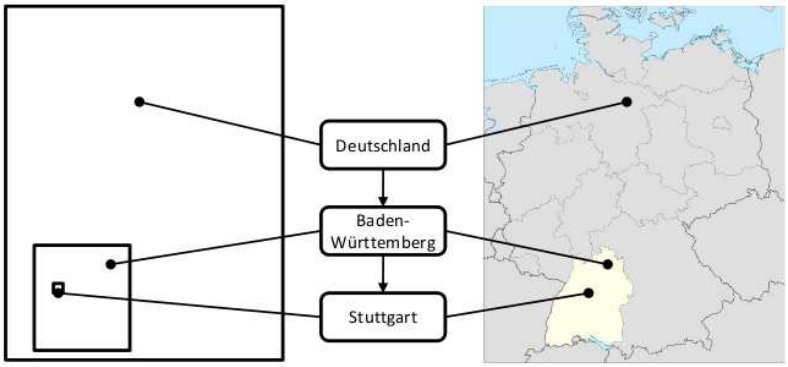

Abbildung 3.2: Kleinere Anfragen liefern Teilergebnisse für größere Anfragen: Durschnittstemperaturen in Deutschland, Baden-Württemberg und Stuttgart

Die Anfragen für das zweite Beispiel werden dagegen mit hoher Wahrscheinlichkeit nicht so stark in ihrer Größe variieren. Stattdessen ist es eher wahrscheinlich, dass es sich um Bereiche ähnlicher Größe handelt, die räumliche Gemeinsamkeiten aufweisen. Das bedeutet, es kommt zu Überschneidungen von einzelnen Anfragen.

Auch abseits der Natur ist die kontinuierliche Überwachung einzelner Messwerte von Bedeutung. In nahezu allen großen Städten existieren Stationen, deren Messungen für die Ausgabe von Ozonwarnungen genutzt werden. Dabei spielt gerade die Umgebungstemperatur eine entscheidende Rolle.

In der bisherigen Situation akzeptieren Produzenten jede Anfrage für eine ausgehende Verbindung, auch wenn die fraglichen Werte bereits an einen anderen Konsumenten übermittelt werden. Die hohe Zahl der ausgehenden Verbindungen führt zu einer starken Belastung der Produzenten. Die Konsumenten müssen oftmals auf aktuelle Werte verzichten, da die Kos-

[2] http://www.bund.net/
[3] http://www.nabu.de

tenmodelle der Dienstleister, die diese Sensordaten bereitstellen, sich an der Verfügbarkeit und Aktualität der Daten orientieren.[4]

Ein System, das die Gesamtmenge aller Anfragen verwaltet, kann durch bestimmte Strategien zur Informationsverteilung für eine gleichmäßige und faire Lastverteilung sorgen. Das führt dazu, dass die Produzenten nicht mehr den Flaschenhals der Anwendung darstellen und die Konsumenten wesentlich häufiger Aktualisierungen für die von ihnen gewünschten Daten erhalten können.

[4]Quelle: Deutscher Wetterdienst, http://www.dwd.de

Kapitel 4

Verwandte Arbeiten

Um die Unterschiede des in dieser Arbeit vorgestellten System zu bereits bestehenden Stream Processing Systemen klar einordnen zu können, werden zunächst die prominenten Anwendungen zur Verarbeitung von Datenströmen aus globalen Sensornetzen beschrieben. Dazu zählen, insbesondere aber nicht ausschließlich, Stream Processing Systeme, die die Informationen aus GSNs für eine Masse von Anwendern verfügbar machen.

Da das hier konzipierte System einen verteilten R-Baum zur internen Organisation nutzt, wird im Folgenden dargelegt, welche verschiedenen verteilten Ansätze auf Basis von R-Bäumen bereits existieren und worin der entscheidende Unterschied zu dem in dieser Arbeit vorgestellten Verfahren liegt.

Die Verwendung von Vorhersagetechniken hat im Bereich der Sensornetze bereits einige Aufmerksamkeit erfahren. Die in diesem Bereich grundlegenden Arbeiten werden ebenfalls erwähnt und gegen die in dieser Arbeit verwendete Lösung abgegrenzt.

4.1 Globale Sensornetze und Stream Processing Systems

Hourglass [SPL+04] stellt eine Publish/Subscribe-basierte Infrastruktur bereit, um Daten aus mehreren heterogenen Sensornetzen zu beziehen. Die verwendete Infrastruktur, an die konkrete Anforderungen bezüglich Ausfallsicherheit und die Verfügbarkeit von Ressourcen gestellt wird, ist dabei vom Dienstanbieter selbst im Netzwerk zu installieren. Mit dem verwendeten Publish/Subscribe-System ist es für einen Interessenten nun möglich, an einem dieser Infrastruktursysteme Aktualisierungen für bestimmte Datensätze zu beantragen. Wird von einem Produzenten ein neuer Datensatz erzeugt, veranlasst Hourglass die Zustellung an alle Konsumenten, die Daten dieses Typs abonniert haben. Hourglass erstellt eine Infrastruktur zur Vermittlung von Datenströmen, setzt dabei allerdings die Verfügbarkeit von dedizierten Systemen voraus, die sich nur um die Vermittlung und Verteilung der Datenpakete kümmern. Das in dieser Arbeit vorgestellte System nutzt hingegen die Ressourcen der Konsumenten selbst, um die Vermittlung zu den Produzenten und die Verteilung der Datenpakete vorzunehmen.

Die Autoren von [AHS06] entwerfen mit Global Sensor Networks – oder kurz GSN – eine Middleware, die mit einer Standardisierung der Kommunikation zu einzelnen Sensornetzen auf eine Beschleunigung der Entwicklung neuer Anwendungen, die Sensordatenströme nutzen, abzielt. Um diese Middleware vom bereits in anderem Kontext gebrauchten Begriff der globalen Sensornetze abzugrenzen, wird diese fortan als GSN-Middleware bezeichnet. Dazu werden die Datenquellen zu virtuellen Sensoren abstrahiert, was eine von der tatsächlichen Sensorhardware unabhängige Anfragesprache ermöglicht. Diese Anfragesprache wird in XML ausgeliefert und bietet neben den gängigen Operatoren auch z. B. Möglichkeiten, die Lebensdauer einer Anfrage zu beschränken.

Der auslösende Gedanke für ein System wie die GSN-Middleware, das Zusammenführen einzelner Sensornetze zu einem globalen Verbund zu ermöglichen und diese Menge an Daten vielen Interessenten zur Verfügung zu stellen, spiegelt sich auch in dem in dieser Arbeit entworfenen DSPS wider. Während sich die Autoren der GSN-Middleware jedoch vorwiegend mit den Details zur Kommunikation mit den einzelnen Sensornetzen und den Fragen beschäftigen, wie eine standardisierte Adressierung beliebiger Datenströme zu erreichen ist, wird diese Möglichkeit in dieser Arbeit bereits vorausgesetzt. Die weitergehenden Probleme, wie die Entlastung der Produzenten durch Vermeidung redundanter Datenströme oder die Realisierung der Echtzeitverarbeitung auch auf Verbindungen mit hohen Latenzen, die diese Arbeit behandelt, finden in der GSN-Middleware keinen Platz.

Aurora [CcC+02, ACc+03] ist einer der ersten Vertreter in der Klasse der zentralistisch aufgebauten Stream Processing Systems. Das definierte Ziel ist es, den speziellen Anforderungen für Stream Processing gerecht zu werden und ein effizientes System zur Verarbeitung von Datenströmen zu beschreiben. Die von den Interessenten gestellten Anfragen werden auf einem zentralen Server gespeichert und verwaltet.

Für Aurora werden eine Reihe von Operatoren definiert, die inzwischen als Standard gelten, unter anderem Filter, Map, Union und Aggregate. Im Aurora-System werden bereits Optimierungen bezüglich der Anfragen durchgeführt. Dafür orientieren sich die Autoren an traditionellen Datenbanken, die bereits eine Neuordnung der Elemente einer Anfrage durchführen, wenn dadurch Ressourcen eingespart werden können. Im Gegensatz zu Aurora ist das in dieser Arbeit vorgestellte System verteilt organisiert und verfügt über eine Vielzahl von verschiedenen Sensornetzen als Eingabequelle, während Aurora von einer einzelnen Datenquelle ausgeht.

In [CBB+03] wird eine Kombination von mehreren Aurora-Systemen beschrieben: Aurora*. Das Ziel der Autoren ist es, die Skalierbarkeit und Verfügbarkeit von Aurora zu stärken. Außerdem führen die Autoren mit Medusa eine verteilte Infrastruktur zum Zugriff auf Sensornetze ein, die von Aurora* genutzt wird. Durch die in Aurora* eingeführten Änderungen kann das Stream Processing System erstmals auf mehrere Einzelsysteme verteilt werden. Die Verteiltheit in Aurora* wird also durch das Aufbauen mehrerer Instanzen von Aurora-Systemen erreicht. Dafür sind wiederum speziell dedizierte Systeme erforderlich, die dem Gesamtsystem bekannt sind und vom Dienstanbieter bereitgestellt werden müssen. Im Gegensatz dazu wird in dieser Arbeit ein Stream Processing System vorgestellt, das schon in seiner Grundstruktur verteilt aufgebaut ist.

In [AAB+05] wird mit Borealis der Nachfolger der beiden Aurora-Systeme vorgestellt. Die Autoren bezeichnen Borealis auch als Stream Processing System der zweiten Generation, da es als eines der ersten Systeme die beiden Probleme behandelt, sowohl aus einer sehr großen Zahl von Sensoren Daten zu extrahieren als auch extrem große Datenströme in adäquater Zeit verarbeiten zu können. Die Autoren nennen diese beiden Optimierungsarten *sensor-heavy* bzw. *server-heavy*. Borealis spaltet die auf den Datenströmen auszuführenden Operationen in Einzeloperatoren auf und versucht, diese möglichst gut im Netzwerk zu platzieren. Dabei spielen vor allem Bandbreite und Latenz der dafür zur Verfügung stehenden Systeme eine Rolle.

Zur Laufzeit wird die Verarbeitung der Datenströme mit einem Prioritätskonzept optimiert: Pakete mit höherer Priorität werden an den einzelnen Operatoren auch bevorzugt behandelt. Weiterhin bietet Borealis die Möglichkeit, Vorhersagen über die Dienstqualität zu machen, um diese Daten für eine dynamische Priorisierung von Datenpaketen zu nutzen.

Eine *time-travel* genannte Funktionalität ermöglicht es, aufgezeichnete Datenströme wiederzugeben und eine Korrektur an alten, fehlerhaft übermittelten Daten vorzunehmen. Die Operatorenmenge von Borealis umfasst zusätzlich zu den bereits aus anderen Systemen bekannten noch frei konfigurierbare Filter, deren Bedingungen noch zur Laufzeit angepasst werden können. Trotz der Vielzahl an Optimierungen ist die Skalierbarkeit von Borealis durch die Anzahl der ausgehenden Verbindungen für die Datenquellen begrenzt. Da Gemeinsamkeiten zwischen den existierenden Datenströmen nicht erkannt und ausgenutzt werden, steigt die Zahl der zu den Produzenten aufgebauten Datenverbindungen zusammen mit der Zahl der vom System bedienten Konsumenten. In dieser Arbeit wird dagegen ein System entworfen, das durch das Vermeiden von Datenströmen mit gleichen Inhalten eine stark verbesserte Skalierbarkeit und ein reduziertes Datenaufkommen bietet.

In [PLS+06] führen die Autoren ein stream-based overlay network, kurz SBON, ein, das beliebige DSPS ergänzen kann. Zunächst werden die Operationen, die die anfragenden Anwendungen auf die zur Verfügung gestellten Datenströme anwenden wollen, in Einzeloperatoren aufgespalten. Anschließend erlaubt SBON eine flexible Platzierung dieser Operatoren. Auf Basis von Bandbreite und Latenz werden die Gesamtkosten für die Informationsübertragung innerhalb des Systems minimiert. SBON bietet außerdem noch eine Lösung für ein weiteres Problem in DSPS: Die redundante Datenübertragung. Der verwendete Algorithmus erlaubt es, gleiche Operatoren zu erkennen und zu verschmelzen. Operatoren gelten als gleich, wenn sie die gleiche Operation auf Datenströmen aus denselben Quellen durchführen. Durch das Verschmelzen von Operatoren bzw. ganzen Ketten von Operatoren wird also erreicht, dass die gleiche Information nicht mehrfach im Netz zwischen Produzenten und Konsumenten übertragen wird. In SBON wird der Verbindungsaufbau vom Erkennen gleicher Datenströmen entkoppelt. Das in dieser Arbeit vorgestellte System hingegen führt diese beiden Tätigkeiten gleichzeitig aus: Schon während des Verbindungsaufbaus einer neuen Anfrage werden Quellen, die die gleichen Daten bereits erhalten, entdeckt.

4.2 Verteilte Lösungen auf Basis von R-Bäumen

Der NR-Tree [LLL05] stellt eine auf dem R-Baum basierende Peer-to-Peer-Struktur zur effizienten Unterstützung von kNN-Anfragen[1] dar. Er setzt eine Super-Peer-Architektur ein, wie sie auch in den verbesserten Versionen des Gnutella-Protokolls eingesetzt wird. Das Prinzip hierbei ist die Ungleichverteilung von Aufgaben und Last auf die verschiedenen Peers.

Die zu Grunde liegende Architektur ist CAN [RFH+01], wobei die verschiedenen Zonen hier Cluster genannt werden. In jedem Cluster wird ein Verwalter für dieses Gebiet, der sogenannte Clusterhead, bestimmt. Dieser stellt Informationen über die in seinem Cluster vorhandenen Daten nach außen zur Verfügung. Auf die Daten eines Clusters kann nur direkt über den Clusterhead zugegriffen werden. Jeder normale Peer eines Clusters indiziert seine Daten in einem R-Baum und übermittelt das MBR des obersten Knotens an seinen Clusterhead. Der Clusterhead ordnet die von seinen Peers erhaltenen MBRs wiederum in einem R-Baum an und hat somit schnellen Zugriff auf die von den jeweiligen Peers verwalteten Datensätze.

Das Vorgehen bei Bereichsanfragen ist folgendermaßen: Die Anfrage selbst wird als Rechteck formuliert. Dann wird der Zentroid dieses Rechtecks bestimmt und mit den aus CAN bekannten geographischen Routingalgorithmen in den dafür zuständigen Cluster weitergeleitet. In diesem Cluster wird die Anfrage zum Clusterhead geleitet, der in seinem R-Baum Informationen über die ihm unterstellten Peers abfragen kann. Es besteht jedoch die Möglichkeit, dass auch die Nachbarn dieses Clusters in den Einzugsbereich der Anfrage fallen. Um darüber Gewissheit zu erlangen muss der Clusterhead alle seine Nachbarn befragen, ob sie in ihrem R-Baum Treffer für die vorliegende Anfrage finden. Die Anfragen wird also mit einer geringen Lebensdauer (engl. time to live, TTL) im System geflutet.

Die Autoren des NR-Trees machen verschiedene Annahmen, die nicht oder nur bedingt auf das in dieser Arbeit vorgestellte System übertragbar sind. So ist die Unterstützung von Bereichsanfragen, dem wichtigsten Anfragentyp, durch Fluten des Systems realisiert. Im NR-Tree werden nur die Super-Peers mit diesen Anfragen belastet; die Existenz von Systemen mit ausreichenden Ressourcen ist allerdings nicht gewährleistet. Derartige Voraussetzungen an die Infrastruktur und die teilnehmenden Peers werden vom Systemmodell (siehe Kapitel 3.2) nicht gestützt.

Auch der Peer-Tree [DF03] verwendet eine Super-Peer-Architektur, um die Sensoren innerhalb eines Sensornetzes zu verwalten. Der Fokus dieses System liegt auf der effizienten Unterstützung von mehrdimensionalen NN-Anfragen[2]. Der Peer-Tree integriert einzelne Sensoren in einen Baum, um in diesem effizient Anfragen nach dem Objekt stellen zu können, das einer bestimmten Position am nächsten ist. Das Ziel ist es, nur die Sensorknoten ausfindig zu machen, die zur Beantwortung einer Anfrage notwendig sind.

[1] k Nearest Neighbours, findet die k nächsten Nachbarn eines Knotens
[2] Nearest Neighbour, findet den nächsten Nachbarn eines Knotens

Analog zum NR-Tree werden die Sensoren in Cluster unterteilt, allerdings nicht mit einem darunterliegenden CAN. Stattdessen wird die Welt anhand der Anzahl der Sensoren dynamisch in Rechtecke partitioniert. Anschließend wird für jeden Cluster ein Clusterhead bestimmt, der an der Konstruktion der rechteckigen Cluster des nächsten Levels teilnimmt. Die Daten selbst lassen sich nur an den Sensorknoten auf dem untersten Level des Baumes abrufen, auf den anderen Levels des Baumes nehmen die Knoten nur Verwaltungsaufgaben wahr.

Eine Anfrage kann an jeden am System beteiligten Knoten gestellt werden. Dieser leitet die Anfrage so lange weiter nach oben im Baum, bis ein Knoten gefunden wird, in dessen Verwaltungsbereich die Anfrage fällt.

Obwohl der Peer-Tree direkt das Thema der Sensornetze anspricht, ignoriert er das Problem der Datenverteilung. Er wurde erstellt, um effizient die Sensoren finden zu können, die zur Beantwortung einer bestimmten Anfrage geeignet sind. Die Autoren begründen die Notwendigkeit für eine solche Struktur damit, dass wegen der stetig wachsenden Größe der Sensornetze eine zentralisierte Lösung zur Verwaltung eines Netzes nicht geeignet sei.

Da der im weiteren Verlauf dieser Arbeit vorgestellte Algorithmus die Kommunikation innerhalb eines Sensornetzes nicht beachtet, finden die Konzepte des NR-Trees zum Großteil keine Anwendung. Der Gedanke der Anfragenweiterleitung innerhalb des Baumes findet sich jedoch in Kapitel 5.3.1 wieder.

Der Distributed R-Tree, oder kurz DR-Tree [BFG07], ist eine verteilte Variante des R*-Baumes. Er wurde für inhaltsbasierte Publish/Subscribe-Systeme entworfen. Diese Struktur wurde gewählt, um die Verteilung der Inhalte möglichst effizient zu gestalten und dabei das Vorkommen von *falschen Positiven* (ein Teilnehmer bekommt ein Ereignis zugestellt, an dem er nicht interessiert war) zu minimieren und *falsche Negative* (ein Teilnehmer bekommt ein Ereignis nicht, obwohl es seinen Abonnement-Parametern entsprach) gänzlich zu verhindern.

Die Teilnehmer werden durch Hyper-Rechtecke im mehrdimensionalen Raum dargestellt. Die Achsen entsprechen den Attributen, nach denen der Inhalt gefiltert werden kann. Für den zweidimensionalen Raum heißt das: Die Fläche, die ein Teilnehmer aufspannt, entspricht dem Themenbereich, an dem er interessiert ist. Teilnehmer mit ähnlichen Interessen werden zu größeren Rechtecken zusammengefasst, wodurch eine Hierarchie aus MBRs gebildet wird. Man spricht auch von *semantischen Gemeinschaften*, die durch diese Zusammenfassungen entstehen. Die Verteilung der Information erfolgt dann von der Wurzel des durch die Hierarchisierung entstandenen Baumes aus.

Jeder Knoten in der Baumstruktur verfügt nur über beschränktes Wissen (gespeichert werden die Kindknoten sowie der Vater). Also wird jedes MBR durch ein System in der realen Welt repräsentiert, das Informationen an eine Menge von anderen Systemen verteilt. Der Informationsbedarf aller unter ihm liegenden Systeme wird zu einem bestimmten Bereich zusammengefasst – hier dargestellt durch das entsprechende Rechteck. Jeder Abonnent tritt mit seinem eigenen MBR an genau einer Stelle als Blatt im Baum auf. Weiterhin werden aus den Blättern einige Knoten ausgewählt, die die Blätter in ihrer Nähe verwalten sollen. So entsteht eine Hierarchisierung, die auch dazu führt, dass die Knoten, die zur Verwaltung ausgewählt werden, sich selbst enthalten müssen. Der Bereich innerhalb eines MBRs eines

Knotens, der nicht von seinen Kindern abgedeckt wird, stellt die oben erwähnten *falschen Positiven* dar, die das System zu minimieren versucht.

Der DR-Tree beschäftigt sich als einziges der hier vorgestellten auf R-Bäumen basierenden Systeme mit der tatsächlichen Verteilung der Daten. Die übrigen Verfahren dienen lediglich dem Auffinden von Daten, bzw. Sensoren, die die gewünschten Daten liefern, nicht dem tatsächlichen Datentransport selbst.

Wie im Peer-Tree wird hier auch jedes MBR durch einen physikalischen Knoten repräsentiert, was dazu führt, dass verschiedene Instanzen eines physikalischen Knotens an verschiedenen Stellen im Baum auftreten können. Was nicht behandelt wird, sind Mechanismen zur Replikation. Die Autoren des DR-Trees beschränken sich darauf, nur ein vorher angekündigtes Verlassen von Knoten zu betrachten. Der fehlerbedingte Ausfall einzelner Knoten oder gar Churn wird nicht berücksichtigt.

Das wichtigste Prinzip des DR-Trees, das Eingang in den in dieser Arbeit vorgestellten Algorithmus findet, ist die Beteiligung der Informationsempfänger an der Weiterverteilung der Information an weitere Interessenten. Während dies bei Publish/Subscribe-System bereits zum Standard geworden ist, wird dieser Ansatz im Bereich der Sensornetze noch nicht verfolgt. Dass die Verteilung der Daten von der Wurzel aus abwärts erfolgt, ist dem Anwendungszweck Publish/Subscribe geschuldet. Das in dieser Arbeit vorgestellte Verfahren wird zum Verteilen von Sensorinformationen eingesetzt und geht den umgekehrten Weg: Die Sensordaten werden aufsteigend von unten nach oben im Baum propagiert.

Der TPR*-Tree [TPS03] stellt eine Weiterentwicklung des TPR-Trees dar und stattet den Benutzer mit einem effizienten Werkzeug zur Beantwortung von kontinuierlichen kNN-Anfragen aus. Die Struktur des TPR-Baumes wurde für Anfragen entworfen, die sowohl eine Raum- als auch eine Zeitkomponente enthalten (engl. spatio-temporal queries) was bedeutet, dass die indexierten Objekte sich nicht an einer Stelle befinden, sondern vielmehr einer stetigen Bewegung unterworfen sind. Hier findet ein interessanter Aspekt Eingang in die R-Baum-Struktur: Neben den üblichen MBRs werden auch Velocity Bounding Rectangles – sogenannte VBRs – erzeugt, deren Aufgabe es ist, die momentane Bewegung eines Objektes abzubilden.

Mit Hilfe dieser VBRs lässt sich voraussagen, wo sich ein Objekt zu einem bestimmten Zeitpunkt befinden wird. Damit kann der TPR*-Tree für jedes indexierte Datum beantworten, welche Objekte seine nächsten Nachbarn in z. B. fünf Minuten sein werden.

Der Vorhersageaspekt des TPR*-Trees ist insofern interessant, als in keinem der anderen hier vorgestellten Baumstrukturen die Konstruktion eines Modells erfolgt, um bestimmte Eigenschaften der Datenobjekte verfolgen und im Voraus bestimmen zu können. Allerdings beschränkt sich diese Modellierung auf die Geschwindigkeitskomponente und bietet keinerlei Möglichkeit, auf andere Sensordaten ausgedehnt zu werden.

4.3 Vorhersageverfahren

Einer der ersten skalierenden vohersagebasierten Ansätze, die Kommunikation in Systemen zur Verarbeitung von Datenströmen zu reduzieren, wurde in [JCW04] vorgestellt. Erstmals wurde das Vorhaben, weniger Informationen vom Sender zum Empfänger verschicken zu müssen, als Filterproblem betrachtet. Während die vorherigen Ansätze vor allem verschiedene Caching-Methoden nutzten, um die Aktualisierungen der Daten seltener durchführen zu müssen, oder aber versuchten, ein zentrales Weltmodell zur Vorhersage zu erstellen, zielt der Dual-Kalman-Filter-Ansatz darauf ab, möglichst viele Pakete aus den Datenströmen der einzelnen Quellen zu filtern, ohne ein gewisses Maß an Ungenauigkeit zu überschreiten.

Die Autoren benutzen das Kalman-Filter [Kal60], um sowohl auf Empfänger- als auch auf Senderseite den Verlauf der Sensordaten zu modellieren. Die Datenquelle sendet nur dann Aktualisierungen, wenn der vom Filter vorhergesagte Wert um mehr als einen gewissen Grenzwert vom tatsächlichen Wert abweicht. Daraufhin müssen die Parameter des Filters auf beiden Seiten aktualisiert werden. Der Empfänger – ein zentraler Server, der die Anfragen eventueller Interessenten nach außen kapselt – installiert so viele Kalman-Filter, wie es Datenquellen gibt, um den Verlauf der einzelnen Quellen bei sich vorhersagen zu können.

