Distributed Stream Processing in a Global Sensor Grid for Scientific Simulations

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
und dem Stuttgart Research Centre for Simulation Technology
der Universität Stuttgart zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Andreas Martin Georg Benzing

aus Stuttgart – Bad Cannstatt

Hauptberichter:	Prof. Dr. rer. nat. Kurt Rothermel
Mitberichter:	Prof. Dr. Frank Leymann
	Prof. Dr. Pedro José Marrón

Tag der mündlichen Prüfung: 19.10.2015

Institut für Parallele und Verteilte Systeme

Universität Stuttgart

2015

Danksagungen

Ich danke allen Menschen, die mich während meines Promotionsvorhabens begleitet und unterstützt haben.

Zunächst gilt mein Dank meinem Doktorvater Prof. Dr. Kurt Rothermel, für zahlreiche wertvolle Hinweise, konstruktive Diskussionen und die Möglichkeit, meine Forschungen am Lehrstuhl Verteilte Systeme durchführen zu können. Prof. Dr. Frank Leymann, meinem Mitbetreuer, gilt mein Dank für seine hilfreichen Kommentare. Prof. Dr. Pedro José Marrón danke ich für sein Interesse an meiner Arbeit als Gutachter. Für die Betreuung während meines Aufenthalts in Toronto danke ich Prof. Dr. Hans-Arno Jacobsen und den Mitarbeitern der Middleware Systems Research Group.

Allen meinen Kollegen am Lehrstuhl danke ich für eine sehr angenehme Arbeitsatmosphäre. Dr. Boris Koldehofe stand mir immer als Ansprechpartner zur Verfügung und hatte bei jedem Problem ein offenes Ohr. Explizit nennen möchte ich auch meine Begleiter seit Studienbeginn Beate Ottenwälder und Björn Schilling. Dr. Marcus Reble, Dr. Jan Hasenauer und ihre Kollegen vom Institut für Systemtheorie und Regelungstechnik zeigten mir mit ihrer Sicht auf die Welt neue Perspektiven. Dr. Tudor Ionescu vom Institut für Kernenergetik und Energiesysteme danke ich für die konstruktive Zusammenarbeit. Darüber hinaus hat mich die Vielfalt an Themen im ganzen Exzellenzcluster SimTech immer inspiriert und motiviert, wofür ich ebenfalls allen Beteiligten danken möchte.

Genauso wichtig wie das fachliche Umfeld war für mich immer das persönliche: meine Eltern Margot Benzing und Martin Benzing sowie Michael Müller, meine Geschwister Julia Klenk und Marc Benzing, meine Freundin Dani Hildebrand und meine Neffen Felix und Fabian. Herzlichsten Dank für eure Unterstützung, eure Geduld und eure Liebe.

Vielen Dank an Bernd Oczko und Katrin Hagelstein für das Korrekturlesen der Arbeit. Schließlich möchte ich Christian Heigele, Dr. Stefanie Wöhrle, Marco Völz, Ulrich Gösele und allen anderen Freunden für viele anregende und motivierende Gespräche danken.

Contents

List of Acronyms

ALM

application layer multicast.

API

application programming interface.

CDN

content distribution network.

DSP

distributed stream processing.

EXC

Cluster of Excellence.

GPS

global positioning system.

GSG

Global Sensor Grid.

GSGM

Global Sensor Grid Middleware.

GSN

global sensor network.

ISP

Internet service provider.

LMS

least mean squares.

MBR

minimum bounding rectangle.

MST

minimum Steiner tree problem.

SARIMA

seasonal autoregressive integrated moving average.

SDN

software defined networking.

SN

sensor network.

SRC

Stuttgart Research Center.

WSN

wireless sensor network.

Glossary of Terms

broker

A dedicated server which executes an instance of the GSGM. 63

diagnostic simulation

A diagnostic simulation continuously integrates measurements and runs in real-time to create a digital representation of the physical world. 57, 71, 110

direct sensor query

A query for directly monitoring measurements of a certain geographic location and sensor type. 65

full sensor stream

A continuous stream containing a regular grid of data points equally distributed over a geographic region in each update. 63, 64, 72, 110, 111

relay broker

A broker which participates in distributing full sensor streams. 63, 143, 150

simulation query

A query for requesting a full sensor stream from a diagnostic simulation which preprocesses measurements. 66, 68, 109

source broker

A broker which generates full sensor streams from sensor measurements. 63, 64, 147

target broker

A broker which delivers full sensor streams to simulation clients. 63, 143, 147

Abstract

With today's large number of sensors available all around the globe, an enormous amount of measurements has become available for integration into applications. Especially scientific simulations of environmental phenomena can greatly benefit from detailed information about the physical world. The problem with integrating data from sensors to simulations is to automate the monitoring of geographical regions for interesting data and the provision of continuous data streams from identified regions.

Current simulation setups use hard coded information about sensors or even manual data transfer using external memory to bring data from sensors to simulations. This solution is very robust, but adding new sensors to a simulation requires manual setup of the sensor interaction and changing the source code of the simulation, therefore incurring extremely high cost. Manual transmission allows an operator to drop obvious outliers but prohibits real-time operation due to the long delay between measurement and simulation. For more generic applications that operate on sensor data, these problems have been partially solved by approaches that decouple the sensing from the application, thereby allowing for the automation of the sensing process. However, these solutions focus on small scale wireless sensor networks rather than the global scale and therefore optimize for the lifetime of these networks instead of providing high-resolution data streams.

In order to provide sensor data for scientific simulations, two tasks are required: i) continuous monitoring of sensors to trigger simulations and ii) high-resolution measurement streams of the simulated area during the simulation. Since a simulation is not aware of the deployed sensors, the sensing interface must work without an explicit specification of individual sensors. Instead, the interface must work only on the geographical region, sensor type, and the resolution used by the simulation. The challenges in these tasks are to efficiently identify relevant sensors from the large number of sources around the globe, to detect when the current measurements are of relevance, and to scale data stream distribution to a potentially large number of simulations. Furthermore, the process must adapt to complex network structures and dynamic network conditions as found in the Internet.

The Global Sensor Grid (GSG) presented in this thesis attempts to close this gap by approaching three core problems: First, a distributed aggregation scheme has been developed which allows for the monitoring of geographic areas for sensor data of interest. The reuse of partial aggregates thereby ensures highly efficient operation and alleviates the sensor sources from individually providing numerous clients with measurements. Second, the distribution of data streams at different resolutions is achieved by using a network of brokers which preprocess raw measurements to provide the requested data. The load of high-resolution streams is thereby spread across all brokers in the GSG to achieve scalability. Third, the network usage is actively minimized by adapting to the structure of the underlying network. This optimization enables the reduction of redundant data transfers on physical links and a dynamic modification of the data streams to react to changing load situations.

Deutsche Zusammenfassung

Verteilte Datenstromverarbeitung in einem Global Sensor Grid für wissenschaftliche Simulationen

1 Einleitung

Die genaue Erfassung ihrer Umwelt fasziniert Menschen bereits seit langer Zeit. Diese Faszination verdeutlicht sich vor allem in der systematischen Aufzeichnung von Wetterdaten über die letzten Jahrhunderte. Mit dem Beginn des Informationszeitalters rückte die automatisierte Verarbeitung von Sensordaten in den Vordergrund, jedoch bedeutet eine manuelle flächendeckende Installation von Messstationen in entfernten und schwer zugänglichen Gebieten, sofern überhaupt möglich, erhebliche Kosten. Drahtlose und drahtgebundene Sensornetze bieten hier eine vielversprechende Option und standen daher in den letzten zwei Jahrzehnten zunehmend im Fokus der Forschung [ASSC02a]. Dabei werden kleine und kostengünstige Sensorknoten mit begrenzten eigenständigen Ressourcen [DAW+08; DGMS07] so programmiert, dass ein selbstorganisierendes Netzwerk entsteht. Solche Sensornetze können dann Daten von allen integrierten Sensorknoten bereitstellen.

Mittlerweile werden Messdaten zunehmend mit Hilfe von komplexen Simulationsmodellen zur Analyse [CHPP09] und Echtzeitvorhersage [IPS+09] genutzt. Der aktuelle Trend geht dabei zu sogenannten datengetriebenen Ansätzen, bei denen die Messdaten im laufenden Betrieb in die Simulation integriert werden können. Bei solchen Anwendungen ist eine schnelle Bereitstellung der Messungen notwendig, um eine zeitnahe Reaktion bzw. einen erweiterten Vorhersagehorizont zu ermöglichen. Durch die begrenzten Energiereserven von Sensorknoten ist allerdings eine Abwägung zwischen der Betriebsdauer eines Sensornetzwerks und dessen Reaktionsfähigkeit notwendig. Existierende Lösungsansätze [DGV04; LHRM09] behandeln jedes Sensornetz als eigenständiges System, wodurch eine großflächige Simulation die Interaktion mit mehreren Sensornetzen erfordert. Eine

weitere Vergrößerung der Sensornetze zur Vermeidung dieses Problems führt dabei zu zusätzlichen Schwierigkeiten [ARE+05]. Obwohl neue Ansätze für hierarchische Algorithmen [GGE+05; IS09b] und verteilte Indexstrukturen erarbeitet wurden, bleibt die Größe von Sensornetzen im Moment limitiert.

Im Hinblick auf eine globale Sensorinfrastruktur wurden neue Methoden hauptsächlich in zwei Gebieten entwickelt: Zum einen sind dies Ansätze zur Kopplung isolierter Sensornetze als Folge von deren Größenbeschränkung, um eine globale Abdeckung zu erreichen. Diese Ansätze konzentrieren sich auf die Entwicklung eines Sensor-Web, welches auf etablierten Web-Technologien [GKK+03] und Grid-Computing [FHW+08] beruht, sowie auf Middleware-Lösungen [AHS06], welche die Installation von virtuellen Sensoren vereinfachen. Zum anderen wurden, auch im Zuge der allgemein gestiegenen Informationsflut, neue Methoden zur Verarbeitung extrem großer Datenmengen vorgestellt. Dabei sind besonders Systeme zur verteilten Verarbeitung von Datenströmen [ACÇ+03] zu beachten.

Insgesamt fehlen für eine funktionierende globale Sensorinfrastruktur wichtige Teile, besonders im Hinblick auf die Bereitstellung von Sensordaten für wissenschaftliche Simulationen. Zunächst fehlt die Möglichkeit einer hocheffizienten Überwachung von großen Regionen, welche möglicherweise diverse Sensornetze überspannen und eine Vielzahl von Sensoren beinhalten. Für die nahtlose Integration von Sensordaten in Simulationen fehlt darüber hinaus eine Anfrageschnittstelle, die es ermöglicht, hochauflösende Sensordatenströme anzufragen. Das zugrundeliegende System muss folglich große Spitzenlasten bewältigen, welche durch diverse Überwachungsanfragen parallel ausgelöst werden können. Um die Skalierbarkeit, trotz stark schwankender Netzwerkkapazitäten, zu gewährleisten und eine Überlastung des Netzes zu verhindern, muss das System außerdem dynamisch auf Veränderungen im Netzwerk reagieren und die Netzwerklast, entsprechend der genutzten physikalischen Netzwerkstruktur, minimieren können.

Diese Arbeit leistet daher die folgenden fünf Beiträge zu einem Global Sensor Grid (GSG), welche in der Global Sensor Grid Middleware (GSGM) integriert sind:

- Identifikation von Anforderungen an eine globale Anfrageschnittstelle für aggregierte Sensordaten zur Überwachung beliebiger Regionen.

- Hocheffiziente Echtzeitdatenerfassung aus Sensornetzen mittels verteilter Aggregation von Messwerten mit integrierter Prädiktion.

- Identifikation von Anforderungen an eine Anfrageschnittstelle für hochauflösende Echtzeitsensordatenströme.

- Skalierbare Bereitstellung von hochauflösenden Datenströmen durch gerichtete Lastverteilung.
- Minimierung der gesamten Netzwerklast durch gezielte Anpassung an das zugrundeliegende physikalische Netzwerk und dessen aktuelle Auslastung.

2 Hintergrund

Kapitel 2 umfasst Hintergrundinformationen zu den drei Kernthemen dieser Arbeit, welche nach dem Top-Down-Prinzip abgehandelt werden: Von den Workflows wissenschaftlicher Simulationen über die verteilte Datenstromverarbeitung zu drahtgebundenen und drahtlosen Sensornetzen. Darüber hinaus beinhaltet das Kapitel einen kurzen Überblick über das „Stuttgart Research Center (SRC) for Simulation Technology“ bzw. den Exzellenzcluster „SimTech“.

Die Workflow-Technologie wurde ursprünglich im Kontext von Geschäfts- und Produktionsprozessen entwickelt, um wiederkehrende Aufgabenmuster zu organisieren und zu automatisieren [LR99]. In letzter Zeit wurden Workflows zunehmend für die Ausführung von komplexen Simulationen in der Forschung benutzt, um die benötigten Tools und Programme bei der Datenakquise und -vorverarbeitung, der eigentlichen Simulation, sowie bei der Nachbearbeitung von Ergebnissen und deren Visualisierung zu orchestrieren. Dadurch ist eine neue Art von Workflows für wissenschaftliche Simulationen entstanden, die die zunehmende Komplexität der benutzten Werkzeugketten berücksichtigt [ABJ+04; BH08; GSK+11; MSTW04]. Während die grundlegenden Mechanismen bei der Workflowverarbeitung für Simulationen ähnlich denen von klassischen Workflows sind, stellen die großen Datenmengen, die in heutigen Simulationen anfallen, grundlegend neue Anforderungen [GDE+07] an die genutzten Webservices und die Integration mit diesen.

Webservices kapseln die einzelnen Werkzeuge bzw. Komponenten, die für einen kompletten Workflow benutzt werden, ab und stellen diese über eine standardisierte Schnittstelle bereit. Sie können daher auch in einem verteilten System [DSS+05; LMJM06], Grid [CAWK08; DBG+04] oder in einer Cloud [AIG12; WHF+13] ausgeführt werden. Darüber hinaus werden für die Nachweisbarkeit des Ursprungs [MSTW04] und der Qualität [RBKK12] von Ergebnissen spezielle Methoden für die systematische Überwachung und Speicherung von Ausführungsparametern der Webservices von wissenschaftlichen Workflowsystemen unterstützt. Für die Integration von externen Datenquellen mittels Webservices fehlt bisher jedoch eine ausreichende Unterstützung, weshalb aktuell dar-

an geforscht wird [AEM13; CAWK08; GMW+04]. Speziell strombasierte Datenquellen erfordern eine grundlegende Anpassung der Interaktion zwischen Webservice und Workflowsystem.

Datenströme werden in verteilten Anwendungen verarbeitet, neben hochauflösenden Sensordaten zum Beispiel in der hochfrequenten Ereignisverarbeitung oder bei der Bereitstellung von multimedialen Live-Daten. In diesen Anwendungsszenarien hat sich die verteilte Datenstromverarbeitung [CBB+03; CCD+03; CM12] als Hauptmethode zur Bewältigung von großen Mengen kontinuierlich produzierter Daten etabliert. Eine der größten Herausforderungen bei der verteilten Datenstromverarbeitung ist die Einhaltung einer gegebenen Servicequalität in Bezug auf die maximale Verzögerung vom Dateneingang bis zur Verfügbarkeit des Ergebnisses. Um diese Verzögerung während Lastspitzen gering zu halten, zeichnen sich viele Systeme durch Verfahren zum Lastabwurf [AJS+06] aus.

Da Simulationen jedoch einen durchgehenden Strom von Daten benötigen, stehen für diese Arbeit die Optimierungsansätze in der Verteilung von Datenströmen gegenüber denen bei der Verarbeitung im Vordergrund. Hierbei stehen zwei Themen im Fokus: Die simultane Zustellung von Datenpaketen an mehrere Empfänger (Multicast) über das Netzwerk auf der Anwendungsschicht und der Gebrauch von spezialisierten Netzwerken zur Verbreitung von Inhalten. Multicast auf Anwendungsebene [CRSZ02; HASG07; KS10; YLE04] wurde entwickelt, da Multicast auf IP-Ebene, aufgrund von technischen und administrativen Hindernissen, bis heute nicht flächendeckend eingeführt werden konnte [DLL+00]. Die Schwierigkeit bei der Optimierung liegt darin, passende Verteilstrukturen innerhalb der komplexen Struktur des Internets zu finden und eine Überlastung von einzelnen Anwenderknoten zu verhindern. Dedizierte Netzwerke zur Verbreitung von Inhalten wurden dagegen in den letzten Jahren zunehmend errichtet, um Daten an verschiedenen Rechenzentren bereit zu halten. Neue Benutzeranfragen können so an Server zugewiesen werden, dass die Last über die Server verteilt und gleichzeitig der genutzte Netzwerkpfad möglichst kurz wird. Im Gegensatz zum Multicast auf Anwendungsebene liegt die Schwierigkeit hier bei der Auswahl der Position der Server im Netzwerk [RGE02] sowie der Auswahl der zu replizierenden Inhalte [LZS04]. Neben der Verteilung von Daten ist die Platzierung von Operatoren im Netzwerk ebenfalls ein integraler Bestandteil der verteilten Datenstromverarbeitung und ein eigenständiges Forschungsfeld. Die verschiedenen Aspekte und Optimierungsziele werden daher in Bezug auf die vorliegende Arbeit detailliert erläutert.

Im weiteren Verlauf des Kapitels wird auf die Sensornetze [ASSC02a; ASSC02b] eingegangen, welche die Rohdaten bereitstellen. Ein Großteil der Forschungsarbeit in

diesem Bereich konzentrierte sich bisher auf die Optimierung von drahtlosen Sensornetzen, welche hauptsächlich durch deren beschränkte Energieversorgung [LMMR07] geprägt sind. Spezielle Betriebssysteme [DGV04; LMP+05] und Frameworks zur Datenverwaltung [MLM+05] wurden zu diesem Zweck entwickelt. Diese systemweiten Ansätze wurden durch eine Vielzahl spezieller Methoden und Protokolle erweitert. Als größter Energieverbraucher lag dabei ein starker Fokus auf der Reduktion der Kommunikation zwischen Sensorknoten. Indem Kommunikation durch zusätzliche Berechnungen ersetzt wird kann der Energieverbrauch um Größenordnungen verringert werden [PK00]. Ein Beispiel für einen solchen Austausch ist die modellbasierte Erfassung von Sensordaten [DGM+04; LCLC04; MFHH03], welche Messungen nur bei ungenauem Modellzustand überträgt. Solche Modelle zur Vorhersage von Messewerten werden auch als Prädiktoren bezeichnet und können neben der Datenreduktion auch zur Erstellung von Schätzungen in Echtzeit verwandt werden. Die ersten Prädiktoren wurden auf Basis von Videokompressionsverfahren vorgeschlagen [GI01] und später mit verschiedenen Modellen aus der Datenreihenanalyse [DGL+05; LGS06; TM06] sowie Regelungstechnik [WB06] erweitert. Allerdings haben sich einfache Modelle [LLG13; RCM+12; SR06] in jüngster Zeit als vergleichbar effektiv erwiesen. Durch Verfahren zur Wegbestimmung [CM09; KDHS10; KK00; SML+06] kann ebenfalls dazu beigetragen werden, den Kommunikationsaufwand zu minimieren.

Um die Sensordaten einem bestimmten Ursprungsort zuordnen zu können, wurden zudem einige Verfahren zur energieeffizienten Positionierung von Sensorknoten erarbeitet [CPH06; HHB+03; LR03; PAK+05; SM10]. Neben dem Energieverbrauch stellt die Verwaltung der gesamten Sensorinformation in einem Netzwerk eine komplexe Aufgabe dar. Der billige und dadurch immer größere Speicher [LMG+07] auf einzelnen Knoten hat dabei zur Entwicklung von neuen Architekturen geführt [DGMS07; GEH03; GJP+06]. Bei diesen speicherorientierten Architekturen stellt die gezielte Auffindung von bestimmten Messwerten [DF03; IS09a; RKY+02; WL09] eine neue Herausforderung dar. Diese Herausforderung stellt sich konsequenterweise auch für globale Sensorsysteme [GKN+07; NLZ06] mit Echtzeitdaten [LKNZ08].

Zum Abschluss wird noch ein kurzer Überblick über das „SRC for Simulation Technology“ bzw. den Exzellenzcluster „SimTech“ [SimTech] gegeben. Die gesamte Struktur des Forschungszentrums wird dargelegt und dessen fächer- und themenübergreifende Ziele vorgestellt. Die Einordnung des Global Sensor Grid Projekts in die Forschungsnetze und -bereiche von SimTech erlaubt eine Identifikation der angrenzenden Forschungsgebiete. Eine Vorstellung der Projektpartner schließt die Übersicht ab.

3 Systemübersicht

Die Systemübersicht leitet zunächst die Anfrageabstraktionen her, die innerhalb des GSG sowie für die Bereitstellung von Daten an Simulationen gebraucht werden. Bisher wurde für die Interaktion zwischen Simulationen und Sensoren entweder eine manuelle Datenübertragung oder eine hart kodierte Verbindung benutzt. Obwohl diese Art der Datenakquise sehr robust ist und für eine geringe Anzahl an Sensoren überschaubaren Aufwand bedeutet, braucht ein System wie das GSG eine Abstraktionsschicht, welche Anwendungen die flexible Spezifikation von gewünschten Sensordaten erlaubt. Um die Regionen für eine genauere Anfrage auszuwählen, wird außerdem eine Schnittstelle benötigt, die die effiziente Überwachung von unverarbeiteten Messwerten erlaubt.

Die beiden wichtigsten Eigenschaften bei der Auswahl von Sensoren für eine Anfrage sind deren geographische Position und der Sensortyp. Während der Sensortyp sich durch einen einfachen Vergleich filtern lässt, sind für die Position verschiedene Spezifikationen denkbar. Um eine effiziente Anfrageverarbeitung zu ermöglichen, wurde daher für das GSG eine rechteckige Form für die Anfrageregionen gewählt, welche durch deren Eckpunkte in Längen- und Breitengrad spezifiziert wird. Um mögliche Mehrdeutigkeiten zu beseitigen, werden für die direkte Überwachung von Sensoren alle verfügbaren Daten in der Region aggregiert.

Im Gegensatz zu direkten Anfragen an die Sensoren benötigen Simulationen ein Gitter von Datenpunkten und somit, wie bereits erwähnt, eine weitere Schnittstelle. Die Einschränkung auf ein lückenloses Gitter von Messpunkten liegt in der Struktur von Simulationen begründet, welche verhindert, dass isolierte Messwerte von einzelnen Sensoren ohne Probleme integriert werden können. Da sich verschiedene Simulationen nicht nur in der simulierten geographischen Region und den zugrundeliegenden Sensortypen unterscheiden, werden weitere Informationen für die Schnittstelle benötigt. Zum einen muss die Auflösung von Datenpunkten innerhalb der Region spezifiziert werden, zum anderen muss die Aktualisierungsrate der Daten bekannt sein. Die Schnittstelle für Anfragen von Simulationen beinhaltet daher neben den Sensortypen und der Region ebenfalls die räumliche und zeitliche Auflösung für den gewünschten Datenstrom.

Der zweite Teil des Kapitels behandelt den Fluss der Datenverarbeitung, von der Erfassung von rohen Messdaten über die Vorverarbeitung mittels diagnostischen Simulationen bis zur eigentlichen anfragenden Simulation. Wie bereits erwähnt nutzt das GSG die bestehenden Funktionen von Sensornetzen zur Akquise von Sensordaten, weshalb die Verarbeitung innerhalb dieser Netze nicht betrachtet wird. Die verfügbaren Daten werden

dann auf leistungsstarken sogenannten Brokern gesammelt, wobei die rohen Messwerte bereits vorverabeitet werden.

In einem zweiten Schritt werden die gesammelten Daten über eine zusätzliche Anpassung in diagnostische Simulationen eingepflegt. Diagnostische Simulationen stellen eine spezielle Art von Simulationen dar, welche die reale Welt in Echtzeit darstellen und daher die direkte Integration von Messdaten ermöglichen. Ein Beispiel, welches im Rahmen der Kooperation in SimTech betrachtet wurde, ist die Berechnung von Windfeldern [RSMF88; Sea00; WWG+05] aus Daten von Wetterstationen, welche im Simulationsmodell mit topographischen Daten kombiniert werden. Neben der Berechnung von Windrichtung und -geschwindigkeit im Beispiel für beliebige Orte, erlauben diagnostische Simulationen generell den Zugang zu Daten, welche nicht direkt zugänglich sind. Dazu gehören neben Daten für Orte, an denen keine direkten Messungen verfügbar sind, auch Messgrößen, für die keine oder nur ungenaue Messverfahren bekannt sind. Der interne Zustand der diagnostischen Simulation kann dann extrahiert und als hochauflösender Datenstrom zur Beantwortung von Anfragen verwandt werden.

Der wichtigste Schritt im Hinblick auf diese Arbeit ist die Verteilung der resultierenden hochauflösenden Datenströme in Echtzeit über das Netzwerk. Zu diesem Zweck müssen zunächst die Anfragen an den Quellbroker geleitet werden, der die diagnostische Simulation mit den angefragten Daten ausführt. Um eine schnelle Auslieferung zu erreichen, werden die Broker über eine flache Struktur organisiert. Entsprechend werden die Anfragen dann direkt an den zuständigen Quellbroker weitergeleitet. Der Quellbroker verschickt daraufhin für jede Anfrage einen gesonderten Datenstrom, was wiederum zur Überlastung eines einzelnen Brokers durch eine hohe Anzahl an datenintensiven Anfragen führen kann. Um dieses Problem anzugehen, werden in dieser Arbeit neue Methoden vorgestellt, um die Ressourcen von allen Brokern des GSG zu nutzen und eine gerichtete Lastverteilung zu erreichen. Dabei werden mehrere sich überschneidende Anfragen durch einen einzelnen Datenstrom an einen anderen Broker ausgelagert, um dessen verfügbare Bandbreite mit zu nutzen. Zusätzlich wird die Skalierbarkeit des Gesamtsystems durch die Minimierung der Netzwerklast verfolgt. Diese Minimierung wird erreicht indem die Datenverteilung an die physikalische Netzwerkstruktur und die aktuelle Netzauslastung angepasst wird.

4 Echtzeitdatenerfassung

In Kapitel 4 wird beschrieben, wie Daten effizient von einer Vielzahl von Sensoren verschiedenen Typs in großen geographischen Regionen erfasst werden können. Zunächst

erfolgt eine detaillierte technische Beschreibung der verteilten Aggregation von Sensordaten, wie sie im GSG verwendet wird. Um die Übertragungseffizienz weiter zu steigern und Echtzeitschätzungen der aktuellen Messwerte zu generieren, wird der Ansatz zur Prädiktion in Multi-Hop-Umgebungen wie unserer verteilten Struktur erläutert. Eine Evaluation der vorgestellten Verfahren zeigt deren Leistungseigenschaften basierend auf realen Sensordaten. Abschließend werden verwandte Arbeiten vorgestellt und das Kapitel nochmals zusammengefasst.

Die Grundlage für die verteilte Aggregation bilden sogenannte Broker, welche jeweils einem oder mehreren Sensornetzgateways zugeordnet sind, die wiederum die einzelnen Sensoren koordinieren. Dabei werden Anfragen, welche an das System gestellt werden, als Datenquellen mit eingebunden. Auf diese Art wird die Last für die Überwachungsanfragen verringert, da die jeweiligen Teilaggregate mehrfach verwandt werden können, was einen effizienten Betrieb ermöglicht. Die Anfrageschnittstelle abstrahiert die Interaktion mit einzelnen Sensoren und bietet die Möglichkeit, eine geographische Region zu spezifizieren, über die kontinuierlich Sensordaten aggregiert werden sollen.

Um die Daten von einer bestimmten Region aggregieren zu können, wird zunächst auf die Indexierung von Anfragen und das Auffinden von Sensoren eingegangen. Als Basis für die verteilte Indexierung dient der R-Baum [Gut84], der häufig im Datenbankbereich für punktförmige und rechteckige Datensätze verwendet wird. Die Verteilung erfolgt dabei über die Broker im System, wobei jedem Broker ein Rechteck als Blatt zugewiesen wird, welches der geographischen Region entspricht, für die er Daten verwaltet. Die Baumstruktur ergibt sich dann dadurch, dass Rechtecke über größere Regionen, welche mehrere Broker umfassen, als Elternknoten generiert werden, die zusätzlich von bestimmten Brokern verwaltet werden. Dieser Schritt wird wiederholt bis schließlich ein Wurzelknoten, der die ganze Welt umfasst, entsteht. Neue Anfragen werden entsprechend der spezifizierten Region über die Indexstruktur zwischen den zuständigen Brokern weitergeleitet. Wenn die Region komplett oder teilweise außerhalb aller verwalteten Regionen liegt, wird die Anfrage weiter zur Wurzel geleitet. Danach folgt eine Weiterleitung zu Kindknoten, solange die Anfrage komplett in der Region des Kindknoten enthalten ist. Abschließend wird die Anfrage in den Indexierungsbaum eingefügt, um als virtuelle Quelle für weitere Anfragen zu dienen. Während die Anfragen von der Wurzel her nach unten gereicht werden, wird der Index ebenfalls genutzt, um den Weg der Daten von den Sensoren zur Wurzel zu bestimmen. Da die Struktur so aufgebaut ist, dass jede Anfrage genau die Kindknoten besitzt, die einen entsprechenden Teilbereich angefragt haben, können die Daten entlang der Verbindungen aggregiert und dabei bereits berechnete

Aggregate wiederverwendet werden. Neben der hohen Effizienz bietet dieser Aufbau den Vorteil, dass Sensordaten von der Quelle nur einmal geliefert werden müssen und von dort weitergeleitet werden. Dadurch wird eine Überlastung von einzelnen Sensoren oder Gateways vermieden und deren Ressourcen geschont.

Die verteilte Aggregation mit Wiederverwendung von Teilaggregaten erfordert die Weiterleitung über mehrere Netzwerkknoten, was wiederum längere Kommunikationswege und damit eine erhöhte Verzögerung nach sich zieht. Um diesem negativen Effekt entgegenzuwirken und Echtzeitschätzungen bereitstellen zu können, werden Prädiktoren in die gesamte Aggregationsstruktur integriert. Neben Echtzeitschätzungen bieten Prädiktoren außerdem die Möglichkeit, die Datenübertragung signifikant zu reduzieren, wodurch das Datenaufkommen zusätzlich zur Aggregation weiter minimiert wird. Zunächst wird die Datenreduktion über einen Kommunikationsschritt beschrieben, welche die Grundlage für die Anwendung in einer Multi-Hop-Umgebung ist. Generell basiert die Reduktion darauf, dass sowohl Quelle als auch Ziel eine Vorhersage berechnen und, solange die Vorhersage hinreichend genau ist, keine Daten übertragen werden. Um ein Verständnis der Funktionsweise zu ermöglichen, werden zunächst verschiedene Prädiktionsmodelle unterschiedlicher Komplexität vorgestellt, wobei als Repräsentant für einfache Modelle der „least mean squares (LMS)“ Algorithmus [SR06] und für komplexe Modelle sogenannte „seasonal autoregressive integrated moving average (SARIMA)“ Modelle [LGS06] herangezogen werden. Beide Prädiktionsverfahren werden detailliert beschrieben, sowie deren Vor- und Nachteile für die Anwendung in dem zuvor beschriebenen System erörtert. Während einfache Modelle geringe Speicher- und Prozessoranforderungen haben, können komplexe Modelle unter Umständen höhere Trefferquoten bzw. Genauigkeit bei der Vorhersage erreichen. Abhängig davon wo die Berechnungen der Modellparameter und Vorhersagen stattfinden, werden außer den Modellen verschiedene Betriebsmodi unterschieden. Je nach Betriebsmodus werden Messwerte, Parameter oder eine Mischung aus beidem übertragen. In drahtlosen Sensornetzen werden hauptsächlich der quell- und zielbasierte Modus verwendet. Beim quellbasierten Modus werden die Modellparameter für einfache Modelle direkt im Sensornetz berechnet, um die Kommunikation zu minimieren, da in diesem Fall keine Lerndaten übertragen werden müssen. Für komplexe Modelle werden in solchen Umgebungen dagegen die Modelle auf den leistungsstarken Zielknoten berechnet und Modellparameter zurück zu den Sensorknoten übertragen. Im Gegensatz dazu nutzt das GSG die Tatsache, dass alle beteiligten Knoten im Vergleich zu Sensorknoten leistungsfähig sind und berechnet die Modellparameter synchron auf den Quell- und Zielknoten. In Kombination mit einer dynamischen Anpassung des Prädiktionsmodells kann dadurch

auf die gesonderte Übertragung von Modellparametern sowie großen Mengen Messwerten in separaten Lernphasen verzichtet und somit die Effizienz maximiert werden.

Um die beschriebenen Ansätze zu bewerten, wurde die Aggregation über mehrere Ebenen mit integrierter Prädiktion implementiert und anhand von realen Messwerten evaluiert. Dabei wurden unterschiedlichste Sensortypen wie dynamische, schnell veränderliche Windgeschwindigkeiten und sehr träge Luftfeuchtigkeitswerte herangezogen, welche die generelle Anwendbarkeit erfordern. Die Evaluation des Verfahrens zeigt, dass das gewählte Prädiktormodell in Kombination mit der Aktualisierungsstrategie nicht nur eine sehr effektive Datenverringerung ermöglicht, sondern außerdem eine hohe Genauigkeit der Vorhersagen bieten.

Verwandte Arbeiten beziehen sich auf andere Verfahren, um eine Schnittstelle zu Sensordaten auf einer globalen Skala bereit zu stellen. Dabei ist vor allem Publish/Subscribe zu nennen [BKR09; CRW01; EFGK03], das sich im Bereich der Ereignisverarbeitung etabliert hat und daher keine integrierte Aggregation ermöglicht. Eine spezielle Middleware für globale Sensornetze [AHS06] bietet hier flexible Möglichkeiten, beliebige virtuelle Sensoren zu definieren, was wiederum zusätzlichen Aufwand bedeutet. Ebenfalls im Bereich der Datenverarbeitung im Netzwerk sind Systeme zur Datenstromverarbeitung [AAB$^+$05; FJK$^+$05; GKK$^+$03] angesiedelt, welche allerdings keine einfache und flexible Schnittstelle oder die aktive Wiederverwendung von Teilergebnissen unterstützen. Alternativen zu dem vorgestellten Indexierungsverfahren für Anfragen wurden ebenfalls vorgeschlagen [BFG07; DF03; LLL05; TPS03] allerdings ohne das Routing der Daten zu unterstützen. Im Bereich der Prädiktoren wurden bisher nur Ansätze für drahtlose Sensornetze[DGL$^+$05; DGM$^+$04; GI01; LGS06; RCM$^+$12; SR06; TM06; WB06] und damit nur für Punkt-zu-Punkt-Verbindungen vorgestellt. Außerhalb von Sensornetzen wurden nur spezialisierte Systeme zur Datenstromverarbeitung mit ähnlichen Methoden ausgestattet [JCW04; OJW03].

5 Datenstromverteilung

Wenn eine Überwachungsanfrage auf eine Situation hindeutet, die genauerer Untersuchung bedarf, muss das GSG einen hochauflösenden Sensordatenstrom bereitstellen, um diese Untersuchung zu ermöglichen. Damit dabei auch die durchgehende Verfügbarkeit gewährleistet werden kann, wird die in Kapitel 5 beschriebene Datenstromverteilung angewandt. Wie für die Überwachungsanfragen wird zunächst auf die relevanten Teile des Systemmodells eingegangen. Danach wird die Verarbeitung von Anfragen auf ver-

schiedene Auflösungen detailliert betrachtet. Basierend auf der Anfrageverwaltung wird das Verfahren vorgestellt, mit dem die Ressourcen des Brokernetzwerks gebündelt und somit die Skalierbarkeit des Gesamtsystems gewährleistet werden können. Abschließend wird das Verfahren evaluiert und auf verwandte Arbeiten eingegangen, bevor das Kapitel mit einer kurzen Zusammenfassung endet.

Im Gegensatz zu der zuvor beschriebenen Überwachung von aggregierten Sensordaten werden für die Bereitstellung von Sensordatenströmen ausschließlich die Broker im GSG herangezogen. Neben ihrer Rolle als Gateway zu den Sensornetzen übernehmen sie direkt die Verteilung der Datenströme, wodurch die Datenmenge, welche jeder einzelne Broker bereitstellen kann, wichtig wird. Die Anfrageschnittstelle bietet neben der geographischen Region zusätzliche Parameter, um die Auflösung des gewünschten Datenstroms für eine Simulation zu spezifizieren. Zum einen unterscheiden sich Simulationen dabei durch die räumliche Auflösung, also durch die Anzahl an Punkten, die innerhalb der Simulation zur Darstellung der simulierten Region dienen. Zum anderen sind je nach untersuchtem Phänomen unterschiedliche Aktualisierungsraten angebracht, abhängig von der zeitlichen Dynamik des Phänomens. Die Schnittstelle bietet daher die Möglichkeit die Auflösung sowohl in räumlicher als auch in zeitlicher Dimension als Bruchteil der gesamten verfügbaren Menge an Datenpunkten und Aktualisierungsschritte zu spezifizieren.

Die Grundlage für die Bereitstellung der angeforderten Datenströme bildet die Verarbeitung der Anfragen. Um eine hohe Effizienz zu erreichen, wird wiederum eine Indexstruktur benötigt, die es erlaubt die relevanten Quellbroker für eine Anfrage zu finden und weitere Metadaten über eine Anfrage zu extrahieren. Die verwendete Struktur basiert auf dem GBD-Baum [OS90], einer binären Baumstruktur, welche für die Indexierung von hochdimensionalen Daten entwickelt wurde. Die räumlichen Grenzen einer Anfrage werden daher direkt durch den Index erfasst. Für die Abbildung der Auflösung einer Anfrage auf zusätzliche Dimensionen im Index wird eine spezielle Kodierung beschrieben, die es erlaubt die verfügbaren Daten mit Hilfe des Index zu filtern. Wie die gesamte Indexstruktur ist auch die Kodierung rekursiv verschachtelt aufgebaut und erlaubt dadurch eine einheitliche Verarbeitung aller Dimensionen. Basierend auf der Indexstruktur wird dann ein Algorithmus vorgestellt, um die Inhalte von verschiedenen Regionen effizient auf Überlappungen zu prüfen. Dabei wird die hierarchische Struktur des Index genutzt, um den zuständigen Broker für eine Anfrage zu finden und gleichzeitig die Verteilung der Last über das System zu extrahieren. Ist die Quelle für die angefragten Sensordaten identifiziert, wird die Anfrage in Richtung Quelle weitergeleitet und schließlich als Antwort ein kontinuierlicher Datenstrom von der Quelle zum Ziel geschickt. Die

Zwischenschritte bei der Weiterleitung von Anfragen dienen dazu, eine Lastverteilung zwischen den Brokern zu ermöglichen, da in jedem Schritt die Anfrage durch eine Replik der Ursprungsdaten beantwortet werden kann. Zu diesem Zweck prüft jeder Broker zunächst, ob der angeforderte Datenstrom ganz oder teilweise verfügbar ist und leitet nur den Teil der Anfrage weiter, der nicht lokal beantwortet werden kann.

Die Einrichtung von Repliken dient dazu, die Ressourcen von allen Brokern im System für die Datenstromverteilung nutzbar zu machen, auch wenn die Anfragen sich auf eine kleine Region konzentrieren. Das Problem dabei besteht darin, die Menge der beantworteten Anfragen zu maximieren, ohne dass ein einzelner Broker überlastet wird. Daher wird im Falle eines überlasteten Brokers der mehrfach angefragte Teil der Daten nicht direkt an die Ziele versandt, sondern die zugehörigen Anfragen zusammengefasst und an einen anderen Broker mit geringerer Last verschoben. Die zusammengefassten Anfragen werden dann durch einen einzelnen Datenstrom an diesen Broker bedient, der damit eine neue Replik darstellt. Dabei werden zwei Kernfragen behandelt: Zum einen die Auswahl der exakten Region, welche auf eine Replik verschoben werden soll, zum anderen wird ein Broker ausgewählt, auf dem die Replik entstehen soll. Für die Auswahl der Region werden die zuvor erwähnten Informationen zur Lastverteilung aus dem Anfrageindex ausgewertet, um eine möglichst hohe Zahl von Anfragen mit einer Replik zu beantworten und damit die Effektivität der Replik zu maximieren. Um den Faktor weiter zu erhöhen, werden zudem bestehende Anfragen aufgeteilt, falls diese nur teilweise von der Replik beantwortet werden können. Eine übermäßige Zersplitterung von Anfragen wird dadurch vermieden, dass eine Mindestgröße für jede replizierte Region vorgegeben wird. Die Auswahl eines Brokers für eine neue Replik erfolgt unter den Brokern, welche benachbarte geographische Regionen verwalten. Dadurch wird sichergestellt, dass bei der Weiterleitung von Anfragen zunächst eine Replik erreicht wird, bevor ein potentiell überlasteter Broker eine weitere Anfrage beantworten muss. Eine gleichmäßige Belastung der Broker wird erreicht indem der Nachbar mit der aktuell geringsten Auslastung gewählt wird, wozu periodisch Lastinformationen zwischen benachbarten Brokern ausgetauscht werden. Darüber hinaus werden Repliken nicht auf Brokern erstellt, falls dadurch zyklische Weiterleitungen entstehen könnten.

Die Evaluation zeigt die Fähigkeit des Systems, die Last von Zonen unterschiedlicher Größe im gesamten System zu verteilen und somit alle Ressourcen effektiv zu nutzen. Hierfür wurde die maximal mögliche Anfragelast bestimmt und mit der gesamten Kapazität der Broker im System verglichen. Zusätzlich wurde die Belastung durch die Anfrageverarbeitung sowie durch Replikationsströme genau analysiert. Dabei hat sich

gezeigt, dass die Abbildung der Auflösungen der Anfragen signifikante Vorteile bei der Replikationsstrategie bietet, da die Datenströme zu einer Replik auf die notwendigen Daten reduziert werden können.

Im Hinblick auf verwandte Arbeiten sind zunächst Systeme zur verteilten Datenstromverarbeitung [AAB+05; FJK+05; GKK+03] zu nennen, welche die Datenströme zwischen definierten Operatoren optimieren. Im Gegensatz zu diesen Systemen ergeben sich im GSG ständig wechselnde Datenabhängigkeiten, die bisher nicht zufriedenstellend erfüllt werden konnten. Die bereits für die Überwachungsanfragen relevanten Publish/Subscribe-Ansätze [BKR09; CRW01; EFGK03; TKK+10] sind zwar in der Lage, hochdynamisch wechselnde Abhängigkeiten aufzulösen und Daten auszuliefern, jedoch sind sie aufgrund des individuellen Routings einzelner Ereignisse nicht dafür geeignet, hochauflösende Datenströme zu verarbeiten. Um solche Datenströme mit enormen Datenmengen speziell im Bereich von Simulationen zu handhaben, wurden mit DataCutter [BKC+01] und dem Earth System Grid-I [CDK+03b] spezielle Systeme vorgestellt, welche allerdings keine Echtzeitfähigkeit bieten. Neben diesen spezialisierten Lösungen wurden Multicast-Lösungen auf Anwendungsebene [CDK+03a; CDKR02] vorgestellt, die auf die Verbreitung von Videodaten an eine große Zahl von Kunden abzielen. Dabei ist jedoch die Unterstützung für verschiedene Auflösungen auf eine kleine Auswahl beschränkt, was für Simulationsdaten eine nicht tolerierbare Einschränkung darstellt.

6 Minimierung der Netzwerklast

Nachdem im vorhergehenden Kapitel das Verfahren beschrieben wurde, mit dem die Last der Datenstromverteilung auf alle Broker im GSG verteilt werden kann, wird in diesem Kapitel die Netzwerklast minimiert. Die Grundlage für die Optimierung bildet die Identifikation von Überschneidungen zwischen verschiedenen Datenströmen, wie sie bereits für die Replikation genutzt wurde. Durch eine Anpassung an das zugrundeliegende Netzwerk können zudem ähnliche Kommunikationspfade identifiziert und damit die redundante Übertragung von Daten über eine physikalische Verbindung verhindert werden. Neben einer erweiterten Skalierbarkeit ermöglicht der vorgestellte Ansatz die Datenstromverteilung dynamisch an die aktuellen Gegebenheiten im physikalischen Netzwerk anzupassen. Wie zuvor wird zunächst das zugrundeliegende Systemmodell erweitert, die Optimierungskriterien werden dabei in einem eigenen Kostenmodell definiert. Basierend auf dem Kostenmodell wird das formale Optimierungsproblem beschrieben. Die Minimierung der Netzwerknutzung wird dann in drei Schritten beschrieben, beginnend

mit der Optimierung von Verteilstrukturen. Danach folgt die Darstellung der Verwaltung dieser Verteilstrukturen im Hinblick auf die unterliegende Netzwerkstruktur. Zum Schluss wird die heuristische Adaption von Datenströmen zur Laufzeit vorgestellt, wie sie im GSG eingesetzt wird. Das Kapitel schließt mit der Evaluation, gefolgt von einer Übersicht über verwandte Arbeiten, ab.

Zum Systemmodell kommt für die Minimierung der Netzwerklast der genaue Aufbau des Kommunikationsnetzes hinzu, welcher für das Verfahren von elementarer Bedeutung ist. Das Kostenmodell beinhaltet daher, neben der Menge an Daten pro Zeiteinheit, auch die Länge eines Kommunikationspfades für jeden Datenstrom als Parameter. Die Datenmenge kann direkt über die Eigenschaften einer Anfrage, also die Größe der Region sowie räumliche Auflösung und Aktualisierungsrate, bestimmt werden. Obwohl die Länge eines Pfades, also die Anzahl der Verbindungen, in einem realen System praktisch nicht messbar ist, verzichtet das Kostenmodell bewusst auf die Integration von Latenz, um eine exakte formale Beschreibung des Optimierungsproblems zu ermöglichen.

Das Problem, das in diesem Kapitel gelöst wird, wird zunächst in zwei Teilprobleme gegliedert. Zum einen gilt es, aus einer potentiell großen Anzahl von Anfragen die Überschneidungen zwischen diesen Anfragen zu identifizieren, da für jede solche Überschneidung mehrere Ziele mit Daten aus der betroffenen Region versorgt werden müssen. Diese Problematik wurde bereits im Zusammenhang mit der Auswahl einer Region für neue Repliken behandelt. Für die Minimierung ist allerdings eine exakte Berechnung notwendig. Zum anderen muss dann für jede Überschneidung eine optimale Datenverteilstruktur gefunden werden, die die Quelle mit allen Zielen verbindet und dabei, entsprechend unserem Kostenmodell, möglichst wenige Kommunikationslinks gebraucht. Da zwei unterschiedliche Überschneidungen nicht dieselben Daten beinhalten können, kann diese Optimierung unabhängig voneinander erfolgen. Die exakte Lösung einer solchen Optimierung stellt jedoch eine Form des NP-harten Steinerbaumproblems [DW71] dar.

Eine exakte Lösung des Problems ist nicht nur aufgrund der Komplexität und der Größe des Internets nicht praktikabel, sondern auch weil sich die Struktur und der Lastzustand des Netzwerks sich häufig ändern. Deshalb wird im weiteren Verlauf des Kapitels ein Verfahren zur näherungsweisen Lösung vorgestellt. Zunächst wird anhand eines Beispiels die grundlegende Funktionsweise der Verschmelzung von Datenströmen vorgestellt. Anstatt einer komponentenweisen Optimierung von Overlaynetzwerk und Operatorgraph und einer anschließenden Platzierung von Operatoren auf Knoten im Overlay wird dabei ein integrierter Ansatz verfolgt. Die Optimierung beginnt mit der

Abstraktion des physikalischen Netzwerks durch sogenannte Nachbarschaften. Dabei fügt jeder Broker diejenigen Broker zu seiner Nachbarschaft hinzu, die aufgrund der gemessenen Netzwerklatenz mit hoher Wahrscheinlichkeit in topologischer Nähe ansässig sind. Damit die Belastung durch Messungen gering gehalten wird, werden nur ausgewählte Broker überhaupt für die lokale Nachbarschaft herangezogen. Durch Extraktion von Daten bei der Anfrageweiterleitung werden zudem ständig neue Kandidaten auf ihre Eignung geprüft. Die zweite Komponente des Verfahrens besteht in der Auswahl eines Brokers für die Weiterleitung von Daten zu einem bestimmten Ziel. Durch die Weiterleitung über einen bestimmten Broker kann der Pfad eines Datenstroms im Netzwerk beeinflusst und damit die Verteilstruktur aufgebaut werden. Um die Eigenschaften der Auswahlstrategie zu verdeutlichen, werden zunächst einfache Strategien analysiert und deren Vor- und Nachteile identifiziert. Kerneigenschaften einer Strategie sind dabei vor allem die resultierende Pfadlänge sowie die Anzahl der Weiterleitungen auf dem Weg von der Quelle zum Ziel. Während die Pfadlänge direkten Einfluss auf die Kosten und die Netzwerklatenz hat, wird durch eine erhöhte Zahl an Weiterleitungen die Möglichkeit geschaffen auch kurze Pfadüberschneidungen zu berücksichtigen. Idealerweise sollten also möglichst viele Weiterleitungen bei möglichst kurzen Pfaden erreicht werden. Dieses Ziel wird durch eine kombinierte Strategie erreicht, welche die Dehnung der Netzwerkpfade minimiert und dabei Broker, die näher zur Quelle sind, bevorzugt. Abschließend wird die Kombination der beiden Komponenten durch einen neuen Verschmelzungsalgorithmus beschrieben, der die Daten über das Netzwerk mit der Auswahlstrategie integriert und darauf basierend Anfragen oder Teile davon für eine Verschmelzung auswählt. Dabei werden auch die Lastinformationen aus dem Index mit in die Auswahl einbezogen, um möglichst viele Anfragen zu verschmelzen. Die verschmolzenen Anfragen werden dann zusammen mit dem zugehörigen Datenstrom an ihre gemeinsame Weiterleitung geschickt und von dort weiter verbreitet.

Die Leistungsfähigkeit der Minimierung wurde mittels der Simulation aller Netzwerkschichten unter Berücksichtigung von realistischen Netzwerkstrukturen evaluiert. Dabei wurden verschiedene Platzierungen der Broker in der Netzwerkinfrastruktur untersucht, um sowohl Szenarien mit Brokern in Rechenzentren, als auch Broker in den Anschlussnetzen abzubilden. Das Hauptaugenmerk liegt dabei auf der Belastung des Netzwerks durch die verschiedenen Auswahlstrategien. Um die Verteilstrukturen genauer zu analysieren, wurde außerdem die Anzahl der Weiterleitungen zwischen Quelle und endgültigem Ziel untersucht. Schließlich wurde die Verzögerung gemessen, die für die Verteilung der Datenströme anfällt. Da durch die Minimierung die Anzahl der Datenströme signifikant

reduziert wird, müssen weniger Daten pro Aktualisierung zwischengespeichert werden, was wiederum die Verzögerung insgesamt senkt. Zusätzlich wird gezeigt, dass durch die geringere Belastung die Skalierbarkeit erhöht wird.

Zu den verwandten Arbeiten zählen zunächst die Ansätze aus der Gruppe des Multicast auf Anwendungsebene [BBK02; CDKR02; CRSZ02; ZZJ+01], wobei mehrere Übersichten zu diesem Thema existieren [HASG07; KS10; YLE04]. Diese Systeme sind alle auf kanalbasierte Kommunikation wie Fernsehen ausgerichtet, bei der eine große Zahl von Zielen mit einer begrenzten Zahl von Kanälen bedient werden muss. Daher werden die Verteilstrukturen nur selten aktualisiert, aber dabei sehr weit optimiert, um den Overhead gering zu halten. Für das GSG sind diese Ansätze nicht geeignet, da sich die enorme Anzahl an Datenströmen nicht auf die Kanäle abbilden lässt. Auch die Beschränkung auf ein grobes Gitter von möglichen Anfragen [BRT00] stellt keine gangbare Alternative dar. Clustering [TKKR12] und Verschmelzung [BFG07; JJE10], mit denen Publish/Subscribe Systeme die enorme Anzahl an Anfragen unterstützen, führen dazu, dass mehr als die angefragten Daten übertragen werden können, was für hochauflösende Datenströme untragbaren Mehraufwand bedeutet. Die Optimierungen, die für verteilte Datenstromverarbeitung bisher vorgeschlagen wurden [AAB+05; GKK+03; PLS+06; RDR11; SMW05] erfordern die Definition eines Operatorgraphen, welcher im GSG nicht explizit verfügbar ist.

7 Schlussfolgerung

Kapitel 7 schließt die Arbeit mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf zukünftige Forschungsthemen ab.

In den letzten Jahren haben Simulationen einen starken Zuwachs in praktisch allen Forschungsbereichen erfahren. Infolgedessen werden neue Ansätze aus der Informatik benötigt, um mit der gestiegenen Komplexität, beziehungsweise Rechenaufwand, sowie den enormen Datenmengen umzugehen. Besonders die Verfügbarkeit von immer mehr Sensordaten erfordert neue Lösungen, um Messwerte zum Zweck der Parameterbestimmung und Echtzeitsimulation in Simulationen integrieren zu können. In dieser Arbeit wurden daher Methoden für die zwei wichtigsten Anfragemethoden für Sensordaten vorgestellt. Für die Überwachung von bestimmten geographischen Regionen wurde eine Anfrageschnittstelle erarbeitet. Durch den Einsatz einer verteilten Aggregationsstruktur gelang eine hocheffiziente Implementierung, um Daten über diese Schnittstelle bereit zu stellen. Verzögerungen, die dabei durch die zusätzlichen Berechnungsschritte und

verlängerte Kommunikationspfade entstehen können, wurde durch eine Erweiterung des Konzepts der Prädiktoren auf eine Kommunikation über mehrere Hops entgegnet. Die Prädiktoren ermöglichen darüber hinaus eine weitere Reduktion des Datenverkehrs, indem hinreichend genaue Vorhersagen genutzt werden, anstatt neue Werte zu übertragen.

Für die Unterstützung der Integration von Sensordaten in Simulationen während der Laufzeit wurden ebenfalls Lösungen vorgestellt. Zunächst wurden auch hier die Anforderungen an das System identifiziert und eine entsprechende Anfrageschnittstelle definiert. Ein System zur Anfrageverwaltung, basierend auf einem speziell erweiterten Index für die Unterstützung unterschiedlicher Auflösungen, leitet die Anfragen an die zugehörigen Quellbroker im GSG weiter, die die angeforderten Datenströme bereitstellen. Um den Herausforderungen der resultierenden hochauflösenden Echtzeitdatenströme zu begegnen, wurden zweierlei Methoden entwickelt. Zum einen konnte durch gerichtete Lastverteilung erreicht werden, dass die gesamte Kapazität im System zur Datenstromverteilung genutzt werden kann. Dabei wird im Falle der Überlast auf einem Broker die Indexstruktur, die auch zur Zustellung der Anfragen verwandt wird, benutzt, um überlappende Anfragen zu identifizieren und diese in Gruppen an andere Broker auszulagern. Zum anderen wurde die Gesamteffizienz des Systems darüber hinaus durch ein zusätzliches Optimierungsverfahren erhöht. Mittels einer Anpassung an die Struktur und den aktuellen Lastzustand des zugrundeliegenden Netzwerkes wurde die Verteilstruktur der Datenströme so adaptiert, dass die Gesamtlast im Netzwerk minimiert werden konnte.

Im Hinblick auf die zukünftige Entwicklung ergeben sich weitere Fragestellungen. Der Trend zu dynamischen Datenstrukturen in Simulationen muss ebenfalls durch die Sensormiddleware unterstützt werden. Daher müssen Verfahren entwickelt werden, die die adaptive Anpassung von Simulationsgittern innerhalb des GSG nahtlos unterstützen. Momentan sind für solche Anpassungen jeweils neue Anfragen nötig, welche stattdessen direkt mit der Anfrage spezifiziert werden sollten. Konsequenterweise müssen generische Verteilungen der Datenpunkte innerhalb eines Datenupdates ebenso herangezogen werden. Eine engere Integration von Simulationen in das GSG birgt darüber hinaus weiteres Optimierungspotential.

1 Introduction

This chapter provides an introduction to the Global Sensor Grid (GSG) for scientific simulations. In the following section, a motivation for the different topics covered by this thesis is given. Afterwards, an overview to each of the contributions of this thesis is presented. A summary of the overall structure of this dissertation concludes the chapter.

1.1 Motivation

The quantitative observation of the physical world has long been a topic of interest. The most prominent example for this interest is probably the systematic observation of weather data like pressure, wind speed, temperature, and humidity. The first electronic instruments for weather observations were developed several decades ago and have been refined ever since. With the rise of the information age, the data from these sensors has been manually gathered and added to databases for archival purposes. The ongoing integration of electrical components has led to the development of sensors which produce digital signals for automated further processing and storage. Despite a fully digital processing chain, data from different sensors still required manual intervention for gathering data from multiple locations. Since this problem is not limited to sensor data, computer networks quickly gained momentum, eventually leading to the development of the Internet as known today. Such computer networks provide the basis to gather data from a wide range of measurement stations for the sophisticated analysis of numerous sensors.

The deployment of sensors in remote or hostile areas, however, remained prohibitively expensive due to the cost of infrastructure deployment. Over the last two decades, research has therefore been conducted on various aspects in the field of wireless sensor networks (WSNs). The overall goal of this research [ASSC02a] is to create a network of cheap, small, independent sensor nodes which are each equipped with individual sensing, generic processing, wireless communication, and persistent storage capabilities. Sensor nodes can then be mass-deployed by simply placing them in the targeted observation

area. Instead of connecting each sensor using an expensive infrastructure, nodes can then relay data across the network to achieve full connectivity. The result is a self-organizing WSN which can provide measurements from any sensor in the network.

The availability of sensor data from a wide range of sources boosted the development of applications based on measurements [CHPP09]. Most notably, simulations of environmental phenomena integrate a growing amount of sensor data. The underlying simulation models are thereby generated from insights about the physical processes obtained from isolated experiments. The executed simulations then relied on measurements obtained from a small number of expensive sensors, combined with complex filtering and preprocessing of the raw data. Using the newly available large number of sensors, the simulations can gain in two aspects: First, a higher resolution can be achieved using an increased number of data points. Second, by aggregating data from multiple sensors, the noise in input data can be eliminated efficiently, thereby providing high-quality data with low delay. The very large amount of real-time measurements available for integration into simulations is especially beneficial to improve the prediction horizon for localized phenomena. An example for such a phenomenon are wind fields which are required to simulate the dispersion of pollutants or even radiation [IPS+09]. As a result, an increasing number of simulation models follows a paradigm shift from dedicated simulation runs towards a dynamic data-driven approach. In a data-driven simulation, data can be integrated at run-time in each simulation step [Dar04].

Wireless sensor nodes are commonly very limited in terms of energy supply, computational power, storage [DGMS07], and communication [DAW+08] capabilities. The limited energy supply of a node is thereby to be taken into account, since depleted nodes can no longer participate in maintaining the network structure. Different methods have been developed with focus on energy consumption to allow for possibly long operation times of the entire network. Approaches include energy efficient network protocols, self-adaptation of wireless sensor nodes to different energy levels, and limited frequency of sensor readings. Since many of these techniques result in decreased quality of service, a trade-off between the lifetime of a WSN and its performance is required [WKSR04]. Most importantly, queries to a WSN commonly require timely replies to provide real-time capabilities for time-critical applications. To this end, modular operating systems have been developed in order to enable nodes to locally adapt to different requirements [DGV04]. On top of these operating systems, programming abstractions [LHRM09] are provided for application development.

With the high number of sensors globally available today, a global integration of local WSNs has become a primary goal. The continuously growing number of nodes in each WSN thereby leads to considerably increased forwarding load on each node, which leads to additional challenges [ARE+05]. Approaches such as hierarchical storage and query algorithms [GGE+05; IS09b] and specialized distributed indexing techniques [WL09] were proposed to address these scalability issues. However, the size of a single WSN remains limited for now, leading to an increasing emphasis on connecting isolated networks rather than extending them.

Over the last years, the transition to a global sensing infrastructure has consequently been subject to research. New solutions have been investigated to provide scalable access to sensor data which originates from various different networks. To manage the vast amount of data, a versatile query interface is required which allows the filtering of data in the network in order to increase efficiency. Driven by these requirements, two main research directions emerged: On the one hand, methods for building a global federation of wired and wireless sensor networks (SNs) were developed. Approaches formulated a worldwide sensor web based on established web technologies [GKK+03] or frameworks similar to the grid computing concept [FHW+08]. Middleware solutions [AHS06] for the definition of virtual sensors were presented which support the creation of new virtual sensors without extensive programming. On the other hand, approaches for generically handling big data tasks, including measurements from SNs were proposed. Solutions for distributed data stream processing focused on the provision of data streams between manually deployed operators [ACÇ+03]. A data stream is thereby characterized by the transmission of tuples with very high frequency between defined end points. For content rather than endpoint defined communication, event processing solutions were proposed which provide highly flexible interfaces for specifying filters based on the properties of events.

The state of the art with respect to a global sensing infrastructure, however, still misses some key components, especially with regard to the provision of measurements for simulations. First of all, a highly efficient way to monitor large geographical regions in real-time is not available in the literature, yet. Such regions might cover multiple SNs and might include an extremely large number of sensors. This monitoring can alleviate the system from providing a huge amount of raw sensor data in favor of small updates which trigger more detailed investigation. To provide real-time capabilities beyond monitoring, an interface to high resolution sensor data needs to be provided in a way which allows for the automated integration of measurements into the workflow

application. This integration in turn requires the query interface to handle a large number of generated requests rather than only supporting manually posed queries. The dynamics of automated requests has to be considered especially for simulation applications where the requested resolution of data can widely vary in both spatial dimensions and temporal update frequency. Despite the highly dynamic demands in terms of requested data and therefore bandwidth consumption, the system needs to provide a reliable service. Lastly, a global sensing system needs to co-exist with other applications on the network since no dedicated resources are available. Therefore, effective means to minimize the overall bandwidth consumption of the entire global sensing system are mandatory.

1.2 Contributions

This thesis describes a set of methods and algorithms bundled in the Global Sensor Grid Middleware (GSGM). Core tasks are executed by a number of brokers which provide the necessary hardware infrastructure. The combination of infrastructure and GSGM comprises the GSG, which aims to close the previously described gaps. The contributions cover five main objectives which focus on two main application scenarios. The first two contributions provide highly efficient real-time monitoring of potentially large areas. The following three topics focus on the scalable provision of high-resolution sensor data streams for simulations.

1. First, we identify the query interface requirements posed by applications which monitor a global sensing system with a large number of sensors. Based on these requirements, we propose an intuitive interface which allows users and applications to pose queries which monitor regions for a single aggregated value.

2. To provide a highly efficient monitoring facility which enables the provision of our query interface, we introduce a new method for distributed aggregation of sensor data on a global scale. The real-time capability of our monitoring system is provided by integrating a multi-hop prediction mechanism. The combination of prediction and aggregation is able to instantly provide estimations of the current aggregated reading at highly reduced communication cost.

3. Simulation applications, as the main targeted application class, require a full grid of data points for each time step with a specific spatial and temporal resolution. In order to provide access to such measurements, we provide an extended query

interface for high-resolution, real-time data streams. Besides the area of interest, the interface allows to specify the resolution of the resulting data stream in spatial and temporal dimensions.

4. The high-resolution data stream provided by our query interface requires a large amount of network resources using conventional transmission. To be able to serve a possibly large number of requests, we propose a new stream management approach, which distributes the load among different nodes in the network. Areas which observe a large number of queries are effectively served using dynamically established replicas of the requested data.

5. Since a dedicated network infrastructure for distributing measurements incurs high monetary cost, the Internet is often used as a communication substrate. We propose a new optimization strategy for minimizing network usage to support the operation of a global sensing infrastructure. In addition, the optimization allows a distribution of data streams among multiple network paths to increase scalability.

Together, the contributions are integrated in a single middleware to form the GSG. This GSG is capable of integrating data from numerous isolated SNs and providing real-time access to highly efficient monitoring and high-resolution data streams. Preliminary results of this thesis have been published in peer-reviewed conference proceedings [BHKR09; BKR11; BKR14; BKVR10].

1.3 Structure

The remainder of this thesis is structured as follows:

A detailed description of the background for the presented work is given in Chapter 2. The topics covered include the execution of simulations using scientific workflows, the distributed processing of data streams, as well as wired and wireless SNs. A short description of the Stuttgart Research Center for and Cluster of Excellence of Simulation Technology, which provided the environment for this work, completes the chapter.

A system overview of the entire GSG is provided in Chapter 3. The overview generically describes query abstractions required to request data for different usages from a global sensor infrastructure. Based on these query abstractions, an introduction to the data processing workflow of scientific simulations based on real-time sensor data is given.

The technical description of the GSG first covers the acquisition of sensor data from isolated sensors and SNs. Chapter 4 details how data from various sources is aggregated

depending on the queries registered in the system. We present a distributed aggregation scheme which supports the reuse of partial aggregates for multiple monitoring queries as well as real-time notification based on the prediction of measurements with limited error bounds.

In Chapter 5, we detail how the massive amount of measurements globally available can be effectively distributed to a large number of clients. The GSG supports queries for continuous real-time data streams at user specified spatial and temporal resolutions. A new multi-resolution query processing approach is introduced to enable the distribution of query load from single servers to the entire system by identifying overlap in data streams of different resolutions.

Chapter 6 covers our approach to minimize the network usage of the GSG by eliminating redundant data transmission on physical network links. The cost of network usage is defined, providing the basis for the corresponding optimization problem. Subsequently, we describe how the physical network is abstracted to extract required information for our optimization approach. The optimization strategy is described in detail along with the algorithm used for merging redundant data streams.

The thesis is concluded by a short summary, including the main results of this work, and an outlook on future research directions in the field of global sensor networks (GSNs). Both are provided in Chapter 7.

2 Background

This chapter provides a background to the approaches presented in this work. An overview of related work is provided and basic concepts are introduced. The description follows a top-down view starting from the high level abstraction of a scientific workflow. The workflow relies on software components, called web services, which integrate data using the concept of data streams. The management of data streams includes methods for efficient distribution of data in a many to many scenario as well as the placement of data and processing entities in the network. On the lowest level reside the sensors and sensor networks (SNs) which provide the raw measurements. The focus on this level is to reduce the amount of data transmitted to the web service by filtering. Information about the main applications of the Global Sensor Grid (GSG) for simulation scenarios is provided in the context of an overview of the Stuttgart Research Center (SRC) and Cluster of Excellence (EXC) Simulation Technology.

2.1 Scientific Simulation Workflows

An increasing number of scientific experiments are carried out using simulations to verify the correctness of a hypothesis. Such scientific simulation experiments follow a pattern commonly executed multiple times: Set up the experiment, run it, evaluate the results, adjust the parameters and start over, if necessary. In each of these main steps, many different tools and methods are used to transform data between steps and to perform specific tasks.

In the first simulation experiments, all the tasks were not only configured and started manually, but also mostly built from scratch. Most core tasks, however, are now executed by well established simulation software packages. To simplify the configuration and to control the execution, scripts were created. Many combinations of simulation software and complementary tools were integrated to automate the entire process. For every step in the experiment, a script specifies all the information about parameters as well as input and output files and formats for every tool. Therefore, most of these scripts require a lot

of maintenance as parameters change and new versions of tools and simulations become available. In addition, if intermediate data formats change, additional converters have to be integrated for the existing tools to work.

In business and production environments, such recurring patterns of tasks are organized in workflows [LR99]. The traditional paper-based processing of workflows, including forms and defined tasks on the information, has long been transformed to the digital world. Powerful workflow management systems have been developed which allow a very flexible handling of different instances of workflows. A single workflow is composed of multiple tasks and specifies the control and data flow between these tasks. Each automated task is usually carried out by a web service. The specification how the web services interact is typically written using the XML-based Business Process Execution Language (BPEL) [JE07]. While originating from the business domain, workflows technology is increasingly present in many areas of modern life where tasks have to be executed regularly. For example, recent research aimed to support nurses in their daily routine [WHR10]. However, the seamless interaction with humans that execute certain tasks is currently still subject to research.

Recently, workflows and their flexible management have moved into the focus of simulation-based research. Workflow technology has been adapted to make use of the related tools to facilitate management of experiments and results. A new type of scientific simulation workflows has emerged in the face of increasingly complex toolchains with a growing number of programs involved [ABJ+04; BH08; GSK+11; MSTW04]. The core of these systems is still similar to the business domain. However, with the large amount of data involved in todays simulations, the underlying web services and the interactions with them have fundamentally different requirements [GDE+07].

Web Services As described previously, the single tasks of a scientific simulation workflow are provided as web services. Unlike the traditional script-based approach, where a static configuration of the simulation has to be manually configured before executing the workflow, these services can be dynamically allocated. A service is selected for executing the task at hand, mostly depending on its cost and reliability, during runtime. The request including the data is then sent to the service using the standardized web service application programming interface (API).

Using this architecture, additional tools can be made available by providing wrapper which translates the existing interface of a tool to the generic API. The tool can then be integrated into a simulation workflow without having to set up a local instance of a

particular tool. Note that a workflow and the corresponding web services can be executed remotely in a distributed system, grid, or cloud infrastructure [DBG+04; DSS+05]. The control flow between tasks, however, is usually managed by a centralized workflow engine.

Provenance for Simulations Like for traditional experiments, the conditions and results of a simulation run need to be reliable and well-documented. Workflow management systems provide methods to systematically monitor and store the state of a particular instance. For example, financial transactions need to be logged consistently in order to be able to recover from crashes or to investigate insurance claims. These provenance mechanisms have been extended to specifically suit the needs of simulation workflows [MSTW04]. In addition to verifying results, the metadata gathered is also used to ensure the quality of data [RBKK12]. One approach which serves both purposes is the integration of visualization tasks into a workflow environment [CFS+06].

Distributed Workflows As simulation experiments are commonly carried out by various research facilities all over the world, the corresponding web services are also widely distributed. The scientific workflow management systems have supported this distributed nature in multiple approaches [LMJM06; PIS08]. Especially the support for grid infrastructures was integrated early and has been continuously extended [CAWK08; CGH+06; MSTW04]. In recent years, the support for the execution of tasks in the cloud was added as the concept has become more popular [AIG12; WHF+13].

In the face of large datasets for input as well as output to and from simulations, additional challenges for distributed workflows arise. Sending all the raw data and results from one research institution to another poses an extremely large problem. Instead of transmitting the data to the web service, the computation is therefore brought closer to the data. Such approaches select the optimal combination of shipping and processing data. For example, computationally expensive tasks might be carried out in the cloud, while services for data intensive operations are set up close to the storage in the local data center. New communication paradigms have emerged to tackle the problem of connecting web services based on the data they provide rather than their specific address [MYSJ11]. The location of a web service can be chosen freely from any place where sufficient computational resources exist at the cost of transmitting data to that location once. Sensor data, however, is bound to the place of measurement and needs to be continuously transferred for seamless integration.

Data Integration While the data flow inside a workflow can be described according to the specification, the integration of data from external sources is still subject to research [AEM13; CAWK08; GMW+04]. Workflow management systems are not designed to internally handle large amounts of data, but rather to establish a connection between the web services which produce and consume the data, respectively. Such connections are commonly explicitly requested for each set of data whereas most sensing systems trigger a transmission based on the measured data.

Only few approaches have been taken so far to alleviate scientists from manually integrating external data sources. The goal is to unify access to heterogeneous data sources including sensors and SNs by automatic transformations [LAB+06; RRS+11]. These frameworks abstract from the underlying formats by relying on logical data descriptors to choose the correct transformation dynamically. However, as previously described, the approaches extending the workflow itself cannot accomplish location transparency for a web service as far as sensor data is concerned. Scientists still have to manually implement access to sensors by querying and fetching the measurements before they can be integrated using the above frameworks.

Another framework, the OGC® sensor web enablement [BEJ+11; RBDP07], allows for self-describing sensors. Based on an XML description of the sensing device, data integration can be automatically set up with the appropriate formats. But even with such a framework, the resulting data streams have to be established and managed in order to continuously provide the measurements. Additionally, the framework lacks support for handling data generated by simulations, its application is restricted to physical sensing devices. In the following section, we describe basic mechanisms necessary to efficiently transfer large amounts of real-time sensor data.

2.2 Distributed Stream Processing

As previously described, integrating large amounts of data generated at spatially separated locations into scientific simulation workflows poses a difficult problem. The problem itself also occurs in various other domains, for example high volume event processing or multimedia streaming. In these application scenarios, distributed stream processing (DSP) [CBB+03] has emerged as the main method to handle large amounts of continuously generated and processed data.

DSP combines two topics and can be seen as the evolution of both. On the one hand, the continuous growth of databases and the increasing requirement for real-time handling

of data lead to a distribution of data bases and to the need to stream data between instances. On the other hand, the increasing number of distributed systems in general lead to ever growing bandwidth requirements and increasing numbers of application nodes. Both trends have grown together and have merged to what is now known as DSP.

The database community sought for solutions which are capable of handling continuous SQL-like queries in a real-time environment. Centralized query processing systems [BBD+02], however, first have to gather all data to a single location before being able to distribute the results. Distributed frameworks like NiagaraCQ [CDTW00] or TelegraphCQ [CCD+03] focused on the semantics of continuous queries and how data is consumed by each processing step in the network. Similarly, in distributed complex event processing as a related domain applications are programmed entirely based on rules which specify how input events are combined and processed [CM12]. The bottleneck of such systems is still the processing power and memory constraints of the computing nodes rather than bandwidth restrictions. Consequently, load shedding has been introduced to cope with the bursty load situations [AJS+06]. Bursts commonly occur if multiple queries are triggered simultaneously by incoming data tuples.

With the rise of peer-to-peer networks, live streaming of television programs and other on-demand content became a driving force. In contrast to the previously described systems, the emerging distributed applications strongly focused on streaming data rather than processing it [Cha03]. Though the requirements on the hardware are not negligible for such systems, the limitations on network bandwidth pose the main problem. To provide a high quality of service to users, multiple connections are combined to logical overlay links and caches are established to bridge short congestions.

The high density of globally distributed sensors anticipated in this thesis leads to high bandwidth data streams which are delivered to a potentially large number of clients. In addition, each client can specify its interest in sensor data, generating a large number of distinct data streams. Together, these two aspects require highly scalable and flexible data distribution methods. Different subsets of the topic of DSP have evolved which are relevant for this work. To provide generic one-to many communication, application layer multicast (ALM) has proven to be a powerful approach. In data-intensive scenarios, the problem of content distribution arises where the location of caches is crucial for the system performance. For computationally expensive tasks, a set of nodes has to be selected to execute calculations and to distribute results. In the following, we provide a background on each of these topics and point out their relation to this work.

Application Layer Multicast The foundation of most stream processing applications requires the distribution of processed data to a potentially large number of destinations. The mechanism of sending a datagram to multiple destinations is called multicast in comparison to addressing a single destination (unicast). As a fundamental paradigm used in distributed systems the Internet protocol (IP) [RFC791] was soon extended with native multicast support [RFC1112]. Even after multiple extensions of the original specification [RFC2236; RFC3376; RFC4604], numerous other approaches have been proposed [CFD01; Obr98].

The underlying optimization problem for multicast routing is to find a graph which connects the source to all recipients at minimal cost. Such a graph is called a minimum Steiner tree problem (MST) [DW71], the distribution points in the graph are the Steiner Points. Algorithms to calculate the MST have been proposed, however, due to its NP-hard nature, the problem has no efficient solution. Approximate solution approaches are still subject to current research [CBD+12; DSU09; MRR06; OP05; OP11; RZ00].

Successful deployments of IP-multicast have been rolled out inside the networks of Internet service providers (ISPs). The main use case for these deployments are streaming audio or video content to a large number of clients. As a result, these systems scale well to a large number of destinations and provide a robust service. However, many ISPs disable multicast since the resulting traffic of a single transmission is hard to predict and control which results in unwanted load in the local network. Moreover, due to deployment issues across ISPs, IP-multicast has not been adopted throughout the Internet [DLL+00]. To overcome these problems, new solutions have been developed to work on the application layer [CRSZ02; HASG07; KS10; YLE04].

To leverage the local deployment of IP-multicast, hybrid approaches combine the native support of the infrastructure with additional functionality on the application layer [RLSS02; ZWJ+06]. The local deployments are used to disseminate data efficiently inside the network of an ISP. To connect the different multicast-enabled networks, an applications layer multicast protocol is used which relies on basic unicast delivery. These solutions solve the problem of transmitting data between multiple ISPs. While this approach suits the previously mentioned distribution of real-time multimedia content, the large number of sources for DSP in general requires more dynamic approaches.

Most ALM approaches rely on a peer-to-peer key-based lookup and storage infrastructure to form and manage multicast groups. For example, Scattercast [Cha03] is built on an extended Gnutella network [CRB+03], other work relies on CAN [RFH+01; RHKS01]. A third example is Bayeux [ZZJ+01] which relies on the interface provided by

Tapestry [ZHS+04]. The proposed ALM solutions rely on the neighborhood management of the respective underlying peer-to-peer substrate. Multicast groups are created by establishing a rendezvous point, which is stored as an abstract key in the peer-to-peer system. The information of the routing algorithms can then be used to generate a multicast tree, which connects all interested nodes to the source point. Depending on the routing substrate, the tree is adapted to the underlying network structure [KF02; RM99] or built solely on the logical neighborhood relations in the overlay.

To improve efficiency, other solutions directly implement the routing and tree generation algorithms in the ALM middleware. Some of these approaches have been designed to support specific applications like multimedia streaming [PWCS02] or publish/subscribe [BCM+99]. General purpose approaches like Scribe [CDKR02], SplitStream [CDK+03a], Overcast [JGJ+00] and others [BBK02; IBU+11] provide a universal interface for sending multicast messages. As for the peer-to-peer based protocols, information about the exact underlying network topology is not available. Therefore these approaches rely on measurements which are otherwise carried out by the substrate. Based on the estimation of the underlay, the distribution structures are dynamically optimized.

Content Distribution In contrast to the previously described ALM, content delivery systems were developed to serve large amounts of mostly static data. The first content distribution network (CDN) instances were built to support content providers in scaling their services in the world wide web. To achieve this, static content like pictures and videos are stored on a number of servers from which clients can be served without overloading the main web servers. More recently, CDNs also support real-time data streams, for example to scale video distribution.

The main challenge in operating a classical CDN is the placement of replicas of the original content across data centers in the network. A similar problem is known from logistics where the number and location of warehouses has to be determined. Theoretical formulations of the problems exist based on global knowledge of the entire network, including all physical links [BCEL08]. Even though these formulations allow for an exact solution of the problem, a generic solution is not available. The simplifications required to provide a theoretical description of the problem mask important details of today's network infrastructure. In addition to the fact, that network graphs typically contain many nodes and vertices, the resulting problems remain NP-hard [BOO07]. Algorithms therefore commonly rely on simulated annealing [BCEL07] or other heuristic strategies.

Due to the complexity of the underlying problem, several approaches have been proposed which aim to find a suitable solution on a macroscopic level. These approaches try to optimize the CDN without determining all underlying parameters and solving the exact formulation. The reduced complexity of this view requires determination of certain aspects. A common approach is to first determine the location of replicas [RGE02] depending on the network structure and load statistics. This selection comprises the long-term optimization since it involves installation of servers and communication links. Based on the resulting set of replicas available, the actual placement of objects to replicas [LZS04] and allocation of storage capacity [LZS05] is determined in a second step. This adaptation of storage to items is extremely challenging for shared CDN systems organized in peer-to-peer networks [PS06].

Overall, CDN technology has allowed new services on the Internet. While the original design focused on a generic robust best-effort transmission of data packages, content delivery networks focus on the type of data at hand. These types of networks are known as content-centric networks [Pas12]. An important aspect of these networks is to provide streaming capabilities for content distribution based on peer-to-peer and CDN techniques. Although streaming support for CDN has been introduced earlier [AEVW04], recent trends to streaming content, like video platforms, fueled the development of new approaches.

The most important new development is the virtualization of the network infrastructure using software defined networking (SDN). SDN is increasingly used to tailor network characteristics to the needs of particular applications. In particular, an SDN can be set up to efficiently distribute real-time streaming content rather than statically stored media. By exploiting knowledge about the entire network topology, content-aware traffic engineering can be achieved based on SDN [FPS+12; PFS+12]. However, these solutions require detailed knowledge about the underlying infrastructure. Similar to the previously described problems in multicast deployments, each SDN solution is therefore currently limited to its administrative domain. Therefore, the approaches presented in this work rely on the traditional application layer to provide services across different ISPs.

Operator Placement The core goal of the previously described methods is to efficiently distribute unaltered content from multiple sources, in the case of ALM, or a master server, in the case of CDN, to numerous destinations. However, as the name suggests, DSP also covers applying certain operations to a stream of data tuples in a distributed fashion. Many operators, most of all filters, have different incoming and

outgoing data rates. The bandwidth used by a DSP system therefore changes significantly with the position of operators in the network. In order to increase efficiently in both distribution and processing of data, the location of the operators executing said processing is crucial. The topic of operator placement has therefore emerged as an independent research area [LLS08].

The placement of content to nodes in the network for distribution is an integral part of content distribution networks, as mentioned previously. Since large fractions of the content change rather infrequently, e.g. movie or music streaming services, algorithms from logistics have been successfully adapted to solve this problem. However, stream processing data changes dynamically, depending on the input values and number of items to process. The varying required resources in terms of bandwidth and processing power must be considered when placing operators on nodes, which typically have only limited capabilities. The placement itself not only has to satisfy these resource constraints, but also has a great impact on the resource consumption and resulting quality of service. This problem is further complicated by the fact that migrating an operator with a large amount of state information comes at additional cost. Consequently, the approaches for placement of operators in DSP focus on dynamic adaptation to the current load situation for optimization.

Some stream processing systems have been built to optimize performance by considering the placement as part of overall operation. These systems differ in the query interface, the granularity at which the operators can be placed, and the underlying performance metric. For query systems which emerged from the relational SQL interface, an operator graph is built from the query. In other systems, the operator graph can be built using a visual boxes-and-arrows interface or using a special type of XML notation. The resulting operator graph is then mapped on a topology which represents the underlying network.

Since the network topology is usually far more complex than the operator graph, hierarchical node partitioning can be used to decentralize the placement [KCS05]. Network nodes are thereby hierarchically grouped according to network properties like delay and bandwidth. The operator graph is then initially assigned to nodes by recursively dividing the operator graph and matching the parts to increasingly smaller groups of nodes. During operation, the placement is incrementally updated when sub-optimal conditions are detected to adapt to changing network conditions.

A similar separation between initial placement and runtime adaptation is used by most DSP systems. Depending on the main application scenario, different additional optimization strategies have been proposed. For data mining applications, the stream pro-

cessing core focuses on maximizing output rather than overall data throughput [AAB+06; AJS+06]. In contrast to centralized systems, where higher throughput indicates better performance, this metric avoids intermediate nodes which generate data which would later be filtered out anyways. Borealis [AAB+05] groups operators in the operator graph to minimize I/O operations between nodes and simplify placement. The groups are then distributed to minimize bandwidth consumption and provide high quality of service properties to the application. Based on the Borealis engine has also been extended to support SN query processing [Lin06]. The integration of stream processing and sensor query processing provides further potential to optimize the operator graph closer to the source of data.

Apart from these placement strategies, which were developed for specific stream processing systems, generic optimization middlewares aim to support all kinds of stream processing application. Since generic approaches do not have knowledge about the internal state of an operator, their optimization goal depends mostly on network properties. Each operator is therefore treated as a special type of filter which produces data in a rate proportional to the input rate. The stream-based overlay network [PLS+06] abstracts each link between to operators as a spring with the data rate as spring constant and the delay as extension. The placement algorithm then minimizes the overall spring energy in the system. Since the spring energy depends on both parameters, this approach tackles the trade-off between minimizing delay and minimizing bandwidth usage in a stream processing system. A more recent approach aims to directly minimize the bandwidth delay product[RDR10b] for a given operator tree while fulfilling latency constraints [RDR10a; RDR11]. Instead of focusing on a system of springs, each operator locally tries to find its optimal position relative to the adjacent operators in the operator tree. An exact formulation [CBD+12] of the problem has been proposed which supports generic operator graphs instead of trees.

Hybrid approaches combine the generic usability with additional information about the actual application to further optimize the system performance. Such additional information includes execution cost and filter coefficients of single operators. The filter coefficient of an operator specifies how much data units are output for each input data unit. The execution cost details the amount of processing power and memory required in different load situations. With this additional information, the system cannot only optimize the location of single operators with respect to their neighbors, but also consider the joint placement of multiple elements. This allows for the distributed optimization of expensive conjunctive filters and joins [SMW05] with the originally centralized STREAM

engine [ABB+04], for example. Even more generic approaches facilitate the integration of modular placement metrics [PÇJ09] which allow a flexible definition of the operator cost depending on a variety of operator properties.

The minimization of network usage and latency, together with an optimized usage of processing resources, is not the only goal of operator placement. Due to the complexity of some processing steps the problem is to find a placement where all operators can be executed at all. This constraint satisfaction problem is crucial for performance monitoring systems [ZOTW06], which have to deal with a bursty high load without interruptions. Especially in the domain of distributed complex event processing, the placement of correlation rules [SKR11] introduces additional constraints like administrative boundaries or the requirement of specific correlation engines. Non-performance related criteria have also been taken into account for placement strategies. Operators which are replicated for high availability [RK08] need to be placed on different nodes to ensure their independence. To transparently support operator migration across these diverse placement approaches, virtualization methods have been applied to DSP [DRAT11]. The additional abstraction layer allows for a very flexible distributed deployment.

Overall, the problem of deciding where to execute a particular operation of a complex distributed application has many faces. The many different and sometimes contradicting optimization goals of networks and applications together with numerous parameters have to be taken into account when choosing a placement strategy. For global sensing applications, the most common operations are computationally cheap filter and aggregation functions. With the vast number of sensors in place, the primary optimization goal is the reduction of data all the way from the sensors to the final result. The next section will provide an insight to the existing approaches for data reduction in SNs.

2.3 Wired and Wireless Sensor Networks

The GSG aims to provide access to streams of sensor data from virtually any location on earth. The source of such data streams are groups of sensors organized in local SNs. These wired and wireless SNs have been a very active research field for more than a decade [ASSC02a; ASSC02b].

Most work has been done on wireless sensor networks (WSNs) which consist of battery powered self-organizing nodes. As sensor nodes are a kind of embedded systems, there exist operating systems which already provide generic query interfaces. Such operating systems have been developed with the special requirements for energy preserving and scarce

resource management in mind. The most popular representatives are TinyOS [LMP+05] and Contiki [DGV04]. TinyOS is based on an event-driven programming model instead of multi-threading and many projects have been built on top of it. On top of TinyOS, TinyCubus [MLM+05] provides a powerful data management framework which supports optimization using cross-layer interaction. Contiki features run-time loading of additional modules, per-application multi-threading and focuses on IP-enabled sensor nodes.

The constrained resources of single sensor nodes, especially in terms of energy consumption [LMMR07], pose a great challenge for the lifetime of an autonomous WSN. Since communication is the most energy-consuming task inside such networks, most approaches focus on trading in processing time for a reduced overall communication. The benefits of this exchange can result in orders of magnitude reduction of energy consumption [PK00]. Considering the fact that where, when, and how often data is physically acquired can be controlled independently inside each SN, efficient query processors [MFHH03] decouple the physical sampling from the queries to minimize energy consumption.

Besides energy, the most difficult task for a sensing infrastructure is accessing sensor nodes and measurements located on them. The IEEE 1451 standard [SL08] defines a common encoding for networked smart transducers which allows unified access to measurements. The Sensor Modeling Language [BCRG14] provides a standard for describing the sensors themselves and the measurement process. However, efficiently locating measurements for a particular geographic region poses a problem with the growing size of SNs. This problem becomes crucial for the GSG which provides global coverage by enabling a worldwide federation of local wired and wireless SNs.

In the following, we first describe approaches for sensor localization and routing in SNs. The localization is essential to provide location information for measurements and is commonly used to optimize the routing of data. Based on the strong relation between sensor data and geographic positions, indexing schemes for data lookup in SNs have been developed. The second part investigates the applicability for real-time systems of these approaches, especially on a global scale. More generic indexing methods are also presented which have proven to be scalable for geographic databases. The indexing is a core foundation of the main technical work presented in this thesis. Following the lookup methods, the third part of this section provides a more detailed background on the concept of predictors. Predictors are used as one major strategy for data reduction by using the processing capabilities of nodes in a WSNs. The basic technique is to estimate the measurements at the sensor and the destination and to skip transmission of data, if the estimate is close to the measurement.

Sensor Localization and Routing Measurements are only of very limited value without the corresponding location information associated. Consequently, nodes in a SN need to know their own location in order to be able to provide the full information. More importantly, when a SN is queried for data, the client is usually interested for measurements from a particular region of that network. The query needs to be transferred to the corresponding sensor node(s) and the measurements have to be sent back to the origin of the query. This paragraph briefly introduces localization and provides an overview of methods used for routing in SN.

The global positioning system (GPS), which is known from car navigation systems, offers a straightforward source for location information. Unfortunately, the required hardware is expensive and the operation of a GPS sensor requires a lot of power. Numerous alternative approaches have been proposed for energy efficient localization of sensor nodes [CPH06; HHB+03; LR03; PAK+05] and extended to three dimensions [SM10] to circumvent this problem. For the purpose of this work, the details of these localization methods are not relevant, since the implementation of a SN cannot be changed by the GSG. However, the tight coupling of sensors to a geographic location has a significant impact on the design of the system.

As stated previously, the location of sensor nodes is not only relevant for the measurements taken by the nodes, but is also used to determine the routing of information in a SN. The routing itself has considerable impact on the energy consumption, since it directly affects the amount of communication in a WSN [SML+06]. A well-known example is the greedy perimeter stateless routing algorithm (GPSR) [KK00]. GPSR can dynamically adapt to node failures by selecting an available node which is geographically closest to the destination as the next hop of each data packet. This method also avoids the costly gathering of sensor locations and the complex construction of a network topology. Other approaches combine the localization and routing processes [BGJ09] to leverage synergies of the two problems. To support the delivery of data to multiple destinations, hierarchical geographic multicast routing [KDHS10] has been adapted for WSNs.

With the introduction of mobile sensors, the problem of localization and routing has become even more complex. Recent work [CM09] has therefore focused on this extended scenario with additional support for multi-sink delivery of measurements. Building on this work, the researchers built a middleware for service oriented programming with multicast support [CM10]. While such a middleware allows for a better integration of a single SN in the previously described workflow management systems, the problem of providing a unified interface to a global federation of SNs remains unsolved.

Measurement Lookup As previously described, query processors, especially for WSNs, focus on energy efficiency. The most common approach is to use the local processing power to reduce the amount of communication which incurs high cost in terms of energy. Local predicates are derived form queries which can be monitored on single sensor nodes without having to communicate during long periods. If a node detects measurements in the range of interest, the actual data acquisition is triggered and active measurements of other nodes can be initialized. This way, queries can be distributed among the relevant parts of a SN without having to constantly probe all available sensors. However, this approach is not suitable for the new generation of big data applications which use the large amount of constantly gathered data from as many sensors as possible.

To provide access to the full amount of available sensor information, a storage oriented view on WSNs has emerged. The increasing capacity and dropping prices for flash memory have led to increased storage space on sensor nodes [LMG+07]. With this additional capacity, time series of measurements can be stored on nodes which leads to a fundamentally different architecture [DGMS07; GEH03]. The data no longer needs to be instantly sent to the gateway, but can be archived in a distributed fashion across the WSN. When a query is issued, the requested data can be gathered on demand and sent to the corresponding client. Especially with the increasing trend to real-time applications, a fast operation is highly desirable. The lookup of sensor nodes, which store the data of interest, and the efficient transmission of this data to clients are, therefore, core tasks of a sensing framework.

For the lookup task, an indexing structure has to be maintained to make the search for data items from a particular location fast and efficient. The indexing not only provides a structure to quickly find information, but also provides the basis for selecting a storage node for a particular measurement. The popular R-Tree [Gut84] spatial indexing and the advancement R*-Tree [BKSS90] have proven useful in the database domain. The peer-tree [DF03] is a peer-to-peer adaptation for WSNs which exploits the containment relation of the R-Tree to limit query processing to involved nodes. The storage of measurements is thereby assigned to the sensor nodes themselves.

The other extreme approach of measurement storage employs dedicated storage nodes which are organized in a geographic hash table [RKY+02]. Other approaches follow a hybrid approach with a combination of sensor and dedicated storage nodes [GJP+06]. TSAR [DGS05] uses proxy-nodes to improve efficiency on multi-tier SN architectures. The underlying indexing method of TSAR are interval skip graphs which are used to summarize the actual measurement on multiple resolutions. Another approach uses

connected dominating set based indexing [WL09] to divide the entire scheme into three levels: sensing, storage, and index nodes. This additional layer allows for the management of large scale WSNs while balancing the cost of transmitting data to storage nodes against the cost for lookup and delivery of measurements. However, the limited number of levels, or tiers, leads to difficulties as the number of storage and indexing nodes becomes quite large with an increasing overall number of nodes. In a global sensing system, this limitation also affects the granularity at which data is available. For example, a query for a large region might require data from many storage nodes just to get an average measurement.

Hierarchical indexing for storage and search [IS09a] avoids this problem by using a data structure with additional levels, which are added when needed. The different levels can be distributed among the sensor nodes, but also extended to more powerful external gateways or even data centers. The main benefit is the integration of aggregations of data at multiple resolutions [IS09b], which are gathered along the index structure. The same structure can therefore be used efficiently in two directions: top-down to find detailed measurements and bottom-up for aggregation of larger regions. The aggregation structure has previously been introduced for single sensor node storage and search [GGE+05; GGP+03]. On a single node with limited storage, the method allows to keep an overview of data and graceful degradation of older measurements.

However, even though these approaches focus on support for SNs with a large number of nodes, their direct application to the GSG imposes some limitations. The approaches do not work across different SNs and the query processing is targeted for scenarios without a large number of parallel requests. The concept of hierarchical storage has consequently been extended to address global, web-based sensing systems. In these systems the sensor sources are provided by volunteers to enable a public sensing community. While SensorMap [NLZ06] integrates data through a structured interface, SenseWeb [GKN+07] focuses on the data provided by individual participants. Both systems are targeted to support a web-based interface for human users. The massive number of read requests is tackled with multiple levels of caches along the hierarchical index of sensor data. Together with an intelligent update scheme, the COLR-Tree [AN08] provides close to real-time data for a large number of users.

Unfortunately, a fundamental task is not addressed by the approaches presented so far: real-time streaming data. Such real-time data streams are required by applications which further process the measurements and monitor for or react on changing conditions, for example natural disaster warning systems. Although the streaming of data over

geocentric interfaces has been incorporated into SensorMap [LKNZ08], the multiple levels of caching in the previously described setting lead to additional delays from sensors to the application.

Predictors The problem of providing instant estimates of sensor data, despite any delays on the path from sensor to destination, can be addressed using the concept of predictors. Predictors have initially been proposed to save energy in WSNs by using the processing capabilities integrated into sensor nodes to calculate a prediction of the next expected measurement. The basis for this calculation is either a locally generated model of the data or a model that has been computed on the SN's resource-rich gateway and transmitted to the sensor node prior to normal operation. If this prediction is sufficiently close to the measurement, the sensor node does not need to send an update to the gateway and can therefore reduce communication which is the most energy consuming task in a WSN. In the context of the GSG, the predicted sensor readings can be calculated instantly at the destination and therefore be used to provide real-time estimations of measurements. Furthermore, this reduction of data can be used to prevent link overload or to allow for intelligent load shedding.

Predictors for WSNs have first been proposed with the motion prediction of the popular MPEG video compression in mind [GI01]. Every sensor node is treated as pixel of an image and therefore the value patterns of the entire SN over time can be seen as a video. The approach then directly adapts the motion predictors used in the MPEG video compression algorithm and transfers these predictions to the sensor nodes. This way, each sensor node knows the expected time series for a small time span. However, the collection of measurements from all sensor nodes and distribution of the prediction require a lot of time and communication, which makes this approach impractical.

To eliminate the communication overhead, model-based query processors have been developed [DGM+04; LCLC04; SPJA13]. Instead of distributing a predicted time series to each sensor, these approaches decouple the gathering of sensor information from the delivery of query results. A model of the real world is maintained and constantly updated with new sensor readings. The minimum required confidence in the world model can be specified by the query and if the thresholds are not met, additional sensors are queries for new measurements. As few sensors as possible are selected to satisfy requests with the least possible energy. The correlation of spatially close sensors provides the key to reduce the number of active sensors and can also be used to overcome long delays waiting for timeouts in case of failed sensor nodes.

In contrast to the model based approaches that cover spatial correlations, another class of predictors focuses on temporal value patterns. Early approaches to mine these patterns where based on Kalman Filters [WB06] but they have shown to be too complex and inert. A variety of different time series models has also been used for predicting measurements. PAQ [TM06] uses autoregressive time series forecasting which automatically adapts to the measurements which are fed into the model. PRESTO [DGL+05; LGS06] employs a more complex model known from economathematics: the Box-Jenkins seasonal autoregressive integrated moving average (SARIMA) model. Such SARIMA models are well suited for the application in a hierarchical WSN because the verification of the models can be done very cheap on the sensor nodes. The expensive calculation of model parameters is done on a resource-rich node higher in the hierarchy.

On the other end of the complexity scale, the simple and cheap to calculate least mean squares (LMS) algorithm [SR06] has been employed. More recently, a variation of piecewise linear approximation (PLA) [RCM+12] has shown to achieve 99% reduction in data transmission. Optimal online PLA algorithms with bounded deviation are still subject to current research [LLG13]. Both, LMS and PLA, algorithms are simple enough to be run synchronously on the sensor nodes and the gateways. The need for a special update procedure is eliminated as the models are generated and adjusted online on the sensor nodes. The flow of information is directed only from sensors to the gateways and the models are implicitly synchronized with measurement updates between the two entities. However, since these approaches assume a correct model in absence of sensor updates, special care has to be taken to detect link and node failures. With a generic update mechanism, multiple models can be used in a prediction framework. An adaptive model selection for time series prediction [BSB07] allows for the exchange of the underlying model during runtime.

Another application domain, called object tracing sensor networks (OTSNs), requires a third class of predictors which generate a combination of spatial and temporal predictions. Dual prediction-based reporting [XWL04] has been proposed in this context as a possibly simple solution. The sensor node that initially detects a certain object generates a linear prediction of the objects future movement and sends it to the gateway of the OTSN. When the tracked object changes its direction, the prediction is updated locally and again sent to the gateway. As the object moves out of the scope of the sensor node, the gateway sends the prediction to the neighbor that will have the object in sight afterwards. More sophisticated approaches used behavior profiles [GQP06] and temporal moving patterns [TLH08] to further improve predictions in OTSNs.