Die Wahl des Filters wird damit begründet, dass es sich bei Kalman-Filtern um eine flexible und gut erforschte Möglichkeit handele, ein Hauptsignal mitsamt eventuellen Störsignalen zu modellieren. Die verwendeten Kalman-Filter bringen jedoch auch erhebliche Nachteile mit sich: Obwohl sie verglichen mit den von den Autoren aufgeführten Caching-Lösungen wesentlich flexibler sind, weisen auch sie eine starke Anwendungsspezifizität auf. Das bedeutet im Detail, dass die Übergangsmatrizen des Filters anhand eines Modells gewählt werden. Somit muss entweder schon im Voraus ein Wissen über die transportierten Daten bestehen, oder aber ein genügend großer Vorrat an verschiedenen Modellen in den Filter integriert sein, aus dem die passenden Parameter ausgewählt werden können.

In [San06] wird das Prinzip eines adaptiven Filters in den Kontext der Sensornetze eingeführt, um den Verlauf von Messwerten eines Sensors zu modellieren. Das Ziel dieses Filters ist es, möglichst viel an Kommunikation zwischen dem einzelnen Sensor und der Senke eines Sensornetzes einsparen zu können, um Energie für die mit begrenzten und nur sehr geringen Kapazitäten ausgestatteten Sensoren zu sparen und so deren Lebensdauer zu erhöhen.

Sowohl Empfänger – hier eine zentrale Senke für das gesamte Sensornetz – als auch Sender der Sensordaten werden mit einem adaptiven Filter ausgestattet, die jeweils dieselben Berechnungen durchführen. Die Senke installiert dabei ebenfalls so viele Filter, wie sie eingehende Datenverbindungen hat. Der Kern eines solchen Filters ist der Least-Mean-Square Algorithmus, der den nächsten Messwert aus einer Linearkombination der letzten n Werte berechnet. Auf die Verwendung von Kalman-Filtern wird bewusst verzichtet, um den Ansatz so flexibel wie möglich zu halten und nicht auf bestimmtes Vorwissen über die transportierten Daten angewiesen zu sein.

Es wird zwischen drei Betriebsmodi für die Sensorknoten unterschieden: Der *Initialisierungsmodus* dient dazu, auf Empfänger- und Senderseite die jeweils für eine Vorhersage notwendigen Daten zu sammeln. Wurden die notwendigen Daten gesammelt, geht der

Knoten in den *normalen Betriebsmodus*. In diesem Modus werden die gemessenen Daten ganz normal an die Senke übermittelt und das Vorhersagemodell stetig angepasst. Die Messwerte werden außerdem mit der Vorhersage verglichen. Liegt der vorhergesagte Wert fünfmal in Folge innerhalb eines gewissen Toleranzbereiches um den tatsächlichen Messwert, so wechselt der Sensorknoten in den *autonomen Modus*. Dieser Wechsel wird auch von der Senke bemerkt, da auch diese die gemessenen Daten und die Vorhersage vergleichen kann. Im *autonomen Modus* erfolgt so lange keine Aktualisierung mit Messdaten durch den Sensorknoten, bis das Vorhersagemodell um mehr als den vordefinierten Grenzwert von den tatsächlichen Messwerten abweicht. In dieser Zeit verwendet der Empfänger seine Vorhersage, um Anfragen zu beantworten.

Beide Knoten aktualisieren ihr Vorhersagemodell vollkommen selbstständig, weshalb eine ausfallsichere Kommunikation in diesem Szenario von besonderer Wichtigkeit ist. Stellt der Sensor eine zu große Abweichung des Vorhersagemodells fest, so sendet er eine Aktualisierung an den Empfänger und verändert gleichzeitig sein Vorhersagemodell. Geht diese Nachricht verloren, vermutet der Empfänger weiterhin die Korrektheit des von ihm verwendeten Modells und aktualisiert dieses nicht mit dem neuen Messwert. Die Modelle der beiden Parteien laufen auseinander, was zu Inkonsistenzen im gesamten System führt.

Für PRESTO [LGS06] erweitern seine Autoren die Idee des adaptiven Filters auf Prädiktoren. Im Gegensatz zum o. g. Ansatz aktualisieren die beiden Parteien ihre Filter nicht autonom voneinander. Stattdessen wird hier lediglich der Empfänger der Sensordaten dazu benutzt, Berechnungen anzustellen, um den Verlauf der Sensordaten zu modellieren. Die berechneten Parameter teilt er daraufhin dem Sender mit, so dass dieser über das gleiche Modell zur Vorhersage verfügt. Dadurch wird die Abhängigkeit von einer fehlerfreien Datenverbindung reduziert.

In PRESTO wird auf ein S-ARIMA-Modell (eine Instanz eines Box-Jenkins-Modells [BJ90]) zurückgegriffen, das bisher besonders in der Ökonometrie und Finanzmathematik Anwendung findet. In diesem Modell lassen sich Parameter bestimmen, die eine saisonale Schwankung der Daten, zufällige Störwerte („weißes Rauschen“) und eine gewisse Anzahl der vorhergehenden Messwerte berücksichtigen und in die Berechnung des als nächstes folgenden Wertes einfließen lassen.

Hat der Empfänger der Sensordaten die Modellparameter ermittelt, wird sowohl auf dem Sensor als auch dem Empfänger der Sensordaten dieses Modell regelmäßig ausgeführt. Statt den aktuellen Messwert zu übermitteln, überprüft der Sensor nun, um wie viel sich dieser Messwert von dem durch das Modell vorhergesagten Wert unterscheidet. Überschreitet dieser Unterschied einen gewissen Grenzwert, so wird ein Update an den Empfänger gesendet, andernfalls erfolgt keine Kommunikation. Da das Senden von Informationen in Sensornetzwerken im Bezug auf die Energiereserven des einzelnen Sensors wesentlich teurer ist als das Anstellen von Berechnungen, kann durch dieses Vorhersageverfahren die Lebensdauer der Sensoren erheblich erhöht werden.

Während im PRESTO-System lediglich die sensornetzinterne Kommunikation angesprochen wird, erweitert der in dieser Arbeit vorgestellte Algorithmus das Verfahren auf die Kommunikation zwischen Anfragenstellern und den Gateways zu den Sensornetzen. Der

Prädiktor dient dann nicht nur zur bloßen Reduktion der Datenmenge, sondern kann dem Kosumenten auch einen Echtzeitdatenstrom gewährleisten.

Die Verwendung von Prädiktoren hat zur Folge, dass die daraus entstehenden Probleme bezüglich der Erkennung von Knotenausfällen – das Ausbleiben eines Updates impliziert die Korrektheit des Modells und nicht den Ausfall eines Knotens – einer gesonderten Betrachtung bedürfen. Die Autoren von PRESTO argumentieren, die regelmäßig erfolgenden Modellupdates, mit denen die saisonale Komponente des Modells verfeinert wird, müssten nur in adäquaten Abständen erfolgen, um Sensorenausfälle in vertretbarer Zeit erkennen zu können. In ihrem Beispiel setzen sie die Aktualisierungsperiode auf einen Tag, so dass spätestens nach 24 Stunden deutlich wird, dass ein Sensor ausgefallen ist. Während dies für einzelne Sensoren noch tragbar erscheint, muss für das hier entworfene System ein anderes Prinzip gefunden werden, da der Ausfall ganzer Sensornetze wesentlich schwerer wiegt und daher auch schneller erkannt werden muss.

Die Autoren von PAQ [TM06], kurz für Probabilistic Adaptable Query System, erstellen ein Framework, das Autoregressionsmodelle innerhalb eines Sensornetzes einsetzt, um die Menge der gesendeten Daten zu reduzieren. In diesem System sind es die Sensoren selbst, die für die Konstruktion der Vorhersagemodelle zuständig sind. Diese Modelle werden dann an die Senke übermittelt, so dass nicht mehr jeder gemessene Wert, sondern nur diejenigen, die um mehr als einen spezifizierten Fehler von der Vorhersage abweichen, an die Senke gesendet werden.

Außerdem schließt PAQ Sensoren, die nebeneinander liegen, zu Clustern zusammen, um die innerhalb dieser Cluster erwartete Ähnlichkeit der Messwerte ausnutzen zu können. Die Autoren argumentieren, dass Sensoren, die räumlich nah beieinander liegen, mit hoher Wahrscheinlichkeit einen Zusammenhang in ihren Datenverläufen aufweisen. Dadurch wird es auch möglich, den Ausfall von Sensoren durch Interpolation besser abzufangen und fehlerhafte Sensoren mit großer Sicherheit zu erkennen.

Die Autoren von PAQ betonen, dass ihr System in der Lage sei, konkrete Aussagen über die Datenqualität zu machen. Sie können den maximalen Fehler sowie die Fehlerwahrscheinlichkeit für einzelne Sensoren angeben. Wie in den anderen hier beschriebenen Systemen, die Vorhersagemodelle einsetzen, wird diese Technik auch in PAQ hauptsächlich dazu eingesetzt, die Lebensdauer der Sensoren zu erhöhen. In dem in dieser Arbeit konzipierten System dagegen werden Vorhersagemodelle zur Reduktion des Gesamtdatenaufkommens und zur Bereitstellung von Echtezitdaten verwendet.

Kapitel 5

Konzepte

Im bisherigen Teil dieser Arbeit wurde in Kapitel 2 eine Einführung in die Grundlagen von globalen Sensornetzen und Stream Processing gegeben, anschließend wurden die speziellen Anforderungen für Stream Processing Systems im Kontext der globalen Sensornetze erläutert.

Darauf aufbauend konnte in Kapitel 3 gezeigt werden, dass die existierenden Stream Processing Systems Defizite in der Erfüllung der an sie gestellten Anforderungen aufweisen. Zwei dieser Anforderungen wurden herausgegriffen und als Schwerpunkte definiert: Die Vermeidung von redundanten Datenströmen, um eine Entlastung der Produzenten und somit eine verbesserte Skalierbarkeit des Gesamtsystems zu erreichen. Außerdem die Integration von Prädiktoren zur Vorhersage von Werteverläufen auf allen Datenverbindungen des Stream Processing Systems, um den Datenverkehr stark reduzieren und eine Echtzeitverarbeitung von Datenströmen auch über Verbindungen mit hoher Latenz ermöglichen zu können.

In diesem Kapitel wird ein Konzept für ein Distributed Stream Processing System vorgestellt, das sich explizit der Erfüllung dieser beiden Anforderungsschwerpunkte widmet. Dazu werden im nächsten Abschnitt 5.1 nochmals die spezifischen Anforderungen an das hier vorgestellte Distributed Stream Processing System zusammengefasst. Der darauffolgende Abschnitt 5.2 beschreibt die Architektur des Gesamtsystems inklusive der verwendeten Prädiktoren.

Die Beschreibung des Konzeptes erfolgt in zwei Schritten. Zuerst wird in Kapitel 5.3 ein System zur verteilten Verarbeitung von Datenströmen konzipiert. Dieses System wird bereits redundante Datenströme vermeiden können. Der aus diesem Konzept entwickelte Algorithmus wird danach in Kapitel 5.4 im Detail beschrieben. Anschließend wird für den zweiten Teil des Konzeptes in Kapitel 5.5 in das vorher entstandene System ein modellgestütztes Vorhersageverfahren, der Prädiktor, integriert. Mit Hilfe dieser Prädiktoren wird der Datenverkehr reduziert und eine Echtzeitverarbeitung der Datenströme ermöglicht.

5.1 Anforderungen an den Algorithmus

Die in diesem Abschnitt beschriebenen spezifischen Anforderungen beziehen sich auf die beiden in Abschnitt 3.1 ausgewählten Schwerpunkte und auf das in Abschnitt 3.2 definierte Systemmodell mitsamt seinen Einschränkungen. Der Anwendungszweck des in dieser Arbeit konzipierten Systems wurde in diesen Abschnitten auf die Übermittlung von Temperaturdaten zu bestimmten geographischen Bereichen festgelegt.

Vor diesem Hintergrund werden an das System die folgenden Anforderungen gestellt:

A: Dezentrale Organisation Da es aus finanziellen und organisatorischen Gründen nicht möglich ist, die Infrastruktur für ein Stream Processing System globalen Ausmaßes zu errichten, ist eine dezentrale Lösung unerlässlich. Gerade in Fragen der Skalierbarkeit, Verfügbarkeit und Erweiterbarkeit sind verteilte Lösungen einem zentralisierten Ansatz vorzuziehen.

B: Unterstützung von Bereichsanfragen Nachdem der Anwendungszweck vorsieht, dass das System Ströme von Temperaturdaten zu bestimmten geographischen Bereichen übermittelt, müssen einem Bereich, den ein Interessent in seiner Anfrage definiert, die entsprechenden Datenquellen zugeordnet werden können. Das System muss also eine Unterstützung für die im Systemmodell 3.2 definierte Art der Bereichsanfrage bieten.

C: Hierarchisches Erkennen von Überschneidungen Das System soll redundante Datenströme vermeiden können. Dazu ist es wichtig, Gemeinsamkeiten zwischen Anfragen, für die bereits Informationen bereitgestellt werden, und neuen Anfragen erkennen zu können. In Bezug auf Bereichsanfragen stellen Gemeinsamkeiten Überschneidungen der angefragten geographischen Bereiche dar. Die Erkennung dieser Überschneidungen muss hierarchisch erfolgen, da die größtmögliche Überschneidung gefunden werden soll, um auch den für das Gesamtsystem größtmöglichen Nutzen zu erzielen. Das bedeutet auch, dass die Struktur bereits hierarchisch organisiert sein muss.

D: Entlasten der Produzenten und faire Lastverteilung auf die übrigen Teilnehmer Die Belastung der Gateways als Produzenten von Sensorinformation stellt in nahezu allen Distributed Stream Processing Systems einen Flaschenhals dar. Da die Zahl der ausgehenden Verbindungen eines Produzenten stetig mit der Zahl der anfragenden Teilnehmer steigt, begrenzen die Ressourcen der Gateways die Skalierbarkeit des Gesamtsystems und haben somit auch starke Auswirkungen auf dessen Funktionstüchtigkeit. Der Algorithmus entlastet die Gateways, indem er auch die Peers, also diejenigen Systeme, denen die Gateways Datenströme bereitstellen, an der Datenvermittlung beteiligt. Somit wird erwartet, dass der Konsument einer Information gleichzeitig bereit ist, diese Information bei Bedarf an andere Konsumenten zu übermitteln. Nachdem in den vergangenen Jahren das Peer-to-Peer-Paradigma stetig an Bedeutung gewonnen hat, stellt dies eine übliche Forderung an Systeme

zur Informationsverteilung dar. Diese Umverteilung der Last muss jedoch zugleich nach fairen und für die Teilnehmer nachvollziehbaren Kriterien erfolgen. Ist die Mehrbelastung für einen Peer durch den Beitritt zum Gesamtsystem zu groß oder steht sie in keinem Verhältnis zu dem Nutzen, den er aus dem System gewinnt, wirkt sich dies negativ auf die Akzeptanz des Systems aus. Da in Peer-to-Peer-Systemen gerade die Akzeptanz von Mehraufwand ein kritischer Faktor für die Kooperationsbereitschaft der Teilnehmer ist, darf der Aspekt der Fairness hier nicht vernachlässigt werden.

E: Echtzeitverarbeitung von Datenströmen In einem Stream Processing System globalen Ausmaßes ist es auf Grund der zu erwartenden hohen Latenz auf manchen Verbindungen zwischen Produzent und Konsument der Informationen nicht mehr möglich, durch eine reine Vermittlung der Datenströme Echtzeitdaten bereitzustellen. Stattdessen muss das Stream Processing System andere Techniken, beispielsweise ein beim Konsumenten installiertes Vorhersageverfahren, nutzen, um dieses Ziel zu erreichen.

Mit dem bereits in Kapitel 4 beschriebenen DR-Tree existiert eine Lösung, die die Anforderungen A bis C erfüllt. Der DR-Tree ist dezentral organisiert und bietet die Möglichkeit, Überschneidungen zwischen einem neuen und den bereits bestehenden Datenobjekten hierarchisch zu erkennen. Eingefügt werden neue Objekte mit Hilfe von Bereichsanfragen.

Die Anforderung D wird hingegen nicht erfüllt. Der für Publish/Subscribe-Systeme entworfene DR-Tree verteilt neue Datenpakete nur von der Wurzel aus. Neue Objekte werden zum Einfügen auch immer bis zur Wurzel des Baumes geleitet, um dann von oben eingefügt zu werden. Die Knoten auf den oberen Levels des Baumes werden somit überdurchschnittlich stark belastet. Die ungleiche Belastung der Knoten geschieht außerdem nach für den einzelnen Teilnehmer nicht nachvollziehbaren Kriterien. Alle Systeme, die Anfragen an einen DR-Tree stellen, werden auf dem untersten Level des Baumes als Blätter eingeordnet. Zusätzlich müssen einige Systeme als innere Knoten des Baumes vom zweiten bis vorletzten Level Verwaltungsaufgaben und die Weiterleitung von Datenpaketen übernehmen. Obwohl ein System nur einen flächenmäßig kleinen Bereich angefragt hat, kann es in den oberen Levels des Baumes zur Verwaltung eingesetzt werden. Die Mehrbelastung eines Knotens durch seine Mitgliedschaft im Gesamtsystem steht also in keinem Verhältnis zu dem von ihm gewonnenen Nutzen, was deutlich den Ausführungen in Anforderung D widerspricht.

Anforderung E wird vom DR-Tree ebenfalls nicht erfüllt. Die Autoren haben bei ihrer Konzeption der Struktur ihren Fokus auf eine möglichst effiziente, nicht jedoch auf eine zeitnahe Auslieferung von Datenpaketen gelegt.

Durch die Erfüllung von drei der insgesamt fünf Anforderungen stellt der DR-Tree eine gute Basis für ein Distributed Stream Processing System mit den geforderten Eigenschaften dar. Um den Defiziten bezüglich der weiteren Anforderungen zu begegnen und um den Unterschieden zwischen Publish/Subscribe und Stream Processing Rechnung zu tragen, müssen dennoch wichtige Modifikationen am DR-Tree vorgenommen werden, um ihn für den Zweck dieser Arbeit nutzbar zu machen.

Modifikationen für Anforderung D: Um eine Entlastung der Produzenten zu erreichen und deren Last fair auf die anderen Teilnehmer des Systems zu verteilen, werden mehrere Änderungen am DR-Tree vorgenommen.

Während der DR-Tree alle Anfragen als Blätter in seiner Baumstruktur speichert, können in dem Verfahren, das in dieser Arbeit konzipiert wird, *Anfragen auf jedem Level des Baumes* eingefügt werden. Die Gateways als initiale Datenquellen werden weiterhin als Blätter im Baum verwaltet. Dadurch wird es möglich, dass ein Peer auf seiner Suche nach Datenquellen zuerst andere Peers entdeckt, die die gleichen Daten bereits von Gateways erhalten. Die Produzenten müssen also nur eine ausgehende Verbindung verwalten und werden dadurch entlastet.

Durch die Änderung bezüglich der Position der Anfragen wird ein *neues Einfügeverfahren* erforderlich. Dieses muss bestimmen, auf welchem Level des Baumes eine neue Anfrage eingefügt werden soll. Als weitere Modifikation wird im Einfügeverfahren *kontrolliertes Aufsteigen* eingesetzt. Das bedeutet, dass nicht wie im DR-Tree alle neuen Anfragen bis zur Wurzel im Baum aufsteigen, um von dort eingefügt zu werden, sondern dass beim Einfügen eine Anfrage nur dann weiter im Baum nach oben steigt, wenn dies auch wirklich erforderlich ist. Durch diese Änderung werden die Knoten auf den oberen Levels des Baumes entlastet und somit die durch die Baumstruktur entstehende Mehrbelastung fairer auf die Teilnehmer des Systems verteilt.

Eine weitere Änderung, die zur Entlastung der Knoten auf den oberen Levels des Baumes führt, ist der *Verzicht auf das Wiedereinfügen* von Kindern eines übergelaufenen Knotens. Der DR-Tree verfährt nach dem in Kapitel 2.2.1 beschriebenen Verfahren des R*-Baumes. Dabei wird ein Teil der Kinder eines übergelaufenen Knotens wieder von der Wurzel an neu eingefügt. Auch hierbei werden die Knoten der oberen Levels stark belastet. Das DSPS, das in dieser Arbeit konzipiert wird, setzt stattdessen ein lokales Clusteringverfahren ein, um einen Knoten neu aufzuteilen.

Auch wenn es durch die Summe der obigen Modifikationen nicht möglich ist, die Belastung der Teilnehmer vollkommen unabhängig von ihrer Position im Baum zu gestalten, können die Knoten der oberen Levels doch stark entlastet werden. Da die Position eines Peers im Baum von seinem Nutzen – also der Menge der Daten, die er erhält – abhängig ist, kann eine geringfügig höhere Belastung der Knoten in den oberen Levels des Baumes als fair bezeichnet werden.

Modifikationen für Anforderung E: Um eine Verarbeitung der Datenströme auch auf Verbindungen mit hoher Latenz in Echtzeit gewährleisten zu können, werden *Prädiktoren* auf allen Datenverbindungen innerhalb des gesamten Stream Processing Systems installiert. Mit diesen Prädiktoren wird der zukünftige Verlauf von Datenströmen der Messwerte vorhergesagt, so dass nur noch in Ausnahmefällen Datenpakete zwischen Produzent und Konsument der Information übertragen werden müssen. Da die Autoren des DR-Trees in ihrer Arbeit keine Echtzeitanforderungen definierten, ist ein derartiges Konzept für den DR-Tree auch nicht vorgesehen.

Modifikationen für Stream Processing: Da der DR-Tree als Publish/Subscribe-System entworfen wurde, werden einige Änderungen nötig, um ihn an den geänderten Einsatzzweck des Distributed Stream Processings anzupassen. Datenpakete werden im DR-Tree einfach mit denselben Regeln, die auch zum Einfügen neuer Datenobjekte benutzt wurden, im Baum nach unten weitergeleitet. So erhält jeder Interessent genau die Information, für die er sich registriert hat. Da Daten nur von der Wurzel aus verteilt werden, ist ein gesondertes Auffinden der Datenquelle unnötig.

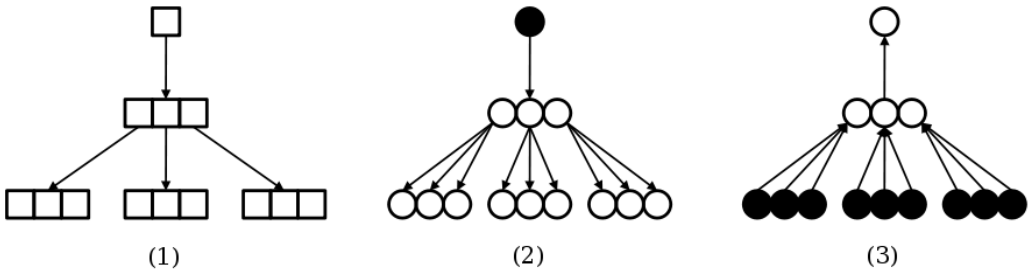

Abbildung 5.1: Vergleich der aus dem R-Baum (1) entstehenden Datenverteilungsgraphen für den DR-Tree (2) und das hier vorgestellte DSPS (3). Die Datenquellen sind schwarz gekennzeichnet

Die Abbildung 5.1 zeigt den Unterschied in der Datenverteilung zwischen DR-Tree und dem in dieser Arbeit vorgestellten DSPS. Dieses muss einer Bereichsanfrage die richtigen Produzenten zuordnen können. Die Gateways werden als initiale Datenquellen als Blätter in der verteilten R-Baum Struktur eingeordnet. Damit die Gateways auch auffindbar sind, wird noch bevor die erste Anfrage an das System gestellt werden kann eine *Initialisierung* durchgeführt. Die Gateways bilden einen verteilten R-Baum und übernehmen zusätzlich zu ihrer Platzierung als Blätter auch noch Verwaltungsaufgaben. Das bedeutet, dass ein Gateway nicht nur als Blatt, sondern gleichzeitig auch als innerer Knoten im Baum auftauchen kann. Die Abbildung 5.2 zeigt die verteilte R-Baum Struktur nach ihrer Initialisierung.

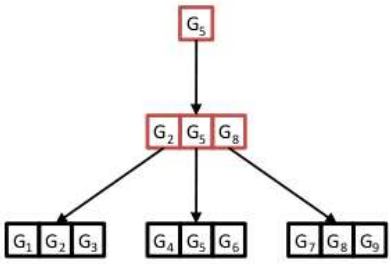

Abbildung 5.2: Die Gateways (schwarz) übernehmen Verwaltungsaufgaben und tauchen erneut als innere Knoten im R-Baum auf (rot)

Nach der Initialisierung können die Datenquellen, die Informationen für die Anfrage eines neu eingefügten Peers liefern, gefunden werden, indem der gesamte Unterbaum des neuen Peers nach unten durchsucht wird. Entweder werden die zugehörigen Gateways oder aber andere Peers, die auf dem Pfad zum Gateway liegen, gefunden.