2.4 Stuttgart Research Center & Cluster of Excellence Simulation Technology

The GSG project is part of the Stuttgart Research Center (SRC) for and Cluster of Excellence (EXC) of Simulation Technology [SimTech]. SimTech itself is a research project funded by the German Research Foundation (DFG) and the German Federal Ministry of Education and Research (BMBF). The goal of SimTech is to bring researchers from different areas together to improve modeling and simulation technology. The wide range of research areas is targeted towards five visions and divided into six research areas and an additional platform of reflexion and evaluation. This section provides an overview of the entire SRC in general and the context of the GSG project in particular.

Overall Vision The mission statement of SimTech is to go "from isolated numerical approaches to an integrative systems science". This mission is approached through five visionary future application areas of simulation technology: Computational material design, virtual prototyping, overall human model, systems biology, and environmental engineering. In the following, we provide a short overview of the first four visions, and a detailed description of the last vision which includes the GSG.

Computational material design aims to use predictive simulations to systematically develop new materials with specific properties. The modeling and simulation of material on a wide range of time scales provides new insights on how specific macroscopic material properties emerge from microscopic structures. The search for new materials can therefore be guided by desired properties rather than the traditional direct search for new materials which properties are initially unknown. The predictive simulations are also key to virtual prototyping, which aims to provide prototypes for testing purposes without having to built them physically. Such prototypes can be built very quickly and help to massively speed up the overall engineering process. In addition, virtual prototypes incur far less cost than their traditionally manufactured counterparts which leads to a significant increase in efficiency of the development of new machines.

A strong emphasis on modeling is the key to integrate the various simulations of single organs into an overall model of the human body. In contrast to the predictive nature of the first two visions, the goal of an integrated model is to gain a detailed understanding of the human body. With the tight connection of different simulation models, the effects of various environmental influences on the human can be investigated. Similarly, systems biology aims to provide models for biological processes based on

engineering based approaches. The combination of classical biology and systems theory leads to new methods on model generation and parameter estimation. Based on this knowledge, new predictive simulations can then be developed to foresee the effects of new drugs.

The vision of using simulations in the domain of environmental engineering focuses on global phenomena and their effects in real-time as well as for long time scales. To provide interactive analysis capabilities, the highly complex models for these large spatial and temporal scales have to be simplified, in order to be able to show the impact of changed input parameters in real-time. Model reduction techniques are being developed to reduce the order of simulation models without sacrificing the precision of the simulation results. At the same time, uncertainty in input parameters has to be considered to quantify the effects of imprecise knowledge of our current ecosystem on the results of simulations. To tackle this imprecise knowledge, the GSG aims to integrate a possibly high number of sensors to provide high resolution measurements as input to diagnostic simulations. These diagnostic simulations continuously monitor the environment and provide a software representation of the global state of our ecosystem. For example, the simulation of the dispersion of pollutants integrates data from the the state of a simulation for wind fields as well as a simulation for precipitation. The GSG therefore also has to provide the interconnections between simulations.

Research Areas & Project Network As previously stated, SimTech is divided into six research areas and an additional platform of reflexion. All projects in SimTech are commonly located at the intersection of two research areas. For the GSG project, these two are research area E for integrated data management and interactive visualization and research area C for systems analysis and inverse problems. The combination of these two forms the field of analysis of sensor data and real-time control. With connections to both research areas, the requirements from real-time simulation can be addressed using knowledge from data management. At the same time, the findings in the domain of real-time simulations can be used to improve the data management in other applications.

In addition to the single projects, nine project networks provide an integration across the research area to exploit the synergies from different domains for a particular group of projects. The GSG project is located in project network 8: integrated data management, workflow and visualization to enable an integrative systems science. The project network focus is to provide an infrastructure to support simulation scientists of different research disciplines in all areas of data management. Simulation workflow technology is developed

to help in the automated set-up, execution, and analysis of simulation experiments. Existing software modules are thereby reused when building new simulations to allow the development of new simulation experiments based on previous solutions. Visualization modules provide easy to use analysis tools for scientists without a computer science background. Data dependencies and the control flow between the components are automatically set up and managed by data provisioning services. In particular, the Global Sensor Grid Middleware (GSGM) handles the provision of sensor data to simulation modules.

Collaborators During the GSG project, active collaboration with the project network and projects from the adjacent research areas has led to a vivid exchange. The integration with workflows and data provisioning systems has been the main task within the project network. New approaches for the integration of streaming push data into workflows have been discussed. Traditionally, the workflow engine selects web services also based on their location in the network. However, since sensor data is globally distributed, the GSG can direct sensor data to any place in the network. This interaction requires a new type of combined web service selection and placement which has also been discussed in the project network.

The main application scenario was developed in close collaboration with the Institut für Kernenergetik und Energiesysteme (IKE). To provide information in an emergency, nuclear power plants are continuously monitored in real-time. The information about current wind and precipitation is fed into a diagnostic simulation. The information of the diagnostic simulation is then combined with real-time measurements of a SN which monitors the current radiation levels around a power plant. The goal of this collaboration is to enable a real-time monitoring with a spatial resolution which is orders of magnitude higher than the current system. In addition, the time steps of the simulation are significantly reduced. While the IKE is working on a more efficient simulation to provide real-time results for more data, the task of providing measurements at high resolution is carried out by the GSG.

Additional real-time scenarios were investigated in manufacturing environments with the help of the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW). The main goal of the project ReFlow is the real-time simulation of complex material flow systems. In this setting, the challenge is a reconfiguration during runtime rather than a high number of sensors. The approach resulting from this collaboration employs two different communication networks: one for management and control of

sensing infrastructure and another for transmission of measurements in real-time. At the interface of distributed systems and control theory, cyber physical systems have emerged to tackle the problem of distributed control tasks. In cooperation with the Institute for Systems Theory and Automatic Control (IST), the controller placement problem has been discussed, which has led to new insights in the theoretical nature of the more generic network placement [CBD+12].

3 System Overview

This chapter provides an overview of the architecture and the operation of the Global Sensor Grid (GSG). First, the components integrated in and related to the GSG are introduced. The properties of each type of component are characterized and additional assumptions are specified. Second, we derive two query abstractions from the requirements of scientific simulations with respect to sensor data. For each abstraction, the primary use case is described and limitations of existing approaches are discussed. Finally, we describe the entire data processing workflow from individual sensors to simulation applications. The context of each processing task is detailed in terms of involved components and the relation to the query abstractions is shown.

3.1 System Components

The GSG comprises three main areas as shown in Figure 3.1. In the bottom area, gateways and associated sensors which capture the measurements are depicted. A network of brokers comprises the core of the GSG, shown in the middle area. The top area contains monitoring clients and scientific simulations which consume sensor data. In the following, the components in each of these parts and their respective roles are introduced in detail.

3.1.1 Sensors and Gateways

Sensors form the foundation of the GSG as they provide measurements of physical parameters in digital form. In order to further process the measurements, the sensors also provide metadata like the geographical location. To make these measurements available to other systems, each sensor is equipped with a means of communication.

While more complex sensors can be directly accessed, simple sensors require an intermediate gateway. In the former case, a sensor provides its measurements and metadata directly over an Internet accessible interface. In the latter case, sensor nodes form a dedicated network which relays the information to a gateway. The gateway then

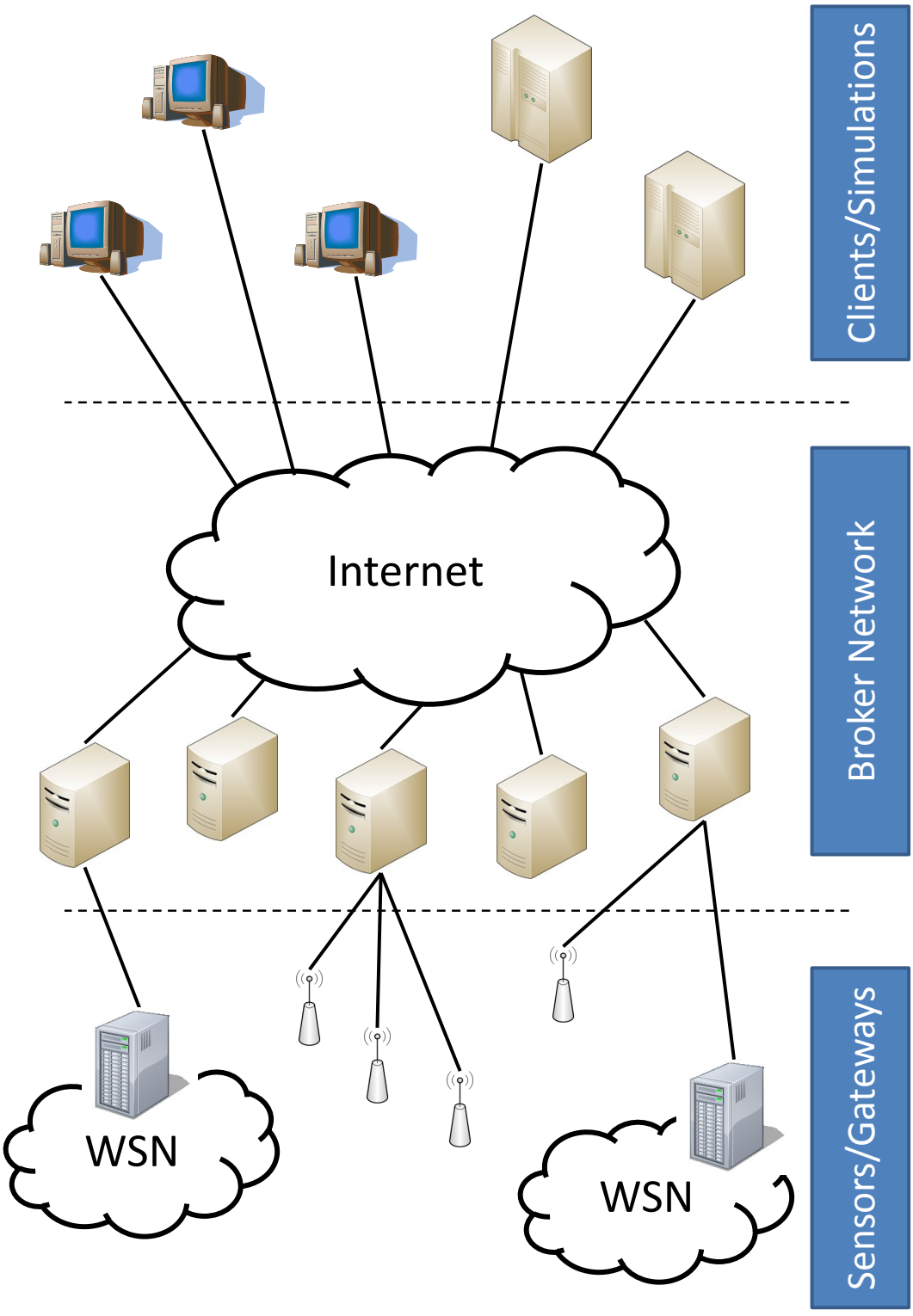


Figure 3.1: Architecture of the GSG.

exposes an interface similar to that of a complex sensor. To access the interface of a sensor or gateway, a corresponding adapter is implemented in the Global Sensor Grid Middleware (GSGM). Generic adapters can be used for interfaces which are described using a standard format like SensorML [BCRG14].

Each sensor or gateway is associated to a single broker depending on its location, as described in the next section. Apart from pushing measurements to the assigned broker, the sensors and gateways do not participate in the operation of the GSG.

3.1.2 Broker Network

The core of the GSG is formed by a network of brokers. A broker is a resource-rich node in the network which executes an instance of the GSGM. In doing so, brokers gather raw measurements from all associated sensors and gateways, process the data, and distribute high-resolution data streams.

To assign these tasks to brokers, the entire globe is partitioned into regions. The partitioning is obtained from the indexing structure presented in Section 5.3.1. Each of these regions is assigned to a broker, which then gathers all measurements from that region. The measurements are further processed to generate a continuously updated, high-density grid of data points, as described in the data processing workflow. Clients and simulations can request the data which is then distributed as full sensor stream. The brokers from which the data originates are also referred to as source brokers.

The computational resources required for processing measurements on a broker correlate with the size of the corresponding region. In contrast to computation, the network load of distributing sensor streams depends on the number and properties of requests. The limiting factor for a scalable GSG is therefore the bandwidth available for distributing data on each broker. Each broker is consequently modeled with a limited bandwidth rather than considering computational or storage resources.

To scale the distribution of measurements with the number and data rates of requests, additional brokers, besides the source brokers, are added to the broker network. The data streams are then distributed over multiple brokers to built a global distribution structure for measurement data. We refer to brokers that forward sensor data streams as relay brokers. Brokers which send data streams to clients and simulations form the end-points of the distribution structure and are called target brokers.

3.1.3 Clients and Simulations

The final destination of measurements are clients and scientific simulations. In order to identify the relevant input data for scientific simulations, monitoring clients request aggregated real-time data from the GSG. These clients, also called peers, participate in the distribution of data to minimize the network load for a potentially large number of monitoring queries. Since the resources required for handling monitoring queries do not pose a significant load on a client, the limits of these resources are not considered in detail.

Once a region has been identified for detailed analysis, the primary use case for the GSG is the distribution of full sensor streams to scientific simulations. As the data might originate from multiple source brokers, multiple streams must first be combined before they are delivered to the simulation. The operator of the simulation thereby has to ensure that the incoming data streams can be handled in terms of bandwidth and computational power, independently of the data source. Therefore, these limitations are not taken into account for the design of the GSG.

3.2 Query Abstractions

To access measurements, the data has to be transferred from sensors to the application at hand. In many current systems, this task is hard coded into the source of the application program and the transmission is carried out using dedicated network links. For weather forecasting, the common approach is even manually entering data into dedicated systems. These approaches have proven reliable and the complexity is acceptable for a few sensors. However, a global sensing system requires an abstraction layer, where interested parties can specify the sensor type and region of interest rather than individual sensors. The sensing system must then take care of gathering the corresponding measurements.

With such a middleware approach, multiple applications can share access to a potentially large number of sensors. Two interfaces are required for this operation: First, the access to sensors is required in order to monitor the measurements of a certain geographic location and sensor type. Requests of this type are called direct sensor queries and are detailed in Chapter 4. Second, the middleware needs to provide access to preprocessed measurements for the supported applications, i.e. simulation workflows in our case. This type, consequently called simulation queries, is covered in Chapters 5 and 6. In the following, we give an introduction to these two main query interfaces considered in this work.

3.2.1 Direct Sensor Queries

The first step in any sensing system is the gathering of data from the physical sensors. This section shows common query and preprocessing abstractions for accessing sensors and sensor networks (SNs) and the variations used by the GSGM for direct sensor queries.

Addressing Sensors First of all, a direct sensor query has to specify which sensors should be selected. Of course, the primary attribute, by which sensors are distinguished, is the quantity to be measured. From an interface point of view, the type of sensor is identified by a single label or a list of labels. If a sensor type is not included in the labels, it will not be considered further in query processing. The filtering of sensors is accomplished by a simple comparison of the labels with the sensor type(s) present on a sensing node. However, the sole limitation to a single sensor type does not provide adequate filtering capabilities to restrict the selection of sensors of interest for a useful query interface.

As previously described in Section 2.3, the geographic location of a sensor is of almost equal importance as the sensor type. A measurement of the right type is obviously hardly of any use, if it is outside the area which is currently monitored. The first extension towards a useful direct query abstraction therefore has to cover the location of sensors of interest. Unlike the selection of a sensor type, location information must be interpreted first. For example, a query can contain location information in different formats like a single point or a geographical region. To prevent ambiguous queries, the GSG only allows queries over rectangular regions.

Especially in the face of energy conservation in wireless sensor networks (WSNs) another property is often used for selecting relevant sensors: the actual value of the measurement. To exploit this additional constraint for energy saving, a query is usually deployed in a SN over an extended period of time. If one of the sensor nodes specified by the main selection criteria registers a value of interest, the single measurement is pushed to the clients instead of continuously polling measurements from all the selected sensors. This basic working principle can also be extended with further preprocessing on the sensors to select measurements based on the change of value over time. In the GSG, this query strategy is used to support real-time monitoring queries on the raw sensor data using predictors.

Basic Preprocessing The most common preprocessing operation used for monitoring queries is aggregation which can combine the possibly large number of relevant measure-

ments into a single date. Another operation, especially for WSNs, are triggers based on observed values allow which save energy by only pushing relevant measurements from sensor nodes. In the following, we briefly describe these basic preprocessing methods.

To monitor a large area, for example an entire forest for a fire, a continuous surveillance of the maximum temperature is sufficient to trigger a more detailed analysis. The average reading of multiple sensors in close proximity can also be used to reduce the noise of the measurements and therefore decrease the uncertainty in the results. This aggregation of data from multiple sensors also has advantages in the energy constrained environment of WSNs. By combining readings from several sensors into a single data packet, energy consuming transmissions can be significantly reduced. As indicated by the previous two examples, different types of aggregation exist. Most query abstractions support average, minimum, and maximum, some approaches also provide the median of measurements. The bias is due to the fact that the first three functions incur very low overhead independent of the number of measurements. The computation of an exact median, however, is much more complex. Consequently, the GSG focuses on simple aggregation functions for monitoring as complex calculations will be performed on the sensor streams provided by the simulation query interface presented in the next section.

The local monitoring of triggers based on the actual values of measurements also provides an efficient way to monitor a large number of sensors with low communication overhead. In some query processors for WSNs, a query can specify extensive calculations to test a predicate on the history of measured values before data is sent. For the GSG, with sensor streams for simulations in mind, the computing capabilities are only used in conjunction with the concept of predictors for direct sensor queries. Unlike the predicates in the previously mentioned query processors, predictors operate transparently on communication links and do not require additional query parameters.

3.2.2 Simulation Queries

In contrast to direct sensor queries which target the raw measurements taken by a sensor, simulations operate on a grid of data points which require a specialized query interface. As the GSG's main purpose is to supply simulations with real-time sensor streams, this simulation query interface is most important for this thesis. The interface provides a complete separation of the data of interest from the physical type and location of sensors on an architectural level. The transformation of data on the processing level is described later in the data processing workflow.

Grid-Based Data Structures The fundamental difference of direct sensor queries and simulation queries can be found in the data structures and the way measurement updates are communicated. This difference has its roots in the nature of a numerical simulation. A simulation maintains its internal state in the form of data points which are distributed over the area which is simulated. Each of these data points is regularly updated according to numerical calculations based on the state of surrounding data points. The underlying functions are thereby modeled to reflect the physical behavior of the simulation. To achieve numerically stable results, the distance between data points is strictly limited.

Because of this internal structure of simulations, updates cannot simply be integrated by setting a physical property of a single data point to a newly observed measurement. If the points were updated this way, the difference to the state of neighboring data points became extremely large, leading to numerical errors in the simulation. In addition, the measurements would require multiple simulation steps to propagate through the entire simulation area. A grid of sensor data is therefore required to update all data points simultaneously in order to prevent disruptions in the internal state of the simulation.

Similar to the problem of isolated updates in the spatial dimensions, an infrequent update with measurements leads to undesired discontinuities in the value patterns over time. The primary requirement for sensor data from the GSG is continuity across the values of a single update as well as across the updates over time. These special requirements for simulations are reflected in the query interface: a simulation requests data updates over the simulated region with a fixed update frequency. The data is actively pushed in a continuous stream to the simulation, which posed the request, instead of being queried repeatedly.

Simulation Resolution The obvious problem arising from the grid-based data structures of simulations is how fine grained the grid of data points is set up. On the one hand, for small scale phenomena like turbulence in wind simulations, the distance of data points must be as small as a single meter. On the other hand, weather simulations to monitor hurricanes have to cover large regions which requires more coarse grained coverage to increase computational efficiency. A similar trade-off needs to be found between high and low update frequencies. These spatial and temporal resolutions also directly affect the amount of bandwidth required to operate the GSG.

With the wide range of simulations to be supported by the GSG, a universally optimal decision on the density of data points and the update frequency cannot be found. Instead,

these spatial and temporal resolutions need to be specified on a per query basis. The most flexible approach for such a specification would be to query for a set of data points, which can be located at arbitrary geographical positions, and to annotate each of these points with an update frequency. However, this approach results in highly complex queries, which are difficult to process efficiently.

The simulation query interface of the GSG focuses on regular grids of data, which can be specified much more concisely. Instead of defining each point of the data grid separately, the simulation specifies the region of interest and the number of data points in this region by a spatial resolution. This resolution determines the density of data points relatively to the maximum density available, which in turn is given by a static system parameter. The update frequency is included in a query in the same fashion as a temporal resolution. The temporal resolution specifies the time steps between each update relatively to the fastest update frequency possible.

Query and Stream Format The simulation query interface is very similar to the monitoring interface to allow for a seamless transition from monitoring to detailed analysis. A query q contains the region R_q and the attribute a_q of interest:

$$q = (R_q, a_q)$$

The attribute specifies the type of data requested, rather than the sensor type, since the diagnostic simulation might also integrate data from other sensors. In contrast to the direct query interface, where a single value is requested for the entire region, a simulation query also contains information about the requested spatial and temporal resolution. The region of a query q is therefore represented by a tuple

$$R_q = (x_{min}, x_{max}, y_{min}, y_{max}, res_x, res_y, res_t).$$

A 2D coordinate representation is used to abstract the physical space and specify the queried regions. x_{min} and x_{max} describe the lower and upper bound for the horizontal axis, i.e. the latitude, of the queried area. y_{min} and y_{max} correspondingly limit the vertical axis, i.e. the longitude. The last three values res_x, res_y, and res_t in the tuple specify the spatial and temporal resolution, respectively. The resolution is specified as a fraction of the maximum resolution provided by the GSG. For example, if $res_t = 0.25$ only every fourth data update is sent to the client.

3.3 Data Processing Workflow

In this section we describe the data processing workflow within the GSG and the attached entities. Based on the interfaces discussed in the previous section, we show the path of measurements from physical sensors over intermediate processing steps to their destination. The chosen path and processing steps depend on the type of query and these differences will be pointed out in the three main steps: First, raw sensor data is acquired from the physical sensors and gathered at specific brokers of the GSG. For direct queries, this data is directly distributed to the interested parties to minimize overhead. Second, the data is preprocessed on these resource-rich brokers by using diagnostic simulations. We present data integration strategies for simulations and provide a more detailed view on the working principle of the simulations. Third, we give an introduction on the generic task of distributing sensor data streams. With the large amounts of data present in the GSG, the main focus of this task is to provide scalability for real-time operation.

3.3.1 Acquiring Raw Sensor Data

The foundation of every sensing system lies at the acquisition of measurements from sensors into the system. In the following, we first provide an overview on the general process of gathering data from a SN. Using this process, a naive solution to a global sensing infrastructure would be to gather all available measurements for further distribution. However, greedy gathering of all raw sensor data at and redistribution from a centralized location obviously does not provide an efficient approach. The GSG therefore operates on a network of brokers to ensure efficient operation. In doing so, each broker gathers data only from a limited geographic region, which limits the number of measurements on each broker and prevents unnecessary transmissions.

General Process Data acquisition starts with the formulation of interest in measurements in the form of a query using the previously described interfaces. This query is eventually forwarded to a broker, which requests the measurements from its associated sensors. An internal interface is provided by complex sensors or gateways, which execute the query processing for a whole SN. Depending on the internal structure of a SN, the query is decomposed into smaller parts for local regions or individual sensors and distributed in the relevant parts of the SN. Active sensors then periodically take measurements and, if all conditions are satisfied, transmit the result to their gateway. As new measurements are taken, updates are pushed towards the gateway, until a query

is changed or canceled. The gateway responds to the broker as the original source of the query with the received data.

Since SNs have been an active research topic over more than the last decade, especially with wireless nodes, a wide range of implementations for query processors exist. The internal details of query processing depend on the SN at hand and are out of the scope of this work. However, the features required for our use of direct sensor queries are provided by all major SN operating systems. The most important feature is the continuous pushing of new measurements on a static schedule or as the observed values change.

Local Monitoring Many applications do not operate continuously and therefore do not require a high resolution sensor stream all the time. Most of the time, monitoring a region for a few aggregated measurements is sufficient to trigger further actions when required. By using a combination of readings gathered from multiple sensors, noise of a single sensor and outliers can be removed in a robust fashion. The GSG efficiently supports such aggregation queries not only over small regions in a single SN, but also on a global scale over possibly many gateways. The additional brokers of the GSG thereby coordinate requests covering multiple gateways.

According to the query interface, a request consists of a region and an aggregation function. To provide high interoperability across many different SNs, the supported aggregation functions are limited to minimum, maximum, and average. The measurements of the entire region are then aggregated using the specified function. The aggregation is carried out in multiple steps and aggregated data is forwarded among participants of the monitoring system. A distributed indexing structure ensures that the sensor sources and existing aggregates can be efficiently identified for a given query. With the increasing number of intermediate steps, however, the transmission delay also grows. This delay is compensated by using our newly developed multi-hop prediction. The full details of this local monitoring protocol are provided in Chapter 4.

Gathering Area-Wide State When a monitoring query indicates an interesting or critical situation, which needs to be investigated in detail, a full sensor stream is requested. The provision of this stream requires preprocessing beyond simple aggregation in order to provide all requested data points in real-time. To transform raw measurements into a high-resolution regular grid of data points, the raw sensor data must first be integrated into the diagnostic simulations running on the brokers of the GSG. Since each broker is

running the simulation for a certain geographic region, the acquisition process needs to transfer raw sensor data from sensors to the brokers for the entire region.

Depending on the density of sensors compared to the resolution of the diagnostic simulation, two different scenarios can occur. If the sensor density is lower than the resolution of the simulation, all sensor nodes in the region of a broker are queried by their respective gateways. The full set of measurements is then gathered at the broker. If the number of sensors is significantly higher than required, the integrated aggregation features of the underlying SNs are used to reduce the number of measurements transmitted. For each data point in the simulation an aggregated value is gathered at the broker. The integration of measurements into the simulation and the possibilities to adapt the sampling rate or the number of data points is presented in the next section.

3.3.2 Preprocessing using Diagnostic Simulations

The task of the GSG is to provide the raw measurements required to steer the diagnostic simulation as previously described. With this approach the problems in gathering data, because of phenomena not measurable or inaccessible locations, can be circumvented. The basic principle of simulations has already been briefly introduced for the simulation queries. Diagnostic simulations form a special type of simulation, which run at the same time scale as the real world, and create a digital reproduction of the physical environment. Note that the details of the modeling and simulation are part of an entire research field and therefore beyond the scope of this thesis. In cooperation within the SimTech research project, the focus was put on simulations for wind fields [RSMF88; Sea00; WWG+05] which can also be run for prognostic purposes [LB98]. However, to keep the simulation from deviating from the physical world over time, the internal state needs to be continuously adapted by integrating correcting measurements. Other approaches for diagnostic simulation focus on a dynamic data driven approach [Dar04; RCM10] to capture these phenomena directly [DLB+06; HFSK06; Hu11] using a large number of sensors. In both cases, the continuous integration of sensor data in real-time rather than processing separate time steps allows to use the full information available.

While simple interpolation between the measurements does not provide additional insight, the processing using simulations integrates knowledge about the system dynamics described by the laws of physics. This way, the temporal aspects as well as spatial relations between measurements are considered. For wind fields, for example, the topology of the simulated area is also included to determine how the air flow is deflected from hills or vegetation. All this information is combined with the internal state of the simulation

and, over time, the behavior of regions without sensor information is forced to adapt to the surrounding conditions. Based on the inertia of physical matter, the current flow of material, and therefore its temporal behavior, is covered, which allows complex behavior to evolve in the simulation. Phenomena, like small turbulences in wind fields, then emerge internally in the simulation, even if they were not directly observed by sensors. The same effect occurs for large scale phenomena, which are driven by sensors in close proximity. The digital state can finally be extracted at virtually any time and any location of the simulation which enables us to provide high resolution data streams for other simulations.

3.3.3 Distribution of Sensor Streams

Similar to the acquisition of raw sensor data, preprocessed sensor data streams are distributed in response to requests issued by simulations. The requests issued by simulations, however, cover a high-resolution, area-wide set of data points. All covered data points must be delivered in regular updates. As a result, high bandwidth consuming data streams have to be organized to serve all requests. The distribution of full sensor streams, as they are extracted from the diagnostic simulation, is the main topic covered in this work. We will therefore only provide an intuition of the overall process and of the main challenges in this context.

The first task is to direct the queries issued by simulations to the broker(s) that provide the requested data. Compared to other processes in the overall workflow, this routing does not require a lot of computing power. The more important resource to consider is the time taken from the initial submission of a query to the first reply. Consequently, the lookup of the corresponding brokers is done using a flat organization structure rather than a hierarchical structure with many levels. In contrast to a centralized entity for query processing, this approach distributes the load among all brokers in the system. The queries are directly forwarded to the source of the requested data streams in normal operation. The broker then sends the continuous data stream directly to the simulation which issued the request to avoid additional propagation delay. However, with a large number of requests for a single broker, the network resources for data distribution might become a bottleneck.

To overcome the problem of overload on a single broker, additional resources of the GSG need to be assigned to help distribute the affected data streams. Since every broker is able to serve multiple clients, the maximum number of clients can be increased by using multiple brokers for the distribution of data from a single region. Multiple clients,

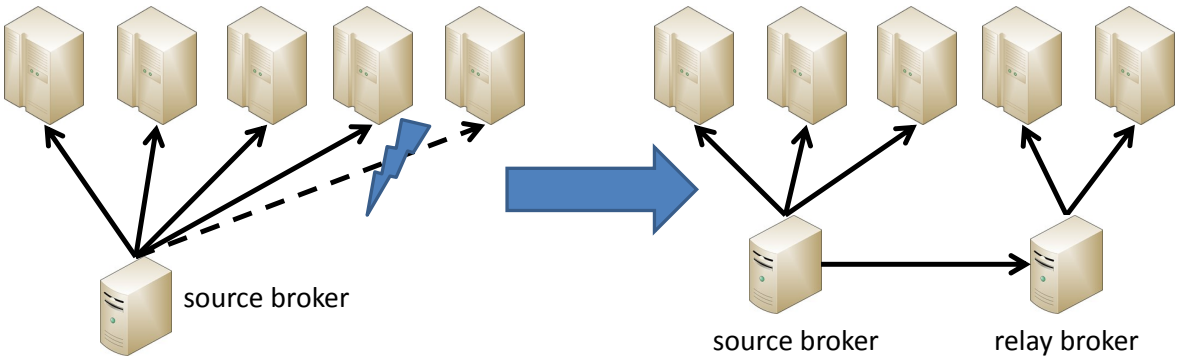


Figure 3.2: Example for offloading data streams using an additional broker. Left: source broker is overloaded by five streams. Right: offloading of two streams with a single outgoing stream.

which require the same data, can be served using a single stream, which is relayed over another broker. Figure 3.2 shows an example how this mechanism can be used to avert overload situations. To offload queries to a remote broker, the original data stream is directed to that broker along with a selected number of queries. That remote broker then serves the queries as if the data of the incoming stream were generated by the local simulation.

The main challenge for this approach is to determine brokers which can be used by an overloaded broker without creating another bottleneck. This challenge is addressed in Chapter 5 of this thesis. Note that the resulting distribution structure is comparable to the multi-hop distribution for monitoring queries. Unfortunately, this also leads to the same problem of increased delay for distribution of data streams as for aggregated measurements. However, unlike for single aggregates, predictors are not suitable for use with the large number of data points in a sensor stream for two reasons: First, the sensor streams are intended for direct integration into simulations which cannot tolerate the deviation incurred by predictors. Second, the computational overhead increases with the number of data points and therefore with the resolution which is not scalable.

To improve the scalability of the GSG, without sacrificing its real-time capability, the bandwidth required for distributing data streams is minimized using a content-aware, network-adaptive approach. This optimization combines the basic principle of relaying data streams over remote brokers with additional information about the underlying network structure as described in Chapter 6. A fine-grained generation of the distribution structure is supported by an improved query processing to efficiently identify overlaps in data streams. In addition, the underlying network structure between brokers is mapped

and the topological location of clients in relation to brokers is estimated. Our approach then selects relaying brokers for data streams in a way which minimizes network load in order to scale to global operation.

4 Real-Time Monitoring of Measurements

In this chapter, we describe how the direct query interface is implemented in the Global Sensor Grid Middleware (GSGM), which allows to acquire data from a wide range of sensor types in large geographic areas for monitoring. We first give an introduction to the general process of data acquisition from sensor networks (SNs) on a global scale. The system model then provides a refined description of the components involved in the monitoring process and their interaction. Based on this model, we then describe the entire process of monitoring as part of the Global Sensor Grid (GSG). We thereby minimize traffic by reusing aggregated data from already existing queries, resulting in the distributed aggregation of sensor data. To provide instant estimates of sensor data and to further reduce the number of transmissions, we present our new measurement prediction method for multi-hop environments. Our evaluations show the performance of our data aggregation and distribution approach and support our choice in the presented prediction model. At the end of the chapter, we discuss related work in the areas of wireless sensor networks (WSNs) and distributed stream processing (DSP). Note that parts of this chapter have previously been published [BKVR10].

4.1 Preliminaries

With the increasing deployment of local SNs, it has become possible to use their data for a wide range of applications on a global scale as part of the GSG. Especially the simulation of natural habitats and ecosystems has gained attention over recent years to better understand global environmental changes. As the popularity of such applications and the number of users increases, the importance to provide a scalable middleware solution becomes evident. A highly efficient continuous monitoring is required to select interesting regions before communication and processing resources are wasted on large scale simulations and the provision of required measurements. The monitoring has to

ensure optimized bandwidth usage and has to avoid the overload of data sources. For this purpose, several DSP systems [AAB+05; FJK+05; GKK+03; SPL+04] that process data streams inside the network were proposed. In these systems, in-network processing, like filtering and aggregation, reduces traffic in the system and therefore improves its efficiency. A generic overview of DSP has been given in Section 2.2.

While these approaches are an important step in contributing to a scalable deployment for applications, which require global sensing, two major challenges remain open. First, the provision of timely data access, despite intermediate processing and communication steps, has to be ensured. Second, large amounts of data need to be handled quickly as many sensors synchronously push updates into the sensing system. These requirements need to be addressed particularly in real-time monitoring, where it is important to directly investigate the impact of environmental disasters on habitats and ecosystems to take countermeasures. However, traditional approaches proposed to enable real-time processing such as network reservation protocols are typically not available at the global scale.

We propose and evaluate an alternative approach to real-time monitoring of sensors based on prediction models. Predictors were originally proposed in the domain of WSNs [GI01] to prolong the lifespan of individual sensors that are only scarcely equipped with resources as described in Section 2.3. To achieve this, the amount of data reported to a WSN gateway is reduced by using predicted values instead of actual readings at the expense of slightly less accurate values. Additionally, an estimated measurement can be obtained instantly without introducing any communication delay. The various prediction models, which have been proposed, widely differ in their employed algorithms and therefore storage and processing complexity.

Existing predictor solutions require a single hop communication between sensors and applications, in order to prevent inconsistent state. However, for the GSG, we present a new approach to hierarchical multilevel sensor data aggregation, which allows efficient reuse of results for multiple recipients. In this setting, the different accuracy in predicted values and applicability of existing predictors remained unclear. Even for SNs there exists no classification, which prediction model is best-suited for a certain type of data (e. g. temperature readings or wind speed). This question becomes even more difficult with in-network processing in a global multilevel context.

We demonstrate our new distributed R-Tree-based aggregation algorithm optimized for data reuse and how multilevel predictions can be integrated into this monitoring system. The performance is evaluated on the basis of temperature sensor readings of the National

Oceanic and Atmospheric Association [NOAA] and the German Weather Service [DWD] taken during a whole month. In addition, the Intel Lab Data set [MITLAB] was used to compare the results to existing approaches. Complex prediction models were expected to perform best since they capture the daily recurring features of sensor data. In contrast, our results show that computationally simple models provide the best performance, with respect to saving bandwidth, due to similar accuracy at reduced update volumes.

4.2 System Model

We start the description of the real-time global monitoring function of the GSG with a refined view on the system introduced in Chapter 3. The monitoring implements the direct sensor query interface and provides data to monitoring clients, simulations are not considered for the monitoring.

Clients participate in the monitoring by sending a query to either a broker or another client of the GSG. Every client that submits a direct sensor query to the GSG is thereby automatically integrated in the monitoring system and contributes to the maintenance. The coordinates of a query are specified as two pairs of latitude and longitude. A query also includes the requested type of sensor data and an aggregation function, which is used to combine multiple measurements to a single reading. This way, the task of data dissemination can be distributed among clients and aggregated data can be reused for matching sensor types and aggregate functions, as will be seen later. With this reuse, the monitoring of sensor values can be achieved at very low communication cost.

To further increase efficiency, predictors are established on every data link in the system. These predictors provide expected future measurements based on the data observed so far. Both sender and receiver agree on the same predictor setup and synchronize their operation. Now, when the predictor on the sender produces an estimation that differs less from the actual measurement than the given threshold, no data needs to be transferred to the receiver. The detailed description of the operation of predictors is given in Section 4.4. Together with the forwarding of reused results between clients, multi-hop prediction links emerge. A simple example for the multi-hop prediction and aggregation is given in Figure 4.1, the details are covered in Section 4.4.

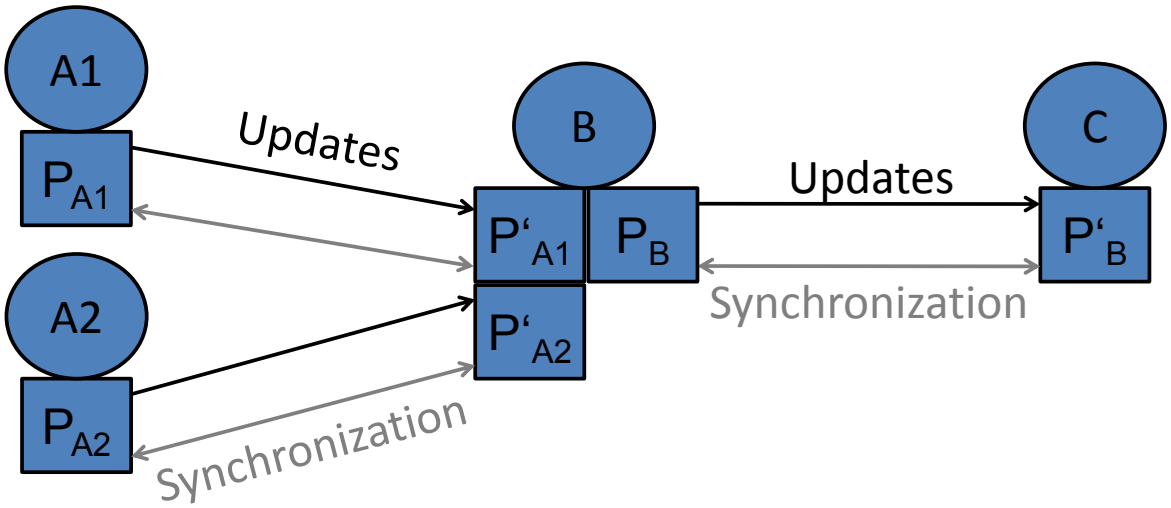


Figure 4.1: Simple example for multi-hop prediction in a distributed aggregation structure. The predictors form the endpoints of each link, node B uses data from P'_{A1} and P'_{A2} and provides the result to P_B.

4.3 Distributed Aggregation of Sensor Data

Based on the refined system model of the previous section we now detail our monitoring which is based on the distributed aggregation of sensor data. We first describe the distributed indexing structure based on the R-Tree, which is used as the foundation for lookup and routing of data. Afterwards, we describe the query process and how data of interest is located in the system. The integration of new clients into the system is then described along with the management of partial aggregates for reuse.

4.3.1 Basic Indexing Structure

Our query model allows for queries to specify a geographical area rather than explicitly stating individual data sources. A distributed index structure is therefore used to efficiently discover relevant sources for a query. We use this structure not only to store the locations of single sensors, but also to store the queries in order to identify their relative data dependencies. The foundation of this index structure is provided by the R-Tree [Gut84], which is commonly used for indexing points and rectangular shapes in database systems.

The basic structure of the R-Tree is initially built up as follows: Each node in the tree represents a minimum bounding rectangle (MBR) of all items in the sub-tree corresponding to that node. The number of children stored in a node is thereby limited to

increase efficiency by maintaining a page aligned memory layout. Starting with a single node, the MBR of that node is extended with newly added items, if required. When the number of items increases beyond the limit, they are separated into multiple groups using a clustering algorithm. Each of the groups is then assigned to a new node and the MBR of that node is updated accordingly. A new parent node, which contains all the data items, is added to store the newly generated nodes as children. In addition to this split operation, which is used to create new nodes into the tree as it grows, data items are re-inserted on overflow of nodes to ensure a balanced distribution of items among nodes.

The resulting tree structure has the following properties: 1) Data items are only stored in the leaf nodes and grouped in a way that tries to minimize the MBR of each leaf. 2) On the way from a leaf to the tree's root, each parent vertex manages a MBR that covers all its children's MBRs. 3) The R-Tree can be distributed by assigning the nodes of the tree to different hosts on the network.

As in the R-Tree, individual sensors are stored as data points in the leaf nodes of the index. In contrast to the R-Tree, however, the structure is not built by adding individual sensors to the index, but by first creating the leaf nodes. Each broker thereby maintains a leaf node with a MBR that corresponds to the area assigned to that broker. One broker is selected to maintain the root of the tree, which represents the entire world. If a single root broker became a bottleneck, the root node could also be replicated among multiple brokers to balance the load among them. The nodes representing the broker regions are then added to the tree. In the case of a split, the newly generated nodes are randomly assigned to one of the brokers whose node is contained in that region. Note that due to the disjoint nature of the regions assigned to brokers, a split operation will result in child nodes with disjoint MBRs.

Figure 4.2 shows an example for the relation between the geographical alignment of MBRs and their corresponding nodes in the tree. On the lowest level, the leaves of the basic tree are formed by the regions maintained by the brokers of the GSG. For the sake of simplicity, the brokers are not depicted in the figure, but rather identified by the index of their region. Each node of regions R_1 through R_4 is therefore assigned to its corresponding broker, B_1 through B_4. The intermediate node representing R_5 can be maintained by either B_1 or B_3, for this running example we choose B_1. The root node represents all items contained in the tree and correspondingly the entire geographic space covered by the index. Any broker could maintain the region R_6 of the root node, for the example, we choose B_2. The regions represented by the children of a node are disjoint

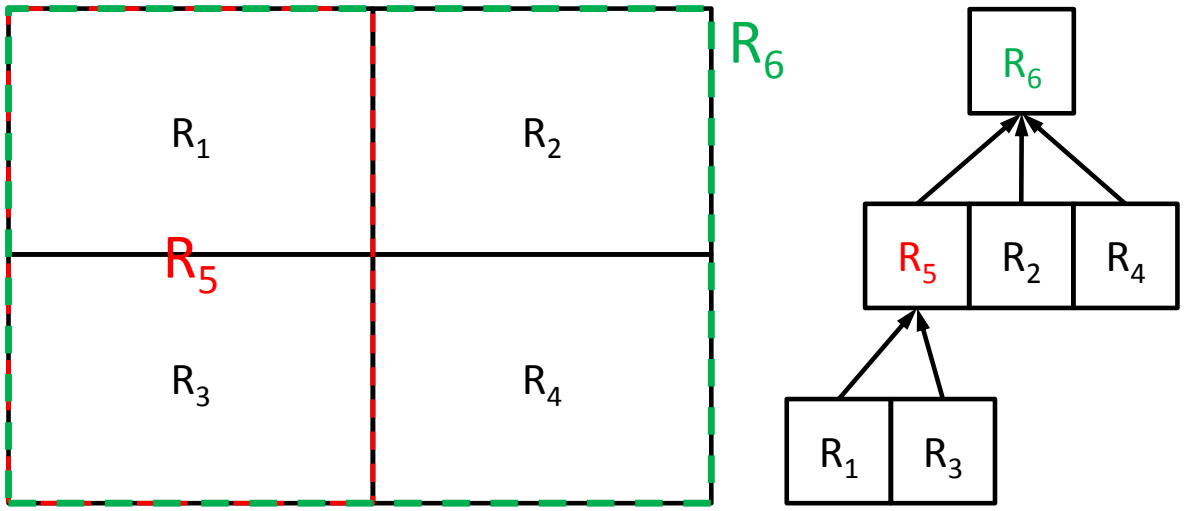


Figure 4.2: The connection between the geographical alignment of MBRs and the position of their corresponding nodes in the basic index tree.

and completely contained in the MBR of that node. The structure can therefore be used to efficiently identify relevant sensors, by traversing the tree top-down.

Besides identifying the sensors relevant for a query, the index structure is leveraged to identify intersections between new and already existing queries. If such an intersection is found, the measurements of the covered sensors are relevant for both the new and the existing queries, resulting in a data dependency. Based on the identified dependencies, clients can reuse the query results which already contain aggregated data. In the following section, we first describe how queries are routed and inserted into the index to identify dependencies. The routing and aggregation of data, based on the dependency information, is detailed thereafter.

4.3.2 Query Routing and Insertion

The containment property of the tree is essential to identify dependencies between queries. We consequently insert queries in the tree at arbitrary levels, rather than storing them in the leaves to identify their relation, as we traverse the structure. However, to maintain the containment property in the presence of arbitrary query intersections, queries are not stored as individual nodes. Instead, a separate R-Tree is attached to the corresponding node, which holds all queries in that region. The results of queries that are already running are then available in the index by their region of interest.

The routing process starts with a new client sending a query to any broker of the GSG. On reception of a new query, the receiver compares the query to the MBRs of the nodes it is currently managing. Recall that in addition to the MBR of its associated SNs, a broker might also manage larger regions on intermediate nodes of the tree up to the whole sensing system at the root. Leveraging the containment property of the tree, multiple local nodes are considered, ordered by increasing size of the respective MBR. If the query is not contained inside any MBR, the receiver passes the query to the parent of the node with the largest MBR in the tree. Due to the structure of the tree, that parent node must be located on another broker.

Eventually, since the regions grow towards the root, the query will reach a node whose MBR completely contains the query. From this point on, the lookup continues down the tree by selecting the child node which contains the requested area. This containment is true as long as the MBR, managed by that child, completely contains the region of the new query. If the child node is located on a remote broker, the query is forwarded to that broker. The lookup is finished, when a node is reached where no child contains the entire region of the new query. This node is called destination node.

The broker maintaining the destination node initiates the insertion as follows: First, consider the case where the destination node is a leaf node of the index. In this case, the maintaining broker can directly identify the relevant sensors for the query. Based on the query region, the attached R-Tree is searched for an existing query. If an existing query is found, the client which posed the new query is appended to the destinations of that entry. If no existing query is found, a new entry is added.

Second, consider the case where the destination node is an intermediate node of the index. Again, the attached R-Tree is searched for an existing query and the corresponding entry is updated, if present. If no existing query is found, the new query is added. However, the relevant sensors cannot be identified locally and, consequently, the query cannot be answered. The query is therefore split, according to the regions of the child nodes in the main structure. A new query is then issued to the brokers maintaining the child nodes for each query part. If required, these parts are further split and forwarded accordingly. Eventually, either all parts are covered by existing queries or the sensors of a query can be determined on a broker maintaining a leaf node as described earlier.

Consider, for example, a query $Q1$ which requests parts of the regions R_2 and R_4 and is sent to broker B_1. As in the previous example, each broker maintains its own region, in addition, B_1 maintains R_5 and B_2 maintains the root node R_6. The final result is shown in Figure 4.3 and built as follows: The broker B_1 first checks region R_1 followed by R_5

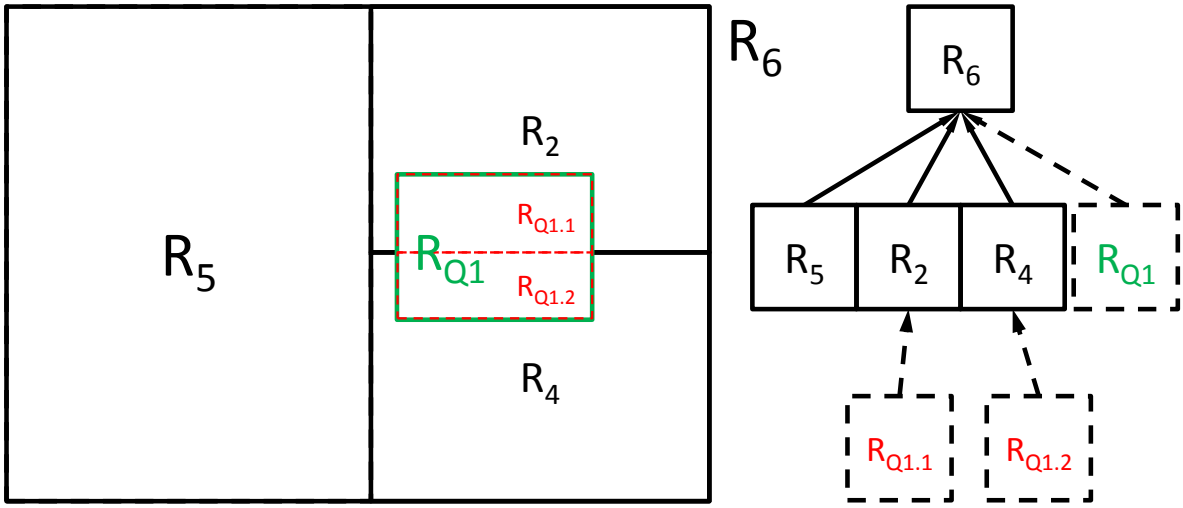


Figure 4.3: Example index structure after inserting a query. Regions R_1 and R_3 of the previous example are omitted for clarity.

which both do not contain R_{Q1}. The query is therefore forwarded to B_2 which maintains the root node. As no child of the root node contains R_{Q1}, B_2 starts the insertion process and adds $Q1$ to the query R-Tree of node R_6. To gather the data from the brokers, one additional query is created for each of the underlying broker regions. The first query, represented by region $R_{Q1.1}$, is pushed down the tree and added at node R_2. The second query, shown as $R_{Q1.2}$, is forwarded to B_4 and then added at node R_4. Both partial results as well as the complete query region can now be identified for reuse during the next insertion process.

4.3.3 Data Routing and Aggregate Calculation

While the query routing procedure traverses the index tree in a top-down fashion, after finding the first matching node, the sensor data needs to be aggregated bottom-up, starting from the sensors. The measurements of individual sensors are gathered by brokers, according to their assigned regions, as described in the system model in Section 3.1. Each broker then calculates the aggregate of measurements requested by the queries stored in the lowest level of the index structure.

Instead of serving all clients listed as destinations for a query directly, the clients are integrated in the distribution process as follows: When the list of destinations for a query is updated, the maintaining broker sends the updated list to each of the clients on the list. By keeping the order of the items in the list constant, a client can therefore identify

the next, as well as the previous, destination in the list. The aggregate computed on a broker is then only sent to the first client of the destination list. Each client then relays the data to its successor until the final client in the destination list is reached. To prevent the breaking of the chain, clients regularly check the state of their predecessor and successor in the destination lists.

The distribution process is also applied for queries stored in intermediate nodes of the index structure. The clients are informed of new destinations as for the queries on a single broker. As the data for these queries originates from multiple brokers, the data for the first client on the destination list originates from the partial queries described in the previous section.

With our approach for the distributed aggregation of sensor data we satisfied the main goals for the monitoring system: The underling index tree structure provides an efficient way to identify data sources for monitoring queries. Sensors of different SNs can be integrated into the global structure, which is distributed among the brokers of the GSG. This distribution provides the scalability necessary for a large number of concurrent monitoring queries. To lower the overall communication footprint of our approach, we exploit the results of running queries to feed larger monitoring queries. Brokers are thereby alleviated of distributing their data to each and every consumer by themselves.

4.4 Prediction in Multi-Hop Environments

While the aggregation structure, introduced in Section 4.3, provides efficient sensor lookup and reuse of data streams, it does not optimize for low latency. Especially with the aggregation over multiple steps and the transmission of partial aggregates over multiple hops, the delays between the reading of a measurement and the final delivery of an aggregate become crucial. To overcome these delays and to provide real-time estimated results, predictors are integrated into our monitoring system. An additional advantage of using predictors for the transmission of data is the possibility to further reduce the amount of data transmissions required to run continuous monitoring queries. In this section, we first describe the basic working principle of predictors for different underlying models and how their capability of providing real-time estimates can be used for data reduction. We then show how predictors are integrated into our distributed aggregation structure and describe our new multi-hop update strategy.

4.4.1 Data Reduction using Predictors

Predictors were initially proposed in the context of WSNs, as described in Section 2.3. The basic idea of a predictor is to use a mathematical prediction model, which captures the change of a sensor reading over time, to estimate the next reading. In the context of WSNs, the main goal is to reduce communication as it is the primary cause of energy consumption. This is achieved by using a predictor as follows: The prediction model is not only maintained at the destination, but also directly at the sensor node. The two models are synchronized, so that both produce the same estimates of future readings. If the sensor now measures a value which is sufficiently close to the estimated reading, no data needs to be sent to the destination which instead relies on the estimate. To keep the two prediction models in sync, the predicted value is also used as an input to the predictor on the sensor.

In the GSG, we use predictors not only for the communication between sensors and brokers, but on all network links used for monitoring. This way, the communication overhead required for operating direct queries is further reduced. In addition, the predictor on a target node can provide and estimation of the current measurements without any delay.

Prediction Models

The core component of a predictor is the mathematical model used to capture the time series of measurements and calculate estimates of expected readings. Virtually all prediction models keep an internal state in the form of a set of model parameters u_i and a set of recently received measurement data m_t, where t denotes the time the measurement was received. The number of model parameters is commonly referred to as the order of a predictor. The time is measured in discrete, equidistant time steps and the value of the next prediction at time t is denoted v_{t+1}.

The main difference between prediction models lies in the way the model parameters are calculated. Two main research areas have focused on the generation of mathematical models which cover the evolution of data over time. From the business background, approaches for generic time series modeling and analysis [BJR08] have been proposed to predict long term trends, for example the value of stock exchange prices. In the context of real-time control, short term deviations need to be predicted in order to differentiate between noise and a change in the observed state. Once the parameters have been

obtained, the next prediction is obtained by combining the model parameters and recent measurements.

We analyzed two representative prediction models from the WSN domain for their applicability in the GSG: The least mean squares (LMS) algorithm [SR06] provides a simple and fast alternative with low memory and computational overhead as required in its originating numerical control domain. In contrast, the seasonal autoregressive integrated moving average (SARIMA) [LGS06] model represents the complex approaches originating from time series analysis, where long term trend analysis is more important than fast generation of model parameters. In the following, we describe parameter and prediction calculation of both models in detail.

LMS Model The LMS algorithm in general is an adaptive filter commonly used for signal processing. The basic method was first presented in 1960 [WH60] in the context of logical switching networks. Since then, the LMS algorithm has been studied extensively in various contexts [HW05; Mac95]. More recently, a predictor based data reduction strategy [SR06] based on LMS has been proposed for WSNs.

The idea behind the LMS algorithm is based on a stochastic gradient descend method. The adaptive algorithm thereby ensures that the model parameters are updated in such a way that minimizes the LMS of errors between the predicted values and the actual measurements. Each update only relies on the error at the current time rather than a previously observed history of measurements or errors. In the application scenario at hand, the target value of the LMS filter is the next measurement rather than an external actuating variable.

In the generic version of the LMS algorithm, multiple input values can be considered. However, for a single sensor, the update strategy can be greatly simplified. As a result, the computational overhead of the LMS algorithm is very low and it can be applied on every single sensor stream in the system. We use the following simplified procedure to update the LMS model parameters in each time step. For each parameter u_i its new value u_i' is calculated based only on the absolute error of the last prediction $e_t = m_t - v_t$ and its corresponding last measurement m_{t-n+i} as follows:

$$u_i' = u_i + \mu e_t m_{t-n+i}$$

n specifies the number of parameters, also called the order of the model. The adaptation speed of the model is controlled by μ. The value of μ is crucial for the stability of the algorithm [CL83] and needs to be set, depending on the variability of the input data.

However, this variability can be adaptively determined during runtime and μ can be updated [MH00], if necessary.

The last n measurements are kept in a FIFO queue not only to calculate the updated model parameters, but also to predict the next value. Based on the model parameters u_i and the last n measurements, the filter output is calculated as a weighted sum:

$$v_{t+1} = \sum_{i=1}^{n} u_i m_{t-n+i}$$

As previously noted, the filter output directly represents the value of the next prediction.

The LMS algorithm can quickly adapt to the observed measurements, especially if a low order model is chosen. Also, depending on the order of the model, the algorithm has very little computational overhead and storage requirements. These two criteria make the LMS algorithm especially suitable for the application directly on the sensor nodes in a WSN or to handle numerous sensor data streams concurrently on a single broker or client. However, the model is not capable of capturing repetitive patterns in measurements, like the change of temperature during the day.

SARIMA Model In contrast to LMS, a SARIMA model is specified by two statistical processes, one of which covers the seasonal part while the other models the general trend. Both processes consist again of an auto-regressive and a moving average part. Individual SARIMA models differ in the order of differencing that is used to estimate the current trend in the observed data for the general and seasonal part. The order of a SARIMA model is consequently given by a set of parameters $(p, d, q) \times (P, D, Q)_S$, where p and q denote the orders of the auto-regressive and moving average processes, respectively. d is the order of differencing. P, Q, and D represent for the corresponding values of the seasonal components. Finally, S represents the period of the seasonal component in terms of the number of observations per season. The model parameters can be estimated using several strategies [BJR08] which are covered by the entire field of statistical modeling. An exhaustive investigation of these strategies is therefore beyond the scope of this work.

We will focus on the description of the actual model considered for use in the GSG. To determine a suitable model for our purpose, consider the following: The wide range of combinations of orders for the seasonal and autoregressive part of the overall model is intended for an equally large number of time series of interest. In a sensing system, especially for the prediction of a single sensor value, the time series to consider only has

a single input parameter, which is the measurement of that sensor. The SARIMA model is therefore restricted to a one-dimensional version.

However, the order of the model and the actual implementation still offer a wide range of different complexities. Note that in general, the parameter estimation, which will be described later, for a SARIMA model takes a considerable amount of computational effort with an increasing order of the model. To employ a SARIMA model for a large number of concurrent sensor streams, the order of the model has to be limited. Fortunately, multiple sensor types, like temperature or pressure sensors, report data which is only changing gradually over time. This observation is supported by previous research [LGS06], which has shown that the measurements of temperature sensors, for example, can be best matched with an order of the SARIMA model of $(0,1,1) \times (0,1,1)_S$. Our research has shown, that this order is also best suited for other common weather measurements, as covered in our evaluations. This model completely eliminates the auto-regressive part and only incorporates a linear change in the two components of the model. The number of measurements S in a seasonal period depends, of course, on the frequency at which measurements are taken. For example, the outdoor temperature might follow a daily pattern due to the course of the sun, resulting in 86.400 measurements if they are taken each second.

The estimation of model parameters poses an inverse problem, as we have to identify the values for the best fit of the model based on a given set of measurements. To describe our implementation of parameter estimation, we therefore first introduce how a new prediction is computed. As described earlier, the computation of a prediction for the SARIMA model takes into account S historical measurements and their respective prediction errors. With the model order of $(0,1,1) \times (0,1,1)_S$, predictions are computed as follows:

$$v_{t+1} = m_t + m_{t-S+1} - m_{t-S} + u_1 e_t - u_2 e_{t-S+1} + u_1 u_2 e_{t-S} \tag{4.1}$$

Note that the calculation of a single prediction comes at little computational cost. However, since an entire season of previous measurements has to be stored, the memory overhead is much higher than that of the LMS model.

The two model parameters for a given time series of measurements are now calculated by minimizing the mean square error (MSE) of the predictions over that time series. As described previously, this poses an inverse problem, since the optimal parameters cannot be identified directly given the time series but requires the evaluation of model parameters and their adjustment. Initially a set of values is chosen for the parameters and the corresponding MSE is calculated. Since the MSE is a convex, two-dimensional

function over the model parameters in our special case, we then choose the next set of parameters in an iterative nested intervals method. This method guarantees a quick convergence of the parameters to their ideal value. If the step size has become sufficiently small and, therefore, the parameters have been identified with an equally high precision, the process is terminated.

Since the sensors continuously report data, an approach for periodically refining the model parameters, to capture qualitative changes in the observed measurements, is required. The parameters are therefore optimized, by using the above method during training phases, which are triggered, if the overall error of the predictions becomes too large. Again, the convex nature of the error in dependence of the model parameters can be exploited to further speed up the identification of new parameters by using the old parameters during the initialization. However, during each training phase, a significant number of measurements is required to generate valid parameters. In terms of the main goal of reduced network communication this leads to the problem of deciding where to calculate the parameters.

Operation Modes

To use the predictor concept to reduce the number of measurements transmitted, the source and the sink of the corresponding link agree on the same prediction model along with the according set of model parameters. They also have to agree on the same history of measurements and prediction errors in order to ensure the same predictions. This can be done during an initial setup phase, where measurements are directly sent to the sink, until a certain history of data is available at both, the source and the sink. Then, during normal operation, they independently estimate the next measurement in each time step. Only if the source detects that the actual measurement deviates more than the user-given threshold from the prediction, it sends an update to the sink. If the prediction was sufficiently precise, it is added to the set of recent sensor measurements and the model parameters remain unchanged.

Although the generation of model parameters is the most expensive part of predictors, with respect to the processing time, this cost is only a limiting factor on battery-driven sensor nodes. All brokers in the GSG, as well as all the clients participating in the monitoring query system, are powerful enough to perform the identification locally. Model parameters can therefore be computed on the source, the sink, or both allowing for the operation modes depicted in the following paragraphs. This freedom also allows a flexible decision on how much information is exchanged for an update between two correspondent

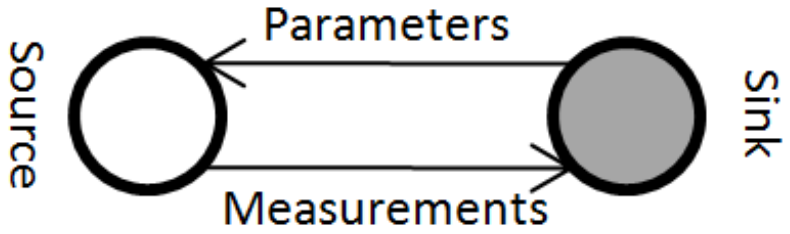


Figure 4.4: Sink-based operation of a predictor. During initialization and readjustment, measurements are sent from the source to the sink, where the prediction parameters are generated and sent back to the source.

predictors. Possibilities range from single measurements over a set of model parameters to a complete history of data. The choice of the operation mode and update extent has of course a high influence on the overall amount of data transmitted for monitoring queries. In the following, we describe the possible operation modes in more detail and we elaborate on their benefits and drawbacks in different scenarios. Section 4.4.3 will later provide the details of the update strategy used in the GSG.

Sink-Based Operation The sink-based operation, shown in Figure 4.4, requires the transfer of measurements to the sink where predictor model parameters are computed. Model parameters are then transferred back to the data source, where the prediction is calculated and compared to the actual measurement. If the prediction does not fit the measurement, the source can send a single measurement to compensate for outliers. After a number of consecutive updates, a new update cycle is initiated, causing a fixed number of measurements to be transferred to the sink.

Obviously, the amount of measurements needed for a full update depends on the predictor at hand. For a model with explicit parameters like SARIMA, a possibly long time series is required, depending on the order of the model including the length of a season. For self adapting predictors like LMS, single measurements can be sufficient to adapt to the qualitative change in input, eliminating the need for the separation between source and sink. This approach is therefore only viable for predictors which are not capable of adapting their model from single measurements.

Sink-based operation is especially suited for the resource-constrained environment of WSNs. The transfer of the expensive computations to the sink of the network allows for the application of complex models by unburdening the sensor nodes. However, for the nodes participating in monitoring using the GSG, computational power is not a limiting

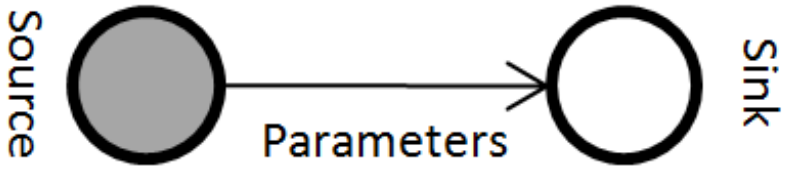


Figure 4.5: Source-based operation of a predictor. Only model parameters are sent to the sink, shifting the computational load to the source node.

factor. In terms of bandwidth usage, the transmission of entire time series *and* model parameters requires highly accurate models to create an overall advantage.

Source-Based Operation Contrary to the sink-based operation, which stems from the asymmetry in computational power on WSN nodes compared to their gateways, the source-based mode depicted in Figure 4.5 follows the flow of data in the GSG. In this case, model parameters are only computed on the data source, where all actual measurements are available. This approach always requires the transmission of the entire parameter set to the sink node. Note that for a seasonal model the last season of measurements is also required to calculate the next prediction. The last season of predictions can also be used for this calculation as a substitute. However, a qualitative change in measurements during a season, which usually triggers an update, requires the transmission of that season along with the model parameters.

Although the full update still requires a considerable amount of data, again depending on the type and order of the predictor, as well as the properties of the measured data, the source-based operation has an important advantage compared to the sink-based version. The source of a predictor link always has all unfiltered measurements available as input. Therefore, multiple models can be simulated concurrently to the the one currently used on the source node and the update type can be selected, depending on their respective behavior. For example, the source node could be running an additional model with updated parameters on an old season of data. If the model performs well, the source node can just send the new parameters and can skip the transmission of the time series.

Symmetric Operation The location of the parameter generation and transfer of measurements for this symmetric operation is shown in Figure 4.6. Similar to the source-based operation, an update of model parameters can be initiated by the source node, commonly after a predefined number of consecutive predictions has failed. Unlike

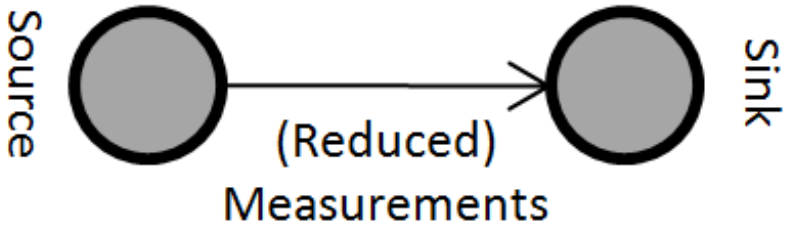


Figure 4.6: Symmetric operation of a predictor on the source and sink node. A reduced number of measurements is transferred to synchronize the predictor models by small adjustments rather than regenerating full model.

the source-based approach, however, the models are synchronized using single to few individual measurements, up to entire time series of data. The source and sink can then both calculate the model parameters from the newly available data.

The symmetric operation thereby further extends the flexibility of the source-based approach. Since updated parameters can be calculated on both nodes, the measurements, which serve as input for the new calculation, can be freely selected and transmitted by the source node, even without triggering an actual update of the model parameters. For example, if the predictions on the source node are increasingly deviating from the measurements, but do not yet violate the predictor threshold, an update can be sent containing any number measurements or model parameters.

The determination of the content of an update, however, becomes extremely difficult. On the one hand, with data that can be predicted with high accuracy over long time periods, updates of larger blocks of measurements can become profitable over single measurements. On the other hand, a continuous slight adaptation with single measurements, which prevents violation of the predictor threshold, can adapt better to changes in observed measurements. A trade-off, which allows the most effective operation of the predictor, has to be found between the reduction of messages and transmitted information.

4.4.2 Integrated Aggregation and Prediction

So far, we have shown how the GSG supports the distributed aggregation of data using a multi-level indexing structure and how predictors can be used on a single link to reduce the amount of data transmitted. To provide a scalable real-time monitoring query interface, the two approaches are combined, by establishing predictors on the

communication links, which are used for the distribution of aggregated data. In this combined form, predictors operate on an aggregate of measurements rather than on a single sensor and the aggregation must be adapted to work with predicted values. The integration of the two methods to the overall monitoring query system of the GSG is described in the following.

Recall that data from a set of input values is combined, using the specified function on each aggregation step. If a new value were sent out for each new incoming information, the aggregation would not result in any reduction of bandwidth use, since the number of outgoing data items would equal the sum of incoming items over all links. Furthermore, this approach prevents the application of predictor models with fixed time steps between updates. Therefore, the system is operated using a fixed update frequency on all aggregation nodes. This frequency is also used to fetch measurements from physical sensors, to avoid excessive sampling. As updates need a certain time to be delivered to the next aggregation, each aggregations results in an additional delay of one cycle. Without predictors, this strategy could therefore lead to large delays when multiple aggregation steps emerge in the distribution structure. Real-time estimations of the current state can, however, always be queried from the local end of a predictor for both incoming single items and outgoing aggregated data in the GSG.

The symmetric operation allows a transparent integration of predictors on all communication links. Instead of using a cached version of the last measurement on each incoming link, the predictor for that link is sampled, when a new outgoing value is calculated. The aggregation is applied exactly as if the incoming values had been received over a regular communication link. The newly calculated output is then fed into the local predictor for the outgoing link. Unfortunately, this approach only partially solves the problem of a growing delay over an increasing number of aggregations. A maximum deviation can only be guaranteed when the updates have been propagated across all predictors. To circumvent this problem, we developed a multi-hop update strategy presented in the next section.

4.4.3 Multi-Hop Update Strategy

As previously described, predictors are employed on every data link between clients in our multilevel aggregation setup. Updates are thereby only distributed after the next aggregation was calculated, due to the transmission delay and fixed update frequency. The maximum deviation of a single predictor can therefore not be guaranteed, if the multi-hop aggregation structure is not considered when sending updates. Furthermore,

the update strategy in our setup has to be bandwidth efficient to enable a monitoring system, which is scalable to a large number of clients. The update strategy not only decides when to actually send a value over a link instead of relying on the prediction, but also specifies what data is sent to update the predictor model and state.

Update Cascades To ensure that predictors do not exceed their allowed threshold, updates must be forwarded immediately, rather than with the next cycle. The situations, in which an update must trigger another update, depend on the effects of the intermediate aggregation steps. Such a sequence of immediate updates is called an update cascade.

Aggregation functions can thereby be classified into two categories. In the first category, a new or updated input of a single value has only little effect on the final result. A common example is the average function, where the influence of a single measurement decreases inversely proportional to the total number of measurements considered. The second category comprises aggregation functions, where a single measurement can directly change the output of the function. The noticeable examples for this category are the minimum and maximum functions.

The average aggregation function equally depends on all incoming measurements. Since the outgoing predictor operates on this aggregated data, the prediction model can benefit from the smoothed input data where one incoming predictor might predict a value above the average while another one follows a decreasing trend. Especially by using the average operator, prediction errors with opposite sign mutually compensate and result in smaller prediction errors on the outgoing data link. Consequently, an update initiated from a single sensor does not represent a qualitative change and therefore does not need to trigger an update cascade.

The direct dependence of the min/max aggregation can intuitively be explained by the fact that a user with such a query is actually looking for a single measurement, rather than aggregated information. This behavior is difficult to predict by the outgoing predictor, since the previous trends of individual incoming values are not known from the result of the aggregation. Therefore, a change in extreme values can generally only be predicted as long as the sensor which reads this extremal value does not change. If an update is triggered due to an excessive prediction error, the deviation indicates that another sensor has observed the new extremal value. In this case, all affected clients must be notified of the qualitative change by triggering an update cascade.

To support both types of aggregation in a uniform way, a generic update strategy is used in the GSG. With each incoming update, the aggregation is evaluated and an update

is sent immediately if the deviation between the new result and the outgoing predictor exceeds the system wide threshold. The predictor model, however, is not updated until the next measurement cycle is over. Although this strategy can result in multiple updates in a single cycle, it is required to ensure that predictions do not deviate beyond the specified upper bound. To enable the presented strategy, predictors are operated in symmetric mode in the GSG.

Update Contents As indicated at the description of the operating modes, updates on a predictor link can contain a wide range of information with the symmetric operation in the GSG. On the side of the spectrum, single measurements can be used to minimize the data per update. On the other side of the spectrum, entire time series can be exchanged to improve future predictions. However, with the update cascades, which can be triggered by single measurements, a large amount of data per update can quickly lead to high and bursty bandwidth consumption. Furthermore, the benefit of updating the entire predictor state, using a full time series of measurements, has shown to never outweigh the cost of transmitting such an update even for a single predictor link. With the goal of the GSG, to provide monitoring capabilities with as low communication overhead as possible, full time series and/or parameter set updates are not a feasible option.

To provide a continuous and burst-free operation, we consequently minimize the amount of data per update by sending only single measurements, if a correction is required. These small updates also help to avoid sudden increase in latency caused by queuing delays on the sending node. The adaptation of the predictor models to qualitative changes in the observed data is ensured by the way updates are triggered. If a single update is not sufficient to adjust a predictor model, the source will trigger another update in the next time step. In a worst case scenario, this strategy results in the transmission of raw measurements until the model is completely adjusted to the incoming data. This way, consecutive deviations in predictions and update cascades lead to gradually increasing data rates instead of large chunks of update data.

4.5 Evaluation

In this chapter, a novel system for real-time monitoring queries in the GSG has been introduced. To show the performance benefits of our distributed aggregation structure with integrated predictors, the system was implemented using the PeerSim network

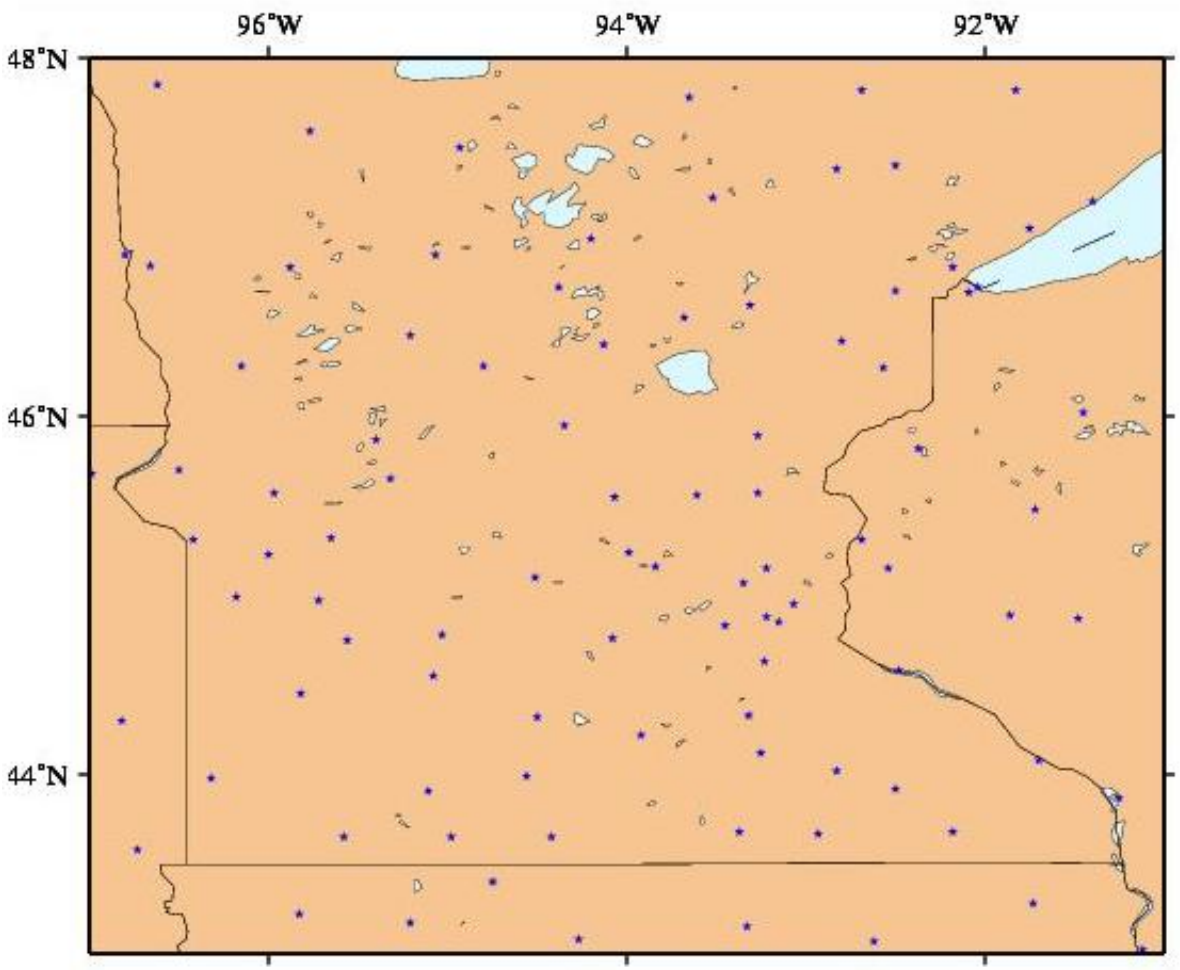


Figure 4.7: The sensor stations considered in the area of Minnesota.

simulator [JMJV]. The evaluation was carried out using real-world datasets from outdoor weather stations of the NOAA and the DWD as well as an indoor set of sensors.

More specifically, we selected a total of 184 sensor stations freely available from the National Oceanic and Atmospheric Administration [NOAA] in the area of 97°W to 91°W and 43°N to 48°N (NOAA dataset). This is a rough approximation of the state of Minnesota, for the concrete placement of sensor stations, see Figure 4.7. The stations were fit into a grid of 9 by 8 fields, one field representing the smallest nonempty query area in our system. The sensor readings were collected over a period of one month, December 2008, on an hourly basis.

Two further sets of data include all of the stations operated by the German Weather Service [DWD] during April 2008. One set includes data from 489 stations on an hourly basis (DWDsy dataset), which provide the main input for the weather prediction. The

other set consists of measurements from 184 (DWDmu dataset) stations that gather data every 10 minutes to be able to provide timely hazard warnings.

To provide a comparison to the originally proposed predictors, the set of Intel Lab Data [MITLAB] measurements (mitlab dataset) was also used for evaluation. The lab was equipped with 52 wireless Mica2Dot sensors which measured humidity and temperature every 31 seconds, starting February 28th, 2004. We chose only the first week of this dataset since the measurements degrade heavily as the battery power decreases. In general, our evaluations show very similar results for all datasets and therefore only the DWDmu dataset is shown in most figures for clarity.

To stress the properties of predictors in this scenario, we set up an aggregation tree for a single query over all available sensors. Measurements from individual sensors were transmitted to a single broker by using predictors. The average of all sensor predictions was then calculated and forwarded by using a second predictor. The overall prediction error was determined for each of the links originating directly from the sensors as well as for the overall average. Although this setup only involved two successive prediction steps, our results clearly show the applicability of different predictor models for a multi-hop scenario.

The two predictor models LMS and SARIMA were chosen for evaluation. To investigate whether the achieved performance really originates from the model and not the general prediction approach, a SIMPLE additional model was used. This simple model always provides the last received measurement as prediction, without considering any history or trend of the data. The state of this model is therefore minimized to a single value and a single update replaces the entire state.

As indicated in Section 4.4.3, the evaluations have shown that none of the predictor models provides a sufficiently high prediction horizon to compensate for the transmission of complete parameter sets or large blocks of measurements. Therefore, only the symmetric operation mode, which provides the intended benefit of reduced traffic and flexible updates, is shown in the results. In particular, only single measurements were transmitted when the desired deviation was exceeded. Note however, that a series of measurements was transmitted, if multiple consecutive predictions deviated more than the allowed threshold.

4.5.1 Data Reduction

To show the ability of the different predictor models to reduce traffic, their hit ratio has been investigated. The hit ratio is defined as the number of sufficiently precise predictions divided by the total number of measurement cycles over the whole experiment.

Dataset	Avgerage Variation	Variation of Aggregated Input
DWD sy	18.4	13.8
DWD mu	17.8	13.9
mitlab	9.5	1.7
NOAA	50.2	33.6

Table 4.1: The reduced variation in temperature measurements clearly shows the smoothing effect of data aggregation.

Dataset	Average Mean Input Power	Mean Input Power of Aggregated Input
DWD sy	77.3	68.7
DWD mu	74.6	66.7
mitlab	477.6	455.9
NOAA	179.3	142.7

Table 4.2: The reduced mean input power in temperature measurements, caused by data aggregation, leads to lower dynamics for the LMS predictor model.

A prediction is sufficiently precise, if it differs less than the user given threshold from the measurements. As only one measurement is transferred each time the deviation is exceeded, the hit ratio corresponds directly to the reduction in traffic. In other words, the hit ratio quantifies the portion of traffic that is saved using a predictor.

Since high and sudden deviation in an observed value usually cannot be covered by predictors, high noise in input data usually leads to poor predictor accuracy and therefore to a low hit ratio. As already described in Section 4.4, smooth input data is therefore expected to lead to better prediction results. Two characteristic properties for the noise in input data were considered: Variance and mean input power. We evaluated the smoothing effect of averaging temperature measurements on a large scale to show the resulting significant reduction in variation and mean input power. Tables 4.1 and 4.2 show the results of the evaluation.

The data was averaged over the complete available data sets to show the overall possible impact. The reduction variation and mean input power, is clearly visible for all datasets. In fact, both values are smaller for the aggregated data than for any single sensor in each data set. Note that for the indoor measurements, the average variation is smaller due to air conditioning. The variation of remaining outliers can be efficiently reduced by aggregation. Of course, this smoothing effect becomes smaller when fewer sensors are considered. However, on a system of global scale, we believe that queries will usually cover many sensors to improve reliability of results and reduce the effect of

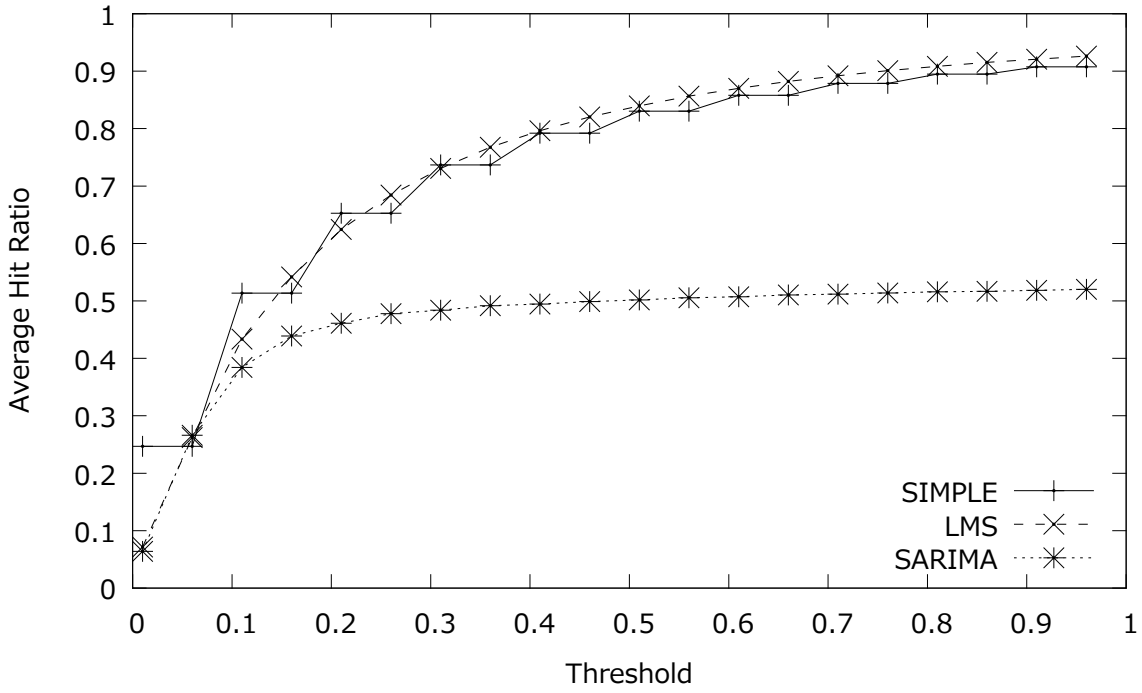


Figure 4.8: Average hit ratio of different predictor models on a single hop using the temperature measurements from the DWDmu dataset.

outlying measurements. The following sections discuss the impact of these results on the different predictor models.

The results on a single link for different predictor models are shown in Figure 4.8. The graph shows the average hit ratio over single hop predictors for all stations of the DWDmu dataset. The LMS model considered was of order 1, which has proven to match the dynamics of temperature data. Higher order models should be able to fit the underlying trend in measurements but have shown to not achieve the same hit ratio. While the LMS model performed best considering the overall hit ratio, the good performance of the simple approach indicates that the measured values only changed slowly over time.

The SARIMA model performed very well for a small threshold in maximum deviation. In contrast, with increasing threshold, the model got too few updates to provide sufficiently precise predictions. This behavior clearly shows that the considered datasets do not have the seasonal nature required for efficient operation of SARIMA predictors due to relatively quickly changing weather conditions. The model is therefore not well suited for larger deviation thresholds.

Figure 4.9 shows the comparison of the SARIMA and LMS models to the simple model for a predictor operating on the data aggregated over all stations of the DWDmu

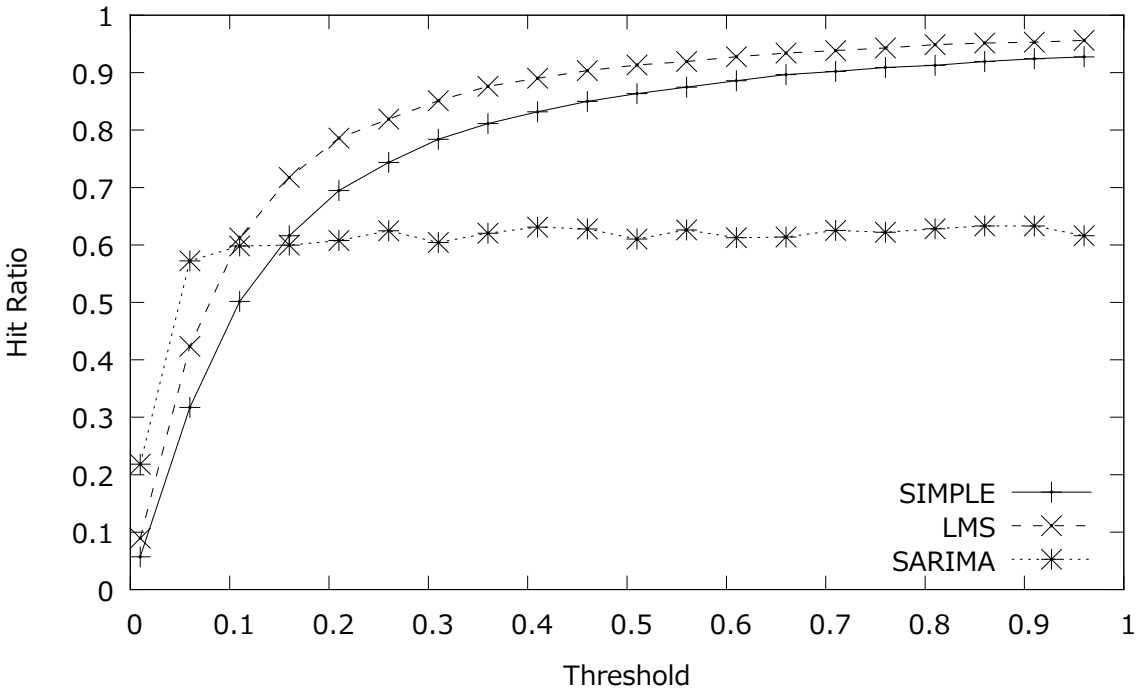


Figure 4.9: Hit ratio of different predictor models on a single hop using the averaged temperature measurements from the DWDmu dataset.

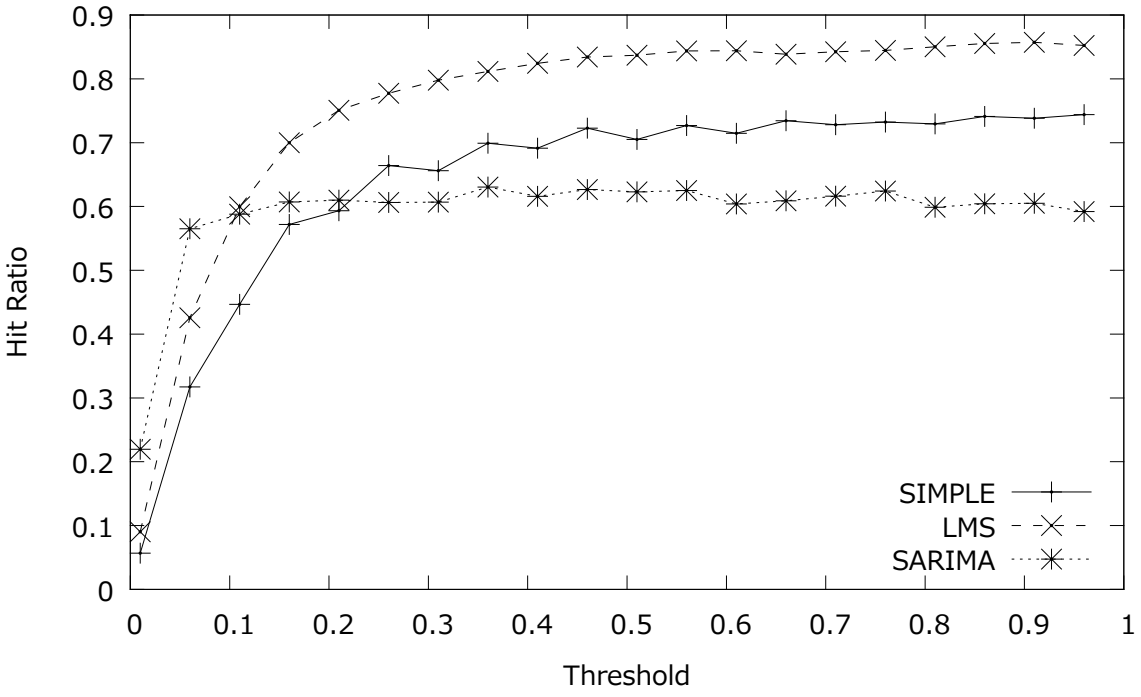


Figure 4.10: Hit ratio of different predictor models in a two hop predictor scenario using the temperature measurements from the DWDmu dataset.

dataset. Although this scenario does not provide any means of practical data reduction, it provides a good insight in the behavior of predictors on averaged data. All predictor models significantly benefit from the smoothed input values and provided a higher hit ratio compared to the input of single sensors.

As expected, the SARIMA model gained most from this effect, especially for low thresholds, since the seasonal trends of the aggregated values were dominant over local anomalies. Nevertheless, the LMS model clearly outperformed the other approaches for larger thresholds in this setting. This advantage is founded in the fact that the current trend in measured data is rather constant considering a sufficiently high sampling frequency of the sensors. By mapping this trend, rather than relying on the last season of measurements, the LMS predictor could therefore achieve the best hit ratio for thresholds higher than $0.1°C$.

After considering the extreme scenarios of a predictor for each single sensors and one for an aggregate over all data, the operation throughout the system was simulated by a combination of the two. Each sensor was monitored using a single predictor and the results of this communication were aggregated and fed to a predictor on a second hop. The hit ratios of the considered prediction models in this two hop setup are shown in Figure 4.10. The SARIMA and LMS models performed comparably well to the single hop operation while the simple approach showed a slight decrease in the hit ratio. The reason for this decrease in performance stemmed from the constant change in temperature caused by the consecutive updates of individual predictors on the first hop. This way, the simple model was only adjusted by one threshold at a time instead of performing larger steps.

In contrast, the SARIMA model can capture the behavior of all the sensors and correspondingly adjust the average overall temperature according to the previous day. The superior performance of the LMS model stems from the fact that it not only considers previous observations for repetitive patters but works on the current trend. This way, not only regular events like sunrise and nightfall are covered but also much more frequent observations like cold fronts traveling across large areas will result in a correct adjustment of the predictor.

Variation of Sensor Types The results of the behavior of the multi-hop prediction have shown that the LMS model is best suited as predictor for temperature data contained in the DWDmu dataset. In a generic monitoring system, however, many types of sensors with different characteristics of the observed data need to be considered. Figures 4.11,

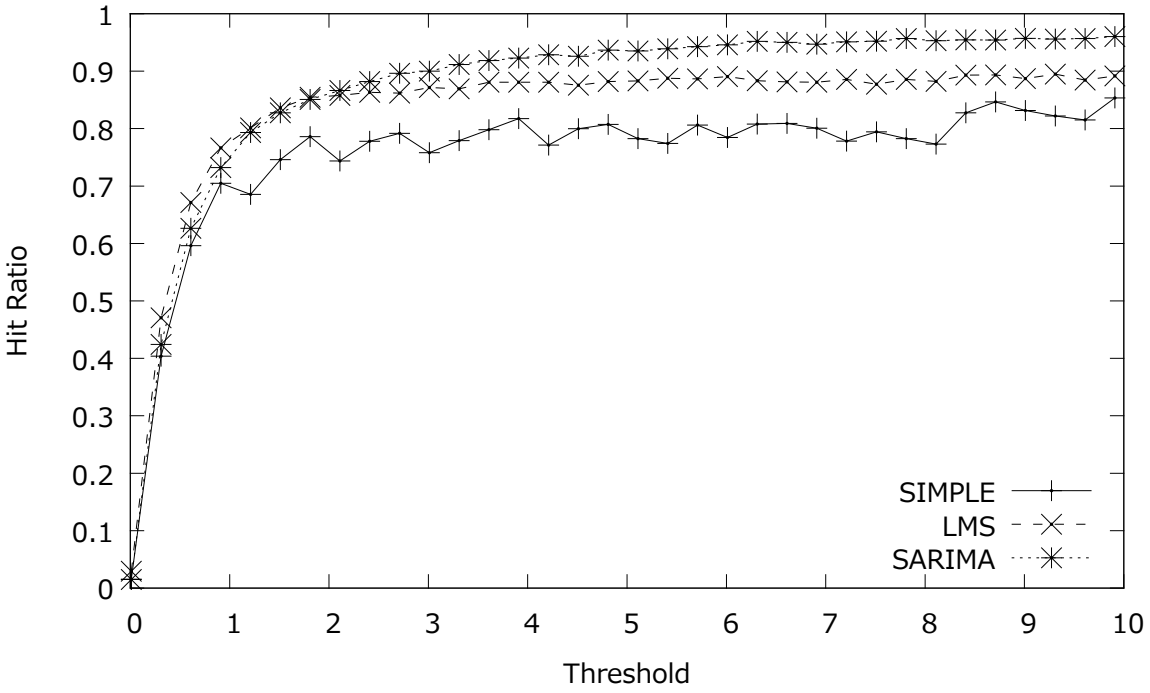


Figure 4.11: Hit ratio for different predictor models in a two hop predictor scenario using the humidity measurements from the DWDmu dataset.

4.12, and 4.13 show the hit ratio for other types of sensor data for this setup. Instead of temperature the measurements of relative humidity, wind speed, and air pressure of the same dataset were used.

The seasonal nature of the very slow change in humidity was best fit by the SARIMA model, especially for larger thresholds. However, the benefit of the considerably more complex model is very low compared to the LMS model. The simple model can still provide approximately 75% reduction in data transmissions for the very low dynamics of humidity measurements. As indicated for the temperature data, the very small but steady change of humidity lead to a constantly increasing error for the predictions of the simple model until a new update is triggered.

In contrast, the highly dynamic, non-seasonal wind speed could only be predicted by the LMS and simple models. Note that the predictions of the simple model were almost as good as those of the LMS model, indicating that the assumption of constant sensor information provides a good fallback if no other predictor models are available. The poor performance of the SARIMA model is not surprising, considering the absence of seasonal patterns in wind speed. The SARIMA model is therefore clearly not suitable for generic usage but needs additional knowledge about the sensor types at hand.

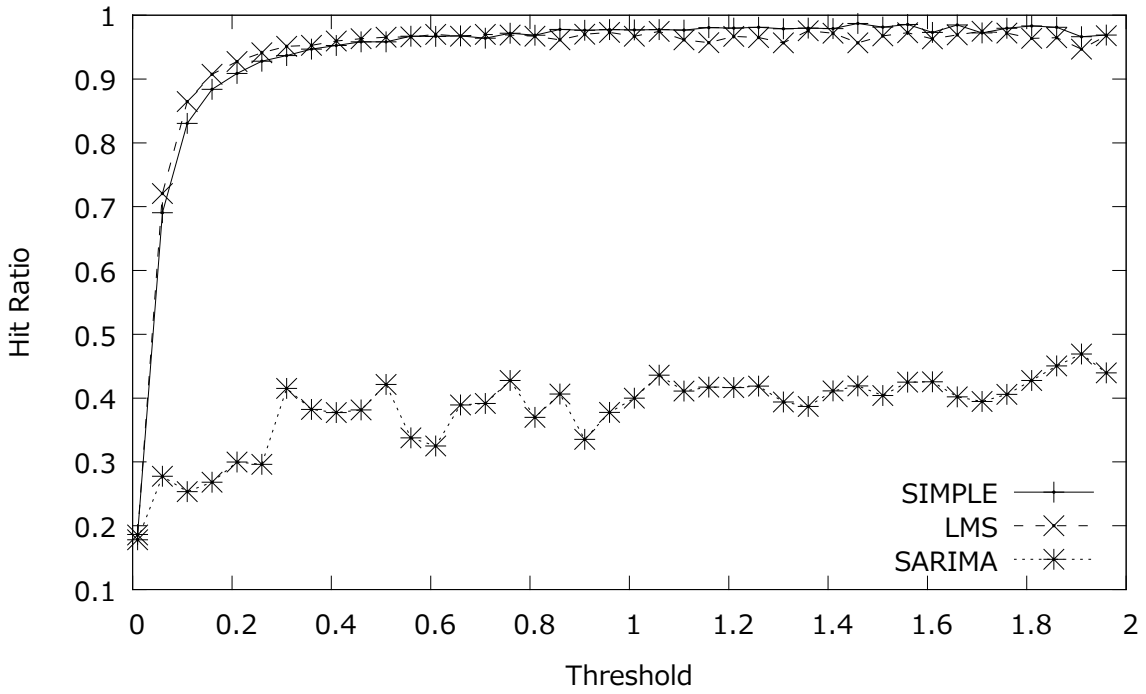


Figure 4.12: Hit ratio for different predictor models in a two hop predictor scenario using the wind speed measurements from the DWDmu dataset.