Die nötigen Anpassungen betreffen im Einzelnen also die folgenden Verfahren:

- Einfügen neuer Anfragen
- Auffinden geeigneter Datenquellen und Verarbeiten des Datenstroms
- Initialisierung der Baumstruktur
- Aufteilen eines übergelaufenen Knotens

Die detaillierten Konzepte zu diesen Punkten werden in Kapitel 5.3 näher beschrieben. Die daraus entstandenen Algorithmen werden dann in Kapitel 5.4 behandelt. Vorher beschreibt der folgende Abschnitt die Gesamtarchitektur des Systems.

5.2 Architektur des Gesamtsystems

Um die in den vorherigen Abschnitten beschrieben Modifikationen des DR-Trees besser in ihren Kontext einordnen zu können, wird in diesem Abschnitt die Gesamtarchitektur für das in dieser Arbeit beschriebene Distributed Stream Processing System erläutert.

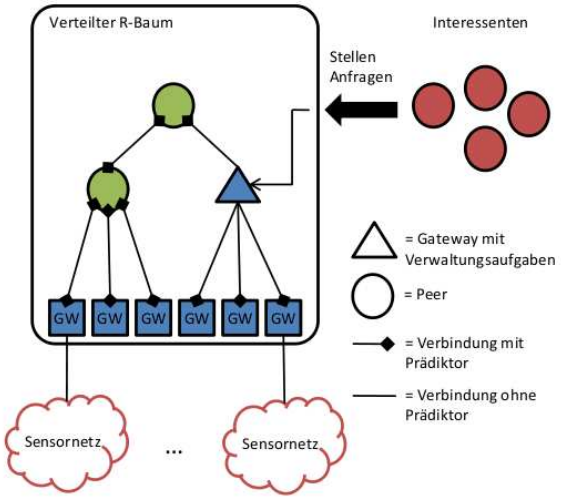

Abbildung 5.3: Die Gesamtarchitektur des konzipierten Systems

Auf Abbildung 5.3 ist das Gesamtsystem abgebildet. Die Sensornetze im unteren Teil werden durch die Gateways abstrahiert und können somit von den weiteren Beschreibungen des Systems ausgenommen werden.

Außerhalb des Stream Processing Systems ist auf der rechten Seite eine Menge an Einzelsystemen abgebildet. Diese stellen die Interessenten dar, die mit einer Anfrage dem Gesamtsystem beitreten können. Dazu kontaktieren sie einen beliebigen bereits am Gesamtsystem teilnehmenden Knoten.

Das Stream Processing System selbst ist als verteilter R-Baum organisiert. Die Gateways als Produzenten befinden sich auf der untersten Ebene: Sie stellen die Blätter des Baumes dar. Dieser R-Baum dient dazu, einem geographischen Anfragebereich die richtigen Datenquellen zuordnen zu können. Da die Konsumenten bzw. Peers auch als Datenquelle dienen sollen, werden sie ebenfalls in die Baumstruktur integriert. Die Position eines Peers bestimmt sich aus der Anfrage, die er an das System gestellt hat. Dieser Anfragebereich entspricht auch genau der Fläche, über die er Sensorinformationen an andere Peers zur Verfügung stellen muss.

Um den Baum zu komplettieren und so eine funktionierende Struktur zu gewährleisten, treten einige Gateways und Peers zusätzlich zu der ihnen zugeordneten Position nochmals als innerer Knoten im Baum in Erscheinung. Die Gateways übernehmen hier lediglich die Aufgabe, die Anfragen bei Bedarf weiter nach unten im Baum zu leiten. Peers, die Verwaltungsaufgaben übernehmen, stellen auch immer Sensorinformationen für den von ihnen verwalteten Bereich für andere Peers bereit. Ein Knoten, der lediglich Verwaltungsaufgaben übernimmt, wird ersetzt, sobald eine Anfrage eingefügt wird, deren Fläche exakt seinem verwalteten Bereich entspricht.

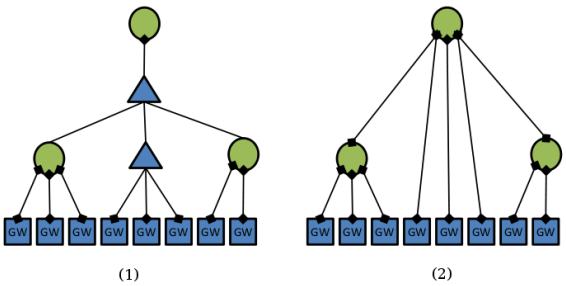

Abbildung 5.4: Der Zusammenhang von verteiltem R-Baum (1) und daraus entstehendem Datenverteilungsgraphen (2)

Der hier aufgebaute verteilte R-Baum stellt die Struktur zum Auffinden von Datenquellen dar, um den Graphen der aufgebauten Datenverbindungen zu sehen, müssen lediglich alle Gateways, die als innere Knoten im Baum auftreten, entfernt werden. Dies wird beispielhaft auf Abbildung 5.4 verdeutlicht.

Diese Tatsache spiegelt sich auch in der Anordnung der Prädiktoren wider. Prädiktoren werden auf jeder Verbindung platziert, über die Datenströme gesendet werden. Daher wird innerhalb des verteilten R-Baumes zwischen zwei Knoten genau dann eine Prädiktorverbindung aufgebaut, wenn keiner der beiden Knoten ein Gateway ist, der lediglich Verwaltungsaufgaben wahrnimmt.

Im nun folgenden Kapitel 5.3 wird das Konzept für das Distributed Stream Processing System auf Basis eines verteilten R-Baumes beschrieben. Danach erfolgt in Kapitel 5.4 eine Darstellung der Algorithmen, die das Konzept umsetzen. Das darauffolgende Kapitel 5.5 beschreibt, wie Prädiktoren in das beschriebene DSPS integriert werden können und welche Folgen sich aus der Verwendung von Prädiktoren ergeben.

5.3 Konzeption eines Distributed Stream Processing Systems

In diesem Teil der Arbeit wird in 4 Abschnitten das Konzept eines Distributed Stream Processing Systems auf Basis eines verteilten R-Baumes beschrieben.

Das wichtigste Element zum Aufbau dieses R-Baumes ist die Integration neuer Objekte: Das Verfahren zum Einfügen neuer Peers. Das dafür verwendete Verfahren wird im direkt anschließenden Abschnitt vorgestellt.

Der darauffolgende Abschnitt beschreibt, was nach dem Einfügen eines neuen Peers geschieht. Der neue Peer muss die Datenquellen finden, die Informationen zu seiner Anfrage liefern. Zu diesen Quellen baut er dann Verbindungen auf, über die die Datenströme übermittelt werden.

Danach wird die Initialisierung der Baumstruktur beschrieben. Diese ist notwendig, um die Auffindbarkeit der benötigten Datenquellen auch in einem System, in dem sich erst sehr wenige Peers befinden, gewährleisten zu können.

Der letzte Abschnitt der konzeptionellen Beschreibung des Distributed Stream Processing Systems befasst sich mit der Aufteilung eines übergelaufenen Knotens.

Wie diese Konzepte algorithmisch umgesetzt werden, wird dann in Kapitel 5.4 behandelt.

5.3.1 Einfügen neuer Anfragen

Wie und wo neue Peers in den verteilten R-Baum integriert werden, ist ausschlaggebend für die Struktur vorgestellten Systems. Die Position eines Peers im Baum bestimmt sich nach dem von ihm angefragten Bereich. Wenn daher im folgenden davon die Rede ist, eine *Anfrage* einzufügen, ist damit gemeint, dass ein *Peer* auf Basis der von ihm gestellten Anfrage in den Baum eingefügt wird.

Interessenten können dem System beitreten, indem sie ihre Anfrage an einen beliebigen am System beteiligten Knoten stellen. Nachdem ein bereits in den Baum integrierter Knoten eine Anfrage von außen erhalten hat, wird diese bearbeitet. Das Einfügen einer Anfrage gliedert sich in drei Phasen, die auch auf Abbildung 5.5 zu sehen sind:

1. Kontrolliertes Aufsteigen der Anfrage im Baum mit einer QueryMessage
2. Absinken der Anfrage mit einer InsertMessage
3. Einfügen der Anfrage in den Baum

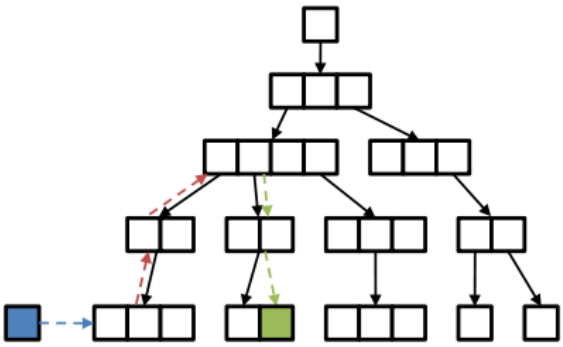

Abbildung 5.5: Der Weg einer neuen Anfrage: Kontaktieren eines am System beteiligten Knotens (blau), Aufsteigen (rot), Absinken (grün) und Einfügen

Das Auf- und Absteigen der Anfrage wird in den nächsten beiden Abschnitten behandelt. So wird der Ort im Baum gefunden, an dem der neue Peer eingefügt werden soll. Der darauffolgende Abschnitt behandelt das Einfügen einer neuen Anfrage und stellt die Konzepte vor, die wesentlich zur Struktur des Baumes beitragen.

Kontrolliertes Aufsteigen der Anfrage

Ein Interessent, der eine Anfrage stellt, sendet eine QueryMessage mit seiner Anfrage an einen beliebigen am System beteiligten Knoten.

Kann der initial kontaktierte Knoten diese Anfrage nicht selbst beantworten, so beginnt die Phase 1: Das *Aufsteigen*. Er leitet die QueryMessage mit der Anfrage weiter im Baum nach oben, bis entweder ein Knoten die Nachricht beantworten kann oder aber die Anfrage ganz oben im Baum angekommen ist. Ein Knoten kann eine Anfrage genau dann beantworten, wenn das von ihm verwaltete MBR den angefragten Bereich komplett umschließt. Das Aufsteigen ist abgeschlossen, sobald ein Knoten gemeldet hat, die Anfrage beantworten zu können, oder aber keine Möglichkeit mehr zur Weiterleitung nach oben besteht.

Die Wurzel wird beim Einfügen neuer Anfragen also je weniger belastet, desto größer die *Lokalität der Ereignisse* ist. Damit ist gemeint, dass Knoten Anfragen stellen, die sich auf Daten in ihrer Nähe beziehen, und dass sie am System beteiligte Knoten kennen, die sich in ihrer Nähe befinden. Dadurch befindet sich der geographische Bereich der Anfrage nahe am Verwaltungsbereich des zuerst mit der Anfrage kontaktierten Knotens und muss nur wenig im Baum nach oben weitergeleitet werden.

Absinken der Anfrage

Ist das Aufsteigen abgeschlossen, wird das *Absinken* der Anfrage im Baum eingeleitet. Der Knoten, bei dem das Aufsteigen der Anfrage gestoppt hat, verfügt über drei Möglichkeiten, weiter zu verfahren. Er kann den neuen Knoten

- zu seinen Geschwistern hinzufügen oder
- zu seinen Kindern hinzufügen oder
- mit einer InsertMessage an ein Kind verweisen und so den Knoten eine Ebene absinken lassen.

Welches Vorgehen gewählt wird, hängt davon ab, wie sich der Anfragebereich des neuen Peers zu dem Verwaltungsbereich des aktuell betrachteten Knotens verhält. Falls der aktuelle Knoten den Anfragebereich umschließt, wird auch für seine Kinder überprüft, ob die Anfrage komplett in einem ihrer MBRs liegt. Solange dies zutrifft, wird die Anfrage rekursiv weiter nach unten geleitet.

Sollte keines der Kinder die Anfrage in seinem MBR beinhalten, fügt der aktuelle Knoten den neuen Peer zu seinen Kindern hinzu. Kommt es an diesem Knoten zu einem Überlauf, da er nun mehr als M_{max} Kinder besitzt, wird ein Verfahren gestartet, um die Kinder auf mehrere neue Knoten zu verteilen. Dieses wird in Abschnitt 5.3.4 beschrieben.

Beim Einfügen wird außerdem eine Liste derjenigen Geschwister des neuen Peers erstellt, die sich mit seiner Anfrage überschneiden. Mit Hilfe dieser Liste werden später die Datenverbindungen aufgebaut, um die neue Anfrage mit Sensorinformationen zu versorgen. Das Auffinden von Quellen und der Aufbau von Datenverbindungen wird in Kapitel 5.3.2 beschrieben.

Wurde das Aufsteigen der Anfrage nicht deswegen gestoppt, weil ein Knoten gemeldet hat, dass er diese beantworten kann, sondern nur, weil die Anfrage bereits das oberste Level im Baum erreicht hat, tritt ein Sonderfall ein. Der aktuelle Knoten kann den neuen Peer auch zu seinen Geschwistern hinzufügen, falls sich bisher weniger als $M_{max} - 1$ Knoten in der Liste seiner Geschwister befinden. Besitzt der Knoten bereits zu viele Geschwister, muss entweder er oder eines seiner Geschwister den neuen Knoten als Kind hinzufügen. Da dazu der Verwaltungsbereich eines Knotens über den ursprünglich von ihm angefragten Bereich vergrößert werden muss, bedarf die Wahl des neuen Vaters eines gründlichen Auswahlverfahrens.

Dieses *Auswahlverfahren* arbeitet auf der Basis von zwei Metriken: Das erste ist die Überschneidung der neuen Anfrage mit den bereits bestehenden. Handelt es sich bei der neuen Anfrage um eine Teilmenge einer bereits im System befindlichen, wird die neue Anfrage in jedem Fall unterhalb dieser Anfrage eingeordnet werden. Anschließend muss die neue Anfrage ihr Ergebnis zur übergeordneten Anfrage weiterleiten. Dadurch wird verhindert, dass ein Gateway die gleiche Information an zwei verschiedene Anfragensteller senden muss. Das Entstehen von Flaschenhälsen sowie eine zu große Belastung für die Gateways und einzelne Teilnehmer wird dadurch verhindert. Der entstehende Nutzen ist dabei umso

größer, je größer die gemeinsame Fläche zwischen den beiden Knoten ist. Die zweite Metrik, die hierbei berücksichtigt wird, ist die Fläche, um die der MBR des Knotens vergrößert werden muss, wenn die neue Anfrage als sein Kind eingefügt wird.

Es wird nun der Knoten als neuer Vater ausgewählt, der das beste Verhältnis von Überschneidungsfläche mit der neuen Anfrage zu benötigter Vergrößerung des MBRs aufweisen kann. Falls sich für zwei Knoten das gleiche Verhältnis ergibt, wird zur Entscheidung die momentan verwaltete Fläche in Relation zur ursprünglich angefragten Fläche gesetzt. Derjenige Knoten, dessen Verhältnis hier näher an 1 liegt, wird ausgewählt. So soll verhindert werden, dass trotz ursprünglich ähnlich großer Anfragebereiche einige wenige Knoten wesentlich vergrößert werden, während andere nicht oder nur kaum zusätzliche Fläche verwalten müssen. Diese Fairness stellt einen zentralen Punkt für die Akzeptanz des Systems – und somit für den Willen, sich am System zu beteiligen – dar.

5.3.2 Auffinden geeigneter Datenquellen und Verarbeiten des Datenstroms

Nachdem das Aufsteigen und Absinken einer Anfrage abgeschlossen und der neue Peer dem Baum hinzugefügt wurde, werden die Beziehungen zu seinen neuen Geschwistern überprüft. Um die Baumstruktur in effizienter Weise zu erhalten, muss die Inhaltsbeziehung unabhängig von der Einfügereihenfolge gewährleistet bleiben. Das bedeutet, dass der neue Knoten alle Geschwister, die vollständig innerhalb seines MBRs liegen, finden und in die Liste seiner Kinder verschieben muss. Diese Knoten waren nur deshalb als Geschwister verfügbar, weil sie eingefügt wurden, *bevor* es auf dieser Ebene einen Knoten gab, der sie vollständig umschloss.

Nachdem ein neuer Knoten hinzugefügt wurde, wird damit begonnen, die Datenquellen für seine Anfrage zu suchen. Dazu sendet der Knoten eine RequestTransactionMessage an jedes seiner Kinder und an die Geschwisterknoten, die sich mit seiner Anfrage überschneiden. Die Knoten werden nach der Größe ihrer Überschneidungsfläche mit der neuen Anfrage sortiert, bevor sie der Reihe nach kontaktiert werden. Auf diese Weise wird die Gesamtzahl der eingehenden Verbindungen für den neuen Knoten so gering wie möglich gehalten.

Ein Peer muss Daten aus mehreren Quellen beziehen, um seine Anfrage vollständig abdecken zu können. Bei diesen Quellen handelt es sich entweder um Gateways auf der Blattebene des Baumes oder aber um Peers, die bereits Informationen von Gateways erhalten.

In der Abbildung 5.6 ist auf Bild (1) zu sehen, wie die Anfrage Q_1 einen Teil für die neue Anfrage Q_2 bereitstellt. Dieser gemeinsame Anteil ist als Q_3 gekennzeichnet. Die restlichen Bereiche $Q_{2.1}$ und $Q_{2.2}$ bezieht die neue Anfrage nun z. B. von seinen Kindern. Der Peer, der die Anfrage Q_1 verwaltet, ist von nun an dafür zuständig, die Sensordaten des gemeinsamen Teilstücks dem anderen Peer zur Verfügung zu stellen. Dadurch wird der Gateway als die ursprüngliche Quelle der Informationen entlastet. Das Datenaufkommen wird außerdem reduziert, da es sich bei den Daten für die Teilanfrage Q_3 bereits um aggregierte Sensordaten handelt. Folgte man der Architektur bisheriger Systeme, würden sowohl die Anfrage Q_1 als auch die Anfrage Q_2 jeweils Rohdaten vom Sensornetz erhalten, auch wenn sie die gleiche Operation darauf ausführen.

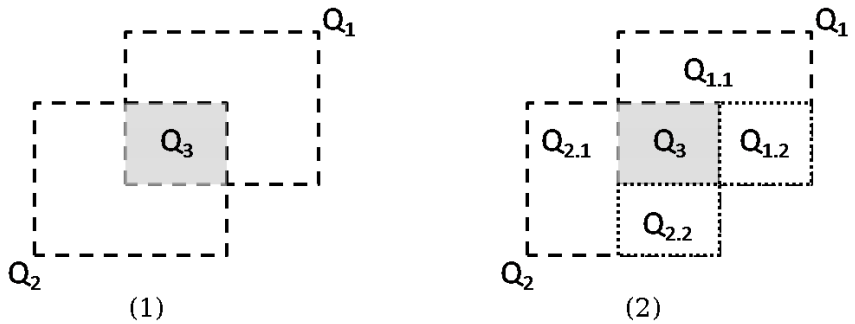

Abbildung 5.6: Unterteilung von sich überschneidenden Anfragen

Jede Anfrage wird also aus verschiedenen Teilergebnissen zusammengefügt. Die Teilanfragen müssen vor ihrer Verarbeitung durch den Aggregationsoperator entsprechend ihres Anteils an der Gesamtfläche gewichtet werden. Da z. B. die beiden Teilflächen $Q_{2.2}$ und Q_3 nur halb so groß sind wie die Fläche $Q_{2.1}$, kann der Durchschnitt über den gesamten Anfragebereich nicht einfach durch ein Aufsummieren der Durchschnittswerte der Einzelflächen und anschließende Division durch drei gebildet werden. Stattdessen müssen die Sensorwerte der Teilfläche $Q_{2.1}$ mit $\frac{1}{2}$ und die Sensorwerte der Teilflächen $Q_{2.2}$ und Q_3 mit jeweils $\frac{1}{4}$ gewichtet werden. Da für jede Verbindung der Anteil der vom Produzenten der Information abgedeckten Fläche an der gesamten Anfragefläche des Konsumenten zur Gewichtung dieser Sensorinformationen berücksichtigt wird, beträgt die Summe der Gewichte über alle eingehenden Verbindungen eines Knotens 1.

Wichtig bei der Unterteilung der Anfragen ist, dass dies vollkommen transparent für die anfragende Anwendung und die übergeordneten Knoten im Baum gehalten wird. Die Interna der Anfragenbehandlung werden durch die verwalteten MBRs der einzelnen Knoten im Baum gekapselt.

5.3.3 Initialisierung der Baumstruktur

Bevor das System bereit ist, Anfragen zu verarbeiten, muss eine Initialisierung erfolgen. Die Gateways der Sensornetze bauen einen verteilten R-Baum auf, um es zu ermöglichen, dass Anfragen an jeden beliebigen Gateway gestellt werden können. Der dabei entstehende R-Baum ist insofern „klassisch“, als die Daten, also die Gateways, die Sensorinformationen bereitstellen, sich auf der Blattebene des Baumes befinden. Zur Konstruktion des Baumes werden die in den vorhergehenden Kapiteln beschriebenen Einfügemechanismen benutzt. Das bedeutet, dass die Gateways auch Verwaltungsaufgaben für die inneren Knoten des Baumes übernehmen müssen.

Dadurch, dass Überschneidungen zwischen Anfragen ausgenutzt und redundante Datenströme werden, sinkt die Belastung für die Gateways der Sensornetze. Da in den bisher

verbreiteten Stream Processing Systems die Gateways selbst dafür verantwortlich sind, ihre Sensorinformationen an alle Interessenten zu verteilen, während sie in dem hier vorgestellten System durch die Peers dabei unterstützt werden, kann man von einer erheblichen Entlastung der Gateways sprechen. Also kann ihnen auch ein geringer zusätzlicher Aufwand für die Verwaltung der inneren Knoten des Baumes zugemutet werden.

Anders als Peers, die eine Anfrage stellen, aggregieren die Gateways jedoch nicht die Sensorinformationen ihrer Kinder. Falls die Anfrage für eine Datenverbindung bei einem durch einen Gateway repräsentierten inneren Knoten des Baumes ankommt, teilt dieser wie in Kapitel 5.3.2 beschrieben die Anfrage auf und sendet sie an seine Kinder weiter.

Tritt ein neuer Peer dem System bei, dessen Anfragebereich genau dem verwalteten Bereich eines Gateways entspricht, kann dieser Gateway seine Verwaltungsaufgaben auf den neuen Peer übertragen und den Baum verlassen. So werden mit zunehmender Laufzeit die Gateways nach und nach von ihren zusätzlichen Verwaltungsaufgaben befreit und treten nur noch als Blätter und Quellen für Sensorinformationen in Erscheinung.

5.3.4 Aufteilen eines übergelaufenen Knotens

Nachdem ein neuer Peer in den Baum eingefügt wurde, kann es sein, dass sein Vaterknoten nun mehr als M_{max} Kinder enthält und somit überläuft. Wie in den Grundlagen zu R-Bäumen in Kapitel 2.2.1 beschrieben, wird dann ein Algorithmus zur Aufteilung des übergelaufenen Knotens gestartet.

Während die Algorithmen der R-Baum-Varianten und die bekannten verteilten R-Bäume alle Verfahren nutzen, die den übergelaufenen Knoten in zwei neue Verwaltungsbereiche unterteilen, wird in dieser Arbeit ein anderer Ansatz verfolgt. Zur Behandlung eines Überlaufs wird hier ein Clustering-Verfahren eingesetzt, dass es erlaubt, die Kinder des übergelaufenen Knotens auf mehr als zwei neue Verwaltungsbereiche aufzuteilen.

Für die Auflösung dieses zwei-Knoten-Paradigmas sprechen mehrere Gründe. Auf Abbildung 5.7 ist zu sehen, wie der übergelaufene Knoten aus Teilbild (1) zunächst horizontal (2) und vertikal (3) in zwei neue Knoten unterteilt werden könnte. Dabei wird deutlich, dass beide Unterteilungen ihre Schwächen haben, da die ungenutzte Fläche innerhalb der einzelnen Gruppen beide Male sehr groß ist. Der deutlich geringere Anteil an ungenutzter Fläche innerhalb der Gruppen auf Teilbild (4) zeigt klar, dass der Nutzen eines Verfahrens, das mehr als zwei neue Knoten zulässt, sehr groß sein kann.

Eine möglichst effiziente Ausnutzung der Flächen innerhalb der MBRs hat zwei für das Gesamtsystem wichtige Folgen:

1. Eine stärkere Clusterung der Knoten führt zu einer geringeren Gesamtfläche.
2. Der Anteil von Bereichen, die nicht von einem Kindknoten angefragt werden und trotzdem weiter nach oben im Baum vermittelt werden müssen, sinkt.