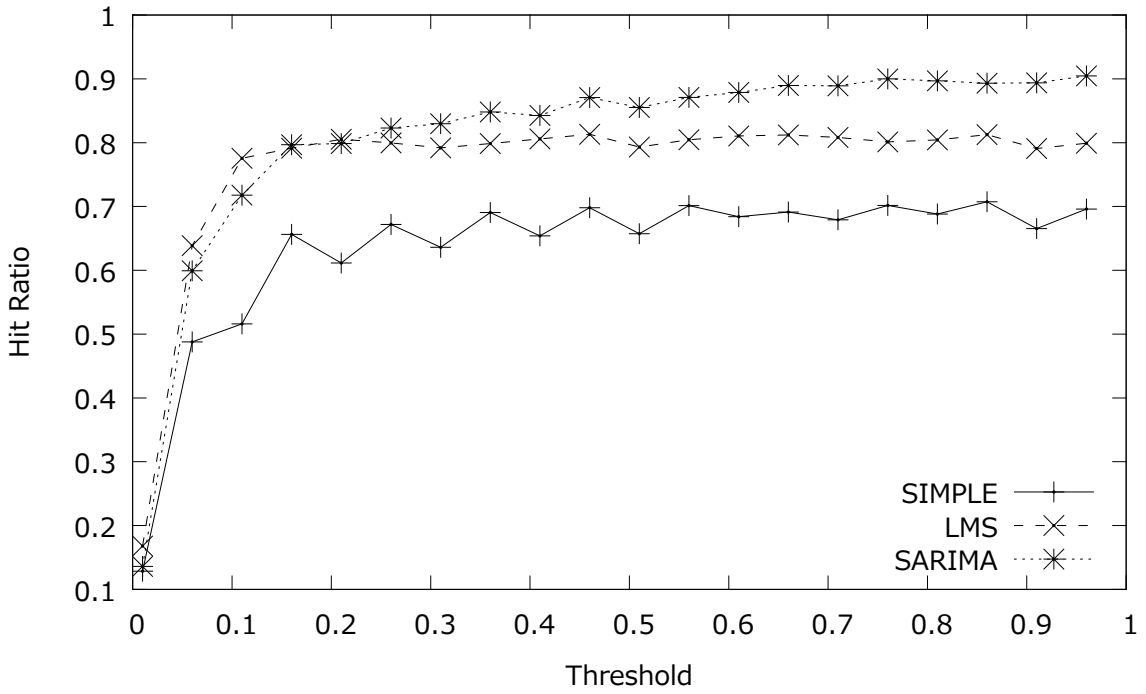


Figure 4.13: Hit ratio for different predictor models in a two hop predictor scenario using the pressure measurements from the DWDmu dataset.

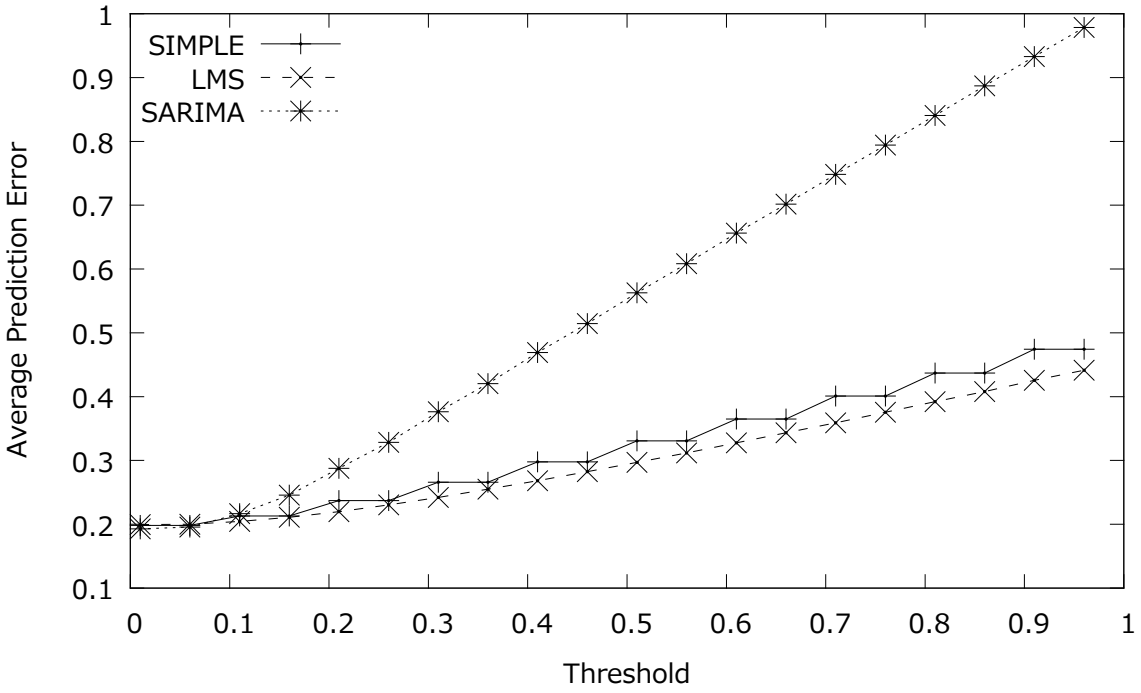


Figure 4.14: Average prediction error of different predictor models on a single hop using the temperature measurements from the DWDmu dataset.

Another measurement type we considered is air pressure, where, most of the time, seasonal behavior can be observed. However, exceptions to this pattern occur much more often than for humidity data. In this case, the simple approach again did not perform well due to the slow and continuous nature of the data. If the threshold was chosen large enough, SARIMA could match the underlying seasonal structure of the data better than LMS which had a slight advantage for very low thresholds.

4.5.2 Prediction Error

While the maximum deviation is given by the user-defined threshold, the actual error experienced by a user of the system is important to assess the quality of the reported data. Similar to the data reduction evaluation, we considered three settings to provide an insight in the properties of the prediction models. First, we compared the models on the first predictor hop, where the input consists of raw sensor data. Second, a single predictor was used on the aggregated data in order to show the behavior for smoothed input. Third, a two hop scenario was investigated to show the overall performance across the system.

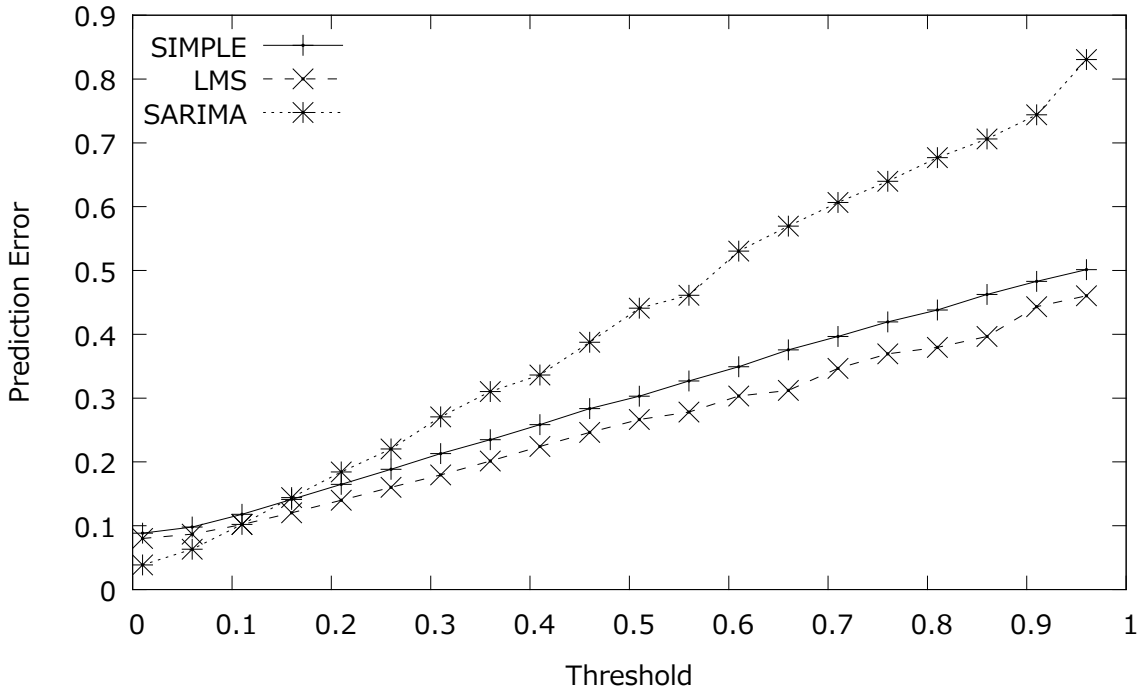


Figure 4.15: Prediction error of different predictor models on a single hop using the averaged temperature measurements from the DWDmu dataset.

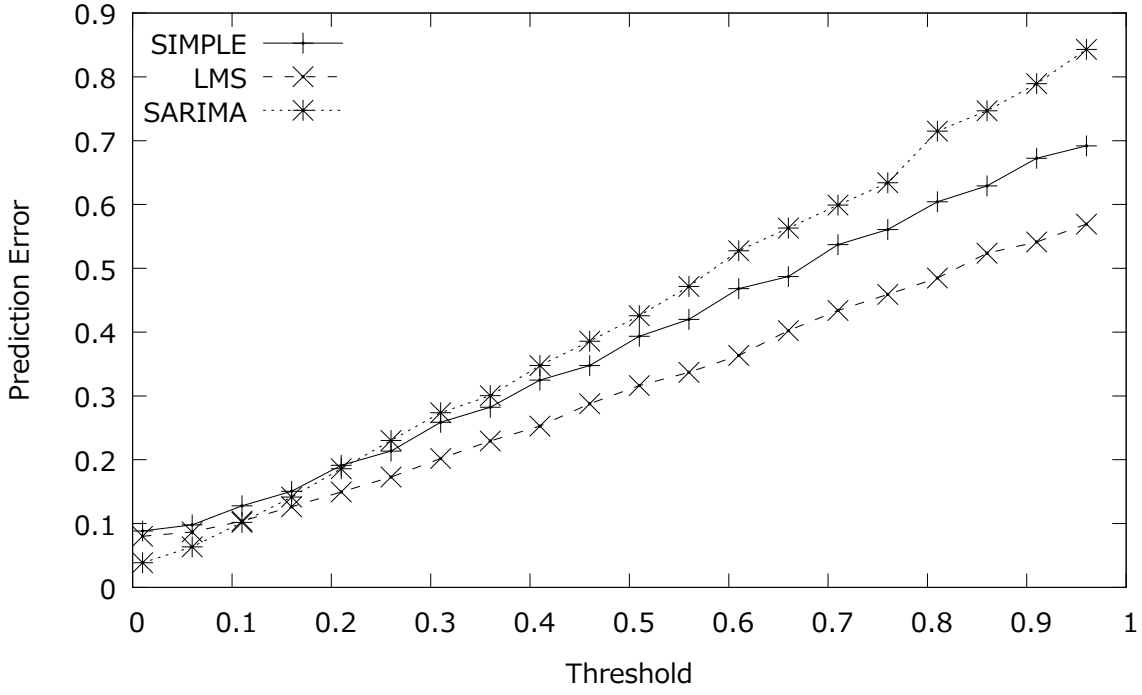


Figure 4.16: Prediction error of different predictor models in a two hop predictor scenario using the temperature measurements from the DWDmu dataset.

The average absolute deviation between the data delivered by a predictor and the corresponding measured, non-aggregated data is shown in Figure 4.14. The values were averaged over all stations in the dataset and all predictions during the experiment. Since the resolution of data provided by the stations is at $0.1°C$, the error does not decrease further even for a threshold below that limit. For larger thresholds the ratio between changes in consecutive measurements and the threshold becomes smaller. The LMS and simple approaches consequently provided a prediction error below the actual threshold. In contrast, the SARIMA model relied on the information from the last season, i.e. the day before. It thereby generated several significantly higher prediction errors as outdoor temperature greatly varied between days.

Figure 4.15 shows the results for the aggregated dataset. Again, the improved performance of predictors can also be seen in terms of reduced prediction error. As in the case of improved hit ratio, the gain heavily depends on the prediction model. The SARIMA model did benefit most when using low error thresholds and provided better prediction accuracy. However, for thresholds above $0.15°C$, which is still close to the sensor resolution, the other two prediction models outperformed SARIMA.

With two hop operation of predictors the prediction error perceived by the user increased faster with the allowed threshold for the LMS and simple prediction models, as shown in Figure 4.16. Note that the SARIMA model did not perform considerably worse as the error is already close to the threshold, indicating that the model does not match the observed measurements with high precision. In contrast, the LMS and simple prediction models observed jumps in incoming data due to the updates received. These jumps caused higher dynamics, which lead to the decrease in prediction accuracy.

4.6 Related Work

In the effort to integrate the huge number of heterogeneous sensor data sources across the globe into a single monitoring system, a number of different approaches have been proposed. The global sensor network (GSN) middleware [AHS06] proposes virtual sensors as an abstraction mechanism. A virtual sensor uses local physical sensors or other remote virtual sensors as a source of measurement which are processed using algorithms which are also specified in the XML-based sensor description. The result of the processing can then be published as new virtual sensor to other users. However, GSN does not support direct queries to sensors without defining a virtual sensor for the set of sensors of interest and lacks the possibility to automatically find sensors of interest.

To support a wide range of queries, publish/subscribe [BKR09; CRW01; EFGK03] has emerged as a generic powerful many-to-many communication paradigm. The main goal of publish/subscribe is the efficient decoupling of data sources (sensor nodes) and subscribers (clients which pose monitoring queries) mostly in event processing systems. More recent approaches in the field also consider quality of service aspects like the delay and bandwidth constraints of subscribers [TKK+10; TKKR09] which provides additional support for monitoring a large number of sensors. Applied in the GSG, however, publish/subscribe lacks the possibility to perform in-network aggregation which is required to reduce the high data rates in order to minimize the network load.

The advantages of in-network processing even beyond aggregation for data reduction are in the focus of several DSP systems like Borealis [AAB+05]. However, the query interface of Borealis requires manual stream modeling and distribution of operators to available network nodes which is not suitable for a large number of monitoring clients. Other approaches like IrisNet [GKK+03] and HiFi [FJK+05] focus on the special properties of sensor data and provide data filtering and preprocessing close to the sensors to reduce bandwidth consumption. Hourglass [SPL+04] provides robust connection abstractions, called circuits, on intermittent connections between nodes to allow for seamless transitions between networks for mobile nodes. Although these systems support information source lookup and automatic routing of data streams, they lack support for the reuse of partial aggregates. Furthermore, their query interface does not support range queries, as required for monitoring global sensor data.

Contrary to these DSP systems, our approach is to exploit the similarities in data streams by actively avoiding redundant data transmission. Recall that in order to reuse data streams, we extended the R-Tree [Gut84] to be organized in a distributed fashion that can be used to locate data as well as to route data streams. With the increasing number of peers in the Internet, other distributed index structures have been previously proposed. Other approaches for distributed R-Trees like NR-Tree [LLL05], Peer-Tree [DF03] or TPR*-Tree [TPS03] are, however, limited to query routing and data source lookup but do not support the routing of the actual data streams. The DR-Tree [BFG07] does handle routing in publish/subscribe to minimize false positives but does not provide means to reuse the partial aggregates.

The concept of predictors as integrated in the GSG was previously introduced in the context of WSN. First successful approaches focused on compression of blocks of multiple measurements before transmitting them. Between the reception of such measurement blocks, the WSN gateway supplies predicted values to the user [LM03].

With the increasing computational power, the predictions could be calculated directly on the sensor nodes using simple models. By simultaneously calculating predictions on the gateways and sensors, only differing measurements actually have to be transferred from sensor nodes to the gateway, thereby reducing communication and hence energy consumption [DGL+05; DGM+04; SR06; TM06]. More complex SARIMA models were later introduced in PRESTO [LGS06]. However, these models are too complex for sensor nodes and have to be computed on the WSN gateway. In addition, recent work has shown that simple models are better suited for real-world applications [RCM+12].

Other projects tried adaptations of methods from control theory like Kalman filters for prediction [WB06]. Although this class of filters is very versatile, the filter setup requires manual modeling for each sensor and its individual characteristics. This is unfeasible for a GSN with a large number of sensors where further challenges arise when using predictors [BHKR09]. Besides the most basic prediction in the form of caching [OJW03], the Kalman filter was the only approach that has yet been applied in DSPS [JCW04].

4.7 Summary

In this chapter, we have shown how a distributed indexing structure can be used to provide a scalable solution for the lookup of sensors. The monitoring queries are thereby integrated into the structure, to allow for running queries to be identified as a source for newly posed queries. This way, the redundant calculation of aggregations is prevented by efficient reuse of data streams. The concept of predictors was integrated into the GSG monitoring system in order to further reduce traffic. With our system, predictors provide estimates for monitoring queries with bounded error by transmitting data only if the prediction model is deviating from the observed measurements.

Our evaluations show the effectiveness of our adapted predictor update strategy for multi-hop operation. The performance of predictors is further increased by the smoothing effect of aggregation which significantly improves the hit ratio and prediction error for all predictor models. The comparison of different sensor types and datasets thereby clearly shows that different models are not equally well suited for the GSG. The LMS predictor outperforms other models not only for temperature measurements, but also for other types of sensors in almost all investigated metrics. A low order of the LMS predictor thereby provides the best results while requiring low computational overhead and little memory at the same time. Considering all available datasets, we therefore chose the LMS model as the generic predictor model in the GSG.

5 Data Stream Distribution

This chapter describes the implementation of the simulation query interface of the Global Sensor Grid (GSG), which can be used to scalably provide full sensor streams to simulations for further analysis. The conversion of raw sensor measurements to a set of data points in a regular grid, by means of diagnostic simulations, is introduced first. We then refine the overall system model to point out the details relevant for data stream distribution. The foundations for handling queries with different temporal and spatial resolution are introduced, followed by the description of the query processing itself. Our approach thereby allows for the identification of data covered by multiple queries with different resolutions. We subsequently detail how load is distributed across the entire broker network of the GSG by using data about the queries in the system. The evaluation shows that the approach can efficiently leverage resources available in the broker network and how bandwidth usage is reduced by incorporating resolution information. The chapter is concluded with a discussion of related work and a short summary. Note that parts of this chapter have previously been published [BKR11].

5.1 Preliminaries

The previous chapter dealt with the real-time monitoring of a huge number of sensors around the globe at preferably low cost in terms of network communication. However, the aggregated information is insufficient to drive a detailed simulation-based analysis. More detailed information has to be provided for further investigation, if a monitoring query indicates a critical or interesting situation. In doing so, the GSG aims to provide data which can be integrated into simulations without further adaptation. To achieve this goal, data points need to be provided in the format used by simulations: a regular grid of data points covering the region of interest with a specific spatial and temporal resolution.

The naive approach of sending all raw measurements from all sensors in the region of interest directly to the simulation clients obviously does not meet the requirements.

Due to the mismatch between data points in the simulation and the measurements taken by sensors, an additional preprocessing step would be required. As described in Chapter 3.3.2, the basic adaptation from raw measurements to a high-resolution grid of data points is therefore achieved using diagnostic simulation in the GSG. A diagnostic simulation relies on two main types of information: First, the measurements of all sensors which provide information relevant to the simulation are integrated into a single model [WB02; Wer02]. Rather than only considering temperature readings, for example, a diagnostic simulation of temperature data also integrates humidity readings to quantify the heat capacity of the atmosphere. Second, gaps in sensor coverage can be filled not only by interpolation between sensors but the simulation model also integrates knowledge from external sources. For example, wind speed and direction is only measured at weather stations but the simulation model also relies on topographical information to calculate the current values at remote locations [LB98; RSMF88]. The combination of all information allows for a highly accurate digital representation of the physical world. The requested information can then be extracted from the internal state of the diagnostic simulation.

The diagnostic simulations thereby maintain a model resolution which is sufficient to provide all data requested by any simulation query. However, not all simulations which request data from the GSG run at such high resolutions. Clearly, the GSG therefore has to ensure that data streams are distributed according to the respective requested resolution rather than send all available data. Otherwise, a significant amount of network bandwidth was wasted by unnecessary transmissions of data. To maximize the overall utilization of the GSG, the load of distributing data streams must be shared among all brokers to avoid bottlenecks.

In this chapter, we propose and evaluate our method for the efficient distribution of full sensor streams in the GSG. The approach consists of two major parts: multi-resolution query processing and load balancing among brokers. During query processing, queries are indexed using the GBD-Tree [OS90] to allow for efficient lookup and reuse of already established data streams. The indexing is extended for the application in the GSG to support different resolutions in all spatial dimensions as well as differing temporal resolutions for individual data streams. Based on this indexing, queries are routed to the corresponding broker which distributes the requested data.

Since brokers have only limited bandwidth available, overload situations can occur if numerous queries request data from a single broker. Using our multi-resolution index structure, the exact regions of highest interest can be identified. Bottlenecks in broker

resources are then avoided by our load distribution method, which replicates the data from regions of high interest to brokers with free resources. The replicas subsequently help to serve requests and contribute to balance the load. The number of replicas is thereby adapted according to the current load situation.

5.2 System Model

As presented in Section 3.2.2, simulations and similar applications can use the simulation query interface to request a continuous stream of data points. The query can be sent to any broker of the GSG, independent of the location of the query source, the relevant sensors, or the broker. The network of brokers will process and route the query to one or more brokers which can provide the requested data. To facilitate the processing of multi-resolution queries, an additional index is created since the structure used for direct sensor queries cannot handle resolution information. The data is finally delivered directly to the query source as a full sensor stream.

The brokers cooperate to deliver the sensor data by replicating data of highly loaded regions from the corresponding overloaded source brokers to additional relay brokers. The regions to replicate are thereby selected using metadata extracted from the multi-resolution index structure. The relay brokers are selected from a local neighborhood of the source broker.

The neighborhood is obtained from the assignment of regions to brokers which is described in Section 5.3.1. Each broker first identifies all regions adjacent to those assigned to itself. The brokers, which these regions are assigned to, are then selected as neighbors. In order to identify the broker assigned to a region, the mapping information is flooded across the entire broker network. Note that the mapping information rarely changes when new brokers are added to the system or regions are reassigned. During regular operation, only the local neighborhood information is used for query routing as described in Section 5.3.2 and to select a relay broker as described in Section 5.4.3.

5.3 Multi-Resolution Query Processing

In this section, we explain how queries are represented internally in the GSG for efficient handling. Based on the internal representation, we show how query regions are handled using an index structure which is extended to accommodate the resolution information.

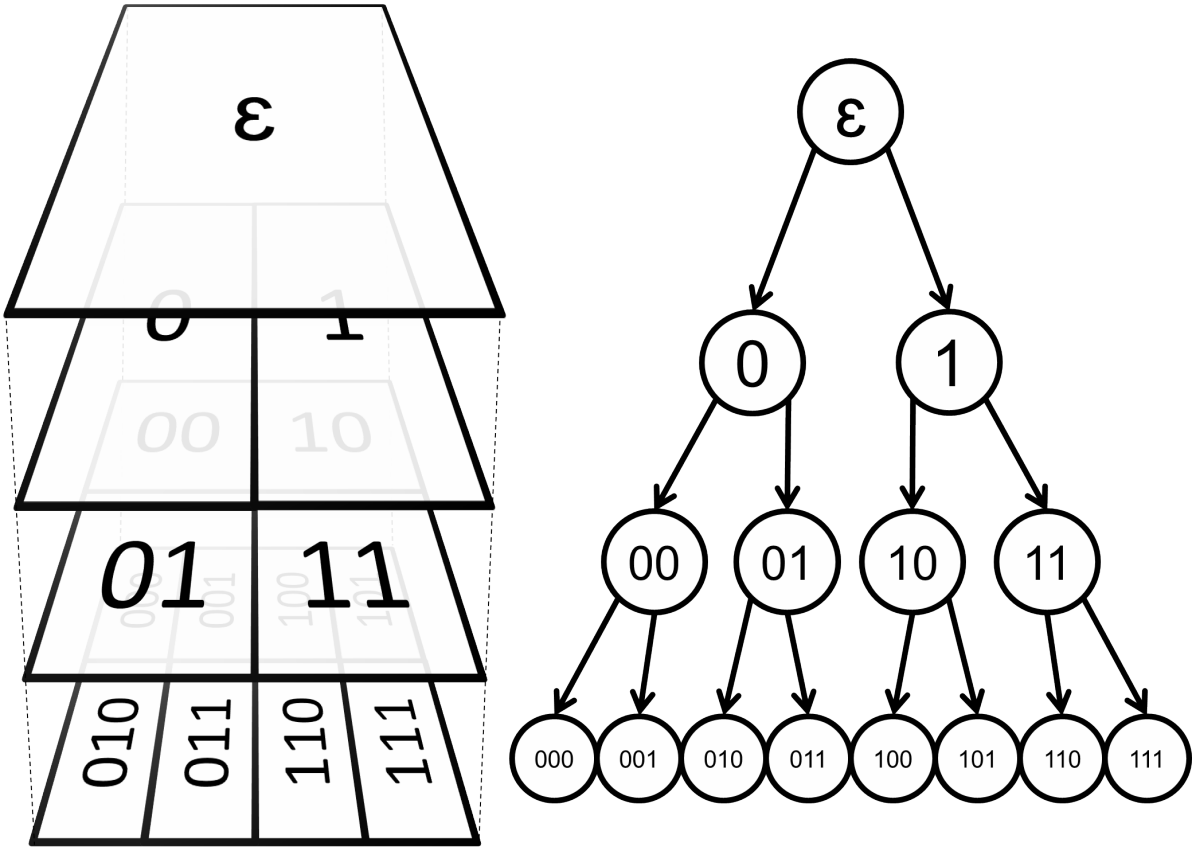


Figure 5.1: Example for generation of DZ expressions using two-dimensional spatial indexing.

These basic operations provided by the internal index structure are in turn used to drive the query routing, which is described at the end of the section.

5.3.1 Spatial Indexing

First, we describe the underlying index structure and basic operations on this structure. We then detail how the additional resolution information, which is specified in the simulation query interface is accommodated by extending the structure.

Basic Structure The foundation for the structured spatial index [GG98] used in the GSG is provided by the GBD-Tree [OS90] (Generalized BD-Tree). The GBD-Tree was originally proposed to index very large spatial databases with high dimensional data. This support for a high number of dimensions will later be used to map the resolution information of queries into the index structure.

An example for a two dimensional structure is depicted in Figure 5.1. The left hand side of the figure shows the entire space represented by the structure during multiple partitioning steps from top to bottom. The right hand side of the figure shows the resulting tree structure with each node labeled according to the region represented by that node.

Nodes labels in the GBD-tree are called DZ expressions. As shown in the figure, DZ expressions are formed by a potentially empty string of characters 0 and 1. An empty DZ expression, denoted ϵ, represents the entire space covered by the index. The node labeled ϵ consequently represents all data available in the GSG.

To create the DZ expressions of the child nodes, the dimensions of the space to be indexed are considered in a static order. In the GSG, dimensions 1 and 2 represent the spatial dimensions x and y of the query region. As indicated previously, additional dimensions will be used to represent the resolution information. The generic indexing for any number of dimensions is shown in Algorithm 1. However, to explain the basic process, we restrict the dimensions on x and y for now.

Starting with the first dimension, i.e. x, the first digit of the DZ expression is determined depending on the extent of the region in the direction of x. If the region is located entirely in the lower half of the dimension, a 0 is appended to the expression. Similarly, if the region is located entirely in the upper half, a 1 is appended. If the child node with the newly created expression does not exist yet, it will be created and assigned the lower or upper half of its parent's region in direction of x, accordingly. The process continues in the corresponding subspace for dimension 2. Depending on the extent of the region in direction of y, a 0 or 1 is appended to the expression. Again, a new child node with a corresponding subregion is created, if necessary. Alternating between the two dimensions, the expression is extended as long as the region fits entirely into the lower or upper half of the remaining range of x or y, respectively. The final DZ expression is obtained when the region can no longer be assigned to a single half on the current dimension.

The partitioning used in each step of the indexing process is also used to generate the partitioning of the world into regions which are then assigned to brokers. An initial partition to a set of regions is thereby obtained by creating the full tree shown in Figure 5.1 down to a predefined level. Each of the regions represented by the leaf nodes is then manually assigned to a broker in the GSG. To add an additional broker to the system, any existing region can be split and one of the new regions is assigned to the existing broker, while the other is assigned to the new broker. As described in the system model,

```
input : R_q: Query Region to Process
        D: Set of Dimensions
        MIN: Set {min_d : d ∈ D} of Lower Bounds of Dimensions
        MAX: Set {max_d : d ∈ D} of Upper Bounds of Dimensions
result : expr: DZ Expression of R_q
Function dz(R_q) begin
    expr ← ε;
    while true do
        for d ∈ D do /* iterate over dimensions                    */
            mid ← (min_d + max_d)/2;
            /* region contained in lower half of interval         */
            if getUpperBound(R_q, d) < mid then
                max_d ← mid;
                expr ← expr + 0;
            /* region contained in upper half of interval         */
            else if getLowerBound(R_q, d) ≥ mid then
                min_d ← mid;
                expr ← expr + 1;
            else /* query region cannot be refined along D         */
                return expr;
            end
        end
    end
end
```

Algorithm 1: DZ Expression Generation Algorithm

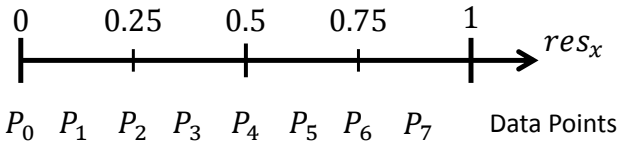
the mapping is flooded across the broker network as a list of pairs, each mapping a DZ expression to a broker address.

The addresses are stored in the corresponding node of the local instance of the index structure maintained on each broker. Hence, the nodes which represent the regions for which data is created locally are annotated with the local address. In addition, a dedicated flag on each node indicates whether the corresponding region is available, i.e. data can be provided for the region, either from a diagnostic simulation or from a relay data stream.

Index Operation In order to store a query in the index, we first generate a DZ expression representing the query. This is done using the query region, which defines the spatial extent of the query and is following the process described above. However, since the indexing stops when one dimension cannot be further divided, certain queries require an adapted indexing process. For example, a query covering a very small range in dimension 1 but a very large range in dimension 2 would be represented using a much larger area than the actual query region. We circumvent this problem by splitting the affected query into multiple parts which are treated as individual queries. Excessive fragmentation is thereby avoided by limiting the depth of the tree structure. At the end of the process, each query is stored in the node identified by the query's expression.

When new data points are available for distribution, the relevant queries have to be found in the index. This lookup is done based on the DZ expression, which represents the geographical region of the data points. The expression itself is either known from the region of the local diagnostic simulation or received along with the data from another broker. Note that a DZ expression represents the path from the root of the index tree to a particular node. Due to the containment property in the tree, all queries in nodes along the path specified by the expression, including the individual node, are considered as a destination for all data points. In addition, queries contained in the subtree below the node are served with a correspondingly filtered subset of data points.

Mutli-Resolution Mapping We now focus on the extensions needed to support queries with different resolutions. Unlike limited intervals like spatial dimensions and attribute ranges, the resolution is not a continuous range of values. Without further extension of the index, the only way to support multiple resolutions would be to request every data point in a separate query. This, however, would result in huge overhead with respect to the number of queries in the GSG. Instead of addressing single data points,



$$res_x = 1:$$
$$[0,1) \sim \{P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$$

$$res_x = 0.5:$$
$$[0,0.5) \sim \{P_0, P_1, P_2, P_3\} \neq \{P_0, P_2, P_4, P_6\}$$

$$res_x = 0.25:$$
$$[0,0.25) \sim \{P_0, P_1\} \neq \{P_0, P_4\}$$

Figure 5.2: Linear mapping of data points along the x resolution dimension. The intervals represented by $res_x < 1$ limit the spatial extent rather than the resolution.

the approach taken in the GSG incorporates the resolution information directly into the index.

The extended index contains five dimensions instead of the previous two while the basic structure remains the same. The indexing algorithm as well as the operations on the index are already capable of handling an arbitrary number of dimensions and, therefore, left unchanged. The properties of a query are mapped to the dimensions as follows: Dimensions 1 and 2 still represent the geographical extent of the query region in the structure. In addition, dimensions 3 to 5 are used to represent the resolutions res_x, res_y, and res_t of a query. Since the indexing operates on ranges rather than single values, the resolutions are mapped to the intervals $[0, res_x)$, $[0, res_y)$, and $[0, res_t)$, respectively.

Intuitively, the additional dimensions can be constrained by a query to limit the spatial and temporal resolutions just as the geographical extent. For instance, by specifying $res_x = 0.25$, a query is constrained to one quarter of the available spatial resolution along the x dimension. In that case only every fourth data point should be delivered to the client in the resulting data stream.

However, aligning data points along the x resolution dimension in a linear fashion as used for the x spatial dimension is not sufficient. Consider Figure 5.2, for example. When evaluating a DZ expression, the interval along one dimension is split in half with every digit of the expression, according to the indexing mechanism described previously. Depending on whether the digit is 0 or 1, the lower or upper half of the region is selected. A DZ expression that covers half of the x resolution would then consequently cover half

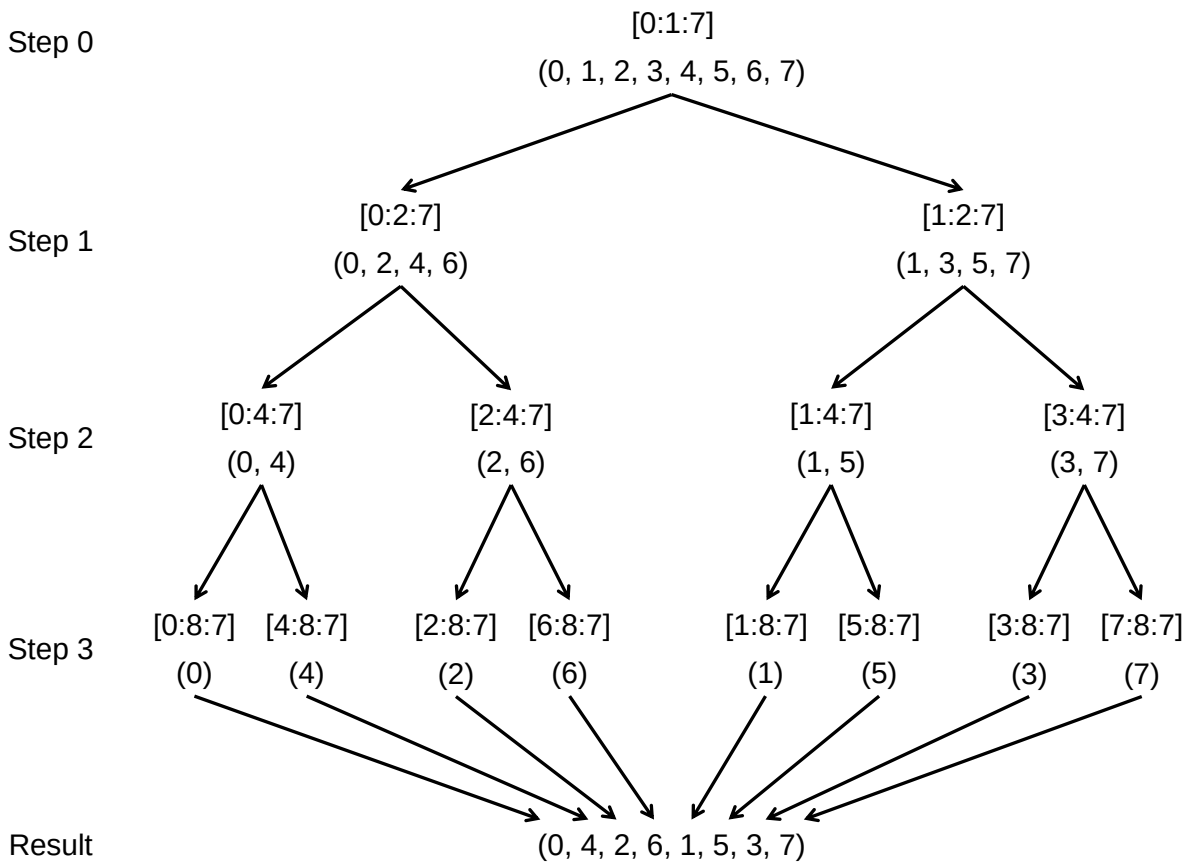


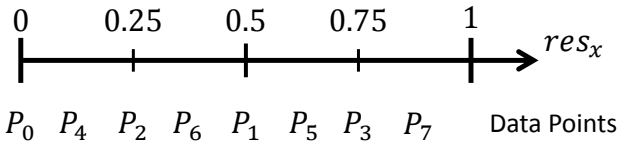
Figure 5.3: Example for the permutation of indices between 0 and 7 for the mapping of data points to the resolution dimensions in the index.

of the area instead of the full area with half as many data points. To achieve the desired behavior, a permutation of data points along the additional dimensions is used in the GSG. This permutation ensures that selecting any contiguous block of the dimension will cover points which are evenly distributed over the whole region.

Since any interval on a resolution dimension is reduced by half during each round of the generation of a DZ expression, the permutation is built so that in each of these reductions, the correct data points are selected. In doing so, the set of all data points is recursively partitioned into a sequence of subsets until only single elements remain. The position in the final sequence of elements provides the desired permutation of each element to its position in the resolution dimension.

To explain the generation of the permutation, we first define the interval $[l : s : u]$ as follows:

$$[l : s : u] = \{n : l \leq n \leq u \wedge n = k \cdot s + l \text{ with } k, n \in \mathbb{N}\}$$



$$res_x = 1:$$
$$[0,1) \sim \{P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$$

$$res_x = 0.5:$$
$$[0,0.5) \sim \{P_0, P_2, P_4, P_6\}$$

$$res_x = 0.25:$$
$$[0,0.25) \sim \{P_0, P_4\}$$

Figure 5.4: Permuted mapping of data points along the x resolution dimension. The intervals represented by arbitrary res_x limit the resolution as intended.

Intuitively, the interval $[l : s : u]$ contains each s^{th} index, starting with l and being smaller than u. With m being the number of data points on a dimension in our scheme, the complete dimension is therefore represented by $[0 : 1 : m]$. Following the basic idea, this interval is first split into the two intervals $[0 : 2 : m]$ and $[1 : 2 : m]$ which represent the data points that should be selected by a DZ expression in the first step, accordingly. In the second step, the interval $[0 : 2 : m]$ is consequently split into $[0 : 4 : m]$ and $[2 : 4 : m]$. Similarly, the interval $[1 : 2 : m]$ is split into $[1 : 4 : m]$ and $[3 : 4 : m]$. In every step, each interval is split into two intervals as follows: The first subinterval starts at the first element of the interval, the second element is chosen as the start of the second interval. The step between elements is doubled with every split to match the binary division of the index. The process is repeated until all data points are divided into m intervals, each containing one single index. The permutation is obtained by aligning the single indices in the order in which they occur in the sequence of intervals after the final step. An example for the permutation of eight data point indices is given in Figure 5.3, the resulting mapping to the dimension is shown in Figure 5.4.

Observe that after step i, the difference between two consecutive indices in any interval is 2^i, i.e. after the first step, the difference is two, after the second step, the difference is four, and so on. The indices are therefore separated into new subintervals by the i^{th} from least significant bit in their binary representation. The permuted position of a single data point can consequently be determined in a non recursive process. Algorithm 2 calculates a binary representation of the path of a particular data point along the subintervals

```
input : b: Number of Bits used for Positions
        P_in: Initial Position
result: P_in: Permuted Position
Function permute(P_in, b)begin
    P_out ← 0;
    for i = 0 to b do /* iterate over each bit                          */
        shiftLeft(P_out);
        /* add value of i^th from least significant bit of P_in to P_out  */
        P_out ← P_out + P_in[i]
    end
    return P_out;
end
```

Algorithm 2: Computation of Permuted Position

Linear Position (Decimal)	0	1	2	3	4	5	6	7
Linear Position (Binary)	000	001	010	011	100	101	110	111
Permuted Position (Binary)	000	100	010	110	001	101	011	111
Permuted Position (Decimal)	0	4	2	6	1	5	3	7

Table 5.1: Linear and permuted positions of data points and their binary representation.

with 0 representing the first subinterval and 1 representing the second one. Due to the construction of the permutation, the sequence of values in the path equals the binary representation of the permuted position of that point. The binary representation of the permuted position also equals the reversed order of the binary representation of the linear position, as can be directly derived from the algorithm. An example is shown in Table 5.1. This equality can be exploited to compute the permutation of indices in a highly efficient manner by using specialized hardware for reversing the bit order. Consequently, we do not take the computational effort for creating the permutation into account throughout the remainder of this thesis.

5.3.2 Query Processing and Routing

The overall query processing can be described with the following overview: Initially, a query q is sent to any broker, say b, of the GSG. The broker b first generates the corresponding DZ expression $dz(q)$ of the query as previously described. Afterwards, the broker checks which parts of the queried area are available locally. If required, the query is further split during the search. The available parts are added to the local index structure. The parts of the query for which data cannot be provided locally are forwarded towards the broker to which the respective region is assigned.

In the following, we describe the two parts of the query processing: First, we detail the local processing of a query including the partitioning into subqueries, each of which can each be answered by a single broker. Second, we describe how a query is forwarded between brokers to ensure its delivery to the corresponding broker.

Local Processing To decide whether all or parts of a query can be answered locally, the available flag of the corresponding nodes in the index is evaluated. Initially, this flag is set for all nodes corresponding to regions for which data is available from the local diagnostic simulation at the broker. The locally available parts of the query are then integrated into the index and the corresponding data is sent on future updates. The remaining parts of the query, if present, are forwarded to one or more remote brokers as described in the next section. A formal description of the process is given in Algorithm 3.

The algorithm `serve()` operates on the local GBD-Tree, starting at the root node which represents all data and, therefore, also must contain q. As the tree provides a strict containment relation between parent and child nodes, the availability of data for a query is checked by descending into the tree. On each node, we check whether the data of the region represented by this particular node is available locally. One of three main cases can occur: 1) the node is marked as locally available, 2) the node is not marked as available and has no children, and 3) the node is not marked as available but has child nodes. The third case in turn is divided into two separate cases: 3a) $dz(q)$ specifies which child to consider next and 3b) the query spans both child nodes. Each case is described in the following.

If the data is available, i.e. case 1) occurs, the query is inserted into the subtree under that node. The insertion process uses the remainder of the DZ expression to successively select child nodes in order to find the target node. After the last digit of $dz(q)$ has been parsed, the query is inserted in the current node of the tree and the process terminates.

Case 2) occurs, if the node is not available and no children are available to continue. Since all nodes that are marked as available are also included in the index, the absence of child nodes indicates that no data for the query is locally available. Therefore, the query is forwarded towards the corresponding broker using the routing process are described in the next section.

If case 3) occurs, the process continues depending on the length of $dz(q)$ which depends on the size of the query. Either the query is entirely contained in one of the child nodes, or the query has to be split and each of the two parts needs to be processed using the corresponding child node. The former case, 3a), happens, if the length of the DZ

```
input : n: Index Node
        q: Client Query to Process
Function serve(n, q)begin
    if isAvailable(n) then
        insert(n, q);   /* available locally, insert q into subtree of n
        */
    else if noChildren(n) then
        forward(q);     /* not available, forward q to another broker */
    else
        if length(dz(q)) > level(n) then /* query in single child node */
            if hasChild(n, dz(q)) then
                serve(getChild(n, dz(q)), q);      /* continue in subtree */
            else
                forward(q);                        /* child not available */
            end
        else /* query spans both children, split required              */
            for bit ∈ {0, 1} do
                if hasChild(n, bit) then
                    forward(split(q, bit));        /* child not available */
                else
                    /* split query, continue on corresponding child     */
                    serve(getChild(n, bit), split(q, bit));
                end
            end
        end
    end
end
```

Algorithm 3: Local Query Processing Algorithm

expression of the query is larger than the level of the current node. Since the expression corresponds to the path in the tree, the next digit can be used to determine the child node, if available. In case 3b), there is no additional digit left to choose the next child node which means that the availability of the query region as a whole cannot be determined. The query is split into two parts q' and q'' of equal size with $dz(q') = dz(q) + 0$ and $dz(q'') = dz(q)+1$, respectively. Each of the parts is then further processed and eventually either stored in the index or forwarded to another broker, depending on the availability of the corresponding child nodes.

Query Routing The query routing implements a greedy forwarding strategy based on the geographic location of the regions maintained by a broker. As introduced in Section 5.2, each broker stores a set of other brokers, called neighbors, which maintain the regions adjacent to the locally assigned regions. If a (partial) query q cannot be served locally, as previously described, a broker forwards q to the neighbor $target$ one of whose regions is geographically closest to the region represented by q. The distance between two regions is calculated based on the center of each region. On receiving the query, broker $target$ performs the local query processing and again forwards the query, if necessary. As the process continues, the query is subsequently forwarded until it eventually arrives at the broker to which the query region is assigned. Algorithm 4 describes the selection for the next hop in the query routing process.

```
input : q: Query Part to Forward
Function forward(q) begin
    target ← null;
    min ← distance(q, this);
    foreach n ∈ neighbors do
        if distance(q, n) < min then
            min ← distance(q, n);          /* update minimal distance */
            target ← n ;                       /* update target broker */
        end
    end
    send(target, q);
end
```

Algorithm 4: Query Routing Algorithm

Note that a query could be directly forwarded to the broker which is stored in the index node corresponding to the query region. However, we chose the iterative routing to enable the delivery to different data sources. In the GSG, with each forwarding step,

a broker checks whether the requested data is available locally. This approach allows for a dynamic distribution of load among brokers by distributing data as described in Section 5.4.

5.4 Maximizing Utility of the Broker Network

If many queries request data from a region assigned to a single broker, the bandwidth resources of this broker might be insufficient for serving all incoming queries. The regions which are requested by many queries therefore have to be replicated to additional brokers. Another broker can then serve queries for the replicated region, thereby distributing the load to more than one broker. This way, the resources of all brokers in the GSG can be leveraged even if queries are concentrated in a small area. In the following, we first formalize the problem of maximizing the utility of the overall broker network. The procedure of choosing the region to replicate and selecting the broker on which the replica should be established is then described in detail along with the integration of this process into the query processing.

5.4.1 Problem Statement

The goal of the method presented in this chapter is to enable the GSG to provide as many clients as possible with sensor data despite highly non-uniform query distribution. Thereby, each broker has only a limited amount of resources available to serve queries. Following the system model introduced in Section 3.1, the bandwidth of each broker is considered as the limiting factor. The basic function of the load balancing in the GSG is consequently to establish replication data streams between brokers, if required, with possibly low overhead. This function is considered optimal, if the total amount of data delivered to client nodes is maximized without exceeding the bandwidth, which is locally available at any single broker.

$$\begin{aligned} &\textbf{maximize} && \sum_{q \in Q} |q| \\ &\textbf{subject to} && \forall b \in B : \sum_{c \in C_b} |c| + \sum_{s \in Q_b} |s| \leq a_b \end{aligned} \tag{5.1}$$

Formally, let B be the set of all brokers in the GSG and let Q_b be the set of queries which are answered by broker b. The complete set of queries in the whole system is then

given by $Q = \bigcup_{b \in B} Q_b$. $|q|$ denotes the size of a query $q \in Q$, i.e. the bandwidth required to serve q. Additionally, let C_b be the set of outgoing data streams, or connections, used to replicate data from broker b where $|c|$ denotes the bandwidth required for a data stream analogously to $|q|$. Since the bandwidth required for queries, connections, and nodes depends on the corresponding region, the term size is used directly for these entities instead of referring to their respective assigned regions for the remainder of this thesis. Finally, let a_b be the total bandwidth available for outgoing data streams at broker b. An optimal solution then provides the maximum total query size without exceeding the local bandwidth at each broker. The linear optimization problem given in Equation 5.1 describes this setup.

5.4.2 Query Load Estimation and Region Selection

When establishing a new replica for load-balancing, two main questions have to be considered: First, which region is replicated? Second, on which node is the replica established? In the following, the details of the replication region selection mechanism are presented. The selection of a broker for the new replica for is described in the next section.

As previously established, the selection of the region to replicate is crucial for balancing the load among brokers. The intuition behind our selection algorithm is to replicate regions with high load, e.g. regions for which data is provided to a large number of clients. The exact load of a region is determined by the overall amount of data which has to be delivered for the region in relation to its size. Regions which are not requested at all consequently have a load of zero while a region which is entirely delivered to n clients has a load of n. Regions that are only partially provided to clients must be taken into account accordingly. The challenge is to determine the load distribution across the locally managed regions in consideration of arbitrary query regions.

Formally, the load of a node and its corresponding region r is defined as

$$load(r) = \sum_{\{q:R_q \subset r\}} |q|/|r|$$

The load is normalized by the size of the considered region and therefore represents the average number of queries for the entire region considering the size of each query. A region r is said to have potential overload if the total bandwidth required to serve

queries in the corresponding subtree is larger than the size of the corresponding region, e.g. $load(r) > 1$.

If data from a certain region is sent out more than once, that region has potential overload. A broker can then reduce the load by sending out a single copy of the data of the overloaded region to another broker. In doing so, the broker delegates the task of distributing the data to multiple clients to that broker. In addition, the region becomes available on the new relay broker during query processing. Consequently, if a broker is experiencing high query load, a list of overloaded query regions for which replication can reduce the load is gathered from the local index.

This search is facilitated by tracking the total size of queries contained in the subtree of each node during insertion and removal of queries to and from the index. The list of regions is then obtained using a breadth-first search over the entire index structure for nodes n containing at least one query and having a $load(n) > 1$. The list is sorted by load in descending order to prioritize the replication of regions with a higher fan-out.

Based on the list of regions, a replication request is sent to another broker for each of the regions, starting at the one with the largest overload. When a replica of a region is successfully established, the associated client queries are relocated to the newly established relay broker. After enough bandwidth has been freed by offloading queries to relay brokers, the broker continues normal operation.

Note that a region which is selected for replication does not need to originate from a certain broker but can also be a replication itself. New replicas for highly loaded regions can therefore be established on additional brokers as second or higher level replicas. This way, the system can use the resources of all brokers in the GSG in scenarios where many users query a small region assigned to a single broker.

To avoid frequent reorganization of replicas due to fragmentation of regions, we restrict the size of a replicated region to a certain minimum. This is done by traversing the index tree only down to a maximum level. If a region r on that level does not contain any queries but satisfies $load(r) > 1$, the region is replicated irrespective of the detailed load distribution in the corresponding subtree. The replicated region can therefore contain regions which are not highly loaded or even queried at all.

More generically, a high load on a certain region does not necessarily indicate that the entire region has been queried multiple times. Instead, an even higher load could be concentrated on a smaller region. If a node corresponding to region r in the index is considered for replication, but has $load(r) < 2$, only part of the region is actually queried more than once. To circumvent the replication of such regions with small overload, large

```
input : n: Index Node
        maxlevel: Highest Level to Push Queries
Function pushDown(n)begin
    if noChildren(n) then
        return ;                                  /* No children, no overload */
    else if level(n) = maxlevel then
        return ;                                     /* Maximum level reached */
    else
        Q ← getQueries(n);
        if count(Q) ≥ 2 then
            return ;  /* Region already covered by at least two queries */
        end
        for bit ∈ {0, 1} do
            child ← getChild(n, bit);
            /* split query, continue on corresponding child              */
            insert(child, split(Q[0], bit));
            pushDown(child);
        end
    end
end
```

Algorithm 5: Query Partitioning Algorithm

queries are split as described in Algorithm 5. On nodes which satisfy $1 < load(r) < 2$, i.e. nodes which only contain a single query but have overload, the recursive algorithm splits the query and pushes the two parts to their corresponding child nodes. The load on the child nodes is updated accordingly and the process continues on each child node. The recursion ends, if a considered node no longer satisfies $1 < load(r) < 2$ or a maximum level of the tree is reached. The restriction on the index level ensures, that queries are not excessively fragmented in the process as described earlier. An example for the core step is given in Figure 5.5.

5.4.3 Directed Load Balancing

Ideally, a replica should be used to answer a query during query processing before the actual, overloaded, broker is involved. To achieve this, the replicas must be set up in a way that a query is first routed to a replica before it arrives at the overloaded broker. Therefore, as described previously, a new replica is established on a broker which is responsible for a neighboring spatial region.

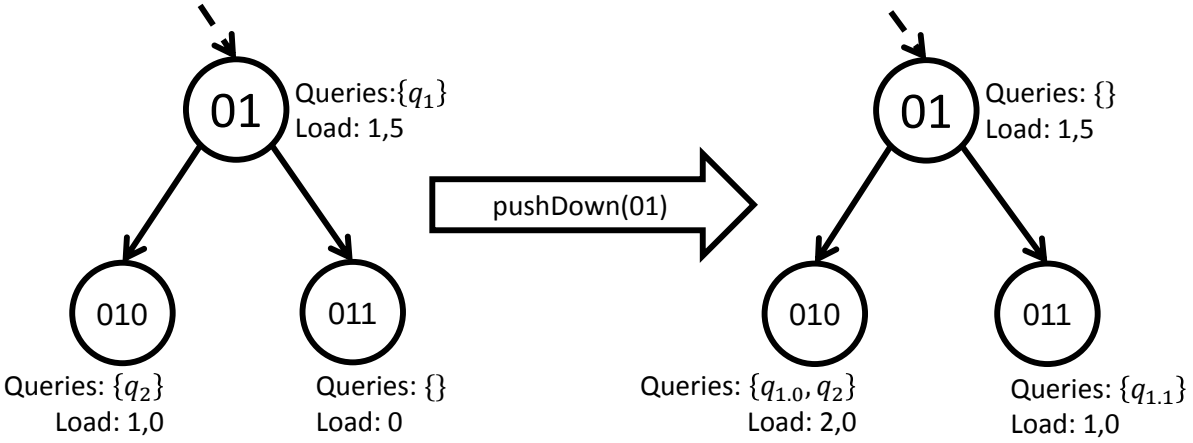


Figure 5.5: Example for Algorithm 5: Query q_1 is split into queries $q_{1.0}$ and $q_{1.1}$ which are added to the children, the load is updated accordingly.

However, not all neighboring brokers are considered as a candidate for establishing a new replica. By restricting the selection of candidates, we ensure a directed load balancing rather than randomly distributing replicas. Foremost, brokers that already provide data to the local broker are excluded. This restriction is essential to prevent unnecessary traffic in the case of two neighboring brokers, which bidirectionally exchange replicated data. Furthermore, only neighbors that are responsible for a region which has a larger distance to the region selected for replication are considered for hosting a replication. By distributing the replicas away from the original source, circular replication requests can be avoided and a directed distribution of data is ensured.

If multiple candidates are found, the neighbor with the highest amount of available bandwidth is selected. That neighbor is most likely to be able to serve the replica without running into an overload situation itself. To facilitate this selection strategy, the brokers periodically exchange information about their available resources with their neighbors. Additionally, with this selection strategy, collisions of replication requests are reduced in presence of multiple highly loaded regions in close proximity. Such collisions can occur as the load propagates to the surroundings of those regions through replication. Eventually, replicas of two different regions can reach adjacent brokers. In this case, new replicas of one region will not be established on the neighbor which holds the replica of another region, due to the increased load on that neighbor.

An example for the aforementioned cases is provided in Figure 5.6. Replications of region 0101 are shown in blue, replications of region 1101 are shown in green. The broker of region 0110 does not replicate the blue region to the broker of region 1100 because it already receives data from that broker. The replication from region 0110 to 0111

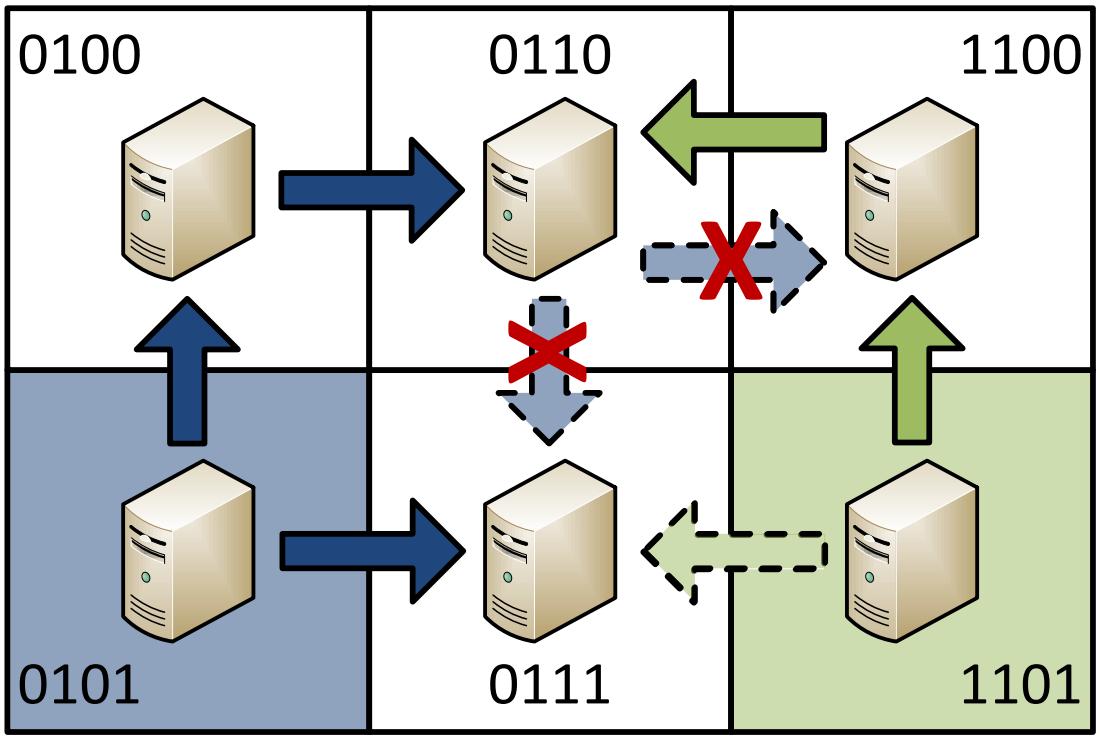


Figure 5.6: Example for the replication of two regions, 0101 and 1101, to multiple brokers.

is prevented because the distance between regions 0101 and 0111 is shorter than that between 0101 and 0110. Finally, since the broker of region 0111 has fewer resources available, due to replicating data from the blue region, the broker of the green region selected the broker of region 1100 to establish its replica.

As queries are terminated, the affected replicas are adapted accordingly. The termination of a query is thereby either actively initiated by the corresponding client or by a connection timeout for the outgoing stream. If the broker, which served the client, is a relay and the load on the relayed region drops below 1, the remaining queries, if any, are transferred back to the relay source and the replica is terminated. The query is then transferred back to the original source and the replica is terminated. This way, unnecessary relays can be removed from the GSG as the load on a region declines.

The replication of regions is triggered when the resources available at a broker are not sufficient to serve a new query. The broker first checks if an existing replica for the query region is available and has sufficient resources available. If such a replica is found, the query is forwarded and no further action is taken. However, if there is no suitable replica, a new replica is created using the process previously described. If no candidates are available to create a new replica, the broker tries to forward the query to an existing replica which does not have sufficient resources. That replica then establishes a new replica to allow the creation of second and higher level replicas.

Properties The presented combination of query routing and replication strategy provides an efficient solution to distribute load among the brokers of the GSG. Any broker can and does replicate regions of high interest to another, less loaded, broker which is responsible for a neighboring spatial region. The replication strategy allows that a replica of the queried region will be contacted by the query routing before putting additional load on the overloaded source broker. As replicas can be established over multiple hops, load from a single broker can be distributed over the entire broker network. This way, the resources of all brokers in the GSG can be leveraged to serve user queries. A low overhead for the replication is ensured by taking the resolutions of interest into account instead of replicating regions at the highest possible resolution. As shown in Section 5.5, our approach can reduce the replication overhead by more than 40%.

While the index structure is maintained on each broker for the overall world, the entries are limited to the locally available regions, thereby limiting the number of nodes. This mechanism also ensures, that additional brokers can be added to the system which consequently reduces the number of nodes in the index tree of affected brokers. Finally,

each broker only stores the queries it is actually serving. The storage required for the index structure is therefore limited by excluding global information.

5.5 Evaluation

The data stream distribution functionality of the GSG was implemented using the PeerSim network simulator [JMJV] for evaluation. 16 brokers were simulated where the entire area was equally distributed among all brokers. In order to trigger the load distribution in the system, each broker was modeled to only have enough bandwidth resources to serve 25 times the entire region for which it is responsible. The ability of the GSG to efficiently distribute high query load in a small region among brokers is shown by using a query hot spot. Using a zipfian distribution for the lower and upper bounds of the queries in the spatial dimensions, we generated a set of queries of varying size with an intentional accumulation of queries. Following the distribution, most of the queries cover a small area around the hot spot, while few queries cover larger areas. The size of the hot spot was modified by choosing different exponents for the zipfian distribution. The center of the hot spot was placed in the center of the overall area. Values for both, temporal and spatial, resolutions were chosen from a uniform distribution.

Queries were posed to the system until it was unable to serve any more requests, i.e. no suitable configuration could be found to accommodate the new request. The focus of the evaluation was to show the ability to distribute the load among brokers in presence of query hot spots. The cost of maintaining the replicas in terms of connections and messages that are needed to establish the connections has been investigated. Our results show that using the resolution information of queries can significantly reduce the overhead required for replication. In addition, the maintenance cost can be flexibly constrained. By limiting the indexing depth of the tree during processing, the replicated regions can be extended in a controllable way.

5.5.1 System Capacity

Overall, the load distribution mechanism employed in the GSG is able to leverage a large fraction of the system wide resources for the delivery of data. Figure 5.7 shows the maximum achievable query load for the system with a varied limit of indexing depth for the large hot spot scenario. The graph shows how far the system could be loaded with queries until no suitable distribution of data streams could be found to answer

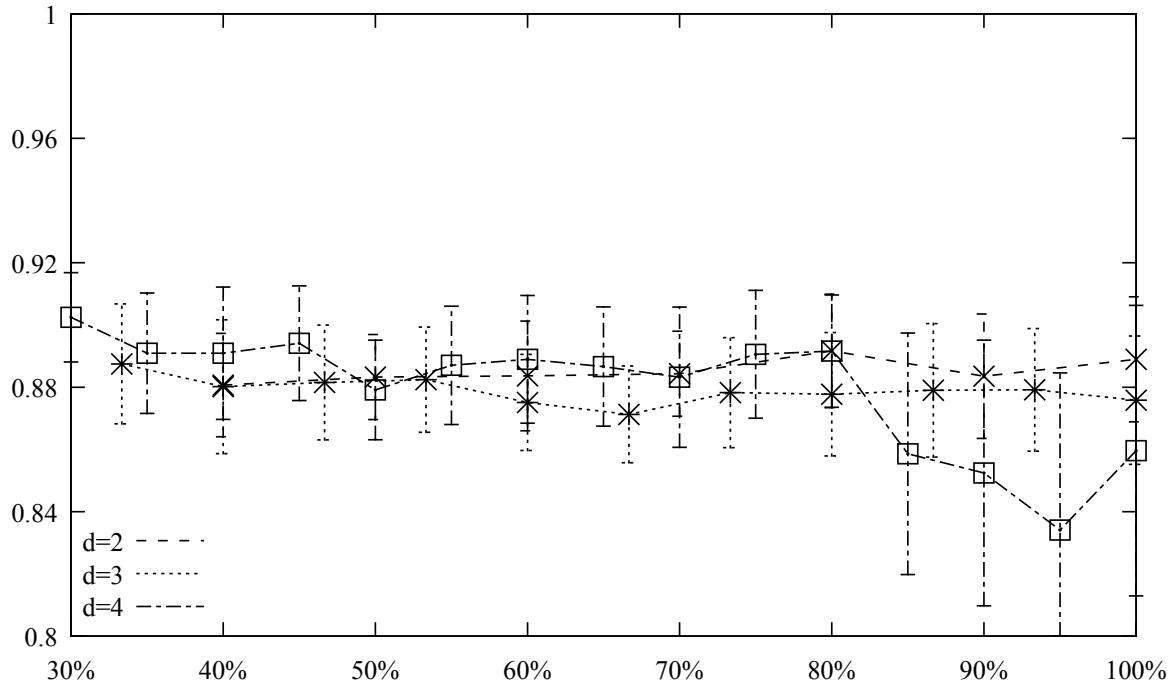


Figure 5.7: The fraction of overall system resources that can be used with respect to the index depth for two, three, and four dimensions, given a large hot spot.

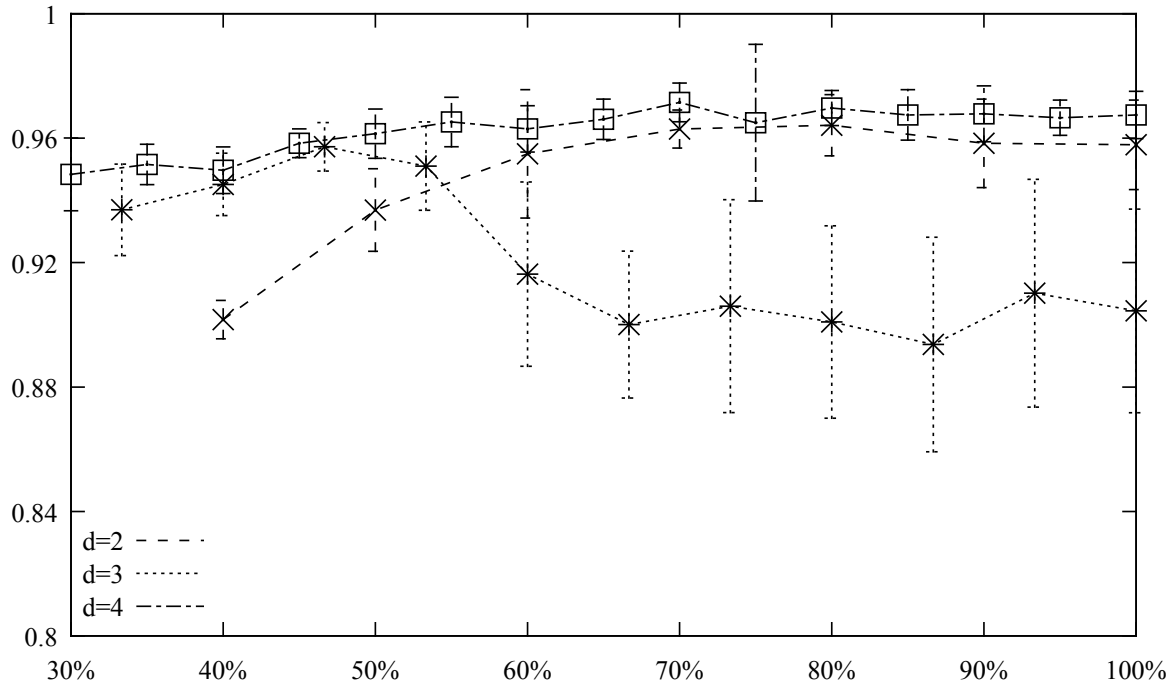


Figure 5.8: The fraction of overall system resources that can be used with respect to the index depth for two, three, and four dimensions, given a small hot spot.

additional queries. Two reasons prevent the system from achieving a ratio of 1, i.e. using all available resources for data distribution. First, bandwidth is required to operate replicas. Second, the continuous nature of the running data streams prevents a complete reorganization of client assignments for optimization.

The constraints on establishing new replicas posed by already running data streams become clear for a large hot spot which covers a region assigned to more than one broker. Since the involved brokers are neighbors in our system, the resolution of conflicts for their respective replication requests becomes difficult. Especially when the full indexing depth is used to minimize overhead, many small partial queries are generated which trigger according replications. Therefore, the results for a high indexing depth show a larger standard deviation, indicating that the maximum achievable load is lower for certain experiments. When only indexing one additional dimension, the system can resolve the conflicts more reliably and therefore serve more data in total for high indexing depths.

For smaller hot spot regions, as shown in Figure 5.8, the directed load balancing can fully exploit the available resources. The regions of such small hot spots are usually assigned to a single broker. In this case, the replications can be established on any neighboring broker as the neighbors do not need to establish replicas themselves. Figure 5.8 also shows that indexing both, spatial and temporal, resolutions results in a high achievable system load over a broad range of indexing depths. Without the additional knowledge about the resolutions, i.e. only maintaining two dimensions in the index, the replication overhead becomes large when using a low indexing depth. The incorporation of of spatial resolution information provides an efficient means to reduce the replication overhead with a low limit on indexing depth. However, the discrepancy of temporal and spatial resolutions in most requests leads to a decrease in achievable load for fine grained indexing. Overall, the results show that the directed load balancing used in the GSG is capable of distributing the load effectively among all brokers in the system and therefore can handle high client load.

5.5.2 Bandwidth Usage

Figures 5.9 and 5.10 show the average number of connections per broker that have been established during an experiment to maintain the replicas for a large and small hot spot area, respectively. The graphs do not include connections to clients, but only the connections required for delivering data between brokers in the GSG. Also note that these figures show the number of logical connections, i.e. relayed regions, rather than unique TCP connections.

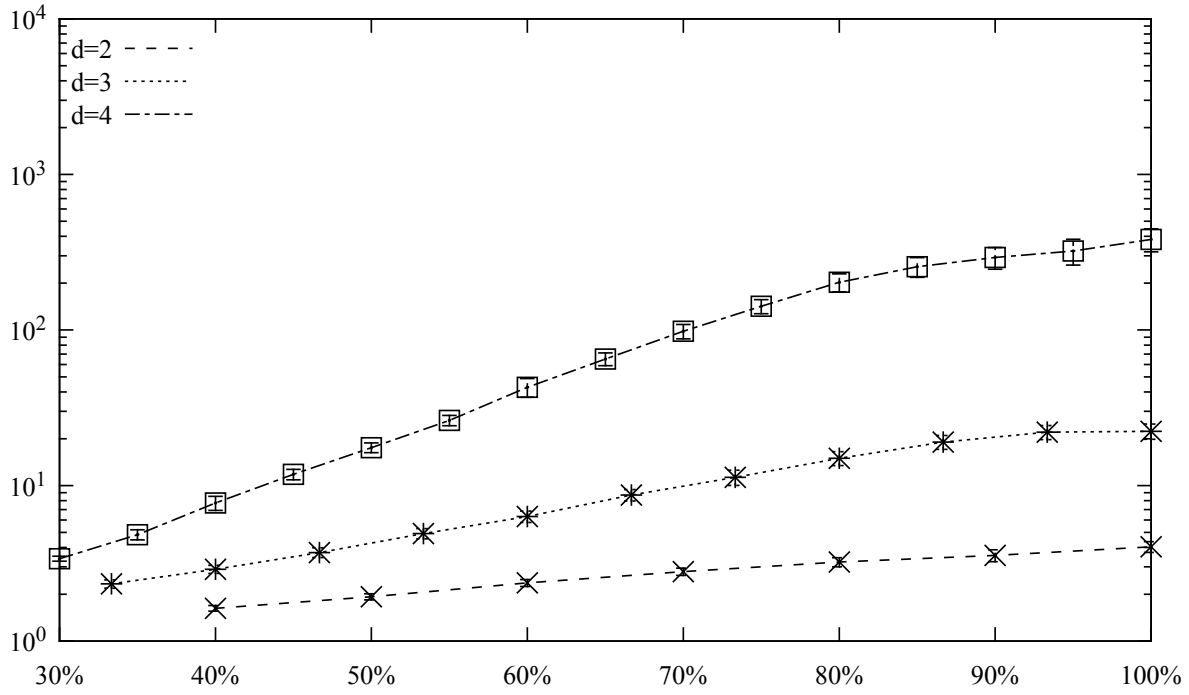


Figure 5.9: Number of logical connections required w.r.t. the maximum processed index depth for two, three, and four dimensions, given a large hot spot.

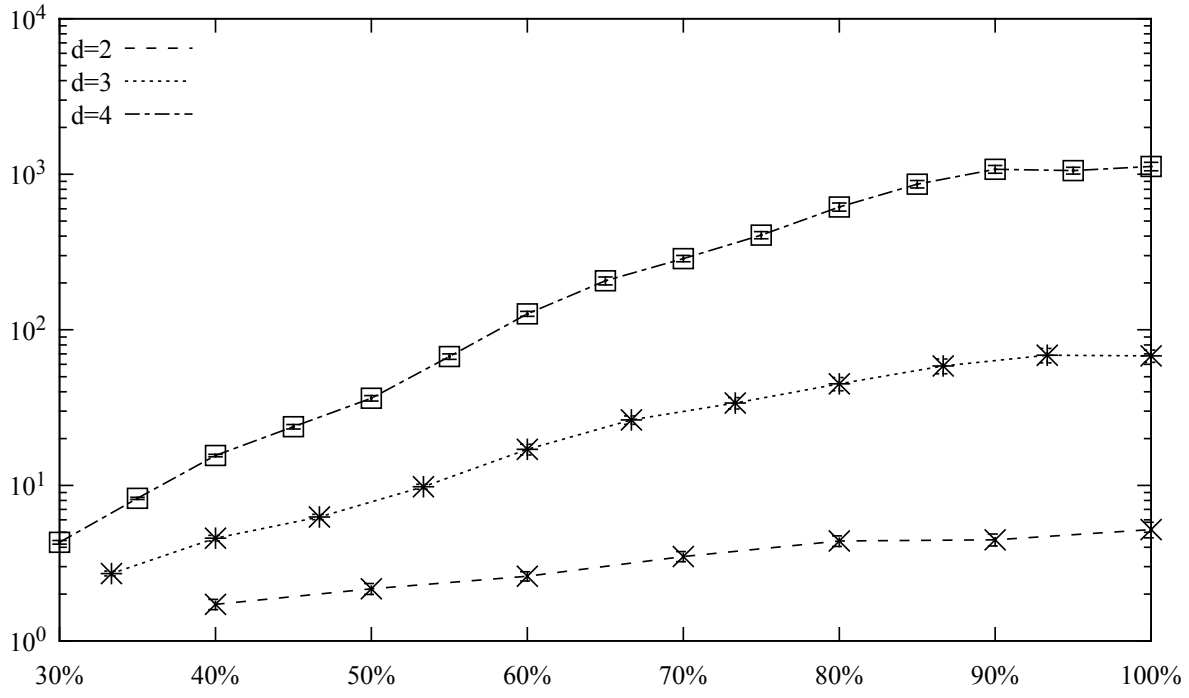


Figure 5.10: Number of logical connections required w.r.t. the maximum processed index depth for two, three, and four dimensions, given a small hot spot.

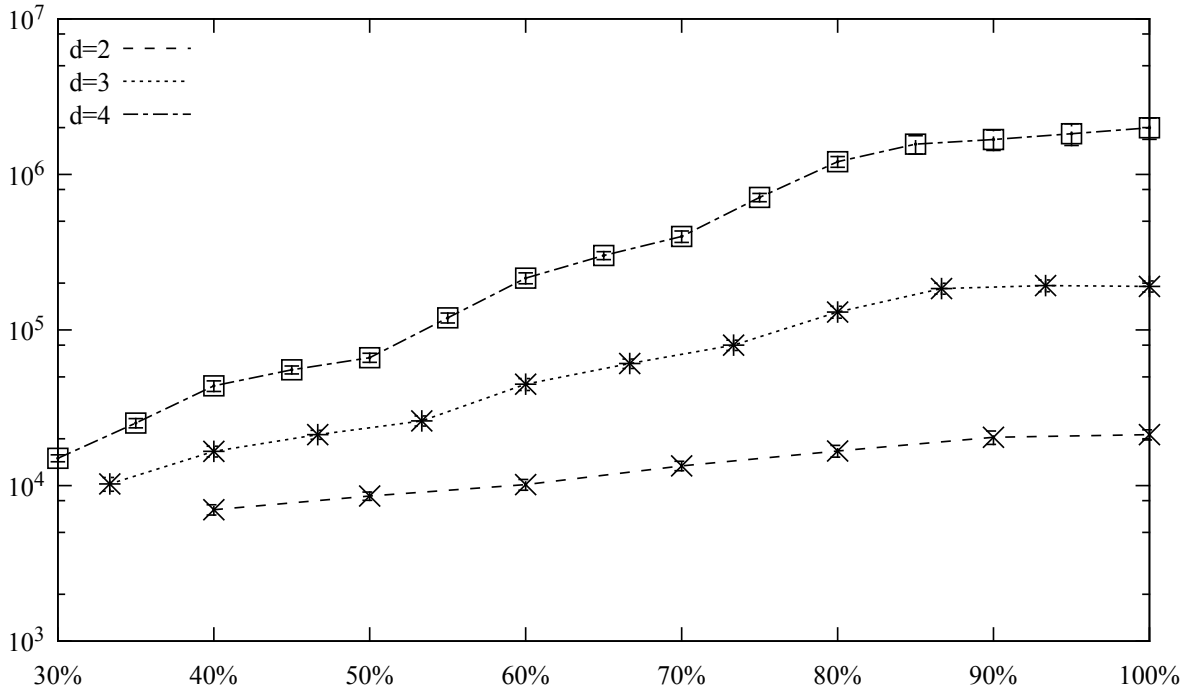


Figure 5.11: Number of messages required with respect to the maximum processed index depth for two, three, and four dimensions, given a large hot spot.

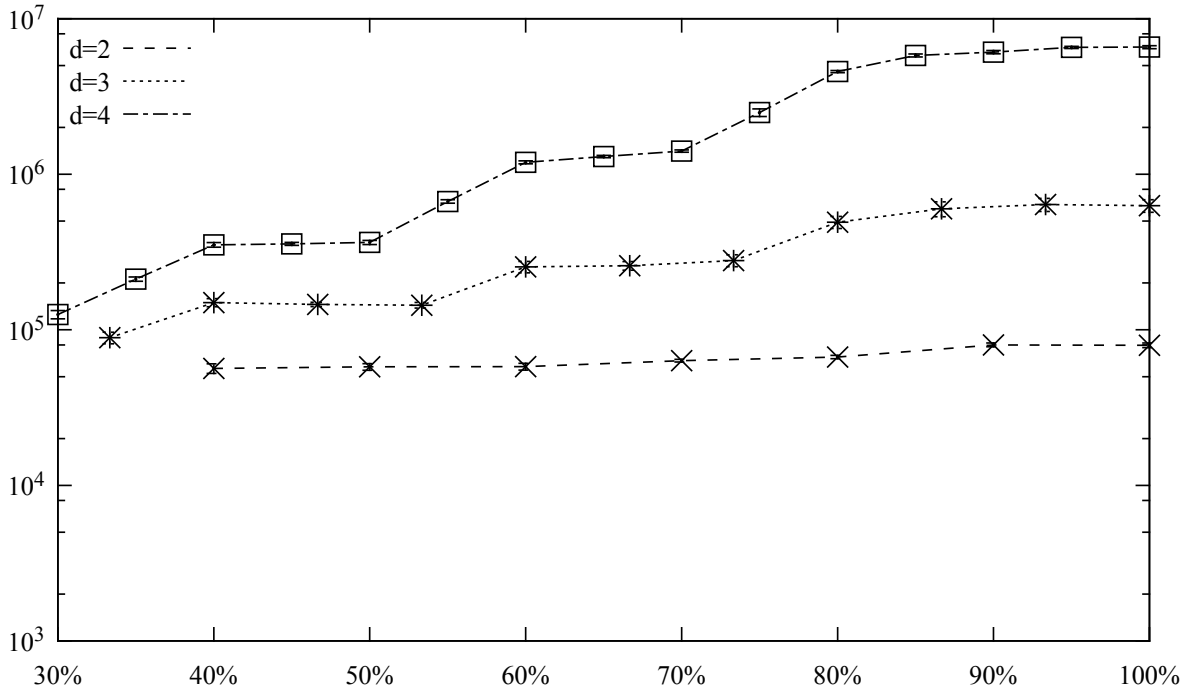


Figure 5.12: Number of messages required with respect to the maximum processed index depth for two, three, and four dimensions, given a small hot spot.

The results show the interdependence between finer indexing and an increased number of connections. The integration of resolution information in our index results in an approximate tenfold increase in connections with each additional resolution dimension. This increase is due to the fact, that queries are split into smaller subqueries to achieve a closer approximation of the actual hot spot region.

Following the increased number of logical connections, the number of messages exchanged during the establishment of these connections is also higher for a larger indexing depth. Again, for the sake of evaluation, we did not consider aggregation of messages. Each request for a single region was considered as a separate message during the experiments. The numbers also only include messages required for establishing, modifying, and removing connections, not the delivery of payload. In a productive system, the amount of messages can be significantly reduced by merging requests for small regions and, in a similar fashion, the data of sensor streams.

The qualitative relation between the indexing depth and the number of messages is depicted in Figures 5.11 and 5.12 for different numbers of dimensions considered and again for a scenario with a large and a small hot spot. We can observe that the number of messages not only increases with indexing depth but that it is also considerably larger for smaller hot spot areas. This is due to the fact that the smaller areas are matched more closely by our replication algorithm, resulting in a higher number of smaller regions replicated.

Another important result of the evaluations is shown by the almost constant number of messages in relation to the indexing depth for a small hot spot and strictly spatial indexing. Even though this behavior is desirable for a scalable system, the lack of additional requests indicates that the system cannot fully leverage the possibility of more fine grained indexing. Only with the integration of the resolution information on additional dimensions, the system can precisely identify and replicate the actual query hot spot.

In contrast to the number of connections and messages, the bandwidth required for maintaining replications was measured in productive operation. The total amount of bandwidth used for replication was determined by measuring the overall data transferred during an experiment and deducting the payload delivered to clients. Again, the experiments were carried out for the two scenarios with a small and large query hot spot and with different dimensions used for indexing, while varying the granularity of the indexing for replication regions.

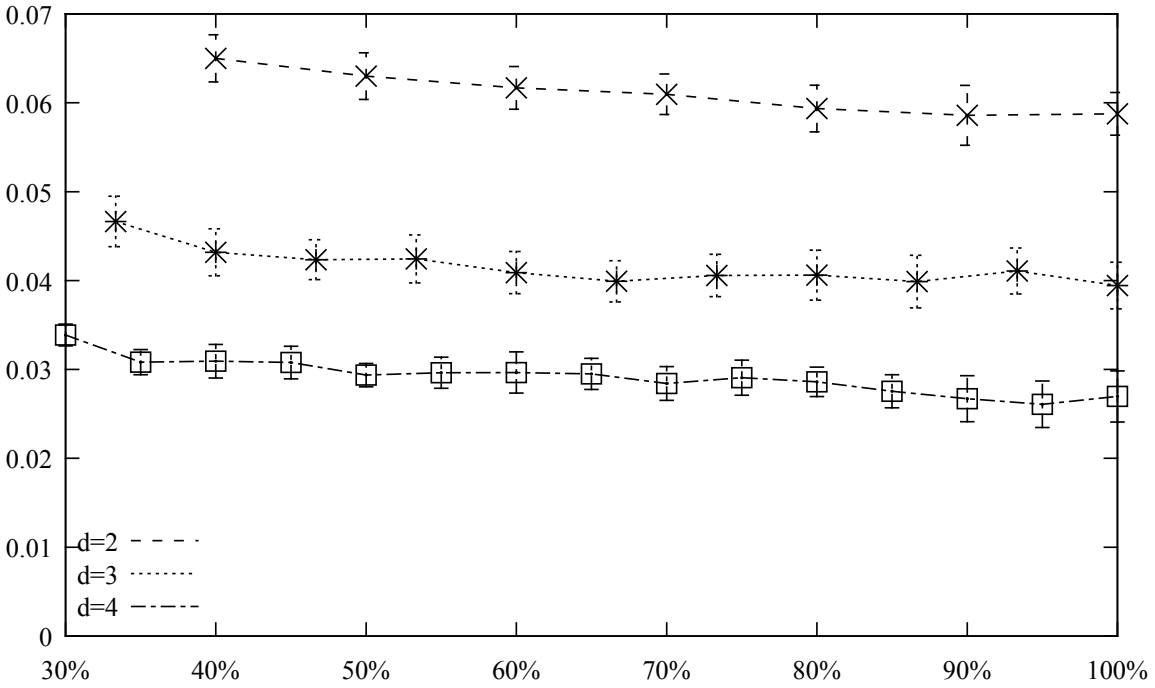


Figure 5.13: Percentage of bandwidth required for the replication with respect to index depth for two, three, and four dimensions, given a large hot spot.

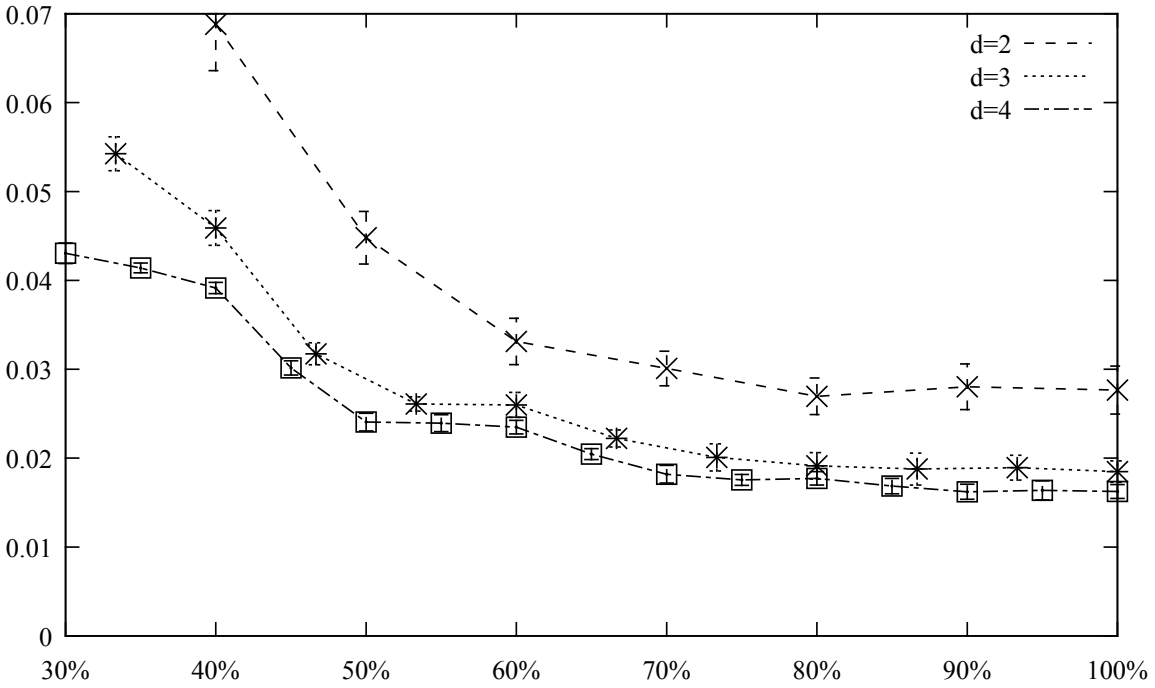


Figure 5.14: Percentage of bandwidth required for the replication with respect to index depth for two, three, and four dimensions, given a small hot spot.

As shown in Figure 5.13, the required overhead can be significantly reduced by increasing the number of indexed dimensions in case of large hot spot areas. This reduction can save over 40% of the payload traffic compared to the pure spatial indexing without resolution information. As large fractions of the hot spot can be covered without fine grained adaptation, the additional gain of increased indexing depth is negligible. The achievable gain of replicating regions with a specific resolution for small hot spots is comparable to that of the large hot spot, as shown in Figure 5.14. However, the graph also indicates that the indexing depth provides an essential parameter for tuning the system. While higher indexing depth can further decrease the bandwidth consumption used for replication, the resulting increase in logical connections and messages leads to additional computational cost.

5.6 Related Work

The major challenge for a global sensing system is the vast amount of data produced by numerous globally distributed sensors. To exploit the advantages of in-network processing for data reduction, several distributed stream processing (DSP) systems like Borealis [AAB+05] have been proposed. However, its query interface requires manual stream modeling and distribution of operators to available nodes. Other approaches like IrisNet [GKK+03] and HiFi [FJK+05] focus on the special properties of sensor data and provide data filtering and preprocessing close to the sensors. Virtual sensors were introduced in the GSN middleware [AHS06] to allow easy provision of preprocessed sensor data. Hourglass [SPL+04] provides robust so called circuits on intermittent connections between nodes. SBON [PLS+06] introduces a layer between the DSP and the physical network to optimize placement for network usage. The approach was improved towards optimal mapping of operators with respect to the underlying network [RDR10b]. Although operators might be placed on the same node, explicit reuse of data streams is not supported by this approach. Although IrisNet, HiFi, GSN, and Hourglass all support information source lookup and automatic routing of data streams, they lack the reuse of streams and intuitive range querying required for simulations operating on global sensor data. The reduction of data rates using a network of adaptive filters has been addressed in [OJW03]. However, the efficient provision of data streams, covering multiple different resolutions of sensor data, to different clients still cannot be provided.

Contrary to these DSP systems, our approach is to exploit the similarities in data streams, thereby actively avoiding redundant data transmission. In this general context,

publish/subscribe [BKR09; CRW01; EFGK03] has emerged as a generic powerful many-to-many communication paradigm. Publish/subscribe is used for efficient decoupling of data sources and subscribers to data streams mostly in event processing systems. More recent approaches also consider the delay and bandwidth constraints of subscribers [TKK+10; TKKR09]. All these approaches, however, accept a certain amount of false positives to provide scalable management of highly dynamic subscriptions. Unlike publish/subscribe, the presented stream distribution method of the GSG is able to split and reunite data streams. This way, the GSG can provide better matching with very few false positives which is required for the high bandwidth used by global sensing systems.

To handle huge amounts of data, especially in the context of climate simulation, systems have been proposed to split gridded data into multiple smaller chunks for distributed processing. DataCutter [BKC+01] provides a network of filters to allow distributed filtering and transformation of data. However, the approach lacks the transparent combination of partial data streams due to its original purpose of making large archives available as data streams. Therefore, new data streams, which are not a subset of existing streams, have to be generated from the source and introduce additional redundancy in data transmission. Another system for climate simulation data is the Earth System Grid-I [CDK+03b]. The system provides data access to archived data rather than real-time data streams and is therefore not suited for a globally distributed sensing system.

Scribe [CDKR02] and SplitStream [CDK+03a] provide application layer multicast (ALM) routing in peer-to-peer systems. Both approaches are related to this work as data is distributed from single sources to many destinations. However, Scribe does not address the problem of providing data at multiple resolutions to the clients. For SplitStream, a suitable encoding is assumed that can be divided to split the actual data stream into smaller chunks. While this approach is well-suited for the dissemination of video streams where distinct channels can be addressed, support for range queries on gridded data is not provided.

5.7 Summary

In this chapter, we have described the data stream distribution system of the GSG for scalable and efficient distribution of grid-based sensor data. The system is capable of providing data concurrently at multiple resolutions to numerous clients while avoiding overload situations at single brokers. To achieve this, we extended the GBD-Tree to allow for the indexing of resolution information. We also provided the required mechanisms to

enable efficient query processing while incorporating the resolution data. As a result, the resources available distributed among the entire broker network can be used highly efficiently independent of the distribution of queries.

Our load distribution approach alleviates single brokers from possible overload situations and thereby remove bottlenecks when query hot spots occur. The directed load balancing approach avoids the high overhead of full replication by selectively replicating data at required resolution. The underlying indexing is scalable with the number of brokers in the GSG, as each broker can locally generate the DZ expression for a query and no centralized organization is required.

6 Minimizing Network Usage

The most important property of the Global Sensor Grid (GSG) is its ability to scale with the increasing number of sensors and clients worldwide. Previously, we have described how the GSG leverages the resources of all available brokers to achieve this scalability. In this chapter, we describe how the system detects and eliminates redundancy in data streams in order to minimize the network usage during operation. This way, the overall resource consumption of the GSG can be minimized to increase efficiency. In order to detect redundant data streams, the underlying network is identified using delay measurements and an approximated representation is generated. The optimization algorithm then uses the combined information of the query index and the underlay abstraction to identify redundant data streams. These streams are then merged to minimize the network usage caused by the GSG. Note that parts of this chapter have previously been published [BKR14].

6.1 Preliminaries

In the previous chapters, we already established that the deployment of sensor networks is making rapid progress all around the globe. Sensors are installed in virtually all areas of everyday life and a huge amount of real-time data has become available. The approach for sensor data distribution taken in the GSG, as shown in Chapter 5, was therefore built to support the large anticipated number of requests for this data. However, the sheer amount of data gathered by the system has not been focused on so far. For example, the Earthscope Project [WFF+10] deployed thousands of sensors on the west coast of the United States. The CeNSE project by HP [Har11] envisions one trillion sensors to enervate the entire Earth. In order to scale not only to a large number of clients but also to the increasing rate at which sensor data is integrated into the GSG, additional measures are required to minimize the usage of network resources.

Previous approaches in this context focused on extending optimization methods developed for local wireless sensor networks (WSNs) to global sensor networks (GSNs) [AN08;

IS09b; SPL+04]. By utilizing in-network aggregation or lossy compression, a significant reduction of sensor data could be achieved. However, the gain of these methods comes at the cost of reduced precision. Many critical applications are highly sensitive to such imprecision. For example, the simulations for the dispersion of pollutants and hazardous substances require precise wind data. Numerical errors, due to compression artifacts, can lead to disturbed data and consequentially wrong predictions.

A promising alternative to reduce communication without loss of precision is to avoid sending information multiple times over a network link. The main challenge of this approach is to find an optimized and dynamically adapting dissemination structure for a large number of requests in the complex Internet. Group communication middleware such as application layer multicast (ALM) [YLE04] or publish/subscribe [EFGK03] tackle the complexity of this problem by reducing the number of requests which need to be organized by a single broker. This reduction is achieved by limiting communication to a fixed number of channels. Clearly, this limit prohibitively restricts expressiveness of sensor queries to statically selected combinations of regions and spatial and temporal resolutions rather than user defined queries. Publish/subscribe offers a high expressiveness for queries over multiple attributes. To achieve the intended simplification of the original problem, the brokers merge similar subscriptions from different subscribers to avoid redundant processing. Since the actual bandwidth requirements for each subscription are masked by the merged subscription, this approach performs sub-optimal for high-bandwidth sensor streams.

Contrary, for global sensor streams, measurements are generated for every point in the region of interest at discrete time steps. Knowledge of the spatial and temporal resolution – as specified, for example, in scientific simulations – allows for exact determination of the amount of data to expect from any given query. Towards this end, the GSG provides an interface to precisely query and methods to scalably deliver real-time sensor data streams. In this chapter, we show how the GSG uses an underlay-adaptive stream management algorithm to minimize traffic by exploiting the predictable network load of each query. Redundant parts of multiple queries are quickly identified using the previously introduced index structure. Identical portions of data streams are merged and sent to dynamically established relay points, where the data is further distributed towards client nodes based on local knowledge.

6.2 System Model

The system model of the previous chapters is extended to incorporate the model of the underlying physical network where each link is considered. In the following the model of the underlay network is presented in detail. We first describe the components of the GSG and their physical connection structure in a realistic network segment. Afterwards, the underlying cost model is described and its formal properties are introduced.

6.2.1 System Components

The basic components and their roles were previously introduced in Chapter 2. The query processing and the interaction of these components for the simulation queries is detailed in the previous chapter. Data is thereby exchanged between brokers and clients using the Internet as communication network. Since, in this chapter, we focus on the minimization of the network usage of the GSG, the phyiscal structure of this network needs to be considered. Figure 6.1 shows an example of the underlying network of a single provider which connects brokers and clients. Note that the network is structured hierarchically, representing a realistic structure [GS08]. Brokers can be placed in the network at different levels of this hierarchy, while clients are usually located at the edge of the network.

As described in the previous chapter, all brokers in the GSG collaborate to support the effective distribution of sensor streams. We refer to brokers that forward sensor data on behalf of other brokers as relay brokers. Relays, which send data to a client, form the end of a data stream inside the GSG and are hence called target brokers for that particular client. In contrast to the basic stream distribution strategy presented in the previous chapter, relays are not only established in case of overload situations. Instead, relays are established to create waypoints along which data streams are forwarded to influence which underlay links are used. This way, we can optimize the network resources used depending on the client requests for sensor data.

6.2.2 Cost Model

The primary goal of the method described in this chapter is to minimize the bandwidth used by the GSG for distributing sensor data streams to clients. Recall the definition of a query region in Section 3.2.2: $R_q = (x_{min}, x_{max}, y_{min}, y_{max}, res_x, res_y, res_t)$. The bandwidth used by each data stream is proportional to the product of the amount of data

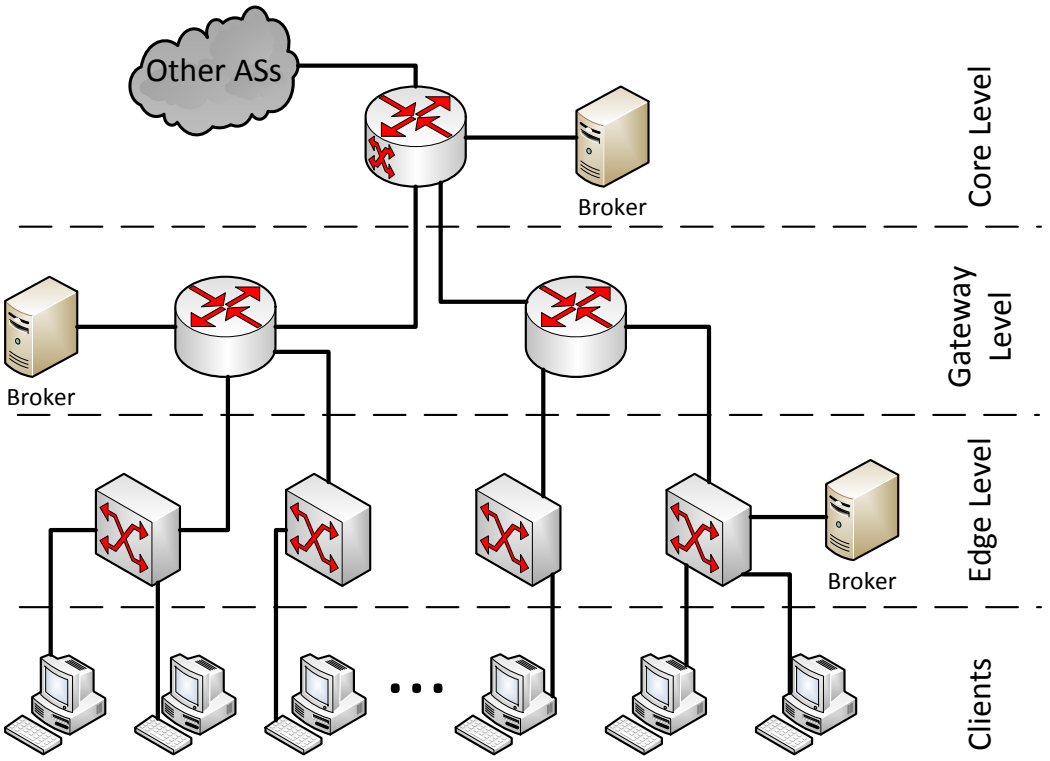


Figure 6.1: System overview for a single administrative domain: Brokers distribute continuous sensor data streams to clients. The underlying network structure needs to be considered when minimizing the resulting network usage.

per update and the update frequency. The amount of data is given by the geographical extent of the area and the spatial resolution. The frequency depends on the temporal resolution. For a data stream serving a query q the corresponding total size $|q|$ is therefore given by:

$$|q| = (x_{max} - x_{min}) \times (y_{max} - y_{min}) \times res_x \times res_y \times res_t$$

While the size of a query determines the bandwidth required to send the data from a broker, the load on the network depends on how data is actually distributed in the network. To serve a query q, the corresponding data stream s_q uses a set L_{s_q} of underlay links, which depends on the chosen path in the physical network. The number of underlay links for this path is denoted by $|L_{s_q}|$. The overall cost of a stream can now be calculated as the product of the number of links and the bandwidth which is identical for each link:

$$cost(s_q) = |L_{s_q}| \times |q|.$$

Note that we intentionally do not take the delay of a particular link into account for the cost. This cost model therefore stands in contrast to other approaches, where the bandwidth-delay product is used as a measure of network usage and cost [JPD03; KHR02; RDR11]. Although the delay has an impact on the amount of data currently in transit, it does not affect the load on a link considering the throughput, i.e. the amount of data transmitted during any given time period. In addition, most of the observed delay in high bandwidth data streams is caused by queuing effects rather than propagation delay [GN12]. The cost of each query is given exclusively by the overall amount of data transmitted by all entities of the network as described before.