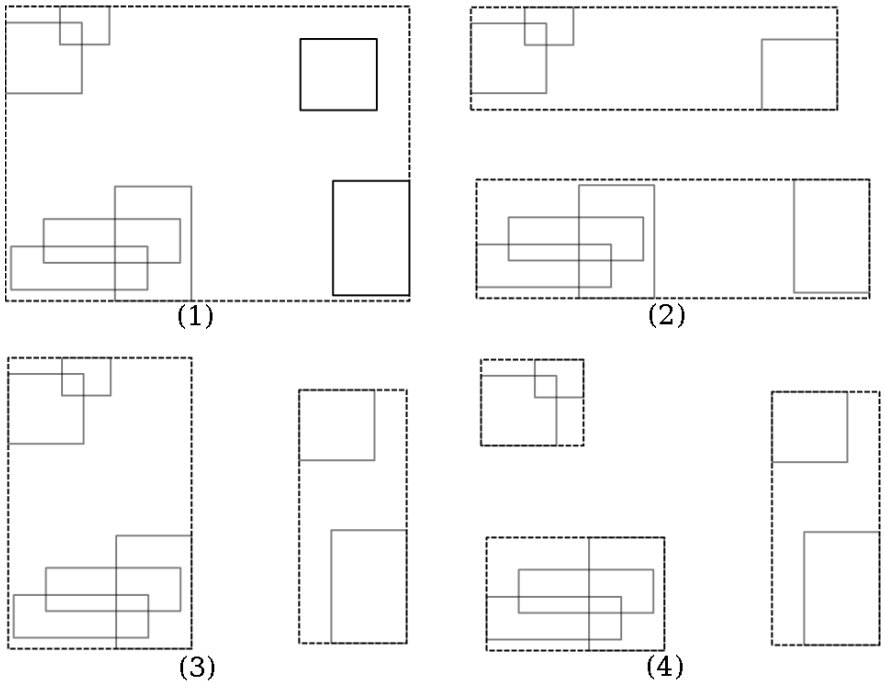

Abbildung 5.7: Aufteilung eines überlaufenden Knotens (1): Horizontal in zwei neue Knoten (2), vertikal in zwei neue Knoten (3), in drei neue Knoten (4)

Punkt 1 ist wichtig, da der von einem Knoten verwaltete Bereich die Fläche darstellt, über die er Sensorinformationen an im Baum über ihm liegende Peers bei Bedarf zur Verfügung stellen muss. Aus Gründen der Fairness sollte das Gesamtsystem die Vergrößerung der Fläche über den Bereich, den ein Knoten ursprünglich angefragt hat, hinaus möglichst gering halten.

Punkt 2 ist ebenfalls eine wichtige Anforderung, da ein Knoten mit jedem Flächenstück, das nicht durch eines seiner Kinder verwaltet wird, mehr Direktverbindungen zu Gateways aufbauen muss. Jeder Bereich, der nicht von Kindern abgedeckt und daher direkt vom Gateway des jeweiligen Sensornetzes abgefragt wird, sorgt für eine zusätzliche Belastung des Systems, da hier nicht aggregierte Daten wie von den Kindknoten, sondern Rohdatenströme übertragen werden müssen.

5.4 Der Algorithmus im Detail

Wie bereits in Kapitel 5.2 zur Architektur erwähnt, dient dem hier vorgestellten Algorithmus der DR-Trees [BFG07] als Grundlage. Die Verteiltheit der Struktur wird dadurch erreicht, dass die lokalen Informationen der einzelnen Knoten beschränkt sind; es existiert an keiner Stelle eine vollständige Sicht auf den Baum. Während im originalen R-Baum die inneren

Knoten des Baumes lediglich Verwaltungseinheiten sind, wird im DR-Tree jedes MBR durch einen Peer oder einen Gateway – in jedem Fall also einen physischen Knoten – repräsentiert. Dadurch sind die Informationen des Baumes auf verschiedene Systeme verteilt. Jeder Knoten speichert Informationen über seinen Vater, seine Kinder – diese werden nach oben im Baum durch das von ihm verwaltete MBR repräsentiert – und seine Geschwister.

Die im Kapitel 5 bereits angesprochenen und in den vorhergehenden Abschnitten 5.2 bis 5.3.3 vorgestellten Konzepte werden nun konkretisiert und algorithmisch dargestellt. Die vorgestellten Algorithmen umfassen

- das Einfügen neuer Anfragen,
- das Auffinden der geeigneten Datenquellen,
- das Verarbeiten von Sensorinformationen und
- das Aufteilen eines übergelaufenen Knotens.

Bei der Beschreibung der einzelnen Algorithmen wird das folgende Schema benutzt: Zuerst erfolgt eine kurze Zusammenfassung der im vorigen Abschnitt gegebenen Problembeschreibung. Dann erfolgt eine algorithmische Darstellung der Lösung, die auch textlich erläutert wird. Anschließend erfolgt eine kurze Analyse bzw. Bewertung des Algorithmus.

Nach der Beschreibung der einzelnen Algorithmen werden die Eigenschaften, die sich direkt aus den Konzepten und deren algorithmischer Umsetzung für das System ergeben, in Abschnitt 5.4.5 zusammengefasst und mit dem DR-Tree verglichen.

5.4.1 Einfügen neuer Anfragen

Eine Anfrage kann an jeden Knoten des Baumes gestellt werden. Der Vorgang zum Einfügen besteht aus vier Phasen: Zuerst steigt die Anfrage den Baum hinauf, um danach über die richtigen Äste abzusinken. Als drittes wird entschieden, auf welchem Level die Anfrage eingefügt wird. Zuletzt werden die geeigneten Datenquellen gesucht und Verbindungen zu ihnen aufgebaut.

Die Anfrage wird anfangs so lange nach oben im Baum weitergereicht, bis ein Knoten sie mit seinem MBR vollkommen umschließt. Dann wird das Absinken eingeleitet. Auf Grund der Tatsache, dass jeder Knoten des Baumes gleichberechtigt Daten speichern kann, muss entschieden werden, auf welchem Level eine neue Anfrage in den Baum eingefügt wird. Neben der vom R-Baum bekannten Frage, welchen Ast ein neu eingefügtes Objekt hinabsteigt, muss also auch entschieden werden, ob ein weiteres Absteigen erforderlich ist oder das Objekt auf diesem Level bereits eingefügt werden soll.

Das Aufsteigen ist sehr einfach realisierbar; der Algorithmus 5.1 zeigt das Verhalten eines Knotens, der eine Benachrichtigung mit einer neuen Anfrage erhält. Umschließt sein MBR den angefragten Bereich, so beginnt der Knoten bei sich selbst damit, die Anfrage absinken zu lassen. Ist dies nicht der Fall, wird die Anfrage solange nach oben weitergeleitet, bis ein

Algorithmus 5.1 Behandlung einer QueryMessage, um eine neue Anfrage einzufügen

```
procedure processQueryMessage(newQuery)
    if (newQuery ⊆ this.MBR) ∨ (this.parent = null) then
        sendInsertMessage(this, newQuery)
    else
        sendQueryMessage(this.parent, newQuery)
    end if
end procedure
```

Knoten den angefragten Bereich beinhaltet oder das oberste Level des Baumes nicht erreicht ist.

Algorithmus 5.2 Behandlung einer InsertMessage, um eine neue Anfrage einzufügen

```
procedure processInsertMessage(insertMsg)
    if (insertMsg.query ⊆ this.MBR) then
        nodeCheckResult ← checkNodesForInsertion(this.children,insertMsg)
        insert(nodeCheckResult)
    else
        nodeCheckResult ← checkNodesForInsertion(this.siblings,insertMsg)
        insertAsSibling(nodeCheckResult)
    end if
end procedure
```

Algorithmus 5.3 Auswahl des besten Kindknotens zum Absinken einer neuen Anfrage

```
function chooseInsertNode
    for all child ∈ this.children do
        rating ← rate(child)
        if (rating > maxRating) then
            maxRating ← rating
            bestChild ← child
        end if
    end for
    return bestChild
end function
```

Der Algorithmus 5.2 zeigt, wie ein Knoten reagiert, an den die Nachricht zum Absinken einer Anfrage gesendet wird. Wenn sein MBR die Anfrage nicht mehr vollständig umschließt, wird dies auch für keines seiner Kinder der Fall sein. Um die Gesamtvergrößerung der MBRs so gering wie möglich zu halten, muss die Anfrage direkt an diesem Knoten eingefügt werden. Der Knoten wird dann als neuer Geschwisterknoten hinzugefügt, falls die Platzbeschränkungen dies zulassen. Falls nicht, wird der am besten geeignete Kindknoten als neuer Vater ausgewählt und in seinem MBR entsprechend vergrößert. Die Auswahl erfolgt mittels Algorithmus 5.3. Dieser wählt das Kind aus, das das beste *Rating* erzielt. Es

sind verschiedene Verfahren denkbar, um eine Bewertung durchzuführen; hier wird eine Kombination aus der Überschneidungsfläche des jeweiligen Kindes mit der neuen Anfrage und der benötigten Vergrößerung des MBRs des Kindes gebildet. Das Rating R_i des i-ten Kindes C_i eines Knotens bezüglich einer neuen Anfrage Q bestimmt sich als

$$R_i = \frac{\text{area}(C_i \cap Q)}{\text{mbr}(C_i \cup Q) - \text{area}(C_i)}$$

Sollten zwei Knoten das gleiche Rating erzielen, wird als dritter Indikator das Verhältnis zwischen angefragter Fläche und momentan verwalteter Fläche gebildet. Der Knoten, der bisher am wenigsten zusätzliche Fläche verwaltet, wird ausgewählt.

Algorithmus 5.4 Überprüft eine Liste von Knoten darauf, ob und wie sie beim Einfügen einer neuen Anfrage hilfreich sein können

```
function checkNodesForInsertion(nodeList,insertMsg)
    insert ← true
    for all (node ∈ nodeList) do
        if (insertMsg.queryArea = node.MBR) then
            if (node dient nur zur Verwaltung) then
                removeMe ← node
            else
                Verbindung aufbauen zwischen insertMsg.sender und node
            end if
        else if (insertMsg.queryArea ⊂ node.MBR) then
            sendInsertMessage(node,insertMsg)
            insert? ← false
        else if (node.MBR ∩ insertMsg.queryArea ≠ ∅) then
            node zur Peer- bzw. Gateway-Ergebnisliste hinzufügen
        end if
    end for
    checkNodeResult zusammenstellen aus Ergebnislisten, insert? und removeMe
    return checkNodeResult
end function
```

Der Algorithmus 5.4 überprüft für die Menge der Geschwister oder Kinder eines Knotens, ob die neue Anfrage von einem der Knoten umschlossen wird. Trifft dies zu, wird die Nachricht zum Einfügen an diesen Knoten weitergeleitet. Findet sich kein Knoten, an den die Anfrage weitergeleitet werden kann, wird versucht, den Knoten auf diesem Level einzufügen.

Außerdem werden alle Knoten aus der gegebenen Menge daraufhin überprüft, ob sie der neuen Anfrage Teilergebnisse zur Verfügung stellen können. Von Algorithmus 5.5 werden anschließend Knoten, die komplett innerhalb der neuen Anfrage liegen, in die Kindmenge des Anfragenstellers verschoben. An Knoten, die zu den Kindern hinzugefügt wurden oder die sich mit der neuen Anfrage überschneiden, wird eine RequestTransactionMessage gesendet, um eine Datenverbindung aufbauen zu können.

Algorithmus 5.5 Anfragen von Datenverbindungen auf Basis einer Ergebnisliste

```
procedure processResultList(resultList,coveredAreas,newPeer)
    for all (peer ∈ resultList) do
        if (peer.Mbr ∩ coveredAreas = ∅) then
            sendRequestTransactionMessage(peer)
            if (peer.Mbr ⊂ newPeer.query) then
                Verschiebe peer in die Kinder von newPeer
            end if
        end if
    end for
end procedure
```

Analyse des Algorithmus Durch das kontrollierte Aufsteigen wird verhindert, dass die Knoten an der Wurzel des Baumes nur auf Grund ihrer Position über die Maßen belastet werden. Sie werden nur kontaktiert, wenn es innerhalb des Pfades, auf dem die Anfrage nach oben gesendet wurde, keinen Knoten gab, der die Anfrage beantworten konnte. Durch die Verwendung der Rating-Funktion und das Hinzuziehen der bisher zusätzlich verwalteten Fläche als Indikator wird vermieden, dass sich Knoten, die bereits schon vergrößert wurden, noch weiter vergrößert werden, während andere Knoten nur den von ihnen angefragten Bereich verwalten. Die Reihenfolge, in der ein Knoten seine Kinder hinzufügt, hat keinen Einfluss auf die entstehende Struktur. Dies wird dadurch erreicht, dass die Geschwisterknoten, die innerhalb des neuen Knotens liegen, nachträglich in diesen verschoben werden. Diese Eigenschaft des Einfügeverfahrens adressiert einen der größten Nachteile von R-Baum Strukturen: Die Abhängigkeit der entstehenden Struktur von der Einfügereihenfolge der Datenobjekte.

5.4.2 Auffinden geeigneter Datenquellen

Nachdem ein Peer dem Baum als neuer Knoten hinzugefügt wurde, versucht er, Verbindungen zu Datenquellen aufzubauen, die Informationen zu seiner Anfrage liefern. Im vorherigen Abschnitt wurde beschrieben, dass von Algorithmus 5.5 an jeden Knoten, den der neue Peer zu seinen Kindern hinzufügt und jeden Geschwisterknoten, der sich mit seinem Anfragebereich überschneidet, eine RequestTransactionMessage zur Verbindungsanfrage gesendet wird.

Erhält ein Knoten eine RequestTransactionMessage, muss er entscheiden, ob er derjenige ist, der die angefragten Daten zur Verfügung stellen kann, oder ob er die Anfrage an seine Kinder vermitteln und eventuell in kleinere Teile aufspalten muss.

Der Algorithmus 5.6 zeigt die möglichen Reaktionen auf eine RequestTransactionMessage. Wenn sich der angeforderte Bereich exakt mit dem Verwaltungsbereich des Knotens deckt und es sich bei dem angefragten Knoten nicht um einen Gateway handelt, der während der in Kapitel 5.3.3 beschriebenen Initialisierung nur eine verwaltende Rolle als innerer Knoten des Baumes übernommen hat, wird die Datenverbindung mit einer GrantTransactionMessage

Algorithmus 5.6 Verarbeiten einer RequestTransactionMessage, um eine Datenverbindung aufzubauen

```
procedure processRequestTransactionMessage(reqMessage)
    if ((this.Mbr = reqMessage.requestedArea) ∧
(Dieser Knoten dient nicht nur Verwaltungszwecken)) then
        sendGrantTransactionMessage(reqMessage.sender)
    else
        for all (child ∈ this.children) do
            if (reqMessage.requestedArea ⊂ child.Mbr) then
                Nachricht an child weiterleiten
                break
            else if (child.Mbr = reqMessage.requestedArea) then
                if (∃ eingehende Datenverbindung zu child) then
                    sendGrantTransactionMessage(reqMessage.sender)
                else
                    Nachricht an child weiterleiten
                end if
                break
            else if (child ∩ reqMessage.requestedArea ≠ ∅) then
                sendRequestTransactionMessage(child,child ∩ reqMessage.requestedArea)
            end if
        end for
    end if
end procedure
```

genehmigt. Handelt es sich bei dem Empfänger der Verbindungsanfrage um einen Gateway, so soll dieser nicht mit dem Transfer von Sensordaten belastet werden. Dieser Knoten untersucht also seine Kinder daraufhin, ob und wie sie einen Teil zu der angefragten Fläche beitragen können.

Ein Knoten fragt auch Teilflächen von anderen Knoten an, die sich nur teilweise mit seiner Anfragefläche überschneiden. Dadurch erhält der angefragte Knoten eine RequestTransactionMessage, die kleiner als der komplette von ihm verwaltete Bereich ist. Der Empfänger einer solchen Nachricht muss also überprüfen, ob er die Verbindungsanfrage nicht nach unten weiterleiten kann. Liegt der Anfragebereich vollständig innerhalb des MBRs eines seiner Kinder, wird die Anfrage an dieses Kind weitergeleitet. Die Beantwortung der Anfrage wird dort rekursiv fortgesetzt.

Entspricht der angefragte Bereich exakt dem Verwaltungsbereich eines Kindes, ergeben sich zwei weitere Möglichkeiten: Es kann sein, dass der aktuell angefragte Knoten bereits alle Daten seiner Kinder erhält und verarbeitet. In diesem Fall kann er die Daten, die er von seinem Kind erhält, bei jeder erhaltenen Aktualisierung auch an weitere Interessenten nach oben im Baum weiterleiten. Sein Kind wird somit entlastet, da es nur an einen Empfänger Daten senden muss. Die zweite Möglichkeit ist, dass der Knoten bisher nicht die Sensordaten seiner Kinder erhält. In diesem Fall wird die Anfrage an das betreffende Kind weitergeleitet.

Überschneidet sich der angefragte Bereich mit dem Verwaltungsbereich eines Kindes, wird die Anfrage aufgeteilt. An jedes Kind, das Informationen für die Anfrage bereitstellen kann, wird eine neue RequestTransactionMessage mit dem jeweiligen Überschneidungsbereich gesendet.

Der Anfragevorgang terminiert, wenn alle Anfragen beantwortet wurden. Somit erhält der gerade dem System beigetretene Knoten eine Reihe von GrantTransactionMessages, auf deren Basis er nun die nötigen Datenverbindungen aufbauen kann. Da es möglich ist, dass sich Knoten überlagern, können auch mehrere Nachrichten eingetroffen sein, die Bestätigungen für den gleichen geographischen Bereich beinhalten. Trifft die Bestätigung für einen Bereich ein, der bereits vollständig durch eingehende Verbindungen abgedeckt ist, wird die Bestätigungsnachricht ignoriert. Ist nur ein Teil der bestätigten Fläche bereits abgedeckt, wird der nun noch benötigte Bereich berechnet und eine eine neue Anfrage formuliert, die an den entsprechenden Knoten gesendet werden kann.

Erhält ein Knoten die Bestätigung zum Datenaufbau für einen Bereich, für den er bisher noch keine Sensorinformationen erhält, erstellt er die für den Datenaustausch zuständigen Komponenten, sogenannte DataManager. Jeder dieser Manager repräsentiert eine eingehende bzw. ausgehende Verbindung. Der Manager für die eingehende Verbindung erhält eine Gewichtung, die, wie in Kapitel 5.3.2 beschrieben, relativ zum Anteil dieser Fläche an der gesamten Anfragefläche erfolgt. Der Manager für die ausgehende Verbindung wird an den Knoten gesendet, der die Bestätigungsnachricht für die Datenverbindung gesendet hatte.

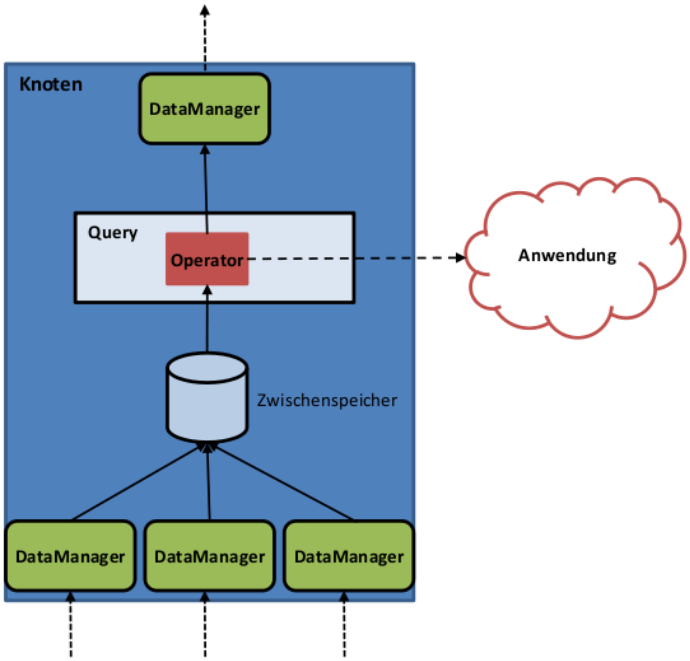

Abbildung 5.8: Architektur eines Knotens

Analyse des Algorithmus Das Prinzip des Algorithmus zum Auffinden der Datenquellen ist relativ einfach: Der neu eingefügte Peer lässt seine Anfrage so lange in Teile aufspalten und nach unten weiterleiten, bis ein anderer Peer oder ein als Blatt auftretender Gateway gefunden wurde, der die gewünschte Information bereitstellt. Anschließend werden die zum Datenverkehr nötigen Komponenten erstellt und übermittelt. Auf Grund der Inhaltsbeziehungen der einzelnen Knoten innerhalb der R-Baum Struktur werden die benötigten Datenquellen immer gefunden. Der MBR eines Knotens entspricht dem kleinsten Rechteck, das alle seine Kinder umschließen kann. Der verwaltete Bereich eines Peers entspricht auch immer dem Bereich, über den er Sensorinformationen anbieten kann. Also wird eine Anfrage, die während ihres Weges nach unten im Baum immer weiter auf die verfügbaren Bereiche der Kindknoten aufgeteilt wird, spätestens dann ihr Ziel erreicht haben, wenn alle Teilanfragen bei den Gateways auf dem untersten Level des Baumes angelangt sind. Da die Gateways die verfügbare Welt vollständig abdecken, wird eine Anfrage in jedem Fall vollständig beantwortet werden.

5.4.3 Verarbeiten des Datenstroms

Wurde ein neuer Peer in den Baum eingefügt und hat die Verbindungen zu den für seine Anfrage benötigten Datenquellen aufgebaut, erhält er in regelmäßigen Abständen aktualisierte Sensormesswerte. Diese Wartezeit zwischen zwei Sensorwerten hat der Peer, wie in Kapitel 3.2 beschrieben, mit dem Parameter τ in seiner Anfrage selbst bestimmt.

Die Datenquellen, zu denen der Peer Verbindungen aufgebaut hat, senden über ihre DataManager eine SensorDataUpdateMessage an die ihnen auf Seiten des neuen Peers zugeordneten Pendants. Dort werden die Sensordaten zwischengespeichert. Jeder Knoten, der eine Anfrage an das System gestellt hat und in regelmäßigen Abständen Sensorinformationen erhält, wendet in der gleichen Regelmäßigkeit seinen Operator auf die Sensordaten an. Die in diesem System verwendete Aggregation ist ein Fensteroperator, was bedeutet, dass auf einer von der Anwendung zu bestimmenden Menge von n aufeinander folgenden Sensorwerten eine Operation ausgeführt wird. Dies kann z. B. das Bilden eines Durchschnittswertes oder das Bestimmen eines Minimums bzw. Maximums sein.

Die interne Verarbeitung der Sensordaten aus den eingehenden Verbindungen erfolgt mittels eines Zwischenspeichers, der einen Array pro eingehende Verbindung verwaltet. Die Größe der einzelnen Arrays entspricht der Breite des Fensteroperators. Sobald der Zwischenspeicher genug Einträge enthält, um den Operator anwenden zu können, nimmt der Knoten pro Verarbeitungszyklus die ersten n Einträge aus dem Speicher und übergibt sie dem Operator seiner Anfrage. Der Rückgabewert des Operators wird der Anwendung und der ausgehenden Verbindung übermittelt.

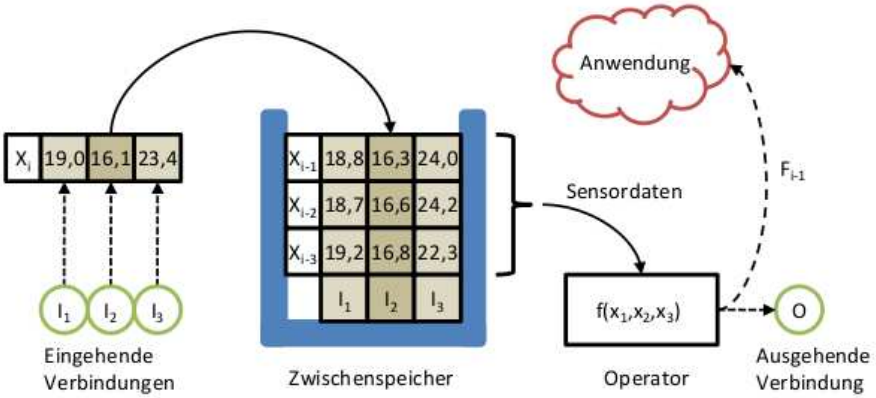

Abbildung 5.9: Anwenden eines Operators

Die Abbildung 5.9 zeigt das Hinzufügen neuer Sensorwerte aus den drei eingehenden Datenverbindungen I_1 bis I_3 zum Zeitpunkt i. Im vorherigen Schritt $i-1$ war aus dem gefüllten Zwischenspeicher der Wert F_{i-1} bestimmt und an die Anwendung und die ausgehende Verbindung des Knotens übermittelt worden. Um Platz für die neuen Messwerte zu schaffen, wird der älteste Datensatz x_{i-3} aus dem Zwischenspeicher entfernt.

Analyse des Algorithmus Dadurch, dass alte Messwerte aus dem Zwischenspeicher entfernt werden, wird immer nur die tatsächlich benötigte Menge an Daten innerhalb eines Knotens gespeichert. Durch die Verwendung des Aktualisierungsparameters τ wird eine effiziente Arbeitsweise garantiert, was bedeutet, dass nur ein neuer Datensatz erzeugt wird, wenn die anfordernde Anwendung einen neuen Wert benötigt.

5.4.4 Aufteilen eines übergelaufenen Knotens

Nach dem Hinzufügen eines neuen Knotens kann der Fall eintreten, dass dessen neuer Vaterknoten überläuft, also mehr als M_{max} Kinder enthält. In diesem Fall werden seine Kinder auf neue Knoten aufgeteilt.

Der hier vorgestellte Algorithmus zur Anfragenverwaltung setzt nicht die im R- und R*-Baum verwendeten Algorithmen ein, sondern verwendet stattdessen das in [BPT02] präsentierte Clusteringverfahren. Dieses lässt es zu, dass mehr als zwei neue Knoten entstehen.

In diesem Verfahren wird ein k-means-Clustering-Algorithmus [HW79] eingesetzt, um $1 < k < M_{max}$ Cluster aus den neu zu verteilenden Kindern zu bilden. Die Idee, Clusteringalgorithmen zu verwenden, geht auf die Überlegung zurück, dass die Wahrscheinlichkeit einer gleichzeitigen Suche in den neu entstandenen Gruppen proportional zu deren Ähnlichkeit zueinander ist. Das Ziel sollte also sein, Gruppen zu erzeugen, die eine große Homogenität aufweisen und sich untereinander möglichst unähnlich sind. Clusteringalgorithmen

maximieren die Ähnlichkeit der Elemente innerhalb der einzelnen Cluster und die Unähnlichkeit zu den Elementen aller anderen Cluster.

Zur Beurteilung der Güte einer berechneten Clusterung $C_k = \{c_i | 1 \leq i \leq k\}$ wird die durchschnittliche Silhouette der entstandenen Cluster, $\bar{s}(C)$, benutzt. Die Silhouette eines Clusters c_i ist wiederum definiert als die durchschnittliche Silhouette seiner Mitglieder. Die Silhouette eines einzelnen Objektes i ist definiert als

$$s(i) = \frac{b(i) - a(i)}{\max\{b(i), a(i)\}}$$

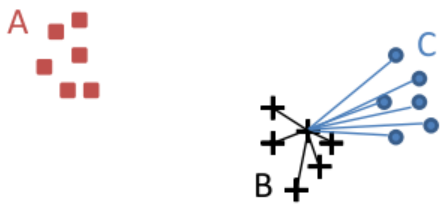

Abbildung 5.10: Bestimmung der Silhouette durch die Abstände zu den Objekten des eigenen und des nächstgelegenen Clusters

Dabei bezeichnet $a(i)$ die durchschnittliche Entfernung (also Unähnlichkeit) des Elements i zu allen anderen Elementen seines Clusters. Mit $b(i)$ wird die durchschnittliche Entfernung (Unähnlichkeit) von i zu allen Elementen des nächstgelegenen Clusters bezeichnet. Die Abbildung 5.10 zeigt die Bestimmung der durchschnittlichen Entfernung innerhalb des eigenen Clusters B und zu allen Elementen des nächstgelegenen Clusters C. Die Silhouette eines Objektes berechnet sich, indem die Nähe zum eigenen Cluster in Relation zur Entfernung zum nächstgelegenen Cluster bestimmt wird. So wird für jedes Element versucht, gleichzeitig die Nähe zum eigenen Cluster und die Entfernung zu den anderen Clustern zu maximieren. Wie aus der Definition 5.4.4 ersichtlich wird, liegt der Wert einer Silhouette im Intervall $[-1, 1]$. Eine Interpretation der Werte kann der Tabelle 5.1 entnommen werden.

Die vom Algorithmus 5.7 als optimal ausgewählte Clusterung OC ist insofern die beste aller berechneten Clusterungen, als für sie gilt:

$$OC = \{clustering_{opt} | \bar{s}(clustering_{opt}) = \max_{2 \leq k \leq M} \bar{s}(C_k)\}$$

$$\text{mit } C_k = \{c_i | 1 \leq i \leq k\}$$

Es wird also in einer Laufzeit von aus allen berechneten Clusterungen diejenige ausgewählt, die die größte durchschnittliche Silhouette aufweist. Es handelt sich hierbei jedoch nicht um die beste aller möglichen Clusterungen, da für jede mögliche Anzahl von Clustern nur jeweils eine Unterteilung vorgenommen wird. Die Berechnung der optimalen Lösung würde zu viel Zeit in Anspruch nehmen, daher wird diese Heuristik verwendet. Es wäre jedoch z. B.

Ermittelter Silhouettenwert	Interpretation
$s(i) = 1$	Die „innere“ Ähnlichkeit ist weitaus besser als die „äußere“ Ähnlichkeit. Dieses Objekt wurde richtigerweise dem Cluster A zugeordnet.
$s(i) = 0$	Die Ähnlichkeiten $a(i)$ und $b(i)$ sind ungefähr gleich. Es ist unklar, welchem Cluster das Objekt zugeordnet werden soll, da es sich direkt dazwischen befindet.
$s(i) = -1$	Die „äußere“ Ähnlichkeit ist größer als die „innere“ Ähnlichkeit. Dieses Objekt ist dem falschen Cluster zugeordnet und sollte stattdessen zu Cluster B gehören.

Tabelle 5.1: Interpretationen der Silhouette eines Objektes

Algorithmus 5.7 Aufteilung eines übergelaufenen Knotens

```
procedure SPLIT(k_max)
    optimalClustering ← KMEANSCLUSTERING(2)
    maxSilhouette ← s̄(optimalClustering)
    for k = 3 to k_max do
        clustering ← KMEANSCLUSTERING(k)
        silhouette ← s̄(clustering)
        if silhouette > maxSilhouette then
            maxSilhouette ← silhouette
            optimalClustering ← clustering
        end if
    end for
    return optimalClustering
end procedure
```

möglich, die Zahl der berücksichtigten Clusterungen zu erhöhen, indem für jedes k nicht nur eine Clusterung, sondern verschiedene mit jeweils unterschiedlichen Seeds bestimmt werden.

Analyse des Algorithmus Die Analysen aus [BPT02] zeigen, dass ein mit diesem Aufteilungsalgorithmus ausgestatteter R-Baum sechs mal schneller neue Elemente einfügen kann als ein R*-Baum. Der verwendete Algorithmus liegt damit auf dem gleichen Geschwindigkeitsniveau wie der des R-Baumes, ohne dieselben Nachteile, wie die Tendenz zu starken Überlappungen zwischen den neu entstandenen Knoten, aufzuweisen. Der Geschwindigkeitsunterschied wird vor allem dadurch erreicht, dass auf ein Wiedereinfügen der Kinder eines übergelaufenen Knotens verzichtet wird. Ein ebenfalls vorgenommener Vergleich der

Algorithmus 5.8 Mit K-Means-Clustering werden n Elemente in einer Laufzeit von $\mathcal{O}(k \times n)$ in k Cluster unterteilt

```
procedure kMeansClustering(k)
    clustering ← chooseSeeds(k)
    repeat
        Berechne die Schwerpunkte aller Cluster neu
        for all child ∈ this.children do
            Verschiebe child in den Cluster mit der minimalen Entfernung
        end for
    until kein Element in einen anderen Cluster verschoben
    return clustering
end procedure
```

Antwortzeiten auf Anfragen zeigt, dass durch diesen Verzicht keinesfalls die Struktur des Baumes geschädigt wird. Der R*-Baum und der R-Baum mit Clusteringalgorithmus liegen für verschiedene Datensätze diesbezüglich gleichauf. Der Algorithmus zum Wiedereinfügen von Knoten eignet sich ohnehin kaum zum Einsatz in verteilten Systemen. Wie in den Geschwindigkeitsvergleichen aus [BPT02] ersichtlich wird, ist der Aufwand für dieses Verfahren schon in einem zentralistischen System sehr hoch. In einem verteilten System fällt noch zusätzlicher Kommunikationsaufwand an, was die Zeitspanne, die zum Einfügen neuer Knoten benötigt wird, nochmals erhöht. Außerdem werden die obersten Knoten des Baumes besonders stark belastet, da die Knoten immer von der Wurzel aus neu eingefügt werden. Das Clustering dagegen ist mit lokalem Wissen durchführbar, was für seine Verwendung in einem verteilten System spricht.

5.4.5 Inhärente Eigenschaften des Verfahrens

Bereits nach der Definition der einzelnen Modifikationen, durch die das beschriebene Distributed Stream Processing System aus dem DR-Tree hervorgeht, können Aussagen über einige inhärente Eigenschaften des Verfahrens getroffen werden. Die Tabelle 5.2 zeigt, in welchen Bereichen sich das in dieser Arbeit konzipierte System gegenüber dem DR-Tree entscheidend verbessern konnte.

5.5 Prädiktoren in einem Distributed Stream Processing System

Das in Kapitel 5.3 vorgestellte Verfahren zur Datenstromverarbeitung erfüllt bereits einen der beiden eingangs gesetzten Schwerpunkte: Es ist in der Lage, redundante Datenströme zu vermeiden und kann somit die Produzenten entlasten. Um auch den zweiten Schwerpunkt, die Echtzeitverarbeitung von Datenströmen, anzusprechen, wird das Verfahren in diesem Kapitel um Prädiktoren ergänzt.

	DR-Tree	Neu konzipiertes DSPS
Belastung der Wurzel beim Einfügen neuer Objekte	Überdurchschnittlich stark, single point of failure	Geringe Belastung bei starker Lokalität der Ereignisse
Einfügegeschwindigkeit	Wie R*-Baum	Wie R-Baum: 6 mal schneller als R*-Baum
Abhängigkeit der Struktur von der Einfügereihenfolge der Objekte	Stark, Re-Insertion nötig	Wesentlich geringer, Verzicht auf Re-Insertion
Aufteilung übergelaufener Knoten	Global	Lokal
Ausgangsgrad des Datenverteilungsgraphen	Zwischen M_{min} und M_{max}	1

Tabelle 5.2: Vergleich des DR-Trees mit dem in dieser Arbeit konzipierten DSPS

Der Prädiktor ist ein sogenanntes *duales Vorhersageverfahren* (engl. dual prediction scheme) [BSB07, XWL04]. Damit wird der Einsatz von Vorhersagemodellen *auf beiden Seiten* einer Datenverbindung bezeichnet, um auf Kosten der Datengenauigkeit den Kommunikationsaufwand auf dieser Verbindung erheblich zu reduzieren. Gleichzeitig sorgt die Verwendung eines Vorhersagemodells auf der Seite des Konsumenten dafür, dass dieser sich genau dann einen neuen Vorhersagewert erzeugen kann, wenn ein neuer Datensatz benötigt wird. Falls sich die vom Modell erzeugte Vorhersage in einem vorher definierten Genauigkeitsrahmen befindet, sind die Daten somit in Echtzeit verfügbar.

Die Idee eines dualen Vorhersageverfahrens hat ihren Ursprung in dem Versuch, die interne Kommunikation in Sensornetzen zu reduzieren (vgl. dazu auch die Beschreibungen der Systeme mit Vorhersagesystemen in Kapitel 4). Das übergeordnete Ziel ist dabei, die Lebensdauer der einzelnen Sensoren zu verlängern. Da die Sensoren in ihren Energievorräten stark beschränkt sind und das Senden von Nachrichten über ihre drahtlose Verbindung die größte Energiebelastung darstellt, richtet sich ein großer Teil der Forschung darauf, die Kommunikation auf ein Minimum zu reduzieren. Um dies zu erreichen, müssen andere Belastungen, wie z. B. zusätzlicher Rechenaufwand, in Kauf genommen werden.

In dieser Arbeit soll nun der Prädiktor als duales Vorhersagemodell eingesetzt werden, um die Kommunikation innerhalb des gesamten Systems – von den Produzenten bis hin zu den einzelnen Konsumenten – zu optimieren und die versendete Datenmenge zu reduzieren. Außerdem ermöglicht der Prädiktor die Verfügbarkeit von Echtzeitdatenströmen auch für Konsumenten, deren Verbindung zu den einzelnen Produzenten mit einer hohen Latenz belastet ist.

Im nächsten Abschnitt werden zunächst verschiedene Vorhersagemodelle beschrieben und bezüglich ihrer Erfüllung gegebener Leistungskriterien verglichen. Der darauffolgende

Abschnitt beschreibt das Konzept des Prädiktors. Dieser bietet die Möglichkeit, beliebige Modelle auf allen Datenverbindungen eines Stream Processing Systems zur Vorhersage von Werteverläufen einzusetzen. In einem abschließenden Abschnitt werden dann die Konsequenzen besprochen, die sich aus der Verwendung von Vorhersagemodellen für das Gesamtsystem ergeben.

5.5.1 Vorhersagemodelle

Die Vorhersage von Sensorwerten basiert auf der Verwendung von Modellen. Abstrakt gesehen stellt ein Modell eine Rechenvorschrift dar, mit der es auf Basis einiger tatsächlich gemessener Werte eines Datenstroms möglich ist, zukünftige Werte vorherzusagen. Ein Modell wird durch eine Reihe von Parametern initialisiert, die die Rechenvorschrift, mit der die Vorhersagewerte erstellt werden, bestimmen.

Die Vorhersagewerte sind immer mit einer gewissen Ungenauigkeit verbunden, was bedeutet, dass der Einsatz von Vorhersagemodellen nur sinnvoll ist, wenn die Konsumenten bereit sind, eine gewisse Abweichung des für sie erzeugten Wertes vom tatsächlichen Messwert zu akzeptieren (vgl. Kapitel 4 für die Spezifizierung der maximal tolerierten Ungenauigkeit in der Definition einer Anfrage durch einen Interessenten).

Durch eine Aktualisierung der oben erwähnten Parameter ist es auch möglich, ein einmal initialisiertes Modell anzupassen, falls die Vorhersage für einen Datenstrom nicht mehr innerhalb der vom Konsumenten spezifizierten Grenzwerte erfolgt.

Modelle unterscheiden sich in der Komplexität und Art der benutzten Rechenvorschrift. Je nach Anwendungszweck und -spezifizität des jeweiligen Modells sind diese Vorschriften zunächst sehr vage und benötigen daher viele Parameter zur Einstellung oder aber bereits sehr spezifisch und können mit nur wenigen Parametern initialisiert werden. Komplexe Modelle, die sehr viele Parameter verwenden, sind somit auch wesentlich flexibler einsetzbar als Modelle, die nur sehr wenige Parameter annehmen. Die Art der Rechenvorschrift ist ebenfalls sehr vom Anwendungszweck abhängig, für den die jeweiligen Modelle konzipiert wurden. Während Modelle zur Benutzung in der Audio- und Videoverarbeitung vor allem Wert auf hohe Ausführungsgeschwindigkeiten legen, spielt dieses Kriterium in der Wirtschaftsmathematik nur eine untergeordnete Rolle. Das führt dazu, dass einige Modelle intern z. B. mit Matrizenmultiplikation arbeiten, während andere stattdessen Integrale einsetzen.

Die Wahl eines Modells hängt also stark vom gewünschten Einsatzzweck und den dafür definierten Leistungskriterien ab. Der Einsatzzweck in dem hier konzipierten System ist zum einen die Gewährleistung von Echtzeitdatenströmen auch über Verbindungen mit hoher Latenz und zum anderen die Reduktion des Datenaufkommens im Gesamtsystem. In Einklang mit diesen Einsatzzielen werden folgende Leistungskriterien definiert, denen das eingesetzte Modell gerecht werden muss:

Kriterium 1: Schnelle Ausführung Das Modell wird unter anderem dazu eingesetzt, auf Verbindungen mit hoher Latenz in Echtzeit verfügbare Datenströme zu gewährleisten. Das bedeutet auch, dass die Erzeugung eines neuen Vorhersagewertes besonders schnell erfolgen muss. Also gilt: Je weniger aufwändige Rechenoperationen ein Modell in seiner Rechenvorschrift einsetzt, desto besser schneidet das Modell in diesem Kriterium ab.

Kriterium 2: Benötigtes Vorwissen Auf Grund der unterschiedlichen Anwendungsspezifizität der Modelle ergibt sich die Frage, wie viel Vorwissen über die Beschaffenheit der Datenströme, deren Verlauf vorhergesagt werden soll, nötig ist. Im Systemmodell wurde in Kapitel 3.2 z. B. die Art der vom System übermittelten Datenströme auf Temperaturdaten eingeschränkt. Vor dem Hintergrund der Erweiterbarkeit des Systems auf andere Arten von Sensordaten wird an das Vorhersagemodell trotzdem die Anforderung gestellt, mit möglichst wenig Vorwissen über die Datenströme selbst auszukommen.

Kriterium 3: Flexibilität Gerade weil von keinerlei Vorwissen über die Art der vorherzusagenden Daten ausgegangen werden soll, wird eine große Flexibilität vom Modell gefordert. Ein Modell schneidet in diesem Kriterium umso besser ab, je breiter sein Einsatzfeld ist und je reichhaltiger seine Einstellungsmöglichkeiten sind.

Kriterium 4: Effizienz Das Modell muss in der Lage sein, den Datenverkehr im System wesentlich zu reduzieren. Dadurch, dass komplexere Modelle jedoch über mehr Parameter verfügen, sind auch die Nachrichten zur Aktualisierung der Modellparameter größer. Um zwei Modelle in ihrer Effizienz vergleichen zu können, darf nicht nur die bloße Anzahl an übermittelten bzw. durch das Modell eingesparten Paketen in Betracht gezogen werden. Stattdessen muss auch die Größe dieser Pakete in die Auswertung miteinbezogen werden. Aus der größeren Flexibilität eines Modells und der damit verbunden Zahl von Parametern folgt, dass jede Aktualisierungsnachricht mit neuen Parametern mehr Daten erfordert, als eine Aktualisierung für ein einfacheres Modell. Somit muss die Ersparnis durch ein komplexeres Modell auch proportional zu seiner Komplexität verlaufen, damit sich die Verwendung des komplexeren Modells auch rentiert. Um den Einsatz eines komplexen Modells zu rechtfertigen, muss dieses also eine wesentlich bessere Approximation des Datenstroms bieten als Modelle geringerer Komplexität. Die Effizienz eines Modells lässt sich also aus der Größe der tatsächlich eingesparten Datenmenge bestimmen.

Will man eine Menge von Modellen auf ihre Eignung für die Verwendung in dem in dieser Arbeit vorgestellten Stream Processing System überprüfen, *bevor* sie implementiert und im laufenden System evaluiert werden, stehen dafür nur die ersten drei Kriterien zur Verfügung. Das vierte Kriterium, die Effizienz, muss durch Messungen innerhalb des laufenden Gesamtsystems bestimmt werden.

In den folgenden drei Abschnitten werden verschiedene Modelltypen vorgestellt, die danach anhand der ersten drei der obigen Kriterien verglichen werden.

Das Kalman-Filter

Einer der ersten Ansätze, ein doppeltes Vorhersageverfahren im Bereich der Sensornetze aufzubauen, setzt auf die Verwendung von Kalman-Filtern [JCW04]. Auf Grund seiner Eigenschaften scheint sich das Kalman-Filter sehr gut für diesen Zweck zu eignen: Sie sind sehr flexibel, berücksichtigen Hintergrundrauschen und versuchen, in einem iterativen Vorgang den mittleren quadratischen Fehler zu minimieren. Auch seine Verbreitung spricht für sich; das Kalman-Filter wird in Form des PLL-Filters vor allem in der Video- und Audioverarbeitung – klassische Beispiele für die Verarbeitung von kontinuierlichen Datenströmen – eingesetzt. Der Nachteil eines Kalman-Filters ist jedoch die Menge an benötigtem Vorwissen. Dazu gehört eine Übergangsmatrix, die dafür verwendet wird, aus einer Reihe der vorhergehenden Messwerte den nächsten zu extrapolieren, und zwei Kovarianzmatrizen, die Messfehler und Übertragungsfehler spezifizieren, die im Voraus angegeben werden müssen. Das bedeutet, dass über die Beschaffenheit der übertragenen Daten und über die Fehleranfälligkeit der Messverfahren und Übertragungskanäle bereits im Voraus Informationen vorhanden sein müssen, um diese Matrizen aufstellen zu können.

Das Konstantenmodell

Die einfachste denkbare Modellfunktion, mit der sich der Sensorwert zum Zeitpunkt t voraussagen lässt, ist eine waagerechte Gerade, die dem letzten gemessenen Wert entspricht. Die Aktualisierung eines Sensorwertes bleibt dann aus, wenn der zum Zeitpunkt t gemessener Wert x_t identisch mit dem Wert zum Zeitpunkt $t-1$ ist. Andernfalls wird ein Update gesendet und das Modell auf den neuen Wert angepasst. Dieses Modell wird auch als *Konstantenmodell* bezeichnet. Allein mit diesem relativ einfachen Modell war es unter realen Bedingungen möglich, zwischen 20 und 65% an gesendeten Paketen einzusparen [BSB07]. Das Konstantenmodell ist das einfachste und schnellste, aber auch das unflexibelste aller denkbaren Modelle, da es lediglich den vorherigen Wert als Parameter speichert.

Das Box-Jenkins-Modell

Es existieren auch wesentlich komplexere Modelle, wie z. B. die unter anderem in der Ökonometrie und Finanzmathematik zur Vorhersage von Bruttosozialprodukt oder Börsenkursen eingesetzten *Box-Jenkins-Modelle* [BJ90]. Diese Modelle sind durch verschiedenste Parameter konfigurierbar und sind somit wesentlich flexibler als das oben angesprochene Konstantenmodell. Das Box-Jenkins-Modell verwendet sowohl einen autoregressiven Anteil (AR) als auch ein gleitendes Mittel (engl. moving average, MA). Der autoregressive Anteil bestimmt deterministisch aus einer gegebenen Anzahl k von vorhergehenden Messwerten x_{t-1} bis x_{t-k} den für den Zeitpunkt t vorhergesagten Wert $\widehat{x}_t$. Der Parameter k ist flexibel; man spricht hierbei auch von der Ordnung des Modells. Das gleitende Mittel wurde in die Modellberechnung miteinbezogen, um zufällige Störprozesse zu berücksichtigen, die die zukünftige Entwicklung der Werte nachhaltig beeinflussen. Der MA-Anteil wird somit stochastisch

	Kalman-Filter	Konstantenmodell	Box-Jenkins-Modell
1.: Ausführungsgeschwindigkeit	Schnell	Sehr schnell	Schnell
2.: Benötigtes Vorwissen	Hoch	Sehr gering	Gering
3.: Flexibilität	Hoch	Sehr gering	Hoch

Tabelle 5.3: Vergleich von Kalman-Filter, Konstantenmodell und Box-Jenkins-Modell

bestimmt und simuliert ein Hintergrundrauschen in den Messwerten. Ein wesentlicher Vorteil des Box-Jenkins-Modells gegenüber anderen Modellen mit vergleichbarer Komplexität ist seine Aufwandsasynchronität. Das bedeutet, dass die Bestimmung der Parameter zwar verhältnismäßig aufwändig ist, die Ausführung des Modells – und damit die Vorhersage selbst – jedoch relativ wenige Rechenoperationen erfordert.

Tabelle 5.3 vergleicht die vorgestellten Modelle noch einmal unter den vorher definierten Leistungskriterien.

Das Kalman-Filter findet auf Grund des benötigten Vorwissens keine Verwendung in dieser Arbeit. Das Konstantenmodell als einfachste Möglichkeit, eine Vorhersage zu verwenden, und das Box-Jenkins-Modell als komplexere, aber auch wesentlich flexiblere Lösung werden für das Stream Processing System verfügbar gemacht. In Kapitel 6 werden diese beiden Modelle auf das vierte der definierten Leistungskriterien, ihre Effizienz, untersucht.

Der folgende Abschnitt beschreibt die Möglichkeit, mit Prädiktoren beliebige Vorhersagemodelle in das in dieser Arbeit konzipierte Stream Processing System zu integrieren.

5.5.2 Integration von Prädiktoren

Der Prädiktor installiert ein duales Vorhersagesystem auf allen Verbindungen innerhalb des Stream Processing Systems, über die Sensordaten übermittelt werden. Das bedeutet, dass mit Hilfe eines beliebigen Modells gleichzeitig auf beiden Seiten einer Datenverbindung die Vorhersage des nächsten Datensatzes durchgeführt wird. Währenddessen ermittelt der Produzent weiterhin regelmäßig aktualisierte Messwerte, überträgt diese jedoch vorerst nicht.