6.3 Sensor Stream Distribution Problem

According to our cost model, the bandwidth usage of the GSG linearly depends on the overall number of underlay hops over which a particular sensor datum is forwarded. To achieve minimal bandwidth usage, it is therefore important to i) identify and remove redundancies between different sensor streams to reduce the amount of sensor data which needs to be forwarded, ii) carefully choose the brokers which contribute in relaying sensor data to ensure short underlay paths. Note that these goals are in general conflicting since longer underlay paths provide higher flexibility in sharing an underlay link between multiple streams.

We address the problem by focusing on the interconnection of two sub-problems. An optimal solution to each of them then allows us to find an assignment of data streams to network links using additional relays which will yield overall minimal bandwidth usage. The first sub-problem is to efficiently partition all available data in the GSG into parts so that each part can be entirely and independently delivered to a set of clients. The second task, and main problem addressed in this chapter, is therefore to reduce the bandwidth consumed by each stream by finding an distribution tree in the underlay at minimal cost.

Clearly, both problems are in general hard to solve: First, the number of intersections between query regions grows quickly with the number of queries in the system, a problem initially known from computational geometry [EM81]. Consequently, the number of parts, which can be indipendently distributed, grows as well. Second, the problem of setting up a distribution tree for each part using relay brokers poses a particular version of the NP-hard minimum Steiner tree problem (MST) [DW71; OP11; PS02].

To formulate the sensor stream distribution problem for a single region as MST, we first describe the relevant nodes in the network. Each client is thereby connected to a target broker which is topologically closest to that client. Now let $V = \{b_i\}_{i \in \{1,\dots,n\}}$ be the set of all brokers in the GSG, b_s be the source broker which provides the data for the region at hand, and R be the target brokers to which the clients are connected. Since a direct connection between any two brokers $b_i, b_j \in V$ can be established using IP-routing, the network is consequently described by a complete digraph $G = (V, E)$ with $E = \{(b_i, b_j) : i \neq j\}$. The edge weights are thereby given by the number of underlay hops, required to transmit data over the corresponding connection, according to our cost model.

In the generic version of the MST, we are given a graph $G = (V, E)$ with a set of nodes V and a set of edges E, one source node $s \in V$, and a set $R \subseteq V$ of required nodes. The problem is to find a graph which connects the root s to all leaf nodes in R using nodes in $V \setminus R$ as Steiner points, i.e. relays, so that the sum of edge weights is minimized. The solution to this problem is the desired tree, which provides the connectivity at minimal cost.

Even though a number of approaches exist, finding the exact solution to the presented version of the MST is infeasible in the real-world GSG for multiple reasons. First of all, the problem has to be solved for a very large graph, i.e. all direct connections between brokers. Since the exact topology of the Internet is not known, determining the exact edge weights is virtually impossible. This knowledge gap is broadened by the continuously

changing network conditions observed in the Internet. Second, a solution is needed for every intersection which results in numerous instances of the problem.

The goal for the GSG is therefore to establish a scalable method that allows for an approximate solution using only measurable information. Each broker must thereby be able to make independent decisions in order to avoid high bandwidth usage due to communication between brokers and high maintenance overhead for a large number of brokers. This requirement also prohibits the frequent collection of an entire underlay topology at a master broker. Furthermore, the number of relay points needs to be controlled despite the distributed operation in order to avoid extremely long network paths from a source broker to its clients.

6.4 Minimizing Overall Network Usage

As previously established, calculating the exact solution of the sensor stream distribution problem is infeasible in general. In this section, we describe a new approach to approximate a solution to the stream distribution problem and show how data streams in the GSG are organized to minimize bandwidth consumption. The approach is divided into four main parts: the query processing, the management of network information, the relay selection algorithm, and the merging algorithm.

6.4.1 Operation Example

To illustrate the main idea of our approach, consider an example of two clients requesting the exact same data. The clients request a sensor stream by connecting to their respective topologically closest target broker. Clients express their interest in sensor data in form of a query, which they send to their target broker. The target broker checks whether the requested data is available and forwards the query, if necessary. A continuous data stream is then sent from the corresponding source broker to the client. With the naive operation shown in Figure 6.2, two separate data streams are sent directly to the clients as it is unaware of the actual path in the network. All data covered by the queries is sent twice over each link of the underlay path except the last hop to the clients.

To address this problem, our approach aims to serve both clients with a single stream which is only split up as close to the destination as possible. The organization of data streams for the previous example is depicted in Figure 6.3. To achieve this organization, the initial steps of the process are similar to the load balancing presented in the previous

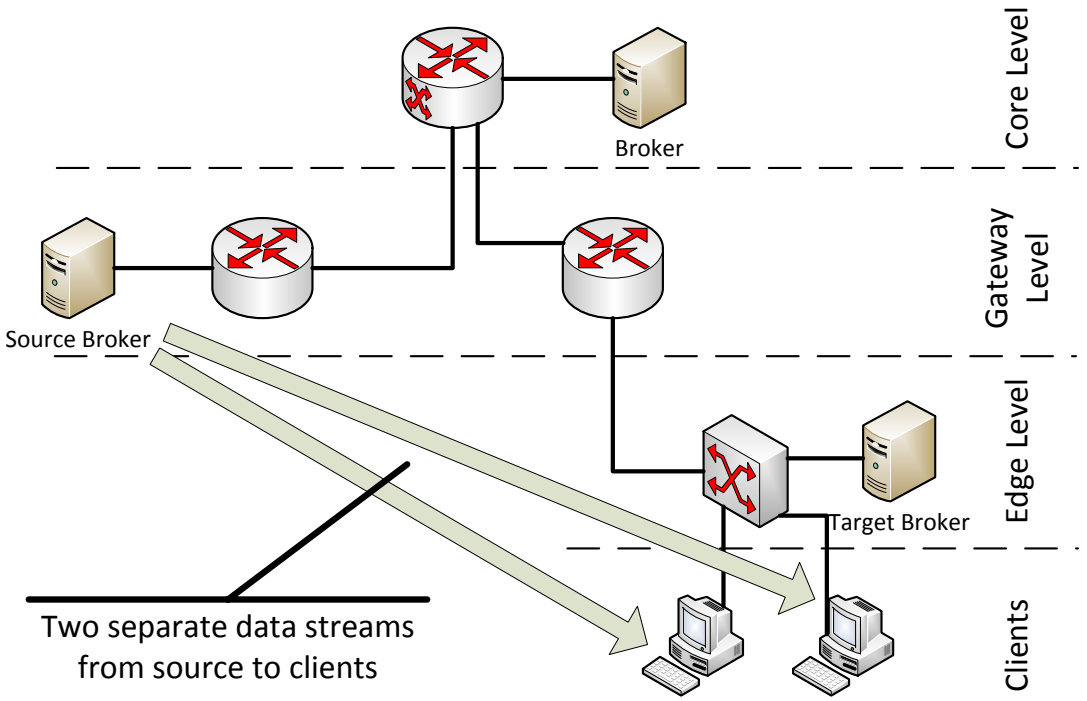


Figure 6.2: Naive operation with a separate data stream per client. Data is transmitted twice, if the clients request the same data.

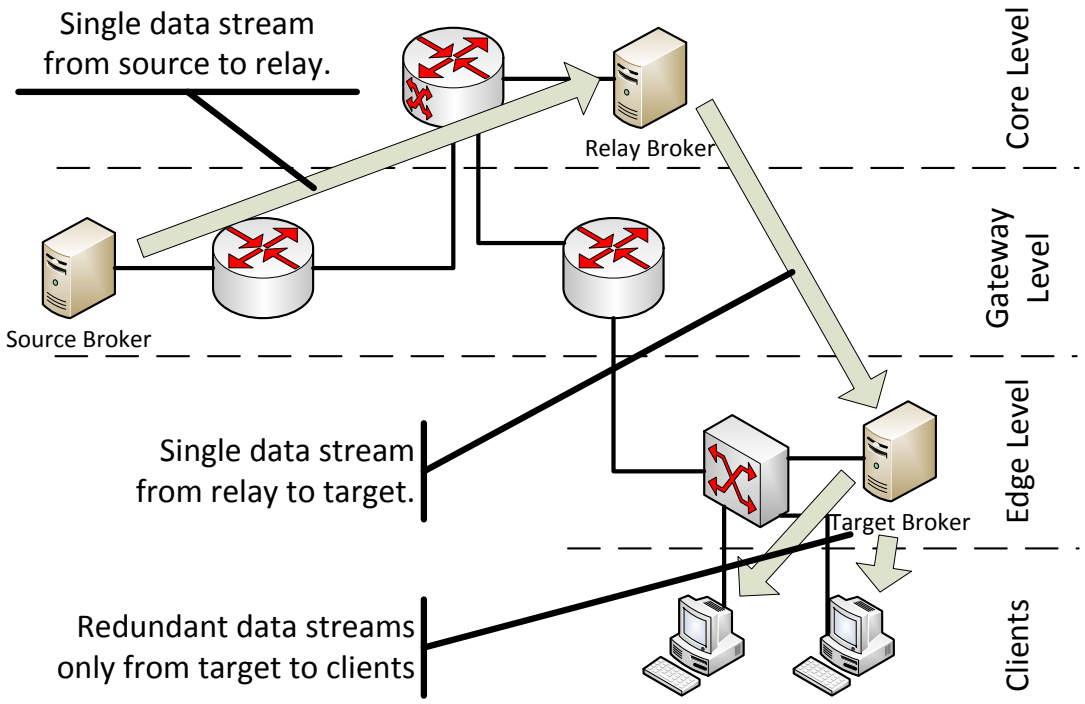


Figure 6.3: Merged data streams using a relay broker in the network. A single data stream is sent over the relay to the target broker which serves both clients.

chapter. The first task is again to efficiently identify the regions requested by multiple, in this case two, queries at the source broker. This function is provided by our query processing framework, which we detailed previously in Section 5.3. Next, we need to determine a relay broker which is closer to both clients than the source broker. To estimate the location of other brokers, we build a local simplified representation, called neighborhood, of the underlying network. Section 6.4.3 shows how the neighborhood management is implemented. Our algorithms then select relay candidates from the neighborhood for all target brokers as described in Section 6.4.3. Finally, the merging algorithm presented in Section 6.4.4 decides how and which outgoing streams are merged and sent to the relay as a single stream. In contrast to the first example, only one data stream is sent over a large fraction of the underlay path. This single stream carries the data for both clients which are served by the target broker. With this merged operation, the load on the links between the routers can be cut down by 50% for this particular example.

6.4.2 Optimizing Distribution Structures

The previously described optimal solution of the stream distribution problem not only involves complete knowledge of the underlying physical network. Rather, additional distribution points have to be dynamically established depending on the currently running queries. At each of these distribution points, a single stream is split into multiple branches resulting in a data distribution graph.

In existing distributed stream-based applications, this overall optimization problem is divided into two components and the task of mapping one to the other as shown in Figure 6.4. First, the physical network is represented by a virtual overlay network. Second, a structure similar to the data distribution graph, typically known as operator graph due to additional processing steps, is either generated manually or derived from the input and output data dependencies of individual operators. The operator graph is then mapped to the overlay network during the placement process.

Overlay Network The overlay network abstracts from the underlying network and provides the information about links required for the optimization metric. Note that the application layer of the network stack provides point to point connections between any two hosts. The overlay management therefore only contains a relevant subset of the possible connections and provides additional information about the available links. For example, when searching for shortest paths, connections using intermediate hops are

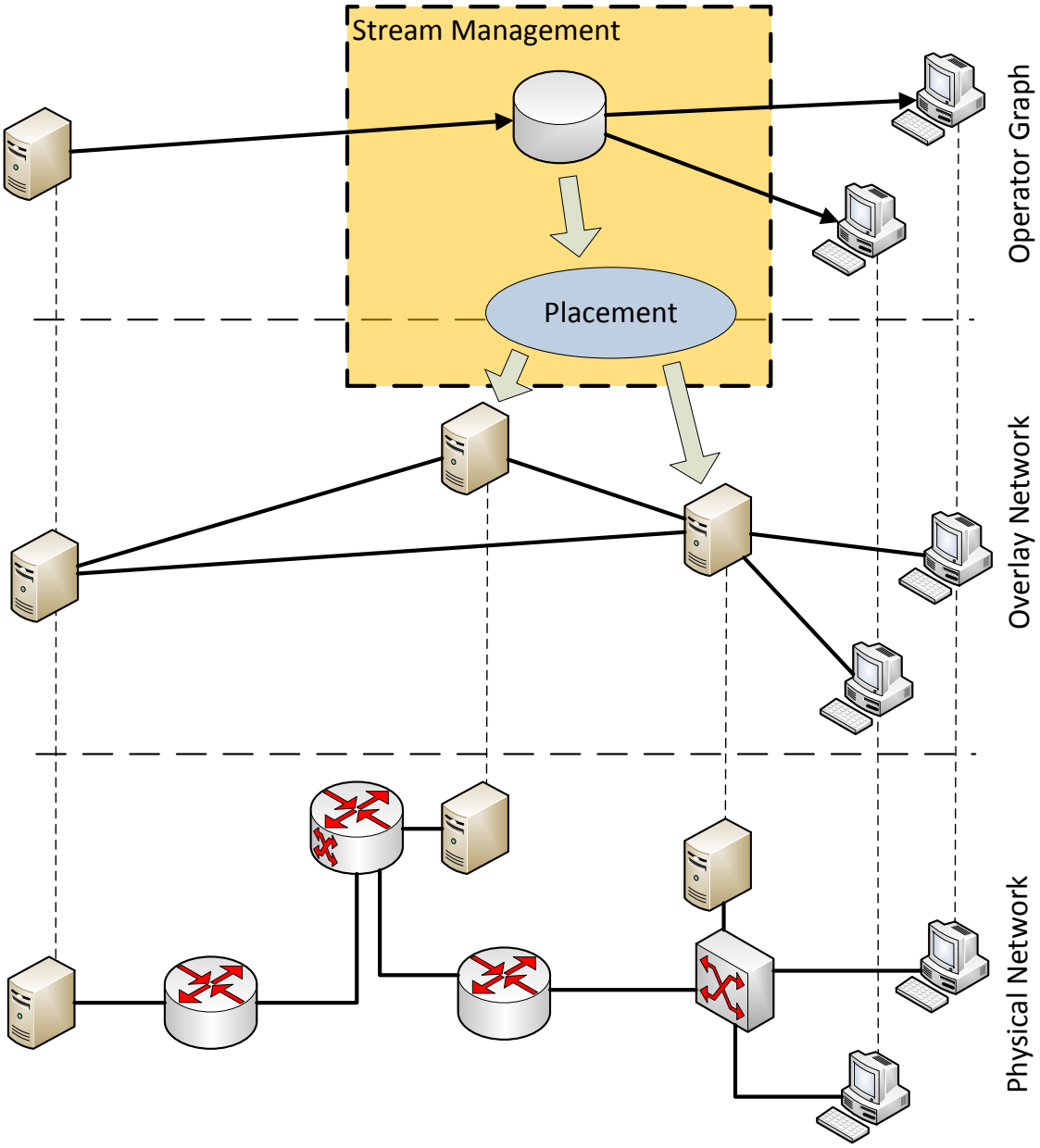


Figure 6.4: Separation of optimization steps in typical distributed, stream-based applications. The application is represented by an operator graph, which is mapped to nodes of the overlay network, which in turn abstracts from the physical network. The stream management of the GSGM combines operator generation and placement.

eliminated in favor of direct links. This approach also reduces the search space of the overall optimization.

Depending on the optimization goal at hand, the metadata gathered includes the available bandwidth, propagation and end-to-end delay, path length, monetary cost of operation, and/or other quantities. Since the optimization depends on these values, their exact determination is crucial to any approach. While some information, like the available bandwidth or monetary cost, is rather static, other data is highly dynamic. The latency, for example, directly depends on the status of queues in the network which in turn correlates with the current load. Apart from the intrinsic problems of the measurement process, the huge number of nodes and links in global overlay networks poses the additional problem of managing all the measurements.

Operator Graph For small applications with a manageable number of operators, the operator graph is explicitly modeled by application developers using a boxes-and-arrows interface. This approach has the advantage that human knowledge about the system can be used for manual optimization. Special cases and skewed distribution of input data can be considered when building the operator graph, even if such exceptions might not be part of the normal operation. However, since an optimal structure comprises all parts of the operator graph, manual optimization is not feasible for large number of operators and dependencies.

Especially in data-centric systems like the GSG, users formulate their abstract interest in data rather than providing detailed instructions on how to gather and process the data. Therefore, an automatic generation of the operator graph from a query or query language is required. This generated graph can then be optimized according to certain restructuring rules which preserve the semantics across transformations in the case of a query language. For example, filters are executed first to reduce the network traffic and computational load on subsequent operators. The particular possibilities for optimizing the operator graph strongly depend on the query interface or language used.

Independent of the way the operator graph is generated, the combination of multiple graphs creates additional difficulties. Redundancies in the combined graph have to be removed, in order to avoid wasting resources. This raises the problem of comparing different parts of the operator graph for semantic equivalence. On the one hand, no normal form of the sub-graph might exist which prevents automatic matching. On the other hand, a globally optimal result might only be achieved by abandoning locally optimal sub-graphs during combination.

Placement The task of mapping the operators of the operator graph to the nodes in the overlay network is called placement. Recall that the previous optimization was carried out on individual data structures. With the placement, the two data structures are finally combined to acquire a working deployment of the application.

In the first step of the placement the nodes which are capable of executing one or more of the operators are selected to limit the search space. In the presence of resource constrained operators, finding a suitable configuration, where all operators can be mapped, poses a problem by itself. For applications, which need to obey additional constraints like matching certain security domains or specialized hardware and software requirements, the resulting constraint satisfaction problem can already be NP-hard [SKR11].

The second step of the placement comprises the generation of a mapping which minimizes the resource consumption of the application. In contrast to the operator graph, where the optimization is done off-line, based on the end-to-end connections between operators, the placement also considers the properties of the network link which is used for that connection during run-time. For example, if the goal is to minimize end-to-end latency, operators are placed to avoid detours over links with a large delay. This way, the placement enables an underlay adaptive operation of the application at hand.

Integrated Optimization To achieve an overall optimal solution, the previously described data structures have to be generated and the mapping has to be executed so that the same optimization goal is met throughout the entire process. Configurations can thereby only be compared if the cost of an individual configuration can be determined in each of the steps, at leas relatively to other configurations. Obviously, this approach works well for applications with a set of operators and network nodes that is fixed during the optimization.

While the GSG experiences only very little churn, since the brokers do not change frequently, the data distribution points are generated dynamically during the optimization process. At this point lies the fundamental difference to other distributed stream-based applications, where the operator graph is either manually modeled or implicitly given by the data dependencies between operators. Each intersection of two query regions can result in an additional distribution point, depending on the location of the clients in the network. In addition, the location of a new distribution point depends on the topology of the overlay network.

Due to the tight integration of the problems in the GSG, we have taken an integrated approach to minimize bandwidth usage of the system. The overlay management is

limited to identifying topologically close brokers, in order to provide information about the structure of the underlying network. The generation of data distribution points is then delegated directly to the appropriate brokers by our stream merging algorithm.

6.4.3 Underlay Aware Overlay Management

In the following, we first describe how we abstract from the physical network by selecting a set of neighbors on each of the brokers. The overlay network of the GSG emerges by maintaining active connections to these neighbors. Afterwards, we describe how potential relay points for each data stream are selected from the neighborhood. Section 6.4.4 then shows how sensor stream distribution points are dynamically established in the network.

Neighborhood Maintenance

Our integrated approach to minimize bandwidth usage of the GSG does not require a complete overlay graph. Instead, each broker maintains a local neighborhood which is then considered for selecting relays for data streams. We now detail the requirements for the neighborhood before describing how each broker maintains its set of neighbors, including the discovery of new nodes.

Recall the basic trade-off between the performance of underlay connections versus the flexibility of relaying data streams: A direct connection, i.e. using IP routing, is based on the shortest path between the end points and therefore provides low latency and high throughput, given sufficient capacity. On the downside, the actual path of a direct connection cannot be influenced by the application and the corresponding stream therefore cannot be used for multiple destinations. In contrast, additional relay points provide increased control over the path but lead to longer paths in the network and therefore an increased number of underlay hops resulting in higher overall bandwidth usage. Consequently, while only very few data streams are present in the system, direct connections provide the best performance. However, for increasing load, multiple streams might carry redundant data over the same underlay link. In such a situation, the flow of these similar data streams needs to be adapted so they can be merged to a single stream to a selected relay point. This adaptation cannot be accomplished using direct connections.

The requirements for the neighborhood management directly result from this trade-off: To minimize the negative effects of adding relay points, we need to consider information about the underlying network topology when selecting neighbors. In other words,

neighbors selected by the GSG should also be topologically close to a broker. Furthermore, the topological distance information has to be provided to the other components of the optimization in order to achieve the same goal throughout the optimization process.

As previously described, each broker maintains a local view on the network to infer the structure of the underlying topology. This view contains a limited number of topographically close brokers, called local neighborhood, as well as additional information about each of these brokers. Two common approaches for estimating the distance between nodes exist: i) by network latency and ii) the geographic distance. While the first approach requires active measurements, the geographic location can be obtained from an external database like GeoIP [Max]. The geographic location correlates with the topological location of a client or broker, if they are part of the same network due to the physical structure of the network. However, two nodes might be located geographically close in the same city while residing in entirely different networks. As a result, high traffic would be exchanged between over long distance links, resulting in high cost w.r.t. our cost model. Latency, in contrast, has shown to be a reliable measure for network proximity [WSS05]. The GSG therefore uses the latency as a measure of proximity in the network.

Initially, brokers from the region assignment messages, which are flooded to all brokers, are added to the neighborhood. The neighborhood is then continuously updated with new candidates during query routing and neighborhood maintenance itself. When a broker learns about another broker, from the region assignment or through an update, that remote broker is added to the neighborhood and delay measurements to the new broker are initiated. Close brokers are kept in the neighborhood, while brokers with higher delay are removed from the active neighborhood and continuous measurements of delay to them are stopped. No additional topology information, for example data from network oracles, is used by the GSG.

To assess the network location of a remote broker with respect to the location of the target of a data stream, the local view contains additional information. For each broker in the local neighborhood, the neighborhood of that broker is periodically fetched together with the respective distance information. This way, each broker can check the network from the point of view of any of the brokers in its neighborhood, in the following called two-hop view. In addition, brokers in the remote neighborhood are considered for the local neighborhood.

Relay Selection

In this section, we describe our approach to relay selection which serves as the basis for merging data streams. Relay selection is executed locally on each broker to determine where to send relay data streams and which clients to serve directly. In the following, we first introduce our notion of data stream similarity which provides an intuitive description of the optimization goal. Simple relay strategies are then discussed to provide an understanding of the underlying effects of relaying. We then present our new strategy for selecting relay candidates to approximate the solution to the sensor stream distribution problem.

Basic Principles The main goal of our approach is to reduce transmission of redundant data, which is accomplished by merging redundant parts of data streams. However, as illustrated in the sensor stream distribution problem, merging data streams will only reduce the bandwidth consumption, if they share at least part of an underlay path. The query index already allows us to efficiently identify regions which are requested by multiple queries. However, it does not provide any information on whether the corresponding streams share any links in the underlay path. We approach this problem by introducing a similarity metric to formalize the path shared by two overlapping streams. Apart from the formalization, the metric provides us an intuition about the problem at hand and a qualitative measure whether or not it is worth merging two streams. Building on this metric we will introduce the relay selection strategies, which aim to find relay points for streams with a high similarity.

The similarity of two data streams carrying the same data is determined by the fraction of underlay links, which they have in common, in relation to the sum of their respective path length. Formally, assume two data streams s_1 and s_2 which carry the same data on paths p_1 and p_2 respectively. In this setting, $|p_i|$ denotes the length of path i and $l_{i,j}$ denotes the j^{th} link on path i. The similarity is then given by:

$$sim(p_1, p_2) = \sum_{l_{1,k}=l_{2,l}} \frac{2}{(|p_1| + |p_2|)}$$

If the paths are identical, a total of $|p_1| = |p_2|$ links will be summed up, resulting in a similarity of one. If the paths have no links in common, the similarity is zero.

Calculating the exact value of similarity would again require knowledge of the entire underlay network. However, as described previously, the network graph is neither known, nor can it be measured exactly. Instead of directly calculating the similarity, we therefore

approach the relay selection problem in the GSG using a heuristic strategy. Relay selection is executed locally for each known target broker since their number is small compared to the large number of distinct data streams (cf. Section 6.3). A good strategy needs to ensure two properties: i) the network stretch is minimized and ii) the probability of selecting the same relay for similar streams is maximized. The former property directly corresponds to a low overhead in terms of communication delay. The latter property is required in order to merge the corresponding streams, i.e. serve all targets by a single stream on the path segment from the originating broker to the selected relay, to reduce the load on the network.

Analysis of Basic Relay Strategies In order to clarify the reasoning behind our approach to relaying data streams, we will first introduce some basic relay strategies. Each of them greedily selects a relay based on a single network property. By identifying their respective limitations in selecting a relay, we identify which properties are essential for a combined strategy to comply to the two target properties presented above. The discussed relay strategies also serve as reference in our evaluations. An example showing the selection of each strategy from a local neighborhood is shown in Figure 6.5.

Two streams forwarding the same data can only be merged if they have the same source and the same destination, which is their common relay point. Without adding any relay points on the path, each datum of a sensor stream is directly transmitted to the corresponding client. In this case, the routing mechanisms of the underlying network provide efficient routes using the shortest path, e.g. the lowest number of underlay hops. To prevent redundant transmission, the simplest relay strategy is to forward data streams directly to a client's target broker. We refer to this strategy as *direct relay*. If the target broker is selected topologically close to the client, as done by the GSG, the introduction of the relay will result in a small additional network stretch. At the same time, however, the direct relay strategy allows to remove redundant transmissions with respect to clients which have overlapping query regions and are connected to the same target broker.

Using additional relays on the path between a source broker and the target brokers will significantly increase the chance of multiple overlapping streams on the same relay link. This interconnection becomes clearly visible when considering the average path length between two relays: More relays result in shorter paths which are more likely to be identical for two streams, i.e. have a similarity of one. However, at the same time, additional relays introduce an additional stretch in the network which results in additionally used underlay links and higher overall bandwidth usage. In general, relying

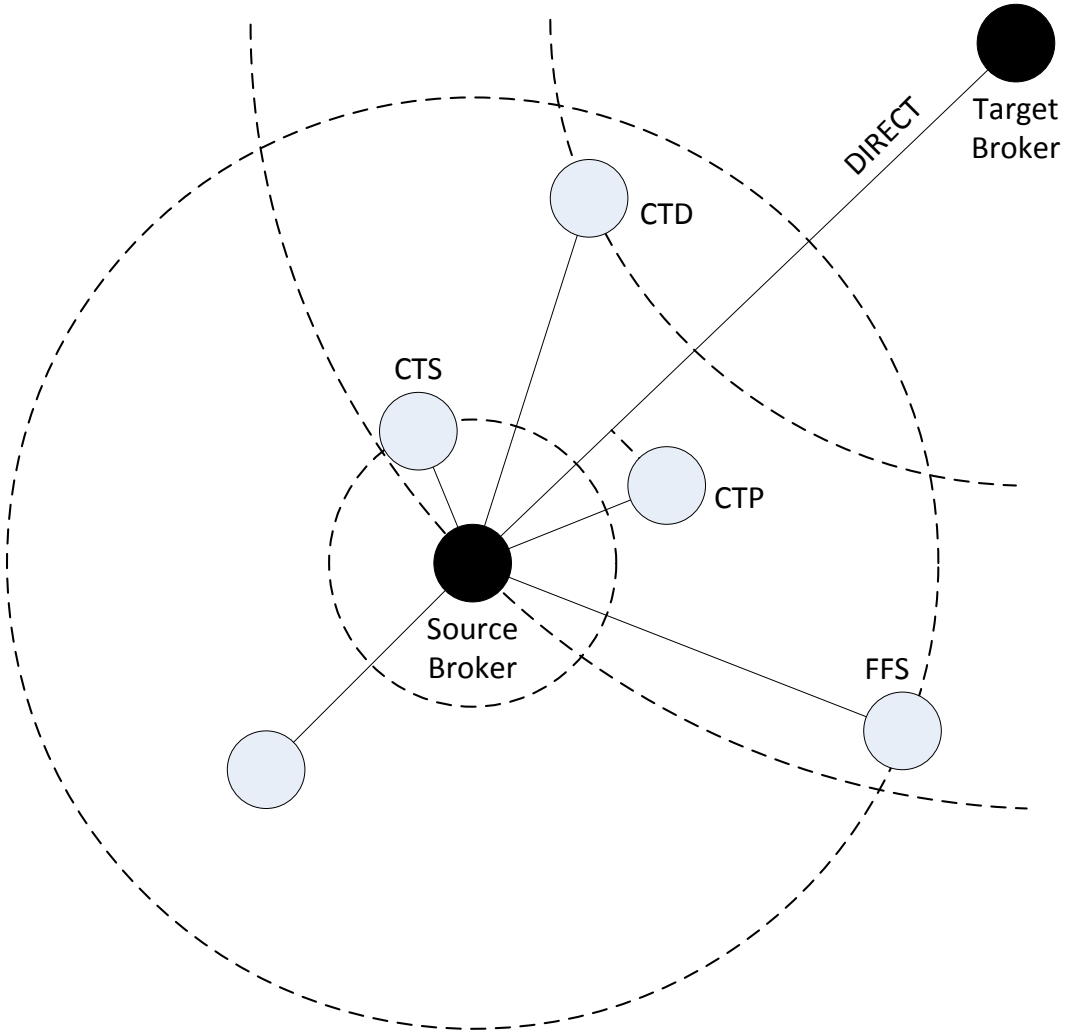


Figure 6.5: Example for selecting a relay from the local neighborhood using basic relay strategies: selection of target (DIRECT), closest to source (CTS), closest to destination (CTD), and farthest from source (FFS). Closest to path (CTP) represents the strategy used in the GSG. The distance between represents the delay.

on the established local neighborhood allows us to limit the stretch introduced by each additional relay. Furthermore, the limited set of relay candidates significantly increases the chance to find a common relay point for overlapping data streams.

Following this basic reasoning, the aim to generate only those relays *closest to destination*, results in a simple greedy strategy. To establish the relay path, the source broker will initially identify the relay point with shortest distance to the respective target broker. The details of the algorithm for the closest to destination strategy are shown in Algorithm 6. The selection process is similar for all basic strategies in a way that all neighbors are checked in a specific order for their location in the network relative to the source and/or target broker. At this point, the two-hop view generated by the neighborhood maintenance is used to select the appropriate broker. Each broker, which is selected as a relay point, will then execute the same selection until the target broker, which is topologically closest to the client, is reached which then forwards the data to the clients. If no broker in the local neighborhood is closer to the target broker, the target broker is selected as a fallback relay to ensure the delivery of the corresponding sensor stream.

```
input : dst: Destination Broker,
        neighbors: Neighbors Sorted by Distance from Source
result : Relay Broker
Function selectRelay(dst, neighbors) begin
    oldDist ← getDelay(this, dst);
    for n ∈ neighbors do /* check each neighbor                    */
        newDist ← getDelay(n, dst);
        if newDist < oldDist then /* n is closer to destination    */
            oldDist ← newDist;
            relay ← n;
        end
    end
    if relay = NULL then /* fallback to target if unsuccessful     */
        return dst;
    else
        return relay;
    end
end
```

Algorithm 6: Closest to destination relay selection algorithm.

While this greedy approach results in a comparably low network stretch, its drawback is the high path length from the source to the first relay in relation to the total distance.

As a result, streams, which have only a short part of their path in common, will be assigned different relay brokers and the merging algorithm fails to join these streams. Only paths which have a comparatively high similarity can be merged.

To exploit a low similarity of data streams by selecting a relay with a possibly low distance, the relays need to be selected *closest to source*. This strategy selects the relay based on the distance to the originating broker rather than based on the distance to the target. First, the neighbors are sorted increasingly by their distance to the originating broker. Then, the first candidate which is closer to the target broker than the direct connection is chosen as relay. The benefit of this strategy is that it is independent of the neighborhood size, as long as one neighbor is closer to the target. As a result, the data streams are routed over a relatively large number of hops. This strategy consequently provides the possibility to merge small common path segments, at the cost of higher average network stretch.

For the sake of completeness, we describe another basic strategy, *farthest from source*. Although this strategy seems to combine the drawbacks of the previous two, the resulting behavior can be beneficial for load-balancing purposes. First, note that advantage of selecting a relay with respect to its distance from the source: more target brokers are accumulated on the same relay. This way, an overloaded broker can delegate multiple sensor streams to a single relay. Second, the negative effect of high increase in network stretch is balanced by the benefit of widely dispersing the load. The additional stretch helps to avoid a concentration of relays in a small region of the network close to the overloaded source. However, the increased path lengths lead to bad quality of service in terms of user perceived application delay.

Combined Strategy for Minimized Network Stretch We conclude the analysis with the strategy used in the GSG for relay selection. The individual strategies show that simplistic approaches tend to extreme emphasis on a single goal. To find a relay which best exploits similar data streams, we need to combine limiting network stretch while simultaneously making use of the entire neighborhood for merging streams by favoring nearby relays. Rather than selecting a position relative to either the source or the target of a data stream, the relay is therefore chosen *closest to the path* between the two with a preference for relays closer to the source.

The pseudo code for the resulting strategy is given in Algorithm 7. Before the relay selection is started, the brokers in the neighborhood are sorted with increasing distance from the source broker. For each candidate we first check whether it is closer to the

```
input : dst: Destination Broker,
        neighbors: Neighbors Sorted by Distance from Source
result : Relay Broker
Function selectRelay(dst, neighbors)begin
    oldDist ← MAX_DIST;
    for n ∈ neighbors do /* check each neighbor                          */
        newDist ← getDelay(this, n) + getDelay(n, dst);
        if getDelay(n, dst) > getDelay(this, dst) then
            continue;
        end
        if newDist + ϵ < oldDist then /* relay has lower network stretch  */
            oldDist ← newDist;
            relay ← n;
        end
    end
    if relay = NULL then /* fallback to target if unsuccessful          */
        return dst;
    else
        return relay;
    end
end
```

Algorithm 7: Closest to path relay selection algorithm.

target broker than the direct connection in order to avoid unnecessary detours. If a candidate is closer to the target, we evaluate the expected network stretch for that relay candidate by summing up the distance to the relay and the distance from the relay to the target. During the process, the broker with the least estimated total path length is kept as the result of the relay selection. Note that in order to prefer relays close to the source, we use an offset ϵ during the comparison. If no suitable relay broker is found in the local neighborhood, the target itself is selected as relay. This direct connection is also used if the target broker is part of the local neighborhood and no candidate with a sufficiently small stretch is found beforehand.

Using this strategy, we gain in both aspects towards our goal. First, the network stretch is minimized, as the relay nodes are selected as close to the direct path towards the target brokers as possible. Referring to our cost metric, the shorter path results in a lower number of underlay links and therefore overall reduced bandwidth usage. Second, the distance to the next relay is limited, not only by the size of the neighborhood, but also by selecting closer brokers with higher priority. This limitation allows for an increased number of relays compared to the greedy closest to destination strategy. Consequently, streams with low similarity can be assigned to the same relay. As a result, the strategy incurs low additional delay while allowing for reduced bandwidth consumption.

6.4.4 Heuristic Online-Adaptation of Streams

The relay selection provides a relay candidate for each target broker and therefore for each outgoing data stream. Based on this selection, each broker has to locally decide how exactly streams are merged towards each relay. The goal of the merging operation is to find possibly large regions for relaying to merge as many streams as possible and keep the number of connections and the resulting overhead low. The basic approach for this operation is to find large regions which carry data for multiple, potentially partial, queries.

As previously described, the query index provides the foundation for quickly identifying the containment relation of query regions. Our novel merging algorithm furthermore exploits the index structure to maximize the number of streams, which are merged for a single relayed region. Queries that partially intersect with that region are thereby split, in order to achieve exact overlap and therefore increase efficiency and flexibility. The maximum fragmentation of queries caused by the algorithm is limited to ensure a low number of relay data streams to minimize connection overhead.

The selection and adaption of regions to relay works in two main steps. First, the potential for merging streams is annotated in each node of the tree similar to the load estimation described in Section 5.4.2. A query q is contained in the sub-tree marked by node n if the region assigned to n covers the region of q. This relation is denoted by $R_q \leq R_n$. Recall the example in Figure 5.1, where a query for region 010 is contained in the nodes marked ϵ, 0, 01, and 010. In addition, we use the result of the relay selection to group queries by their relay candidate. For a target broker t_q of query q the corresponding relay is given by $r = relay(t_q)$. The set of queries for a node n and relay r is now given by $Q_{n,r} = \{q : R_q \leq R_n \wedge relay(t_q) = r\}$. In our algorithm, this set is computed by `getQueries`.

The total amount of data for queries per node n and relay broker r can then be described by the total size of queries in $Q_{n,r}$:

$$size(n, r) = \sum_{q \in Q_{n,r}} |q|$$

Recall that the size of a query is proportional to the bandwidth required to serve it. The potential for bandwidth reduction can therefore be identified by $overlap(n, r) = size(n, r) - |n|$, where $|n|$ denotes the size of the region assigned to n. By calculating $size(n, r)$ and $overlap(n, r)$ for all nodes and relay candidates, we annotate the index with the size of queries per relay candidate in each node.

In the second step, the query index is traversed top-down to identify nodes with preferably large overlap. The first condition in Algorithm 8 checks whether the entire sub-tree of the current node contains queries which can be relayed at all. If $overlap(node, relay) < \Delta$, total overlap is too small to be considered for relaying and the process terminates for the current sub-tree. Δ is a system parameter which controls the minimal relay region size and therefore limits fragmentation due to the split described below.

If a node does not contain any queries, the merging process continues on the child nodes.

If there is only a single query available at a node for the currently considered relay, the query is split. The resulting two parts are added to their two corresponding child nodes and the process continues on each of the two child nodes. Note that query parts are rejoined during the bottom-up gathering of metadata, if possible, in order to further reduce fragmentation.

If a node contains more than one query which can be relayed by the same broker, a new relay is established. All queries assigned to that relay broker in the sub-tree are merged

```
input : node: Index Node,
        relay: Relay Broker
result: Set QF of Queries to Forward,
        Set RS of Regions to Stream to Relay
Function mergeStreams(node, relay) begin
    if overlap(node, relay) < Δ then
        /* not enough overlap in this subtree, return                */
        return;
    end
    num ← count(getQueries(node, relay));
    if num = 0 then
        /* no queries on this level, continue on children            */
        for child ∈ getChildren(node) do
            mergeStreams(child, relay);
        end
    else if num = 1 then
        /* potential for merging, refine queries to sub-trees        */
        pushDown(node, relay);
        for child ∈ getChildren(node) do
            mergeStreams(child, relay);
        end
    else if num ≥ 2 then
        /* entire region is used for at least two queries, relay     */
        QF ← QF ∪ getQueries(node, relay);
        RS ← RS ∪ getRegion(node);
    end
end
```

Algorithm 8: Stream merging algorithm.

into the newly established relay, i.e. added to the set QF of queries to forward. Queries, which have previously been forwarded are also added to QF. The region corresponding to the node which consequently covers all the queries is added to the set of regions to forward.

To establish the relay data stream, all queries in QF are forwarded to the relay. A copy of the queries is kept in local storage to gather metadata for future merging attempts. Other relays, to which queries had previously been assigned, are informed about the removal. Starting with the next update, a single data stream, containing the data of the relayed region, is sent to the relay broker. Once all queries have been sent to the relay broker, the source broker signals the relay broker the end of the relay step. The relay broker then starts the merging algorithm using the queries and relays from its local view.

The merging algorithm is executed periodically on each broker, if no incoming relay triggers the execution. To avoid excessive computational overhead the process is only started, if the contents of the index have changed. In addition, new queries are checked for similarity to all existing data streams and directly forwarded, if applicable.

6.5 Evaluation

The bandwidth minimization approach used in the GSG has been implemented using the OMNeT++ network simulator [Var10] including the INET Framework for IP-traffic simulation. A realistic topology was generated using the tools of ReaSE [GS08]. The topology consists of seven autonomous systems (AS), two of which are set up as transit domains. The AS themselves have a hierarchical structure, similar to Figure 6.3, based on up to three core routers on the top level which are fully connected. The second level consists of up to three gateway routers per core which are connected to the edge routers on the third level. Client nodes are connected to the edge routers using asymmetric DSL. Brokers are placed depending on the scenario.

All evaluations are compared to the operation without any relaying (OFF). As benchmark setups, we chose to add the simple strategies presented in Section 6.4.3, including the DIRECT relay, closest to destination (CTD), and closest to source (CTS) strategies. To show the impact of queuing delay in comparison to increased path length, we included the farthest from source (FFS) relay selection strategy. The main strategy used in the GSG is abbreviated CTP, as it primarily selects a relay closest to path. The numbers attached to the label show the limit of the corresponding neighborhood size, where applicable.

The network setup included 577 clients and an average of 48 brokers selected randomly for each experiment. The brokers where directly connected to routers on different levels of the topology. Each of the clients generated queries in uniformly distributed time intervals between 60s and 300s. Each query lasted between 60s and 300s compared to a total simulated time span of 600 seconds. Each scenario was run 10 times, and the figures show the average results, unless otherwise noted. The queried regions were distributed around a single point with an exponentially distributed query size in all dimensions. This distribution simulates high interest on a small region, where some queries additionally cover the vicinity. Such behavior occurs, for example, in the case of a hurricane, where the center is of high interest, while prediction systems also require data of the surrounding area.

To show the impact of the size of the search space for relay candidates on the performances, we modified the neighborhood size. Three settings were chosen to represent the main operation ranges. A neighborhood size of 4 covers only a subset of the brokers in the local AS. 60 neighbors cover all brokers in our simulation runs. An intermediate value of 20 brokers represents the realistic setting, where the neighborhood is neither complete nor restricted to the local AS.

As described in Section 6.3, the location of brokers in the networks is crucial for the performance of the overall system. Again, three scenarios where considered: For the core placement, brokers where only connected to gateway and core routers to represent placement in data centers. For the edge placement, brokers where only connected to edge routers to simulate only the sensor gateways acting as brokers in the GSG. In the full placement scenario brokers are distributed randomly among all routers.

6.5.1 Bandwidth Usage

To show the effectiveness of our approach, we investigated the overall data transmission load on the simulated Internet caused by the GSG. The basic load, which cannot be influenced by the approach presented in this chapter, was not subject to our evaluations. We limited the observation to data transmitted by core and gateway routers. The usage of the links to brokers, as well as the links to the clients, has consequently not been taken into account. Figure 6.6 shows the total traffic for the different placement scenarios of brokers and different relay strategies.

Overall, all relay selection strategies are able to considerably reduce the total traffic used by the GSG. The narrow differences between the different approaches are partially caused by the reduced size of our evaluation setup. A detailed analysis of control samples

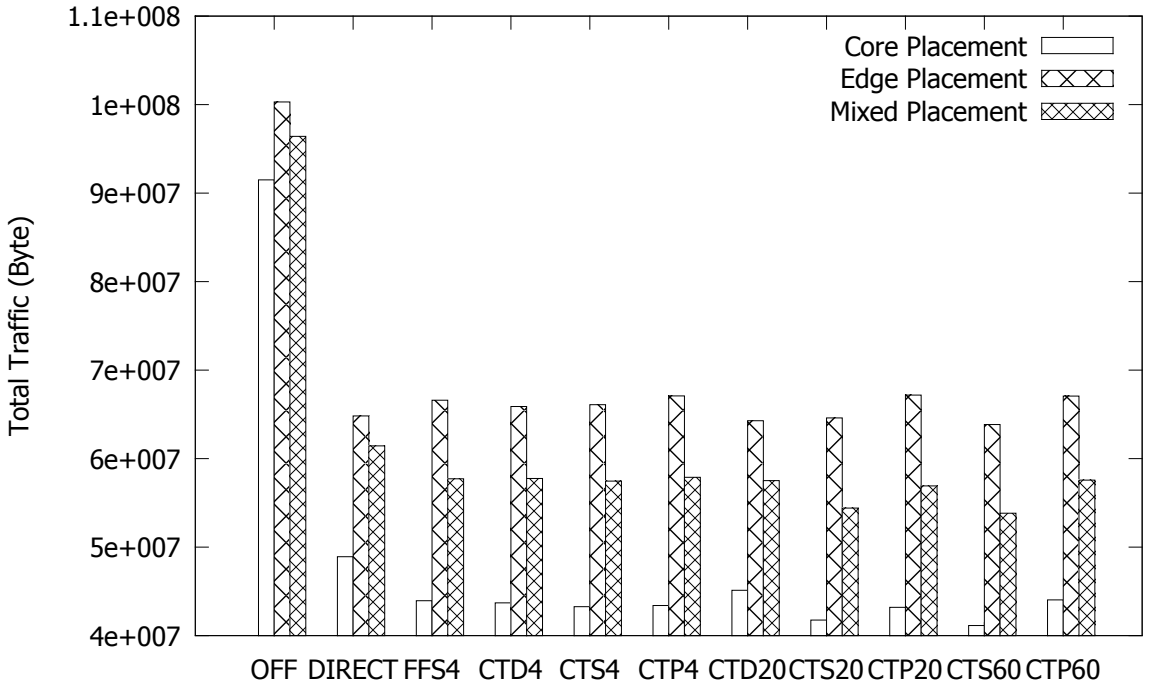


Figure 6.6: Overall traffic used by the GSG.

has shown that especially the simple strategies benefit from the limited number of brokers in the simulation. The detailed properties of the strategies become clear when considering the quality of the results as shown in the following sections.

The results clearly show, that the overall traffic is lower for brokers being located in the network core rather than in topologically remote locations. This effect becomes even stronger in scenarios with a sparser distribution of clients. Shorter paths between core placed brokers in the network lead to efficient use of the network. For the long paths between edge placed brokers, additional relays require more underlay hops, which lead to a higher bandwidth usage, according to our metric. As a result, the DIRECT relay strategy with its few relays performs best for the edge based scenario. However, the DIRECT strategy cannot benefit from additional brokers in the core of the network.

6.5.2 Network Stretch

As previously indicated, while the different strategies perform comparably well in our evaluation scenario considering the bandwidth used, they significantly differ in the emerging structures. A key indicator for these structures is the average depth of the corresponding distribution trees. If each broker distributes data to numerous child nodes,

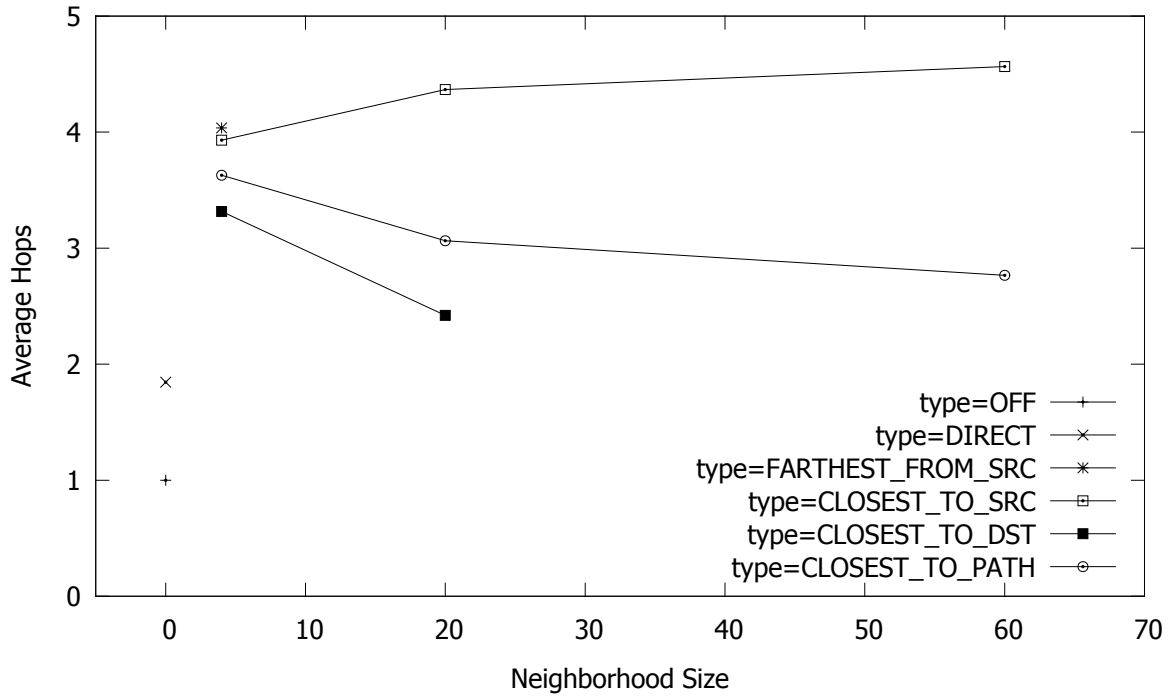


Figure 6.7: Average number of application layer hops for data received by clients for core placed brokers.

shown by a low depth, the merging obviously performs poorly. However, if the number of children is too small, the tree degrades to a linear distribution which results in high depth, indicating prohibitively long paths for data delivery. To show the impact of the relay strategy on the length of the delivery path in the overlay, we recorded the number of brokers which were involved in the delivery of each stream seen by a client. The results are shown in Figure 6.7.