Unterscheidet sich der vorhergesagte Wert um weniger als den vom Konsumenten spezifizierten maximalen Fehler ϵ vom tatsächlichen Messwert, besteht für den Konsumenten keinerlei Handlungsbedarf (vgl. dazu die in Kapitel 4 beschriebene Spezifikation des maximal tolerierten Fehlers in einer Anfrage durch Interessenten). Der Konsument arbeitet mit dem bei ihm ebenfalls ermittelten Vorhersagewert und kann sicher sein, dass er dem von ihm spezifizierten maximalen Fehler genügt. Das Ausbleiben einer Aktualisierung durch den Produzenten implizierte also die korrekte Funktionsweise des verwendeten Modells.

Algorithmus 5.9 Kommunikation mit Prädiktor: Der Produzent übermittelt dem Konsumenten nur dann den aktuellen Sensorwert, wenn dieser sich um mehr als den maximal tolerierten Fehler ϵ vom aktuellen Vorhersagewert unterscheidet

```
procedure receiveNewDataSet(newSensorDataSet)
    newPredictedDataSet ← model. getNextPredictedValue
    if (|newSensorDataSet − newPredictedDataSet| > ϵ) then
        sendSensorDataUpdateMessage(outgoingLink, newSensorDataSet)
    end if
end procedure
```

Erkennt der Produzent einen Unterschied zwischen Vorhersagewert und Messwert, der größer als der vom Konsumenten tolerierte Fehler ist, sendet er eine Aktualisierungsnachricht mit dem neuesten Messwert an den Konsumenten. Dieser erkennt, dass sein Modell nicht im Rahmen des spezifizierten Fehlers arbeitet und verwendet statt des Vorhersagewerts also den erhaltenen Messwert.

Das Verhalten, wann genau Aktualisierungen von Sensorwerten übermittelt werden, wird nochmals in Algorithmus 5.9, der vom Produzenten für jeden neu ermittelten Messwert durchgeführt wird, verdeutlicht.

Abbildung 5.11 zeigt die Integration von Prädiktoren in die in Kapitel 5.2 vorgestellte Gesamtarchitektur. Jeder eingehenden Verbindung eines Knotens wird ein Prädiktor mit Vorhersagemodell zugeordnet. Falls keine Aktualisierung über die eingehenden Verbindungen erfolgt, wendet der Knoten seinen Operator auf die von den jeweiligen Vorhersagemodellen erzeugten Datensätze an. Anschließend erzeugt er mit seinem Vorhersagemodell den gleichen Wert, den auch der seiner ausgehenden Verbindung zugeordnete Knoten generiert. Sollte der Vorhersagewert innerhalb des maximal tolerierten Fehlers um den vom Operator berechneten Wert liegen, wird der Vorhersagewert an die Anwendung übermittelt. Über die ausgehende Verbindung erfolgt keine Aktualisierung.

Wird der Fehlergrenzwert überschritten, wird stattdessen der berechnete Wert an die Anwendung übermittelt und auch in einer Aktualisierungsnachricht über die ausgehende Verbindung gesendet.

Bestimmen der Modellparameter Um ein Modell zur Vorhersage verwenden zu können, muss dies zunächst, wie im vorherigen Abschnitt 5.5.1 beschrieben, mit Parametern initialisiert werden. Diese Parameter müssen jeweils auf beiden Seiten einer Verbindung identisch sein, damit sowohl der Produzent als auch der Konsument dieselben Vorhersagewerte ermitteln. Die Bestimmung dieser Parameter übernimmt der Konsument eines Datenstroms.

Die Parameter werden in zwei Schritten bestimmt: Zuerst beschränkt sich der Konsument auf das Sammeln von Informationen. Während dieser Phase verläuft die Kommunikation wie gewohnt; der Produzent übermittelt für jeden gemessenen Sensorwert eine Aktualisierungsnachricht. Hat der Konsument eine bestimmte Menge von Sensorwerten gesammelt, ermittelt er auf der Basis dieser Werte die Parameter für ein Vorhersagemodell. War dies

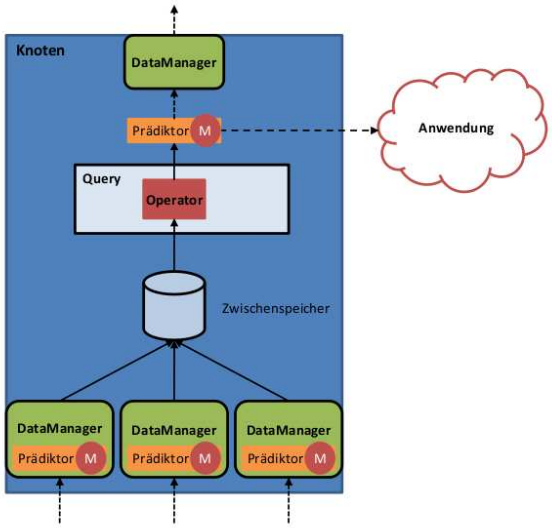

Abbildung 5.11: Die Integration von Prädiktoren mit ihrem Vorhersagemodell M in die Gesamtarchitektur

erfolgreich, übermittelt er die Modellparameter zusammen mit dem durch ihn maximal tolerierten Fehler an den Produzenten.

Instanziierung von Modellen Nachdem die Parameter bestimmt und auf beiden Seiten einer Datenverbindung bekannt gemacht wurden, instanziieren sowohl Konsument als auch Produzent auf ihrer Seite ein Modell. Die Kommunikation verläuft nun nicht mehr direkt zwischen den beiden Parteien ab, sondern findet nun, wie in Algorithmus 5.9 beschrieben, dank der modellgestützten Vorhersage nur noch selektiv statt.

In der Abbildung 5.12 ist nochmals verdeutlicht, wie sich die direkte Kommunikation zwischen Produzent und Konsument auf Bild (1) zu einer modellgestützten Vorhersage auf Bild (2) wandelt. Während anfangs noch dauerhaft Sensordaten übermittelt wurden, sendet der Produzent nun nur noch im Bedarfsfall – das bedeutet nur dann, wenn die Vorhersage um einen zu großen Wert vom tatsächlichen Messwert abweicht – aktualisierte Sensordaten.

Aktualisieren der Modellparameter Wurde die Kommunikation auf das modellbasierte Verfahren umgestellt, sagt der Empfang eines aktualisierten Sensordatenwertes aus, dass das zur Vorhersage verwendete Modell nicht korrekt ist und angepasst werden muss. Der Konsument berechnet unter Berücksichtigung des erhaltenen Sensorwertes aktualisierte Modellparameter und teilt diese dem Produzenten mit. Beide Seiten aktualisieren ihr Modell mit den neuen Parametern und verfügen nun wieder über eine Vorhersage, die innerhalb des spezifizierten maximalen Fehlers arbeitet.

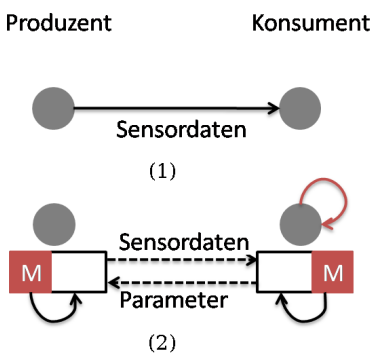

Abbildung 5.12: Der Transfer von Sensordaten ohne Prädiktor (1) und mit modellgestütztem Prädiktor (2)

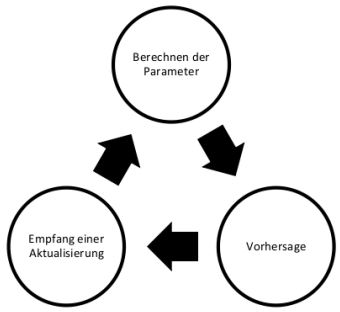

Abbildung 5.13: Der Prädiktionskreislauf

Von hier an wiederholt sich der auf Bild 5.13 dargestellte Kreislauf von Empfang aktualisierter Sensordaten, Berechnung aktueller Modellparameter und Vorhersage.

Nachdem die grundlegende Funktionalität des Prädiktors beschrieben wurde, behandelt der nächste Abschnitt die Auswirkungen des Prädiktors auf das Verhalten des Stream Processing Systems.

5.5.3 Auswirkungen der Prädiktoren auf die Architektur

Die Integration eines Vorhersageverfahrens mit Prädiktoren in das in Kapitel 5.3 vorgestellte Konzept zieht weitere Konsequenzen nach sich.

Das in Kapitel 5.3.1 beschriebene *Verfahren zum Einfügen* neuer Knoten muss leicht modifiziert werden, um den unterschiedlichen Anforderungen der Peers Rechnung zu tragen. Außerdem wird beschrieben, wie diese *Anforderungen an die Vorhersagemodelle*, also insbesondere der maximal tolerierte Fehler, in das System eingebracht werden.

Danach wird das Systemverhalten unter der Verwendung von Prädiktoren genauer analysiert. Dabei wird besonders darauf eingegangen, welche Auswirkungen *Dynamik* – also der Ausfall von Knoten oder deren unerwartetes Verlassen des Systems – auf das Gesamtsystem haben kann.

Als letztes werden noch zwei Punkte behandelt, die sich mit den Vorhersagemodellen selbst beschäftigen: Zum einen stellt sich die Frage, wie die *Modellrümpfe*, also die Grundgerüste eines Modells, die nur noch die Eingabe von Parametern erfordern, um ihre Vorhersage starten zu können, innerhalb des Systems verfügbar gemacht werden. Zum anderen wird die Frage nach der Konvergenz der Modellparameter aufgeworfen.

Auswirkungen auf das Einfügeverfahren Die Anwendung, die die von einem Knoten angefragten Sensordaten weiterverarbeitet, kann die von ihr tolerierte Ungenauigkeit selbst quantifizieren. Das führt dazu, dass verschiedene Knoten zwar den gleichen geographischen Bereich anfragen, jedoch unterschiedliche Fehlertoleranzen spezifizieren. Um die Gemeinsamkeiten zwischen den Anfragen weiterhin ausnutzen zu können, muss der Platzierungsalgorithmus um diese Fehlertoleranzen erweitert werden. Von zwei Anfragen, die den gleichen Bereich anfordern, muss diejenige näher an der Quelle platziert werden, die die geringste Fehlertoleranz spezifiziert hat. Der Knoten, der diese Anfrage verwaltet, kann dann die Daten an den zweiten Interessenten weiterleiten, so dass die eigentliche Quelle der Sensordaten so wie bisher nur eine ausgehende Verbindung besitzt. Die von den Anwendungen spezifizierte Fehlertoleranz wird somit als *Maximalfehler* interpretiert.

Die Interpretation der Fehlertoleranz als Maximalfehler erlaubt es auch, einem Peer genauere Daten bereitzustellen, als dieser in seiner Anfrage spezifiziert hatte. Dies kann erforderlich sein, wenn von zwei Peers derjenige einen höheren Fehler toleriert, der das Ergebnis seiner Anfrage dem anderen Peer als Teilergebnis zur Verfügung stellen sollte. Das bedeutet, dass Knoten, die auf Grund ihrer Anfrage weiter unten in der verteilten R-Baum-Struktur eingefügt wurden, unter Umständen genauere Daten erhalten, als sie angefordert haben.

Anforderungen der Anwendung Ein Interessent tritt mit einer Anfrage dem Gesamtsystem bei und wird damit zum Peer. In dieser Anfrage muss, wie in Kapitel 3.2 beschrieben, der maximal tolerierte Fehler spezifiziert sein. Die Anwendung, die auf dem Peer läuft und den angefragten Datenstrom als Eingabe erfordert, kann ihre Bedürfnisse also in der gestellten Anfrage dem Gesamtsystem vermitteln. Innerhalb des Systems werden diese Bedürfnisse dann vom Konsumenten zusammen mit den Parametern zur Modellinstanziierung an den Produzenten übertragen.

Auswirkungen von Dynamik Die Auswirkungen eines Knotenausfalls innerhalb der verteilten R-Baum Struktur des Stream Processing Systems muss bei einer Verwendung von Prädiktoren neu betrachtet werden.

Da das Ausbleiben eines aktuellen Sensorwertes durch den Produzenten nun eine andere semantische Bedeutung erhält, nämlich die Aussage, das verwendete Vorhersagemodell sei

korrekt, können diese Nachrichten nicht mehr verwendet werden, um Knotenausfälle zu erkennen. Also muss eine *zusätzliche Überprüfung* in das DSPS integriert werden, so dass z. B. in regelmäßigen Abständen *Heartbeat-Nachrichten* unter Geschwisterknoten bzw. zwischen einem Vaterknoten und seinen Kindern ausgetauscht werden.

Der Einsatz von Prädiktoren bietet aber auch Vorteile in Bezug auf Knotenausfälle. Da der Konsument weiterhin ein Vorhersagemodell zur Verfügung hat, das ihm Werte liefern kann, die zumindest eine Annäherung an die tatsächlichen Messwerte darstellen, ist ein Knotenausfall weniger schwerwiegend. Die Reparatur von Knotenausfällen ist also weitaus weniger zeitkritisch als in anderen kollaborativen Systemen.

Sollten auf Grund von Knotenausfällen Nachrichten für eine Aktualisierung der Modellparameter verloren gehen, verwenden Konsument und Produzent Modelle mit verschiedenen Parametern. Nach einer Reparatur der Baumstruktur kann es ebenfalls vorkommen, dass kurzzeitig beide Seiten einer Verbindung nicht mit denselben Parametern und Fehlerdefinitionen arbeiten.

Für die korrekte Funktionsweise des dualen Vorhersagemodells ist kritisch, dass die vorhergesagten Werte auf beiden Seiten der Verbindung gleich sind. Die Parameter des Modells müssen also schnellstmöglich synchronisiert werden. Um solche Situationen zu erkennen, kann eine regelmäßiger Vergleich der Parameter auf beiden Seiten einer Datenverbindung eingeführt werden. Kombiniert man diese mit den oben geforderten Heartbeat-Nachrichten, ist der zusätzliche Kommunikationsaufwand sehr gering.

Gleichen sich zwar die Modellparameter, weist das Modell des Produzenten jedoch eine geringere Fehlertoleranz auf, als der Konsument spezifiziert hat, arbeitet das vorgestellte Verfahren weiterhin korrekt. Der Konsument erhält zwar aktualisierte Sensorwerte, die für die von ihm geforderte Genauigkeit nicht nötig gewesen wären, kann dann aber stets mit korrekten Werten arbeiten.

Verfügt stattdessen der Konsument über eine geringere Fehlertoleranz als das beim Produzenten eingesetzte Modell, bleiben Aktualisierungen aus, die erforderlich gewesen wären. In diesem Fall ist eine schnelle Synchronisation des maximal tolerierten Fehlers nötig, um die korrekte Funktionsfähigkeit des Vorhersageverfahrens wiederherzustellen.

Einbringen von Modelltypen In dem in Abschnitt 5.5.2 beschriebenen Vorgehen zur Instanziierung eines Modells wurde nicht berücksichtigt, *woher* ein Peer das Grundgerüst eines Modells bezieht, das er nur noch mit Parametern ausstatten muss.

Das in dieser Arbeit verwendete Verfahren setzt zu diesem Zweck eine Fabrik-Konstruktion ein, die ein Modellgerüst mitsamt Algorithmus, der aus einer Reihe von Messwerten die Modellparameter bestimmen kann, zurückliefert. Da die Art des eingesetzten Modells zur Zeit nur systemweit bestimmt werden kann, müssen neue Modelltypen nur in die Fabrik eingebracht werden.

Sollen später auch unterschiedliche Modelltypen verfügbar sein und gleichzeitig im selben System eingesetzt werden können, um z. B. die unterschiedliche Beschaffenheit verschiedener

Arten von Sensordatenströmen zu berücksichtigen, muss gewährleistet werden, dass alle Konsumenten eines bestimmten Datenstroms denselben Modelltyp einsetzen. Andernfalls wäre auf Grund impliziter Eigenschaften der verschiedenen Modelltypen unklar, zu welchen Eigenschaften noch Garantien an den verschiedenen Stellen eines Datenstroms auf dessen Weg von Produzent zu Konsument gegeben werden könnten.

Konvergenz der Modellparameter Eine Frage, die es zur Laufzeit des Gesamtsystem zu klären gilt, ist die nach der Konvergenz der Modellparameter. Liegt eine Konvergenz der Parameter vor, bedeutet dies, dass sich die Vorhersage des Modells stetig verbessert. In diesem Fall werden die zeitlichen Abstände zwischen zwei Parameteraktualisierungen immer größer. Damit steigt auch die Menge der pro Zeiteinheit gegenüber direkter Kommunikation eingesparten Daten. Außerdem kann der Konsument mit immer länger zusammenhängenden Echtzeitdaten arbeiten.

Möglich ist auch, dass die Parameter nicht konvergieren, sondern zwischen zwei bzw. mehreren Status wechseln. Eine Oszillation der Modellparameter entsteht dann, wenn der Datenverlauf regelmäßige Schwankungen aufweist, die vom Modell nicht erfasst werden können. Eine Lösung ist, eine Komponente in die Modelle aufzunehmen, die saisonale Schwankungen ausdrücken kann. Mit dem S-ARIMA-Modell [BJ90] ist bereits ein Box-Jenkins-Modell mit saisonaler Komponente verfügbar. Da die Verwendung einer saisonalen Komponente die Komplexität eines Modells erhöht und damit seine Effizienz senken kann, wird in Kapitel 6 analysiert, ob ein solches Modell für das in dieser Arbeit konzipierte System benötigt wird bzw. sinnvoll ist.

Kapitel 6

Evaluation

Für eine Evaluierung des in Kapitel 5 vorgestellten Konzeptes wurde ein Prototyp implementiert. Dieser Prototyp nutzt das in Java geschriebene Peer-to-Peer-Simulationsframework PeerSim [JMJV] als Basis. PeerSim bietet die Möglichkeit, ein Peer-to-Peer-System mit einer großen Teilnehmerzahl zu simulieren. Außerdem sind bereits Implementierungen für wichtige Elemente, wie z. B. die Transportschicht, integriert.

Das Konzept für das Distributed Stream Processing System aus Kapitel 5.3 wurde vollständig implementiert. Aus Zeitgründen konnten die Vorhersagemodelle jedoch nicht wie in Kapitel 5.5.1 beschrieben implementiert werden. Für die Implementierung eines Box-Jenkins-Modells wurde auf die Bibliothek drej[1] zurückgegriffen. In dieser Bibliothek ist ein AR-Modell verfügbar. Dabei handelt es sich um ein Box-Jenkins-Modell, das nur aus einem autoregressiven Anteil besteht. Der MA-Teil, der das „weiße Rauschen" modelliert, wurde weggelassen. Dies muss bei der Bewertung der Simulationsergebnisse berücksichtigt werden.

Als Datenbasis für die Gateways dienten tatsächliche Messwerte für Temperaturdaten aus dem Intel Research Lab in Berkeley[2]. Dort wurden mit 54 Sensoren alle 31 Sekunden zwischen dem 28. Februar und dem 5. April 2004 Messungen vorgenommen. Die Anordnung der Sensoren kann der Abbildung 6.1 entnommen werden.

Die Messwerte dieser einzelnen Sensoren werden als Ergebniswerte eines gesamten Sensornetzes angesehen. Dafür wird je ein Gateway pro vorhandenem Sensordatensatz im Testszenario verwendet.

6.1 Ablauf der Experimente

Die folgenden Evaluierungen wurden jeweils mehrfach in unterschiedlichen Szenarien durchgeführt. Allen Szenarien ist gemeinsam, dass die 54 Gateways zunächst auf einem neun mal sechs Felder umfassenden Netz platziert wurden. Mit den Gateways wurde

[1] http://www.gregdennis.com/drej/
[2] http://berkeley.intel-research.net/

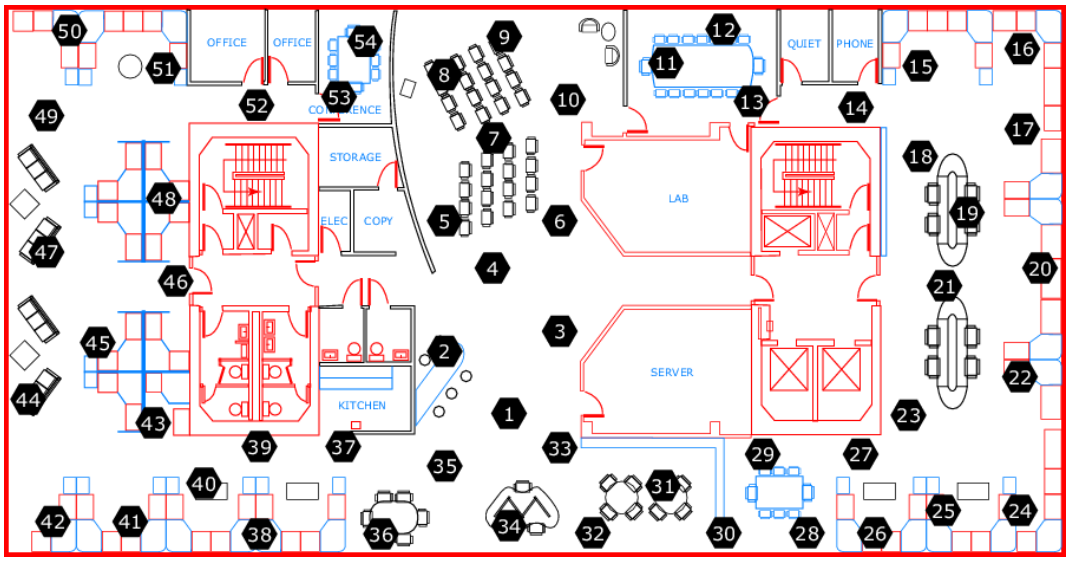

Abbildung 6.1: Anordnung der Sensoren im Intel Research Lab
Quelle: http://db.csail.mit.edu/labdata/labdata.html

anschließend, wie in Kapitel 5.3.3 beschrieben, der Baum initialisiert. Als maximal tolerierter Fehler wurde für die Experimente, die mit Vorhersagemodellen arbeiten, $\epsilon = 0.01$ spezifiziert. Die Ordnung des AR-Modells – also die Anzahl der zur Bestimmung der Modellparameter verwendeten Stützstellen – wurde auf 5 festgelegt.

Die Szenarien unterscheiden sich in der Zahl und Platzierung der an das System gestellten Anfragen. Allen Szenarien ist gemeinsam, dass folgende Parameter mit einem Zufallsgenerator bestimmt wurden:

a. Die Mittelpunkte der Anfragen

b. Die Höhe und Breite des Anfragebereichs

c. Der Gateway, über den Kontakt zum System aufgebaut wird

Die Parameter a und b werden in den drei Szenarien zufallsverteilt. Je nach Szenario wurden der c ebenfalls zufallsverteilt oder aber mit einer hohen Wahrscheinlichkeit auf die nahe Umgebung einiger spezieller Bereiche konzentriert.

6.1.1 Testszenario 1: 10 Anfragen

In diesem Szenario wird der Parameter c gleichverteilt gewählt. Das Konzept wird mit einer minimalen Zahl von Anfragen auf seine Funktionsweise überprüft. Es ist zu erwarten, dass der R-Baum auch nach dem Einfügen der Anfragen zu einem sehr großen Teil aus Gateways

besteht. Ob und wie sich dieser ungewöhnlich große Anteil von Gateways auf das System auswirkt, wird Teil der Analyse dieses Szenarios sein.

6.1.2 Testszenario 2: 100 Anfragen

Auch in diesem Szenario wird der Parameter c gleichverteilt gewählt. Im Gegensatz zum vorherigen Szenario stellt dieser Testaufbau eine moderate Belastung für das System dar.

6.1.3 Testszenario 3: 1000 Anfragen

In diesem dritten Szenario soll die Belastbarkeit des Systems getestet werden. Betrachtet man den Anwendungszweck des in dieser Arbeit vorgestellten Systems, ist dieses Testszenario mit einem Faktor von 20 zwischen Sensornetzen und gestellten Anfragen das realistischste. Daher wurde dieses Szenario gewählt, um noch weitere Modifikationen an den Parametern vorzunehmen: Ein Experiment erfolgt wie auch in den vorherigen Fällen mit gleichverteilten Anfragemittelpunkten, das andere Experiment erfolgt mit einer Konzentration der Anfragenmittelpunkte auf drei Hot-Spots.

Testszenario 3a: Gleichverteilung der Anfragen

Wie auch in der vorherigen Testszenarien wird hier der Parameter a gleichverteilt gewählt.

Testszenario 3b: Hot-Spots

Für feweils $\frac{1}{3}$ der gesamten Anfragenmenge wird der Parameter a normalverteilt um einen zufälligen Erwartungswert gewählt. Das sorgt dafür, dass es drei Gebiete gibt, die besonders häufig angefragt werden, sogenannte *Hot-Spots*. Die weiterhin vorliegende Gleichverteilung der Höhe und Breite der jeweiligen Anfragebereiche sorgt für ein ausgewogenes Verhältnis zwischen kleinen und großen Anfragen.

6.2 Beschreibung der Experimente

Im Folgenden wird für jedes Experiment beschrieben, welche Größen gemessen werden und über welche Eigenschaft des Systems diese gemessene Größe eine Aussage treffen kann. Anschließend werden die Ergebnisse der unterschiedlichen Testszenarien in den jeweiligen Experimenten aufgelistet. Nach der Durchführung aller Experimente werden die Ergebnisse für die Testszenarien zusammengeführt und einer gemeinsamen Bewertung unterzogen. Die neuen Anfragen wurden in jeweils zehn Simulationsschritten an das System gestellt. Das bedeutet, dass in Szenario 1 pro Simulationsschritt eine Anfrage, in Szenario 2 pro

Simulationsschritt zehn Anfragen und in Szenario 3 pro Schritt 100 Anfragen an das System gestellt wurden. Eine vollständige Simulation besteht aus 30 000 Einzelschritten.

6.2.1 Mehraufwand für Peers

Gemessene Größe Für jeden Peer P_i im Baum wird das Verhältnis zwischen der Fläche, die er verwaltet, und der Fläche, die von ihm angefragt wurde, berechnet. Dieses Verhältnis wird als r_i bezeichnet und ergibt sich zu

$$r_i = \frac{A_{angefr}}{A_{verw}} \in (0, 1]$$

Außerdem wird über alle Knoten hinweg das Maximum $r_{max} = \max\{r_i\}$, das Minimum $r_{min} = \min\{r_i\}$ und der Durchschnittswert $\bar{r}$ berechnet. Dabei gilt: Je näher sich r_i an 1 befindet, desto weniger zusätzliche Fläche muss der Peer verwalten.

Ziel Das Ziel dieser Messungen ist es, eine Aussage darüber treffen zu können, wie groß die Mehrbelastung für einen Peer ist, wenn er sich dem System anschließt. Da anders als in bisherigen DSPS jeder Teilnehmer dazu verpflichtet wird, an der Verteilung der Sensordaten mitzuwirken, ist der gegenüber den bisherigen Systemen zu leistende Mehraufwand ein kritischer Faktor für die Akzeptanz des neues Konzeptes. Der Mehraufwand kann aus der Größe der verwalteten Fläche abgeleitet werden, da diese Fläche genau dem Bereich entspricht, über den ein Peer Informationen für andere Peers bereitstellen muss. Der Durchschnittswert $\bar{r}$ gibt Auskunft darüber, ob die Lastverteilung auf die Teilnehmer des Systems nach fairen Gesichtspunkten erfolgt.

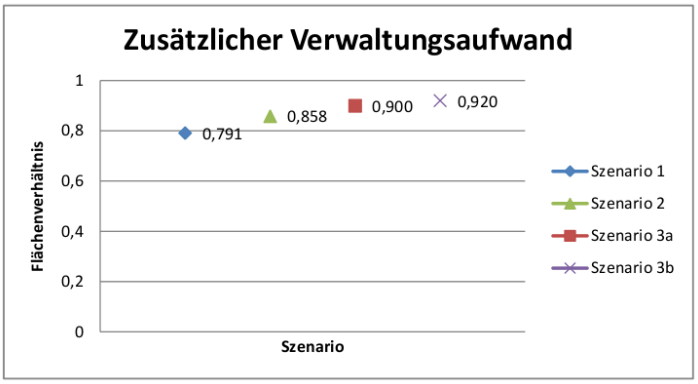

Abbildung 6.2: Der Durchschnittswert $\bar{r}$ gibt das Verhältnis zwischen verwalteter und ursprünglich angefragter Fläche an

Ergebnisse Die Abbildung 6.2 zeigt den Durchschnittswert $\bar{r}$ über alle Teilnehmer des Systems für die Szenarien 1 bis 3b. Zu erkennen ist, dass sich die Durchschnittswerte in allen Szenarien an den Wert 1 annähern. Der Minimalwert für r ergab sich in jedem der durchgeführten Experimente mit $r_{min} = \frac{1}{54}$, was darauf schließen lässt, dass ein Gateway, der während der Initialisierung des Baumes für die Verwaltung der Wurzel zuständig war, diese Position im Baum nicht mehr verlässt.

6.2.2 Lokalität von Ereignissen

Gemessene Größe Eine neu an das System gestellte Anfrage steigt zunächst im Baum auf, danach beginnt das Absinken und Einfügen. Für jede Anfrage Q_i wird das Level L_{ins} im Baum bestimmt, bis zu dem die Anfrage aufsteigen musste. Über alle an das System gestellten Anfragen wird anschließend der Durchschnittswert ermittelt. Das Level der Wurzel wird in diesem Experiment mit 0 angegeben. Je höher der ermittelte Durchschnittswert für L_{ins} also ist, desto geringer ist die Belastung für die Knoten auf den oberen Levels des Baumes.

Ziel Der Hintergrund dieser Analysen ist die Frage, inwieweit das vorgestellte Konzept bereits eine Mehrbelastung der Knoten auf den oberen Levels des Baumes verhindern kann. Auch dieser Wert kann zur Beurteilung der Fairness des Systems herangezogen werden.

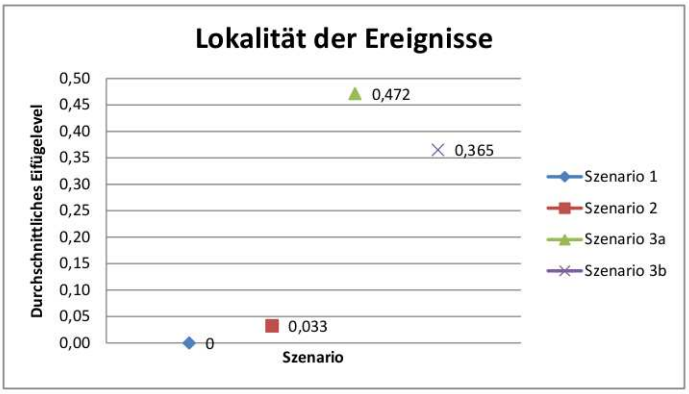

Abbildung 6.3: Der Durchschnittswert für L_{ins} gibt das Level an, bis zu dem neue Anfragen aufsteigen

Ergebnisse Auf Abbildung 6.3 ist zu erkennen, dass der Durchschnittswert für das Level, bis zu dem Anfragen beim Einfügevorgang aufsteigen, in allen Szenarien zwischen 0 und 1 liegt. Auf Abbildung 6.4 ist zusätzlich noch der prozentuale Anteil der Anfragen aufgetragen, die während ihres Einfügevorgangs bis zur Wurzel aufsteigen mussten. Dadurch,

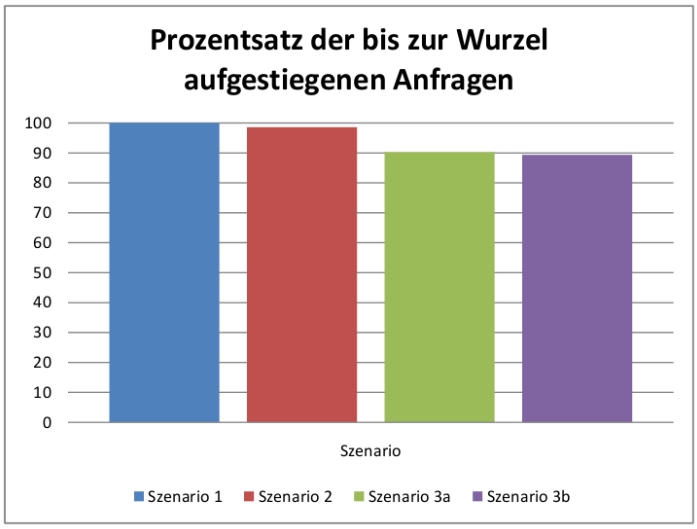

Abbildung 6.4: Der prozentuale Anteil der Anfragen, die bis zur Wurzel des Baumes aufgestiegen sind

beinahe 100 % aller neuen Anfragen über die Wurzel eingefügt wurden, ergeben sich für das durchschnittliche Einfügelevel die geringen Werte < 1. Hier besteht also noch deutliches Optmieriungspotential, um die Last für die Knoten auf den oberen Levels des Baumes zu reduzieren. Durch die Ergebnisse in Experiment 6.2.1 konnte jedoch schon gezeigt werden, dass die Platzierung eines Knotens im Baum sich nach seinem erhaltenen Nutzen richtet. Eine leichte Mehrbelastung der oberen Knoten ist also zu rechtfertigen und schadet der Fairness des Gesamtsystems nicht grundlegend.

6.2.3 Entlastung der Gateways

Gemessene Größe Der Anteil von Gateways im Baum, die noch Verwaltungsaufgaben übernehmen müssen, da sie bisher nicht durch einen Peer ersetzt werden konnten.

Ziel Eines der Hauptziele dieser Arbeit ist die Entlastung der Gateways zu den Sensornetzen. Dies wird dadurch erreicht, dass jeder Gateway nur noch eine ausgehende Verbindung besitzt. Die Gateways müssen jedoch Verwaltungsaufgaben als innere Knoten im Baum übernehmen, was wiederum zusätzlichen Aufwand bedeutet. Dies geschieht auch unter der Annahme, dass die Gateways nach und nach von ihrer Verwaltungsaufgabe entbunden werden, da sie durch Peers ersetzt werden können. Ob und in welchem Maße dies der Fall ist, soll in diesem Experiment überprüft werden.

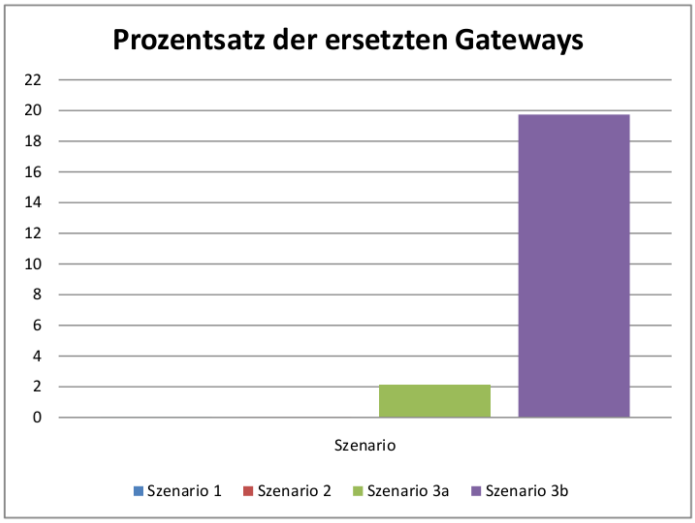

Abbildung 6.5: Der prozentuale Anteil der als innere Knoten agierenden Gateways, die ersetzt wurden

Ergebnisse Der Anteil der Gateways, die von ihren Verwaltungsaufgaben befreit werden konnten, ist für die Szenarien 1 bis 3a sehr gering, wie die Abbildung 6.5 zeigt. Werden die Anfragen auf Hot-Spots konzentriert, ist der Anteil der ersetzten Gateways höher, bewegt sich aber dennoch auf niedrigem Niveau. Dadurch, dass die Zahl der ausgehenden Verbindungen für die Gateways mit der Verwendung des verteilten R-Baumes auf 1 reduziert werden konnte, werden diese im Vergleich zu bisherigen Systemen zur Datenstromverarbeitung erheblich entlastet. Die geringe Mehrbelastung durch Verwaltungsaufgaben innerhalb des Baumes hebt diesen Effekt keineswegs auf.

6.2.4 Effekt von Prädiktoren

In diesem Abschnitt sind Experimente beschrieben, die die Auswirkungen der in Kapitel 5.5 beschriebenen Prädiktoren analysieren. Außerdem wird die Wahl des Vorhersagemodells auf ihre Auswirkungen untersucht.

Einsparungen durch Prädiktoren

Gemessene Größe Die Gesamtzahl der im System gesendeten SensorDataUpdateMessages und ModelDataUpdateMessages in Abhängigkeit der verschiedenen Vorhersagemodelle.

Ziel Um den Effekt der Prädiktoren zu überprüfen, werden vergleichende Messungen durchgeführt. In diesem Experiment wird der Unterschied zwischen einem DSPS ohne Prädiktoren und einem DSPS mit Prädiktoren in Hinblick auf die Anzahl der gesendeten Pakete analysiert. Pakete zur Aktualisierung der Modellparameter eines AR-Modells sind wesentlich größer, als reine Aktualisierungen eines Sensorwertes, da dort drei Parameter statt nur ein Sensorwert versendet werden. Folglich wird eine ModelDataUpdateMessage für das AR-Modell mit dem Faktor 3 gewichtet. Das Konstantenmodell verschickt nur einen Wert zur Aktualisierung, hier ist also keine Gewichtung notwendig.

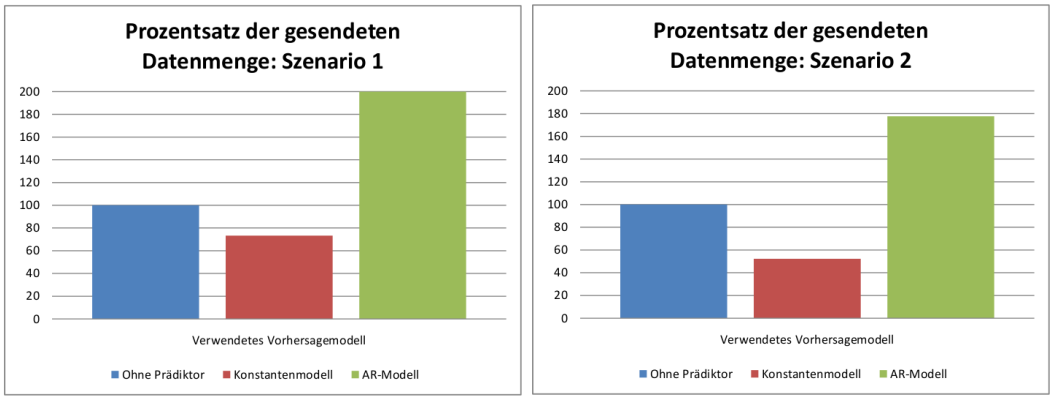

Abbildung 6.6: Der Anteil der unter Verwendung von Vorhersagemodellen in Szenario 1 und 2 gesendeten Daten

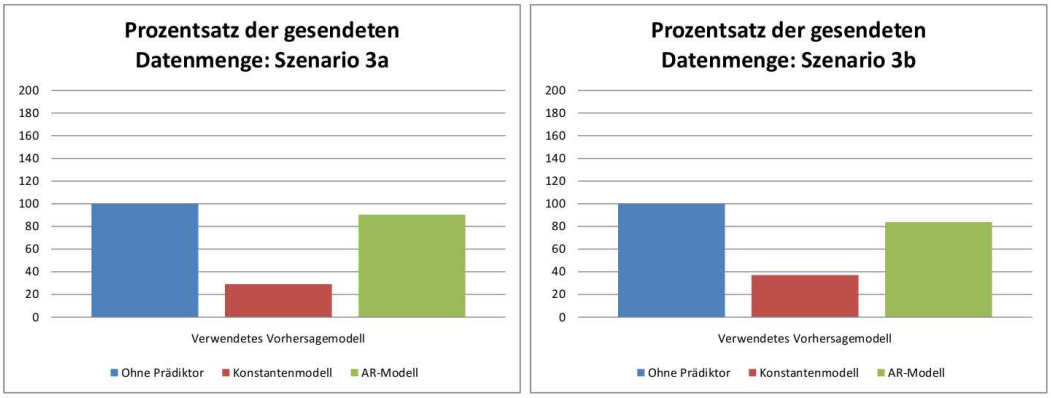

Abbildung 6.7: Der Anteil der unter Verwendung von Vorhersagemodellen in Szenario 3a und 3b gesendeten Daten

Ergebnisse Abbildungen 6.6 und 6.7 zeigen pro Szenario den durchschnittlichen Anteil für die unter Einsatz der verschiedenen Vorhersagemodellen versendeten Pakete in Relation zur

Gesamtzahl der versendeten Pakete ohne Prädiktoren. Die Gesamtzahl für das AR-Modell ist zusätzlich in einer gewichteten Version angegeben. Zu sehen ist, dass das Konstantenmodell wesentlich besser abschneidet, als das AR-Modell. Beide Vorhersagemodelle sind geeignet, die Menge der versendeten Pakete zu reduzieren. Das tatsächliche Datenaufkommen kann jedoch nur das Konstantenmodell reduzieren.

Eignung der Vorhersagemodelle für die Echtzeitverarbeitung

Gemessene Größe Die Intervalle zwischen zwei SensorDataUpdateMessages auf einer Datenverbindung.

Ziel Das Versenden einer SensorDataUpdateMessage bedeutet, dass das verwendete Modell nicht mehr korrekt ist und neu berechnet werden muss. Aus den Daten zur Intervalllänge zwischen zwei solchen Nachrichten ist ersichtlich, ob sich das vorgeschlagene Konzept schon zur Realzeitverarbeitung von Sensordaten nutzen lässt oder ob dafür Erweiterungen nötig sind. Falls das Intervall zwischen zwei Aktualisierungsnachrichten zu gering ist, kann die Latenz zwischen Produzent und Konsument nur in den wenigsten Fällen durch eine Vorhersage egalisiert werden. Außerdem ist aus dem Intervall zwischen zwei SensorDataUpdateMessages erkennbar, ob sich die Modelle im Verlaufe der Simulation in ihrer Vorhersage verbessern können oder ob dafür komplexere Modelle notwendig sind. Außerdem ist mit der Messung des Intervalls erkennbar, wie groß die aus der Berechnung der Modellparameter entstehende Mehrbelastung für die Knoten des Baumes ist. Da sich der Aufwand zur Parameterberechnung für die einzelnen Modelltypen stark unterscheidet, ist auch dies ein wichtiger Faktor bei der Beurteilung der Eignung der Modelle.

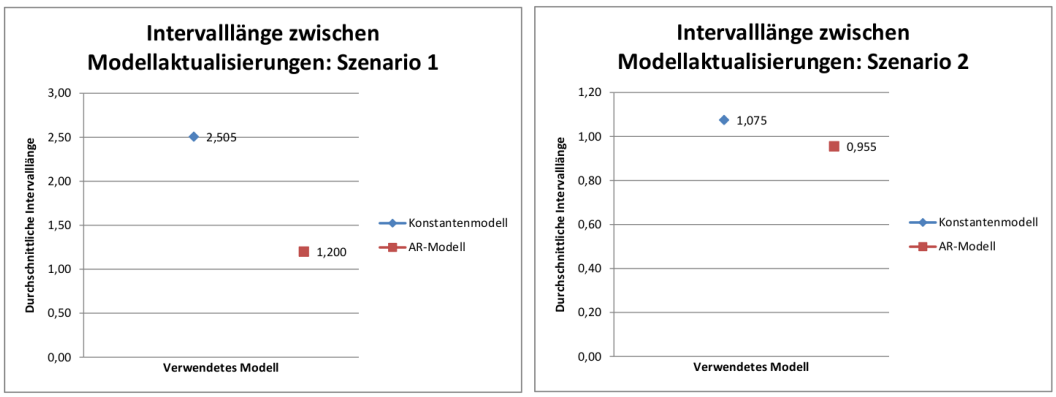

Abbildung 6.8: Das Intervall zwischen Modellaktualisierungen in Szenario 1 und 2

Ergebnisse Für beide Modelle ergibt sich in allen Szenarien ein nahezu identischen Bild: Eine Konvergenz ihrer Parameter ist nicht ersichtlich. Die Vorhersagegenauigkeit der Modelle

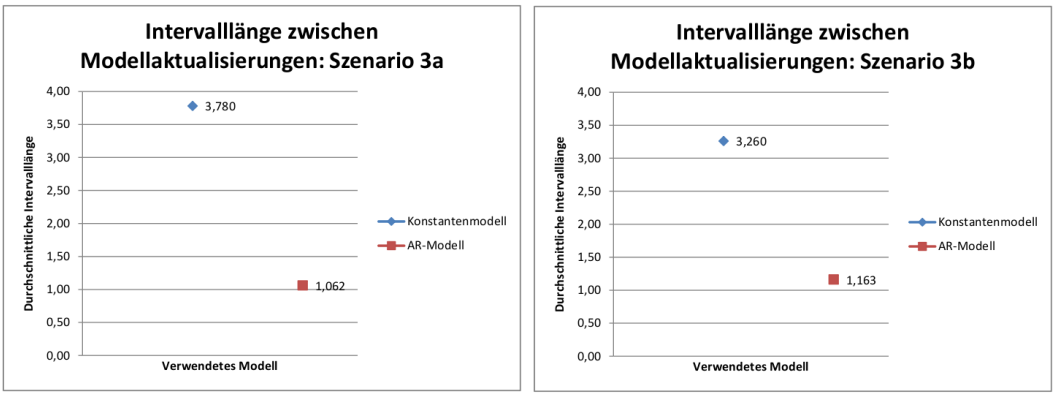

Abbildung 6.9: Das Intervall zwischen Modellaktualisierungen in Szenario 3a und 3b

steigt nicht mit zunehmender Laufzeit des Systems. Auf den Abbildungen 6.8 und 6.9 ist das durchschnittliche Intervall zwischen zwei Modellaktualisierungen für die beiden verwendeten Modelle jeweils pro Szenario zu sehen. Beide Modelle und weisen in den getesteten Szenarien ein ähnliches Verhalten auf. In den beiden Szenarien mit jeweils 1000 Anfragen kann sich das Konstantenmodell weiter vom AR-Modell absetzen, als dies in Szenarien mit weniger Teilnehmern der Fall ist.

6.3 Bewertung der Simulationsergebnisse

Die Auswertungen aus Experiment 6.2.1 zeigen klar: Das durchschnittliche Verhältnis zwischen angefragter und verwalteter Fläche ist mit einem Wert von $0,867$ sehr nahe am Bestwert von 1. Der durchschnittliche Verwaltungsmehraufwand für die Knoten hält sich also in akzeptablen Grenzen. Das regelmäßige Auftreten des Minimalwertes von $0,0185 = \frac{1}{54}$ lässt jedoch darauf schließen, dass einer der Gateways, die sich nach der Initialisierung in der Wurzel des Baumes befinden, dort auch während der gesamten Laufzeit des Systems bleibt. Davon abgesehen kann die Umverteilung der Last von den Gateways, die nur noch den Ausgangsgrad 1 besitzen, auf die Peers als fair bezeichnet werden. Die Leistung, die ein Knoten innerhalb des Systems erbringen muss, richtet sich nach dem Nutzen, den er aus dem System erhält.

Das Experiment 6.2.2 zeigt, dass trotz des in Kapitel 5.3.1 vorgestellten Verfahrens des kontrollierten Absinkens einer Anfrage zwischen 90 und 100 % aller neuen Anfragen bis zur Wurzel des Baumes aufsteigen müssen, um eingefügt zu werden. Das durchschnittliche Einfügelevel ist infolgedessen mit einem Wert von $0,217$ auch sehr nah an der Baumwurzel verzeichnet. Die Fairness bezüglich dieses Punktes ist also verbesserungswürdig. Eine gewisse Mehrbelastung der Knoten auf den oberen Levels des Baumes kann jedoch damit gerechtfertigt werden, dass in Experiment 6.2.1 nachgewiesen werden konnte, dass sich die

Platzierung eines Knotens nach seinem Nutzen aus dem Gesamtsystem richtet. Da in Ermangelung realer Anfragedaten auf zufallsgenerierte Anfragen zurückgegriffen werden musste, stellt dieses Ergebnis die Aufgabe, weitere Nachforschungen anzustellen. Bei Verwendung realer Anfragedaten besteht die Möglichkeit, dass Interessenten ähnliche Daten anfragen, wie die bereits am System beteiligten Knoten, an die sie ihre initiale Anfrage richten. Ist eine solche Lokalität der Ereignisse gegeben, kann das verwendete Verfahren des kontrollierten Aufsteigens vielleicht ausreichen, um die Wurzel nennenswert zu entlasten. Ändert sich die Situation jedoch auch mit echten Anfragedaten kaum, müssen neue Konzepte erarbeitet werden, um eine stärkere Entlastung der Wurzelknoten zu erreichen. Denkbar wäre z. B. die Verwendung einer festen Zahl von zufällig verteilten weitreichenden Kontakten (engl. long-range contacts) für jeden Knoten des Baumes, die eine Anfrage benutzen kann, falls der Verwaltungsbereich des damit erreichbaren Knoten näher an der Anfrage liegt, als der gerade betrachtete Knoten.

Der Anteil von Gateways aus der initialen Baumstruktur, die durch anfragende Peers ersetzt werden konnten, ist äußerst gering, wie das Experiment 6.2.3 zeigt. Der Unterschied zwischen Szenario 3a und Szenario 3b zeigt, dass durch eine starke Konzentrierung der Anfragen auf 3 Hot-Spots die Wahrscheinlichkeit größer ist, mit einer Anfrage genau den Verwaltungsbereich eines Gateways zu treffen. Da die Gateways bereits dadurch entlastet werden, dass die Zahl ihrer ausgehenden Verbindungen auf 1 reduziert werden konnte, kann der zusätzliche Verwaltungsaufwand als innerer Knoten des Baumes gerechtfertigt werden.

Das Experiment 6.2.4 hatte den Zweck, die Wirksamkeit des Prädiktorenkonzeptes zu überprüfen. Es konnte gezeigt werden, dass mit dem sehr einfachen Konstantenmodell durchschnittlich zwischen 27 und 70 % der übermittelten Datenmenge eingespart wird. Es ist also in jedem Fall sinnvoll, Prädiktoren in einem System zur Datenstromverarbeitung einzusetzen, um die Datenmenge zu reduzieren. Der Prozentsatz an eingesparten Daten bei Verwendung des AR-Modells bleibt nicht nur hinter den mit dem Konstantenmodell erzielten Erfolgen zurück, die tatsächlich übermittelte Datenmenge ist teilweise sogar wesentlich größer als ohne Vorhersagemodell. Die höhere Komplexität des Modells kann also nicht durch ebensoviel höhere Einsparungen gerechtfertigt werden. Stattdessen ist die Verwendung des AR-Modells in Szenarien mit wenigen Anfragen sogar kontraproduktiv. Möglicherweise wurden die verwendeten Sensordaten des Intel Research Labs mit einer zu hohen Aktualisierungsrate erzeugt, um das Potential des AR-Modells vollständig ausnutzen zu können. Obwohl die Ordnung des verwendeten AR-Modells mit 5 schon verhältnismäßig hoch gewählt wurde, war das Modell offensichtlich nicht in der Lage, die korrekten Tendenzen in den Datenverläufen abzulesen, da die Änderungen der in Abständen von einer halben Minute erzeugten Sensorwerte nur marginal waren. Dieser Umstand erklärt auch das gute Abschneiden des Konstantenmodells. Während in dem hier vorliegenden Szenario das Konstantenmodell die eindeutig bessere Wahl darstellt, kann noch mit anderen Datensätzen analysiert werden, ob das AR-Modell nicht auch effizient zur Datenreduktion eingesetzt werden kann.

Die Ergebnisse des Experiments 6.2.4 zeigen ebenfalls die Überlegenheit des Konstantenmodells in dem hier getesteten Szenario. Es ist besser in der Lage, Datenströme zur Echtzeitnut-

Experiment	Ergebnis
Durchschnittliches Flächenverhältnis	0,867
Durchschnittliches Einfügelevel	0,217
Prozentsatz der Einfügevorgänge über die Wurzel	96,29 %
Durchschnittlicher Prozentsatz der ersetzten Gateways	5,55 %
Durchschnittlicher Prozentsatz gesendeter Daten mit Konstantenmodell	48,14 %
Durchschnittlicher Prozentsatz gesendeter Daten mit AR-Modell	142,03 %
Konvergenz der Parameter Konstantenmodell	Nein
Konvergenz der Parameter AR-Modell	Nein

Tabelle 6.1: Zusammenfassung der Evaluationsergebnisse

zung zur Verfügung zu stellen, da die Zeiträume zwischen zwei Modellaktualisierungen zu einem Großteil wesentlich länger sind, als die des AR-Modells. Für keines der beiden getesteten Modelle kann eine Tendenz der Parameter entdeckt werden; die Vorhersagegenauigkeit der Verfahren nimmt also nicht mit fortschreitender Laufzeit zu. Die Verwendung des AR-Modells sorgt für eine wesentlich stärkere Rechenlast für die Knoten, als dies bei Verwendung des Konstantenmodells der Fall ist. Während eine Aktualisierungsnachricht für das Konstantenmodell lediglich die Ersetzung eines einzelnen Wertes zur Folge hat, muss für das AR-Modell ein lineares Gleichungssystem gelöst werden. Dieser zusätzliche Rechenaufwand bei einer gleichzeitig nicht niedrigeren Anzahl benötigter Aktualisierungen ist ein weiterer Grund, der gegen die Verwendung des AR-Modells spricht.

Weiterhin konnte gezeigt werden, dass das Gesamtsystem auch in der Lage ist, eine große Zahl von 100 neuen Anfragen pro Simulationsschritt zu verarbeiten. Die Skalierbarkeit konnte also auch in Bezug auf die Menge der anfragenden Interessenten bestätigt werden. Die qualitative Ähnlichkeit der Evaluationsergebnisse bezüglich der unterschiedlichen Szenarien zeigt, dass das getestete System bis zu einer Menge von 1000 Anfragen skaliert, ohne dass eine Verschlechterung des Verhaltens eintritt.

Die Tatsache, dass die meisten Experimente kaum Unterschiede zwischen Szenario 3a mit zufälliger Anfrageverteilung und Szenario 3b mit einer auf drei Hot-Spots konzentrierten Anfrageverteilung aufweisen, spricht ebenfalls für die Fähigkeit des Systems, mit einer stark konzentrierten Menge von Anfragen zurechtzukommen.

Die Ergebnisse der Evaluation sind nochmals in Tabelle 6.1 zusammengefasst.

Kapitel 7

Zusammenfassung und Ausblick

Die Trends im Bereich der globalen Sensornetze zeigen klar in eine Richtung: Die Informationsströme aus stetig wachsenden globalen Verbünden von Sensornetzen, die aus immer mehr Einzelsensoren bestehen, für eine immer größer werdende Masse von Konsumenten verfügbar machen. Durch die enorme Größenordnung und das globale Anwendungsszenario ergeben sich gerade in den beiden Bereichen Skalierbarkeit und Echtzeitverarbeitung Probleme, die von den bisher existierenden Lösungen zum Distributed Stream Processing nicht oder nur ungenügend gelöst werden.

Als Antwort auf diese Defizite wurde in dieser Arbeit ein System konzipiert, das es auch unter den obigen Bedingungen ermöglicht, eine Realzeitverarbeitung von Datenströmen in globalen Sensornetzen zu erreichen. Die Skalierbarkeit des Systems konnte gestärkt werden, indem der Flaschenhals bisheriger Stream Processing Systems, die Produzenten, wesentlich entlastet wurden. Die Echtzeitverarbeitung konnte durch die Integration von Prädiktoren zur Vorhersage von Datenverläufen gewährleistet werden.

Konkret wurde dafür ein Distributed Stream Processing System auf Basis eines verteilten R-Baumes konstruiert. Dieses System entlastet die als Produzenten auftretenden Gateways der Sensornetze, indem diese ihre Datenströme nicht wie bisher üblich an alle Interessenten verteilen, sondern die Interessenten selbst in den Verteilungsprozess einbezogen werden. Die Entlastung der Gateways kann erreicht werden, da die R-Baum-Struktur des entwickelten DSPS es ermöglicht, Gemeinsamkeiten in den an das System gestellten Anfragen schon während der Suche nach den zu einer Anfrage gehörigen Datenquellen zu erkennen. So wird von vornherein vermieden, dass mehrere Datenströme aufgebaut werden, die die gleichen Informationen übertragen. Mit der Kopplung von Datenquellensuche und Verhinderung der Mehrfachübermittlung gleicher Datenströme geht dieses System einen anderen Weg als die bisher in der Literatur vorgeschlagenen Lösungsansätze. Dort wurde mit Systemen wie SBON [PLS+06] vor allem versucht, DSPS durch eine flexible Operatorenplatzierung zu ergänzen und dadurch reaktiv Gemeinsamkeiten zwischen den angeforderten Datenströmen zu entdecken und auszunutzen.

Anschließend wurde das hier konzipierte DSPS mit einem dualen Vorhersagemodell ergänzt. Das Einsatzgebiet von Prädiktoren beschränkte sich in bisherigen Ansätzen nur auf die Kom-

munikation innerhalb einzelner Sensornetze. In dieser Arbeit werden Prädiktoren auf jeder Verbindung für die Vermittlung der Datenströme im gesamten DSPS eingesetzt. Durch den Einsatz von Prädiktoren konnte das Datenaufkommen reduziert und eine Verfügbarkeit der Datenströme in Echtzeit auch für Konsumenten mit einer hohen Latenz zu den Produzenten realisiert werden.

Die Effektivität der in dieser Arbeit vorgestellten Konzepte konnte in einer Evaluierung mit realen Sensordaten gezeigt werden. Durch die schon inhärent in der Struktur vorhandene Reduzierung der ausgehenden Verbindungen auf 1 (vgl. auch Kapitel 5.4.5 für die Eigenschaften des vorgestellten Verfahrens), konnte eine Entlastung der Produzenten und somit eine stark verbesserte Skalierbarkeit gewährleistet werden. Die Umverteilung der Last erfolgt nach fairen Kriterien, die sich am jeweiligen Nutzen, den die Teilnehmer aus dem Gesamtsystem ziehen, orientiert. Dies ist am ermittelten durchschnittlichen Verhältnis zwischen verwalteter und angefragter Fläche zu erkennen, das mit $0,867$ einen Wert nahe des Idealwertes 1 aufweist.

Weiterhin konnte gezeigt werden, dass die Integration von Prädiktoren in ein System zur Datenstromverarbeitung zu einer starken Reduktion der übermittelten Datenmenge führen kann. Auch die Verfügbarkeit von Echtzeitdatenströmen kann mit Prädiktoren erreicht werden. Allerdings hat die Wahl des Vorhersagemodells entscheidenden Einfluss auf die Effizienz der Prädiktoren. Während in dem hier evaluierten Szenario mit dem Konstantenmodell sehr gute Ergebnisse bezüglich der Reduktion der Datenmenge und dem Intervall zwischen zwei Modellaktualisierungen erzielt werden konnten, schneidet das AR-Modell in allen Kriterien wesentlich schlechter ab. Ob sich das Modell generell nicht für die Verwendung im Umfeld des Stream Processings eignet oder ob die Beschaffenheit der verwendeten Sensordaten das Konstantenmodell bevorteilte, konnte nicht abschließend geklärt werden.

Zusätzlich ergeben sich für das hier konzipierte System noch Fragen und Forschungsfelder, die in einer Arbeit dieses Umfangs nicht betrachtet werden konnten. Außerdem können manche im Systemmodell dieser Arbeit vorgenommenen Beschränkungen aufgehoben oder erweitert werden, um den hier vorgestellten Ansatz zu generalisieren.

Zum Abschluss folgt eine kurze Zusammenfassung ausgewählter Fragen und weiterführender Forschungspunkte. Die Fragestellung wird dabei jeweils kurz skizziert, um anschließend Möglichkeiten zur Modifikation des in dieser Arbeit vorgestellten Systems zu beschreiben.

Proaktive Aktualisierungen der Modellparameter Nicht analysiert werden konnte die Auswirkung eines Aktualisierungsschwellwertes δ, der kleiner als der von einer Anwendung spezifizierte tolerierte Fehler ϵ ist. Der Aktualisierungsschwellwert δ muss dabei in Abhängigkeit der Verzögerung zwischen Produzent und Konsument gewählt werden. So ist es möglich, eine Echtzeitverarbeitung eines Datenstroms zu erzielen, der sich bei gleichbleibenden Umweltbedingungen mit sehr hoher Wahrscheinlichkeit innerhalb eines garantierten Intervalls um den tatsächlichen Messwert befindet.

Erweiterungen des Systemmodells Das hier vorgestellte DSPS wurde exemplarisch mit Temperaturdaten evaluiert. In einem realen Anwendungsszenario sind nicht nur Sensordaten eines Typs, sondern z. B. zusätzlich zu Temperatur- auch noch Luftdruckdaten und Luftfeuchtigkeit sowie eventuell Windrichtung und -geschwindigkeit innerhalb des gesamten DSPS verfügbar. Diese Sensordaten können durch eine Erhöhung der Dimensionen in die verteilte R-Baum Struktur integriert werden. Die von den Knoten vertretenen Bereiche werden dann zu Hyper-Rechtecken erweitert, indem für jede neue Datenart eine Dimension hinzugefügt wird.

Ausfallsicherheit und Verfügbarkeit Da durch die Verwendung von Prädiktoren das Ausbleiben von Aktualisierungen bedeutet, dass das verwendete Vorhersagemodell korrekt arbeitet, muss ein zusätzliches Verfahren entwickelt werden, das Knotenausfälle erkennen kann. Dies ist ein wichtiger Punkt für die Zuverlässigkeit und somit auch die Praxistauglichkeit des Systems.

Eine konkrete Möglichkeit, eine solche Ausfallerkennung zu integrieren, wäre die regelmäßige Überwachung der Geschwisterknoten untereinander durch Heartbeat-Nachrichten. Stattet man die Knoten zusätzlich mit einer bestimmten Anzahl von weitreichenden Kontakten aus, könnte dies für eine schnelle Reparatur der Baumstruktur ausreichen. Diese weitreichenden Kontakte könnten gleichzeitig zu einer weiteren Entlastung der Knoten auf den oberen Levels der Baumstruktur beitragen.

Literaturverzeichnis

[AAB+05] Abadi, D.J. (Hrsg.) ; Ahmad, Y. (Hrsg.) ; Balazinska, M. (Hrsg.) ; Cetintemel, U. (Hrsg.) ; Cherniack, M. (Hrsg.) ; Hwang, J.H. (Hrsg.) ; Lindner, W. (Hrsg.) ; Maskey, A.S. (Hrsg.) ; Rasin, A. (Hrsg.) ; Ryvkina, E. (Hrsg.) u. a.: *The Design of the Borealis Stream Processing Engine*. 2005 (Zitiert auf den Seiten 18, 27 und 37)

[ACc+03] Abadi, Daniel J. ; Carney, Don ; Çetintemel, Ugur ; Cherniack, Mitch ; Convey, Christian ; Lee, Sangdon ; Stonebraker, Michael ; Tatbul, Nesime ; Zdonik, Stan: Aurora: a new model and architecture for data stream management. In: *The VLDB Journal* 12 (2003), Nr. 2, S. 120–139. `http://dx.doi.org/10.1007/s00778-003-0095-z`. – DOI 10.1007/s00778–003–0095–z. – ISSN 1066–8888 (Zitiert auf den Seiten 18 und 36)

[AHS06] Aberer, K. ; Hauswirth, M. ; Salehi, A.: The Global Sensor Networks middleware for efficient and exible deployment and interconnection of sensor networks / EFPL. 2006 (006). – Forschungsbericht (Zitiert auf Seite 36)

[ARE+05] Arora, Anish ; Ramnath, Rajiv ; Ertin, Emre ; Sinha, Prasun ; Bapat, Sandip ; Naik, Vinayak ; Kulathumani, Vinod ; Zhang, Hongwei ; Cao, Hui ; Sridharan, Mukundan ; Seddon, Nick ; Anderson, Chris ; Herman, Ted ; Trivedi, Nishank ; Zhang, Chen ; Shah, Romil ; Kulkarni, Sandeep ; Aramugam, Mahesh ; Wang, Limin: ExScal: Elements of an Extreme Scale Wireless Sensor Network, 2005, S. 102–108 (Zitiert auf Seite 14)

[Bay72] Bayer, Rudolf: Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms. In: *Acta Inf.* 1 (1972), 290-306. `http://dblp.uni-trier.de/db/journals/acta/acta1.html#Bayer72` (Zitiert auf Seite 21)

[BFG07] Bianchi, S. ; Felber, P. ; Gradinariu, M.: Content-Based Publish/Subscribe Using Distributed R-Trees. In: *LECTURE NOTES IN COMPUTER SCIENCE* 4641 (2007), S. 537–548 (Zitiert auf den Seiten 39 und 58)

[BJ90] Box, George Edward P. ; Jenkins, Gwilym: *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. – ISBN 0816211043 (Zitiert auf den Seiten 42, 73 und 80)

[BKSS90] Beckmann, Norbert ; Kriegel, Hans-Peter ; Schneider, Ralf ; Seeger, Bernhard: The R*-tree: an efficient and robust access method for points and rectangles. In: *SIGMOD Rec.* 19 (1990), Nr. 2, S. 322–331. http://dx.doi.org/10.1145/93605.98741. – DOI 10.1145/93605.98741. – ISSN 0163–5808 (Zitiert auf Seite 22)

[BM02] Bayer, Rudolf ; McCreight, E.: Organization and maintenance of large ordered indexes. (2002), S. 245–262. ISBN 3–540–43081–4 (Zitiert auf Seite 21)

[BPT02] Brakatsoulas, Sotiris ; Pfoser, Dieter ; Theodoridis, Yannis: Revisiting R-Tree Construction Principles. In: *ADBIS '02: Proceedings of the 6th East European Conference on Advances in Databases and Information Systems*. London, UK : Springer-Verlag, 2002. – ISBN 3–540–44138–7, S. 149–162 (Zitiert auf den Seiten 66, 68 und 69)

[BSB07] Borgne, Yann-Aël Le ; Santini, Silvia ; Bontempi, Gianluca: Adaptive model selection for time series prediction in wireless sensor networks. In: *Signal Process.* 87 (2007), Nr. 12, S. 3010–3020. http://dx.doi.org/10.1016/j.sigpro.2007.05.015. – DOI 10.1016/j.sigpro.2007.05.015. – ISSN 0165–1684 (Zitiert auf den Seiten 70 und 73)

[CBB+03] Cherniack, M. ; Balakrishnan, H. ; Balazinska, M. ; Carney, D. ; U. Cetintemel, S. Z. Y. Xing X. Y. Xing: Scalable Distributed Stream Processing CIDR'03, 2003 (Zitiert auf den Seiten 15, 18 und 36)

[CcC+02] Carney, Don ; Çetintemel, Uğur ; Cherniack, Mitch ; Convey, Christian ; Lee, Sangdon ; Seidman, Greg ; Stonebraker, Michael ; Tatbul, Nesime ; Zdonik, Stan: Monitoring streams: a new class of data management applications. In: *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, VLDB Endowment, 2002, S. 215–226 (Zitiert auf den Seiten 18 und 36)

[DF03] Demirbas, Murat ; Ferhatosmanoglu, Hakan: Peer-to-Peer Spatial Queries in Sensor Networks. In: *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*. Washington, DC, USA : IEEE Computer Society, 2003. – ISBN 0–7695–2023–5, S. 32 (Zitiert auf Seite 38)

[Gut84] Guttman, Antonin: R-trees: a dynamic index structure for spatial searching. In: *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 1984. – ISBN 0–89791–128–8, S. 47–57 (Zitiert auf Seite 21)

[HBR+03] Hwang, Jeong hyon ; Balazinska, Magdalena ; Rasin, Alexander ; Stonebraker, Michael ; Zdonik, Stan: A Comparison of Stream-Oriented High-Availability Algorithms / Brown CS. 2003. – Forschungsbericht (Zitiert auf Seite 27)

[HBR+05] Hwang, Jeong-Hyon ; Balazinska, Magdalena ; Rasin, Alexander ; Cetintemel, Ugur ; Stonebraker, Michael ; Zdonik, Stan: High-Availability Algorithms for Distributed Stream Processing. In: *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2285–8, S. 779–790 (Zitiert auf Seite 27)

[HCZ07] Hwang, Jeong-Hyon ; Cetintemel, U. ; Zdonik, S.: Fast and Reliable Stream Processing over Wide Area Networks. In: *Data Engineering Workshop, 2007 IEEE 23rd International Conference on* 1 (2007), April, S. 604–613. http://dx.doi.org/10.1109/ICDEW.2007.4401047. – DOI 10.1109/ICDEW.2007.4401047 (Zitiert auf Seite 18)

[HW79] Hartigan, J. A. ; Wong, M. A.: A K-Means Clustering Algorithm. In: *Applied Statistics* 28 (1979), Nr. 1, 100–108. http://www.jstor.org/stable/2346830. – ISSN 00359254 (Zitiert auf Seite 66)

[IA06] International Associations, Union of: *Yearbook of International Organizations: Statistics on international organizations*. http://www.uia.org/statistics/pub.php. Version: 11 2006 (Zitiert auf Seite 32)

[JCW04] Jain, Ankur ; Chang, Edward Y. ; Wang, Yuan-Fang: Adaptive stream resource management using Kalman Filters. In: *SIGMOD 04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 2004. – ISBN 1–58113–859–8, S. 11–22 (Zitiert auf den Seiten 28, 41 und 73)

[JMJV] Jelasity, Márk ; Montresor, Alberto ; Jesi, Gian P. ; Voulgaris, Spyros: *The Peersim Simulator*. – http://peersim.sf.net (Zitiert auf Seite 81)

[Kal60] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME – Journal of Basic Engineering* (1960), Nr. 82 (Series D), S. 35–45 (Zitiert auf Seite 41)

[KF94] Kamel, Ibrahim ; Faloutsos, Christos: Hilbert R-tree: An Improved R-tree using Fractals. In: *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994. – ISBN 1–55860–153–8, S. 500–509 (Zitiert auf Seite 22)

[LGS06] Li, Ming ; Ganesan, Deepak ; Shenoy, Prashant: PRESTO: feedback-driven data management in sensor networks. In: *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*. Berkeley, CA, USA : USENIX Association, 2006, S. 23–23 (Zitiert auf den Seiten 28 und 42)

[LLL05] Liu, Bin ; Lee, Wang-Chien ; Lee, Dik L.: Supporting Complex Multi-Dimensional Queries in P2P Systems. In: *Distributed Computing Systems* (2005), S. 155–164. http://dx.doi.org/10.1109/ICDCS.2005.75. – DOI 10.1109/ICDCS.2005.75 (Zitiert auf Seite 38)

[PLS+06] Pietzuch, Peter ; Ledlie, Jonathan ; Shneidman, Jeffrey ; Roussopoulos, Mema ; Welsh, Matt ; Seltzer, Margo: Network-Aware Operator Placement for Stream-Processing Systems. In: *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2570–9, S. 49 (Zitiert auf den Seiten 19, 27, 37 und 93)

[RFH+01] Ratnasamy, Sylvia ; Francis, Paul ; Handley, Mark ; Karp, Richard ; Schenker, Scott: A scalable content-addressable network. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM, 2001. – ISBN 1–58113–411–8, S. 161–172 (Zitiert auf Seite 38)

[San06] Santini, Silvia: An adaptive strategy for quality-based data reduction in wireless sensor networks. In: *in: Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS 06*, 2006, S. 29–36 (Zitiert auf den Seiten 28 und 41)

[ScZ05] Stonebraker, Michael ; Çetintemel, Uğur ; Zdonik, Stan: The 8 requirements of real-time stream processing. In: *SIGMOD Rec.* 34 (2005), Nr. 4, S. 42–47. http://dx.doi.org/http://doi.acm.org/10.1145/1107499.1107504. – DOI http://doi.acm.org/10.1145/1107499.1107504. – ISSN 0163–5808 (Zitiert auf Seite 16)

[SPL+04] Shneidman, J. ; Pietzuch, P. ; Ledlie, J. ; Roussopoulos, M. ; Seltzer, M. ; Welsh, M.: Hourglass: An Infrastructure for Connecting Sensor Networks and Applications / Harvard University. 2004 (TR-21-04). – Forschungsbericht (Zitiert auf Seite 35)

[SRF87] Sellis, Timos K. ; Roussopoulos, Nick ; Faloutsos, Christos: The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In: *VLDB '87: Proceedings of the 13th International Conference on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1987. – ISBN 0–934613–46–X, S. 507–518 (Zitiert auf Seite 22)

[TM06] Tulone, Daniela ; Madden, Samuel: PAQ: time series forecasting for approximate query answering in sensor networks. In: *In EWSN*, 2006, S. 21–37 (Zitiert auf den Seiten 28 und 43)

[TPS03] Tao, Yufei ; Papadias, Dimitris ; Sun, Jimeng: The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In: *In VLDB*, 2003, S. 790–801 (Zitiert auf Seite 40)

[XWL04] Xu, Y. ; Winter, J. ; Lee, W.-C.: Dual prediction-based reporting for object tracking sensor networks. In: *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on* (2004), Aug., S. 154–163. http://dx.doi.org/10.1109/MOBIQ.2004.1331722. – DOI 10.1109/MOBIQ.2004.1331722 (Zitiert auf Seite 70)

Alle URLs wurden zuletzt am 22.12.2008 geprüft.

Danksagung

Dank geht an Professor Rothermel für die Möglichkeit, diese Arbeit in der Abteilung Verteilte Systeme durchführen zu können. Besonders möchte ich Andreas Benzing und Boris Koldehofe danken, die mich mit guter Betreuung, dem richtigen Maß an Freiheiten, interessanten Diskussionen und wertvoller Kritik unterstützten und berieten. Für Korrekturen, Kritik und wichtige Anmerkungen geht Dank an Anja, Benny, Daniel und Janni. Danke auch an Lea für die Motivation zur richtigen Zeit. Nicht zuletzt möchte ich meiner Familie danken. Die jahrelange Unterstützung in jeglicher Hinsicht war und bleibt unbezahlbar. Mama, Papa und Tobi, ihr seid großartig.

Erklärung

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

(Marco Völz)