The strategies which select a relay with respect to its distance to the source, i.e. CTS and FFS, lead to an increase in the number of hops due to the higher fan-out in distribution. With an increased number of brokers in the neighborhood, these strategies further increase path length. Especially the farthest from source strategy depends on the limits provided by the neighborhood size to avoid extreme path lengths. Contrary, the number of hops drops with increased neighborhood size for the CTD and CTP strategies. However, for the CTD strategy, the probability of selecting the target broker as a relay increases with the size of the neighborhood. As a result, the behavior of CTD quickly converges to the direct relay strategy as the neighborhood size increases. In contrast, the CTP strategy benefits from the larger number of potential relays without resorting to

direct shortcuts. Overall, the CTP strategy achieves the best balance between merging more streams at a time and increased network stretch.

6.5.3 Client Perceived Delay

The reduced bandwidth usage of in the GSG also has a positive effect on the user perceived quality of service in terms of delay. In a high bandwidth streaming system, the user perceived delay is significantly caused by queuing. By distributing fewer data from each individual broker and leveraging more relay nodes, the resulting smaller fan-out helps to reduce the queue sizes. However, the additional processing required during routing and the increased propagation delay due to longer paths lead to increased client perceived delay for each additional overlay hop. The combined effects can be seen in Figure 6.8.

Without using relaying, the many outgoing data streams on the source broker lead to extremely high queuing delay for all broker placements. For the edge placement, the negative effect of longer paths is clearly visible compared to brokers placed in the network core. However, for the DIRECT strategy, the low fan-out also results in increased queuing delay in the core placement scenario. Overall, the comparison shows that the CTP strategy performs best.

6.5.4 Scalability Improvement

The previous results showed the performance of the system for a fixed set of queries rather than the increase in maximum load that can be handled using our approach. Without the merging of data streams, the outgoing bandwidth required to serve data can quickly become a bottleneck if many queries are requesting data from a single broker. As introduced in Chapter 5, serving multiple queries with a single merged data stream can avoid these bottlenecks.

To show the improved capacity of the GSG, queries were added to the system for a single source broker over a time span of five minutes. The relaying was triggered every 30 seconds. We measured the increasing user perceived delay over time to show the impact of increasing load. As Figure 6.9 shows, the system is able to serve all queries for approximately 150 seconds without relaying. The system is operating robustly albeit causing an average delay of over 200ms. After that point, packets are lost and the maximum delay rises abruptly.

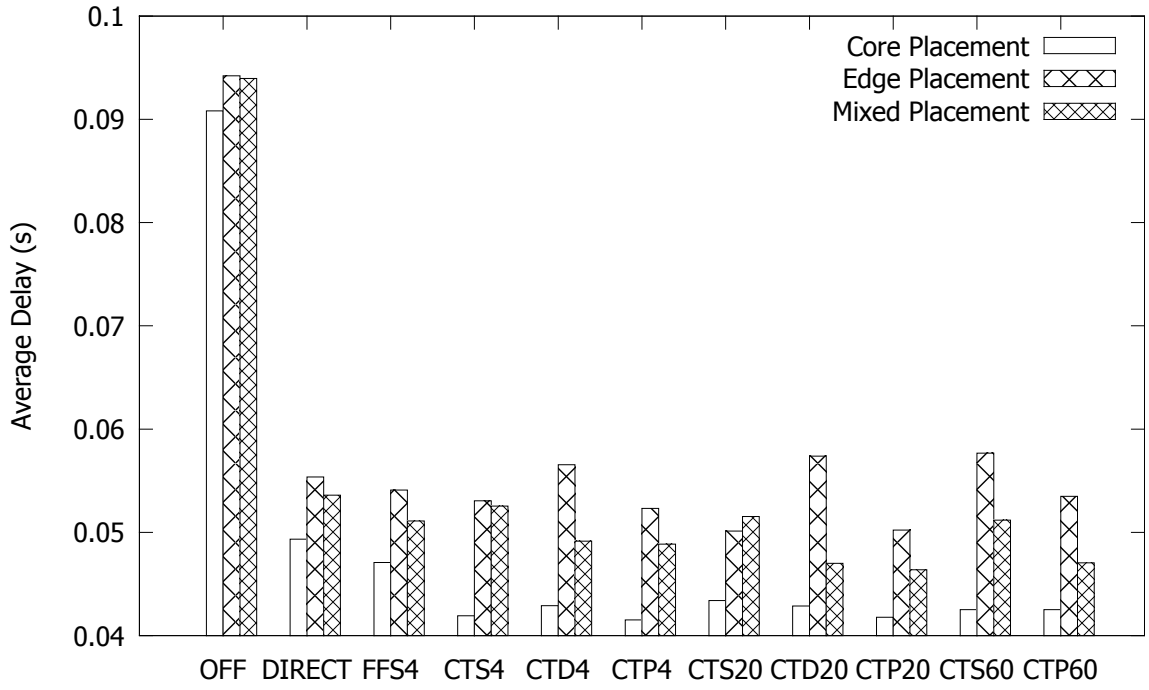


Figure 6.8: Average delay observed by the clients of the GSG.

With stream merging enabled, the average delay rises as quickly as without until the relaying is triggered. By combining data streams, the load on the source broker is reduced and the average and maximum delay are kept on a low level. Note that the size of client queries is limited in all cases by the bandwidth of links to the client, to avoid any effects from queuing delay and packet drops on the last mile.

6.6 Related Work

While data deduplication is commonly used to eliminate redundancies in centralized storage systems [WJZF10], multiple categories have been addressed in distributed applications: The solutions of the first category, ALM [BBK02; CDKR02; CRSZ02; ZZJ+01], have been proposed to handle redundant queries for identical data. The wide range of approaches has also been covered in multiple surveys [HASG07; KS10; YLE04]. These systems are tailored to channel-based communication, like TV-broadcasting, where a rather limited number of channels with a high number of clients need to be served. Consequently, the common underlying approach relies on building a distribution tree where the clients are attached as leaves. The expensive reorganization of the core is only done very infrequently. However, the content of the sensor data streams provided by the

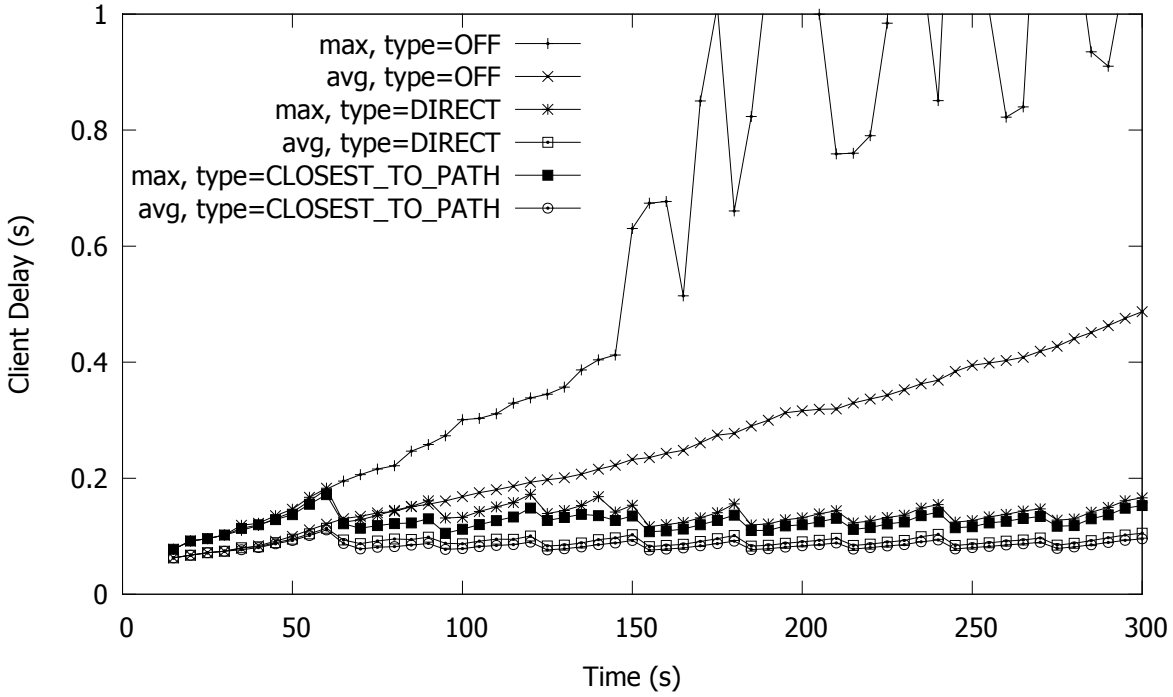


Figure 6.9: Client perceived delay with increased load over time for core placed brokers.

GSG cannot be divided into channels but is rather determined individually by the users' queries. Although extensions have been proposed to map queries on a coarse grid and thereby limit the number of query intersections [BRT00], the resulting false positives incur prohibitively high load on the network.

Content-based publish/subscribe systems [EFGK03] as the second category aim to establish a containment relationship between the data sources and sinks, i.e. source brokers and individual clients. Publish/subscribe systems provide a flexible subscription interface, which is capable of handling arbitrary query intersections. Typically, such systems use clustering [TKK+11; TKKR12] or merging of subscriptions [BFG07; JJE10] to reduce the processing overhead at the cost of increased bandwidth usage. For the intended application scenario, distributed event processing, the amount of false positives remains low, as critical events that cause additional traffic rarely occur. This is especially true for systems which consider the underlying network structure [TKR13] similar to the approach presented in this chapter. In a GSG, however, sensor data is sent out in high volume for all queried regions. Therefore, these approaches are not suitable to serve arbitrarily intersecting queries for sensor data and their resulting high bandwidth data streams.

A third related category, distributed stream processing (DSP) systems [AAB+05; GKK+03; PLS+06; RDR11; SMW05], move processing tasks to optimize application performance and minimize network usage. Based on the underlying assumption, that some of the processing tasks which can be moved are filters or otherwise reduce the bandwidth used, these tasks are placed closer to the data sources. The remaining tasks are placed in a way to minimize the amount of data that is currently in transit on the network. However, these systems require a definition of an operator graph as indicated in Section 6.4.2. The dynamic generation of a data distribution structure from individual queries, as supported by the approach presented in this chapter, is not supported.

6.7 Summary

In this chapter we presented our approach to minimize bandwidth usage for measurement distribution in the GSG. Using the previously introduced index structure, we were able to efficiently identify distinct regions for which streams can be independently managed. In doing so, the underlying physical network structure is first approximated by gathering delay information to neighboring brokers. In a second step, our relay selection algorithm decides on the next hop for each data stream. Based on the metadata extracted from the index and the selected relays, the presented merging algorithm finally selects and combines data streams towards relays for further distribution. The result of the merging operation is an optimized distribution structure for data streams.

The presented approach has been evaluated on an Internet like network topology. To account for a realistic broker deployment, the evaluations covered placement of brokers at the network edge, to represent sensor network gateways executing the Global Sensor Grid Middleware (GSGM), as well as in the core of the network, e.g. in data centers, and a combination of both extremes. The results show that our strategy yields at least 33% reduction in overall traffic. Furthermore, our approach can exploit brokers placed in the core of the network, achieving a total reduction of traffic of over 50%. The evaluations also show the positive effects on user perceived delay due to reduced queuing.

7 Conclusion

This chapter first provides a summary on the topics covered in this thesis along with concluding remarks on the obtained key results. Subsequently, remaining open research questions are discussed, to guide future work in the domain of global sensor networks (GSNs) is given.

7.1 Summary

Over the last years, simulations have gained a strong momentum across virtually all scientific domains. The increased resolution and complexity of the underlying simulation models require new approaches in the field of computer science. On the one hand, computational resources are required to support the calculations for large scale simulation models. These resources include workflow solutions, to maintain the complexity of interacting software components, as well as high performance computing for large processing demands. On the other hand, an increased amount of measurements of the physical world is required to devise simulation parameters, verify results, and simulate future events. To this end, additional sensor data sources have to be made available to simulations in real-time.

In this thesis, we presented the Global Sensor Grid (GSG) which provides methods to address the key issues for integrating real-time sensor data into simulations. We discussed the overall data integration workflow used for scientific simulations. From the sensor perspective the workflow starts with continuously monitoring certain regions for defined environmental conditions. If the monitoring indicates data of interest, a full grid of data is requested as a continuous data stream for the simulation to further analyze the situation. For both types of queries, we derived requirements for the corresponding query interface which have to be provided by a global sensing infrastructure. A monitoring query is primarily defined by the region to monitor and the type of sensor. The queries for a high-resolution stream, in contrast, require the interface to include the specification of the exact resolution not only in space but also in update frequency.

The GSG requires a highly efficient implementation of the monitoring query interface to support a large number of concurrent, long-running queries. The reduction of data in our approach is achieved by aggregating data of the sensors in the observed region. Aggregated results of existing queries are thereby made available to be reused by other queries. Partial aggregates are used to serve queries where possible to avoid the repeated transmission of data from individual sensor nodes. To coordinate the partial aggregates, we developed a distributed index structure based on the R-Tree. The index structure serves two purposes: On the one hand, queries are routed along the branches of the index tree towards the leafs to quickly find relevant partial aggregates. On the other hand, the measurements are aggregated on their way to the root of the index tree, resulting in highly efficient communication. The reuse of partial aggregates, however, results in additional processing steps and increased communication delay. Nevertheless, real-time capability of monitoring queries in the GSG is provided using the concept of predictors. We have successfully developed a new update strategy, which allows the efficient operation of simple prediction models over multi-hop communication. The result is a monitoring component, which can scale to a large number of queries with a very low communication footprint.

Once a monitoring query indicates data of interest, a stream consisting of a regular grid of data points, which are periodically updated, is required to feed measurements into a simulation for further processing. The previously derived query interface for these high resolution data streams provides the possibility to specify the density of data points as well as the update frequency in addition to the region of interest. Note that a potentially large number of data streams can be triggered simultaneously by monitoring queries. In this case, existing sensor query systems lack the capability to provide data at multiple resolutions as required by our interface. To close this gap, we developed a multi-resolution query processing framework. The resolution information is thereby mapped to additional dimensions of an index structure. The flow of data starts at the individual sensor nodes which are located in isolated sensor networks. Measurements are then gathered by dedicated brokers, which preprocess the raw data and provide the requested streams. Our query processing uses the metadata from the index to quickly identify the data requested by each query. As a result, we are able to efficiently detect the overlap between different queries. Using the extracted information about overlapping queries, we built a scalable stream distribution system. Regions with high load, which are covered by a high number of queries, are dynamically replicated to additional brokers in the GSG to spread the load and eliminate bottlenecks.

Since the GSG uses the Internet for communication, the properties of the network cannot be easily obtained let alone controlled. The physical network structure changes constantly, as network operators extend their infrastructure. Numerous other applications in the network cause dynamically changing load on links, which consequently underly high volatility in terms of available bandwidth and delay. The system therefore has to adapt to changes in the network topology and to the current load, which is caused by other applications. At the same time, the load caused by the GSG should be minimized in order to increase scalability. We addressed this challenge with an underlay adaptive data stream management, which builds on the query index and replication mechanism. The underlying physical network structure between brokers is measured and abstracted to extract the required topological information. This knowledge is used to relay data streams among the brokers of the GSG, which allows us to control the network path of each data stream. An optimized distribution of data streams is then achieved by merging data streams, which contain identical data, depending on the topological location of clients. The result is a distribution structure, which adapts to the current load in the network and minimizes the transmission of redundant data on physical network links.

Overall, we have proposed a system which provides scientific simulations with sensor data in real-time. The interfaces provided support the continuous monitoring, as well as the provision of sensor data streams with defined spatial resolution and update rate for real-time simulations. Our stream management achieves scalable operation by distributing the load among the entire system and minimizing network usage.

7.2 Future Work

While the presented work provides the foundation for a sensing infrastructure tailored to simulations, we have identified areas which require further research. The recent trend to more dynamic data structures used in simulations poses new challenges on the query interface to sensor data and the provisioning of the resulting data streams. Adaptively refining meshes, for example, require separate queries for each resolution used in the entire simulation, which currently results in additional delay due to query processing on mesh change. To prevent this query overhead, the refinement conditions need to be directly integrated in the query interface. The resolution of a data stream can then be adapted automatically by the GSG and pushed to the simulation at hand. In addition to adaptive mesh refinement, simulations might use an arbitrary distribution of data points rather than a regular, hierarchical structure. Additional mesh structures besides rectangular

grids and individual specification of data points therefore need to be supported by the query interface. New methods need to be developed, to enable the optimization of data streams resulting from such an extended interface.

The challenge of reducing network usage will become even more important as the number of sensors keeps growing and simulation models are refined to higher resolutions. To this end, the separation of the simulation and sensing system has to be reconsidered. If simulations are executed directly in the GSG, data streams can be drastically reduced, since the large amount of raw sensor data can reduced to few high level results at the source. Such a tight integration, however, requires new placement strategies, which consider the sensor data requested by simulations, as well as the resources required to execute them. Similar to the overlap detection used in the GSG, common components of simulations can then be merged, in order to optimize their execution. The integration of various simulation models on a global scale will be the first step to create a virtual globe by means of real-time simulation.

List of Figures

List of Tables

Bibliography

[AAB+05] D.J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.H. Hwang, W. Lindner, A.S. Maskey, A. Rasin, E. Ryvkina, et al. "The Design of the Borealis Stream Processing Engine". In: *CIDR '05: Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research*. Jan. 2005. URL: http://www.cs.harvard.edu/~mdw/course/cs260r/papers/borealis-cidr05.pdf (cit. on pp. 24, 27, 30, 48, 76, 106, 137, 172).

[AAB+06] Lisa Amini, Henrique Andrade, Ranjita Bhagwan, Frank Eskesen, Richard King, Philippe Selo, Yoonho Park, and Chitra Venkatramani. "SPC: A Distributed, Scalable Platform for Data Mining". In: *DMSSP '06: Proceedings of the 4th international workshop on Data mining standards, services and platforms*. Philadelphia, Pennsylvania: ACM, 2006, pp. 27–37. ISBN: 1-59593-443-X. DOI: 10.1145/1289612.1289615 (cit. on p. 48).

[ABB+04] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. *STREAM: The Stanford Data Stream Management System*. Techreport (Technical Report) 641. Department of Computer Science, Stanford University, Mar. 2004. URL: http://ilpubs.stanford.edu:8090/641/1/2004-20.pdf (cit. on p. 49).

[ABJ+04] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. "Kepler: an extensible system for design and execution of scientific workflows". In: *16th International Conference on Scientific and Statistical Database Management*. 2004, pp. 423–424. DOI: 10.1109/SSDM.2004.1311241 (cit. on pp. 17, 40).

[ACÇ+03] Daniel J. Abadi, Don Carney, Uğur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. "Aurora: a new model and architecture for data stream management". In: *The VLDB Journal* 12.2 (2003), pp. 120–139. ISSN: 1066-8888. DOI: 10.1007/s00778-003-0095-z (cit. on pp. 16, 35).

[AEM13] Omar Asad, Melike Erol-Kantarci, and Hussein T. Mouftah. "A Survey of Sensor Web Services for the Smart Grid". In: *Journal of Sensor and Actuator Networks* 2.1 (2013), pp. 98–108. ISSN: 2224-2708. DOI: 10.3390/jsan2010098 (cit. on pp. 18, 42).

[AEVW04] J.M. Almeida, D.L. Eager, M.K. Vernon, and S.J. Wright. "Minimizing delivery cost in scalable streaming content distribution systems". In: *Multimedia, IEEE Transactions on* 6.2 (Apr. 2004), pp. 356–365. ISSN: 1520-9210. DOI: 10.1109/TMM.2003.822796 (cit. on p. 46).

[AHS06] K. Aberer, M. Hauswirth, and A. Salehi. *The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks*. Tech. rep. 006. EFPL, 2006. URL: http://lsirpeople.epfl.ch/salehi/papers/LSIR-REPORT-2006-006.pdf (cit. on pp. 16, 24, 35, 105, 137).

[AIG12] Mohamed Abouelhoda, Shadi Issa, and Moustafa Ghanem. "Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support". In: *BMC Bioinformatics* 13.1 (2012), p. 77. ISSN: 1471-2105. DOI: 10.1186/1471-2105-13-77 (cit. on pp. 17, 41).

[AJS+06] L. Amini, N. Jain, A. Sehgal, J. Silber, and O. Verscheure. "Adaptive Control of Extreme-scale Stream Processing Systems". In: *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. 2006, pp. 71–71. DOI: 10.1109/ICDCS.2006.13 (cit. on pp. 18, 43, 48).

[AN08] Y. Ahmad and S. Nath. "COLR-Tree: Communication-Efficient Spatio-Temporal Indexing for a Sensor Data Web Portal". In: *ICDE '08: Proceedings of the 24th International Conference on Data Engineering*. Apr. 2008, pp. 784–793. DOI: 10.1109/ICDE.2008.4497487 (cit. on pp. 53, 141).

[ARE+05] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, Hongwei Zhang, Hui Cao, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, M. Nesterenko, R. Shah, S. Kulkami, M. Aramugam, Limin Wang, M. Gouda, Young-ri Choi, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker. "ExScal: Elements of an Extreme Scale Wireless Sensor Network". In: *RTCSA '05: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Aug. 2005, pp. 102–108. DOI: 10.1109/RTCSA.2005.47 (cit. on pp. 16, 35).

[ASSC02a] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". In: *Computer Networks* 38.4 (2002), pp. 393–422. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(01)00302-4 (cit. on pp. 15, 18, 33, 49).

[ASSC02b] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. "A survey on Sensor Networks". In: *IEEE Communications Magazine* 40.8 (Aug. 2002), pp. 102–114. ISSN: 0163-6804. DOI: 10.1109/MCOM.2002.1024422 (cit. on pp. 18, 49).

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. "Models and issues in data stream systems". In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '02. Madison, Wisconsin: ACM, 2002, pp. 1–16. ISBN: 1-58113-507-6. DOI: 10.1145/543613.543615 (cit. on p. 43).

[BBK02] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. "Scalable Application Layer Multicast". In: *SIGCOMM '02: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Pittsburgh, Pennsylvania, USA: ACM, 2002, pp. 205–217. ISBN: 1-58113-570-X. DOI: 10.1145/633025.633045 (cit. on pp. 30, 45, 170).

[BCEL07] T Bektas, J-F Cordeau, E Erkut, and G Laporte. "A two-level simulated annealing algorithm for efficient dissemination of electronic content". In: *Journal of the Operational Research Society* 59.11 (Sept. 2007), pp. 1557–1567. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2602490 (cit. on p. 45).

[BCEL08] Tolga Bektaş, Jean-François Cordeau, Erhan Erkut, and Gilbert Laporte. "Exact algorithms for the joint object placement and request routing problem in content distribution networks. Part Special Issue: Telecommunications Network Engineering". In: *Computers & Operations Research* 35 (12 2008), pp. 3860–3884. ISSN: 0305-0548. DOI: 10.1016/j.cor.2007.02.005 (cit. on p. 45).

[BCM+99] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman. "An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems". In: *ICDCS '99: Proceedingsof the 19th IEEE Interna-*

tional Conference on Distributed Computing Systems. 1999, pp. 262–272. DOI: 10.1109/ICDCS.1999.776528 (cit. on p. 45).

[BCRG14] Mike Botts, Simon Cox, Alex Robin, and John Greybeal. *OGC® SensorML: Model and XML Encoding Standard*. OGC® Publicly Available Standard 12-000. Open Geospatial Consortium, Feb. 2014. URL: https://portal.opengeospatial.org/files/?artifact_id=55939 (cit. on pp. 50, 63).

[BEJ+11] Arne Bröring, Johannes Echterhoff, Simon Jirka, Ingo Simonis, Thomas Everding, Christoph Stasch, Steve Liang, and Rob Lemmens. "New Generation Sensor Web Enablement". In: *Sensors* 11.3 (2011), pp. 2652–2699. ISSN: 1424-8220. DOI: 10.3390/s110302652 (cit. on p. 42).

[BFG07] Silvia Bianchi, Pascal Felber, and Maria Gradinariu. "Euro-Par '07". In: Springer, 2007. Chap. Content-Based Publish/Subscribe Using Distributed R-Trees, pp. 537–548. DOI: 10.1007/978-3-540-74466-5_57 (cit. on pp. 24, 30, 106, 171).

[BGJ09] Jehoshua Bruck, Jie Gao, and Anxiao (Andrew) Jiang. "Localization and Routing in Sensor Networks by Local Angle Information". In: *ACM Trans. Sen. Netw.* 5.1 (2009), pp. 1–31. ISSN: 1550-4859. DOI: 10.1145/1464420.1464427 (cit. on p. 51).

[BH08] Adam Barker and Jano Hemert. "Scientific Workflow: A Survey and Research Directions". In: *Parallel Processing and Applied Mathematics*. Ed. by Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Wasniewski. Vol. 4967. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 746–753. ISBN: 978-3-540-68105-2. DOI: 10.1007/978-3-540-68111-3_78 (cit. on pp. 17, 40).

[BHKR09] Andreas Benzing, Klaus Herrmann, Boris Koldehofe, and Kurt Rothermel. "Identifying the Challenges in Reducing Latency in GSN using Predictors". In: *WowKiVS 2009: Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*. Ed. by Tiziana Margaria, Julia Padberg, and Gabriele Taentzer. Vol. 17. Electronic Communications of the EASST. Kassel, Germany: EASST, Mar. 2009. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2009-04 (cit. on pp. 37, 107).

[BJR08] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Ed. by David J. Balding, Noel A. C. Cressie, Garrett M. Fitzmaurice, Iain M. Johnstone, Geert Molenberghs, David W. Scott, Adrian F. M. Smith, Ruey S. Tsay, Sanford Weisberg, Vic Barnett, J. Stuart Hunter, and Jozef L. Teugels. 4th. John Wiley & Sons, Inc., 2008, p. 746. ISBN: 0816211043 (cit. on pp. 84, 86).

[BKC+01] Michael D. Beynon, Tahsin Kurc, Umit Catalyurek, Chialin Chang, Alan Sussman, and Joel Saltz. "Distributed processing of very large datasets with DataCutter". In: *Parallel Computing* 27.11 (2001), pp. 1457–1478. ISSN: 0167-8191. DOI: 10.1016/S0167-8191(01)00099-0 (cit. on pp. 27, 138).

[BKR09] Jorge A. Briones, Boris Koldehofe, and Kurt Rothermel. "SPINE : Adaptive Publish/Subscribe for Wireless Mesh Networks". Englisch. In: *Studia Informatika Universalis* 7.3 (Oct. 2009), pp. 320–353. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=ART-2009-15 (cit. on pp. 24, 27, 106, 138).

[BKR11] Andreas Benzing, Boris Koldehofe, and Kurt Rothermel. "Efficient Support for Multi-Resolution Queries in Global Sensor Networks". In: *COMSWARE 2011: Proceedings of the 5th International Conference on Communication System Software and Middleware*. COMSWARE '11. Verona, Italy: ACM, 2011, 11:1–11:12. ISBN: 978-1-4503-0560-0. DOI: 10.1145/2016551.2016562 (cit. on pp. 37, 109).

[BKR14] Andreas Benzing, Boris Koldehofe, and Kurt Rothermel. "Bandwidth-Minimized Distribution of Measurements in Global Sensor Networks". English. In: *Distributed Applications and Interoperable Systems*. Ed. by Kostas Magoutis and Peter Pietzuch. Vol. 8460. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 156–170. ISBN: 978-3-662-43351-5. DOI: 10.1007/978-3-662-43352-2_13 (cit. on pp. 37, 141).

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". In: *SIGMOD Rec.* 19.2 (1990), pp. 322–331. ISSN: 0163-5808. DOI: 10.1145/93605.98741 (cit. on p. 52).

[BKVR10] Andreas Benzing, Boris Koldehofe, Marco Völz, and Kurt Rothermel. "Multilevel Predictions for the Aggregation of Data in Global Sensor Networks". In: *DS-RT '10: Proceedings of the 14th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. IEEE, Oct. 2010, pp. 169–178. DOI: 10.1109/DS-RT.2010.26 (cit. on pp. 37, 75).

[BOO07] Tolga Bektas, Osman Oguz, and Iradj Ouveysi. "Designing cost-effective content distribution networks". In: *Computers & Operations Research* 34.8 (2007), pp. 2436–2449. ISSN: 0305-0548. DOI: 10.1016/j.cor.2005.09.013 (cit. on p. 45).

[BRT00] A. Boukerche, A. Roy, and N. Thomas. "Dynamic grid-based multicast group assignment in data distribution management". In: *DS-RT '00: Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*. 2000, pp. 47–54. DOI: 10.1109/DISRTA.2000.874063 (cit. on pp. 30, 171).

[BSB07] Yann-Aël Le Borgne, Silvia Santini, and Gianluca Bontempi. "Adaptive model selection for time series prediction in wireless sensor networks". In: *Signal Processing* 87.12 (2007). Special Section: Information Processing and Data Management in Wireless Sensor Networks, pp. 3010–3020. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2007.05.015 (cit. on p. 55).

[CAWK08] Emrah Ceyhan, Gabrielle Allen, Christopher White, and Tevfik Kosar. "A grid-enabled Workflow System for Reservoir Uncertainty Analysis". In: *CLADE '08: Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*. Boston, MA, USA: ACM, 2008, pp. 45–52. ISBN: 978-1-60558-156-9. DOI: 10.1145/1383529.1383537 (cit. on pp. 17, 18, 41, 42).

[CBB+03] Mitch Cherniack, Hari Balakrishnan, Magdalena Balazinska, Don Carney, Uğur Çetintemel, Ying Xing, and Stan Zdonik. "Scalable Distributed Stream Processing". In: *CIDR '03: Proceedings of the 1st Biennal Conference on Innovative Data Systems Research*. 2003. URL: http://www.cidrdb.org/cidr2003/program/p23.pdf (cit. on pp. 18, 42).

[CBD+12] B.W. Carabelli, A. Benzing, F. Dürr, B. Koldehofe, K. Rothermel, G. Seyboth, R. Blind, M. Bürger, and F. Allgöwer. "Exact convex formulations of network-oriented optimal operator placement". In: *CDC 2012: IEEE 51st*

Annual Conference on Decision and Control. Dec. 2012, pp. 3777–3782. DOI: 10.1109/CDC.2012.6426790 (cit. on pp. 44, 48, 59).

[CCD+03] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et al. "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World". In: *CIDR '03: Proceedings of the 1st Biennal Conference on Innovative Data Systems Research*. 2003. URL: http://www-db.cs.wisc.edu/cidr/cidr2003/program/p24.pdf (cit. on pp. 18, 43).

[CDK+03a] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. "SplitStream: high-bandwidth multicast in cooperative environments". In: *SIGOPS Oper. Syst. Rev.* 37 (5 Oct. 2003), pp. 298–313. ISSN: 0163-5980. DOI: 10.1145/1165389.945474 (cit. on pp. 27, 45, 138).

[CDK+03b] Ann Chervenak, Ewa Deelman, Carl Kesselman, Bill Allcock, Ian Foster, Veronika Nefedova, Jason Lee, Alex Sim, Arie Shoshani, Bob Drach, Dean Williams, and Don Middleton. "High-performance remote access to climate simulation data: a challenge problem for data grid technologies". In: *Parallel Computing* 29.10 (2003), pp. 1335–1356. ISSN: 0167-8191. DOI: DOI:10.1016/j.parco.2003.06.001 (cit. on pp. 27, 138).

[CDKR02] M. Castro, P. Druschel, A.-M. Kermarrec, and A.I.T. Rowstron. "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure". In: *IEEE Journal on Selected Areas in Communications* 20.8 (Oct. 2002), pp. 1489–1499. ISSN: 0733-8716. DOI: 10.1109/JSAC.2002.803069 (cit. on pp. 27, 30, 45, 138, 170).

[CDTW00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. "NiagaraCQ: A Scalable Continuous Query System for Internet Databases". In: *SIGMOD Rec.* 29.2 (2000), pp. 379–390. ISSN: 0163-5808. DOI: 10.1145/335191.335432 (cit. on p. 43).

[CFD01] Lu Henrique M. K. Costa, Serge Fdida, and Otto Duarte. "Hop by hop multicast routing protocol". In: *SIGCOMM '01: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. San Diego, California, United States: ACM, 2001, pp. 249–259. ISBN: 1-58113-411-8. DOI: 10.1145/383059.383079 (cit. on p. 44).

[CFS+06] Steven P. Callahan, Juliana Freire, Emanuele Santos, Carlos E. Scheidegger, Cláudio T. Silva, and Huy T. Vo. "VisTrails: visualization meets data management". In: *SIGMOD '06: Proceedings of the ACM international conference on Management of data*. SIGMOD '06. Chicago, IL, USA: ACM, 2006, pp. 745–747. ISBN: 1-59593-434-0. DOI: 10.1145/1142473.1142574 (cit. on p. 41).

[CGH+06] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. "Programming scientific and distributed workflow with Triana services". In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), pp. 1021–1037. ISSN: 1532-0634. DOI: 10.1002/cpe.992 (cit. on p. 41).

[Cha03] Yatin Chawathe. "Scattercast: an adaptable broadcast distribution framework". In: *Multimedia Systems* 9 (1 2003), pp. 104–118. ISSN: 0942-4962. DOI: 10.1007/s00530-002-0082-z (cit. on pp. 43, 44).

[CHPP09] C.-H. Chen-Ritzo, C. Harrison, J. Paraszczak, and F. Parr. "Instrumenting the planet". In: *IBM Journal of Research and Development* 53.3 (2009), pp. 1–16. URL: http://www.research.ibm.com/journal/rd53-3.html (cit. on pp. 15, 34).

[CL83] C. Caraiscos and Bede Liu. "A round-off error analysis of the LMS adaptive algorithm". In: *ICASSP '83: IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 8. Apr. 1983, pp. 29–32. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1172165 (cit. on p. 85).

[CM09] Gianpaolo Cugola and Matteo Migliavacca. "A Context and Content-Based Routing Protocol for Mobile Sensor Networks". In: *Wireless Sensor Networks*. Ed. by Utz Roedig and Cormac Sreenan. Vol. 5432. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, pp. 69–85. ISBN: 978-3-642-00223-6. DOI: 10.1007/978-3-642-00224-3_5 (cit. on pp. 19, 51).

[CM10] Gianpaolo Cugola and Alessandro Margara. "SLIM: Service Location and Invocation Middleware for Mobile Wireless Sensor and Actuator Networks". en. In: *International Journal of Systems and Service-Oriented Engineering* 1.3 (2010), pp. 60–74. ISSN: 1947-3052. DOI: 10.4018/jssoe.2010070104 (cit. on p. 51).

[CM12] Gianpaolo Cugola and Alessandro Margara. "Processing flows of information: From data stream to complex event processing". In: *ACM Comput. Surv.* 44.3 (June 2012), 15:1–15:62. ISSN: 0360-0300. DOI: 10.1145/2187671.2187677 (cit. on pp. 18, 43).

[CPH06] Jose A. Costa, Neal Patwari, and Alfred O. Hero III. "Distributed weighted-multidimensional scaling for node localization in sensor networks". In: *ACM Trans. Sen. Netw.* 2.1 (Feb. 2006), pp. 39–64. ISSN: 1550-4859. DOI: 10.1145/1138127.1138129 (cit. on pp. 19, 51).

[CRB+03] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. "Making gnutella-like P2P systems scalable". In: *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '03. Karlsruhe, Germany: ACM, 2003, pp. 407–418. ISBN: 1-58113-735-4. DOI: 10.1145/863955.864000 (cit. on p. 44).

[CRSZ02] Yang-hua Chu, S.G. Rao, S. Seshan, and Hui Zhang. "A case for end system multicast". In: *IEEE Journal on Selected Areas in Communications* 20.8 (Oct. 2002), pp. 1456–1471. ISSN: 0733-8716. DOI: 10.1109/JSAC.2002.803066 (cit. on pp. 18, 30, 44, 170).

[CRW01] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. "Design and Evaluation of a Wide-Area Event Notification Service". In: *ACM Transactions on Computer Systems* 19.3 (2001), pp. 332–383. ISSN: 0734-2071. DOI: 10.1145/380749.380767 (cit. on pp. 24, 27, 106, 138).

[Dar04] Frederica Darema. "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements". In: *ICCS 2004: Computational Science*. Ed. by Marian Bubak, Geert van Albada, Peter Sloot, and Jack Dongarra. Vol. 3038. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, pp. 662–669. ISBN: 978-3-540-22116-6. DOI: 10.1007/978-3-540-24688-6_86 (cit. on pp. 34, 71).

[DAW+08] Mathilde Durvy, Julien Abeillé, Patrick Wetterwald, Colin O'Flynn, Blake Leverett, Eric Gnoske, Michael Vidales, Geoff Mulligan, Nicolas Tsiftes, Niclas Finne, and Adam Dunkels. "Making Sensor Networks IPv6 Ready". In: *SenSys '08: Proceedings of the 6th International Conference on Embedded Networked Sensor Systems, poster session*. Raleigh, North Carolina, USA,

Nov. 2008. URL: http://www.sics.se/~adam/durvy08making.pdf (cit. on pp. 15, 34).

[DBG+04] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. "Pegasus: Mapping Scientific Workflows onto the Grid". In: *Grid Computing*. Ed. by MariosD. Dikaiakos. Vol. 3165. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 11–20. ISBN: 978-3-540-22888-2. DOI: 10.1007/978-3-540-28642-4_2 (cit. on pp. 17, 41).

[DF03] M. Demirbas and H. Ferhatosmanoglu. "Peer-to-peer spatial queries in sensor networks". In: *P2P 2003: Third International Conference on Peer-to-Peer Computing*. Sept. 2003, pp. 32–39. DOI: 10.1109/PTP.2003.1231501 (cit. on pp. 19, 24, 52, 106).

[DGL+05] Peter Desnoyers, Deepak Ganesan, Huan Li, Ming Li, and Prashant Shenoy. "PRESTO: A Predictive Storage Architecture for Sensor Networks". In: *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Santa Fe, NM: USENIX Association, 2005, pp. 23–23. URL: http://portal.acm.org/citation.cfm?id=1251146 (cit. on pp. 19, 24, 55, 107).

[DGM+04] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. "Model-Driven Data Acquisition in Sensor Networks". In: *VLDB '04: Proceedings of the 30th international conference on Very large data bases*. Toronto, Canada: VLDB Endowment, 2004, pp. 588–599. ISBN: 0-12-088469-0. URL: http://portal.acm.org/citation.cfm?id=1316741 (cit. on pp. 19, 24, 54, 107).

[DGMS07] Yanlei Diao, Deepak Ganesan, Gaurav Mathur, and Prashant Shenoy. "Rethinking Data Management for Storage-centric Sensor Networks". In: *CIDR '07: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*. Asilomar, California, Jan. 2007. URL: http://www.cs.umass.edu/~dganesan/papers/CIDR07-StonesDB.pdf (cit. on pp. 15, 19, 34, 52).

[DGS05] Peter Desnoyers, Deepak Ganesan, and Prashant Shenoy. "TSAR: A Two Tier Sensor Storage Architecture Using Interval Skip Graphs". In: *SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked*

Sensor Systems. San Diego, California, USA: ACM, 2005, pp. 39–50. ISBN: 1-59593-054-X. DOI: 10.1145/1098918.1098923 (cit. on p. 52).

[DGV04] A. Dunkels, B. Gronvall, and T. Voigt. "Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors". In: *LCN 2004: 29th Annual International Conference on Local Computer Networks*. Nov. 2004, pp. 455–462. DOI: 10.1109/LCN.2004.38 (cit. on pp. 15, 19, 34, 50).

[DLB+06] C.C. Douglas, R.A. Loader, J.D. Beezley, J. Mandel, R.E. Ewing, Y. Efendiev, Guan Qin, M. Iskandarani, J. Coen, A. Vodacek, M. Kritz, and G. Haase. "DDDAS Approaches to Wildland Fire Modeling and Contaminant Tracking". In: *WSC 06: Proceedings of the Winter Simulation Conference*. Dec. 2006, pp. 2117–2124. DOI: 10.1109/WSC.2006.323011 (cit. on p. 71).

[DLL+00] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. "Deployment issues for the IP multicast service and architecture". In: *Network, IEEE* 14.1 (2000), pp. 78–88. ISSN: 0890-8044. DOI: 10.1109/65.819174 (cit. on pp. 18, 44).

[DRAT11] Michael Duller, Jan Rellermeyer, Gustavo Alonso, and Nesime Tatbul. "Virtualizing Stream Processing". In: *Middleware 2011*. Ed. by Fabio Kon and Anne-Marie Kermarrec. Vol. 7049. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, pp. 269–288. ISBN: 978-3-642-25820-6. DOI: 10.1007/978-3-642-25821-3_14 (cit. on p. 49).

[DSS+05] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. "Pegasus: A framework for mapping complex scientific workflows onto distributed systems". In: *Scientific Programming* 13.3 (Jan. 2005), pp. 219–237. URL: http://iospress.metapress.com/content/84H5Q70AWX6FAU0W (cit. on pp. 17, 41).

[DSU09] Lúcia M. A. Drummond, Marcelo Santos, and Eduardo Uchoa. "A distributed dual ascent algorithm for Steiner problems in multicast routing". In: *Networks* 53.2 (2009), pp. 170–183. ISSN: 1097-0037. DOI: 10.1002/net.20276 (cit. on p. 44).

[DW71] S. E. Dreyfus and R. A. Wagner. "The steiner problem in graphs". In: *Networks* 1.3 (1971), pp. 195–207. ISSN: 1097-0037. DOI: 10.1002/net.3230010302 (cit. on pp. 28, 44, 146).

[DWD] Deutscher Wetterdienst. *Homepage*. URL: `http://www.dwd.de` (visited on 03/17/2014) (cit. on pp. 77, 95).

[EFGK03] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. "The Many Faces of Publish/Subscribe". In: *ACM Comput. Surv.* 35.2 (2003), pp. 114–131. ISSN: 0360-0300. DOI: `10.1145/857076.857078` (cit. on pp. 24, 27, 106, 138, 142, 171).

[EM81] H. Edelsbrunner and H.A. Maurer. "On the intersection of Orthogonal objects". In: *Information Proc. Letters* 13.4, 5 (1981), pp. 177–181. ISSN: 0020-0190. DOI: `10.1016/0020-0190(81)90053-3` (cit. on p. 146).

[FHW+08] G. Fox, A. Ho, Rui Wang, E. Chu, and Isaac Kwan. "A Collaborative Sensor Grids Framework". In: *CTS 2008: International Symposium on Collaborative Technologies and Systems*. May 2008, pp. 29–38. DOI: `10.1109/CTS.2008.4543909` (cit. on pp. 16, 35).

[FJK+05] M.J. Franklin, S.R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong. "Design Considerations for High Fan-in Systems: The HiFi Approach". In: *CIDR '05: Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research*. Asilomar, California, Jan. 2005. URL: `http://www.eecs.harvard.edu/~mdw/course/cs260r/papers/hifi-cidr05.pdf` (cit. on pp. 24, 27, 76, 106, 137).

[FPS+12] Benjamin Frank, Ingmar Poese, Georgios Smaragdakis, Steve Uhlig, and Anja Feldmann. "Content-aware traffic engineering". In: *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '12. London, England, UK: ACM, 2012, pp. 413–414. ISBN: 978-1-4503-1097-0. DOI: `10.1145/2254756.2254819` (cit. on p. 46).

[GDE+07] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. "Examining the Challenges of Scientific Workflows". In: *Computer* 40.12 (Dec. 2007), pp. 24–32. ISSN: 0018-9162. DOI: `10.1109/MC.2007.421` (cit. on pp. 17, 40).

[GEH03] Deepak Ganesan, Deborah Estrin, and John Heidemann. "DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks?" In: *SIGCOMM Comput. Commun. Rev.* 33.1 (2003), pp. 143–148. ISSN: 0146-4833. DOI: `10.1145/774763.774786` (cit. on pp. 19, 52).

[GG98] Volker Gaede and Oliver Günther. "Multidimensional access methods". In: *ACM Comput. Surv.* 30.2 (1998), pp. 170–231. ISSN: 0360-0300. DOI: 10.1145/280277.280279 (cit. on p. 112).

[GGE+05] Deepak Ganesan, Ben Greenstein, Deborah Estrin, John Heidemann, and Ramesh Govindan. "Multiresolution Storage and Search in Sensor Networks". In: *Trans. Storage* 1.3 (2005), pp. 277–315. ISSN: 1553-3077. DOI: 10.1145/1084779.1084780 (cit. on pp. 16, 35, 53).

[GGP+03] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. "An Evaluation of Multi-resolution Storage for Sensor Networks". In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. Los Angeles, California, USA: ACM, 2003, pp. 89–102. ISBN: 1-58113-707-9. DOI: 10.1145/958491.958502 (cit. on p. 53).

[GI01] Samir Goel and Tomasz Imielinski. "Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG". In: *SIGCOMM Comput. Commun. Rev.* 31.5 (2001), pp. 82–98. ISSN: 0146-4833. DOI: 10.1145/1037107.1037117 (cit. on pp. 19, 24, 54, 76).

[GJP+06] Omprakash Gnawali, Ki-Young Jang, Jeongyeup Paek, Marcos Vieira, Ramesh Govindan, Ben Greenstein, August Joki, Deborah Estrin, and Eddie Kohler. "The Tenet Architecture for Tiered Sensor Networks". In: *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. Boulder, Colorado, USA: ACM, 2006, pp. 153–166. ISBN: 1-59593-343-3. DOI: 10.1145/1182807.1182823 (cit. on pp. 19, 52).

[GKK+03] P.B. Gibbons, B. Karp, Y. Ke, S. Nath, and Srinivasan Seshan. "IrisNet: An Architecture for a Worldwide Sensor Web". In: *IEEE Pervasive Computing* 2.4 (2003). Ed. by B. Karp, pp. 22–33. ISSN: 1536-1268. DOI: 10.1109/MPRV.2003.1251166 (cit. on pp. 16, 24, 27, 30, 35, 76, 106, 137, 172).

[GKN+07] W.I. Grosky, A. Kansal, S. Nath, Jie Liu, and Feng Zhao. "SenseWeb: An Infrastructure for Shared Sensing". In: *Multimedia, IEEE* 14.4 (Oct. 2007), pp. 8–13. ISSN: 1070-986X. DOI: 10.1109/MMUL.2007.82 (cit. on pp. 19, 53).

[GMW+04] Mark Gaynor, Steven L. Moulton, Matt Welsh, Ed LaCombe, Austin Rowan, and John Wynne. "Integrating Wireless Sensor Networks with the Grid". In: *IEEE Internet Computing* 8.4 (2004), pp. 32–39. ISSN: 1089-7801. DOI: 10.1109/MIC.2004.18 (cit. on pp. 18, 42).

[GN12] Jim Gettys and Kathleen Nichols. "Bufferbloat: Dark Buffers in the Internet". In: *Commun. ACM* 55.1 (Jan. 2012), pp. 57–65. ISSN: 0001-0782. DOI: 10.1145/2063176.2063196 (cit. on p. 145).

[GQP06] O. Garcia, A. Quintero, and S. Pierre. "Profile-based energy minimisation strategy for Object Tracking Wireless Sensor Networks". In: *WiMob 2006: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. June 2006, pp. 372–379. DOI: 10.1109/WIMOB.2006.1696370 (cit. on p. 55).

[GS08] Thomas Gamer and Michael Scharf. "Realistic simulation environments for IP-based networks". In: *Simutools '08*. Marseille, France: ICST, 2008, 83:1–83:7. ISBN: 978-963-9799-20-2 (cit. on pp. 143, 165).

[GSK+11] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. "Conventional Workflow Technology for Scientific Simulation". English. In: *Guide to e-Science*. Ed. by Xiaoyu Yang, Lizhe Wang, and Wei Jie. Computer Communications and Networks. Springer London, 2011, pp. 323–352. ISBN: 978-0-85729-438-8. DOI: 10.1007/978-0-85729-439-5_12 (cit. on pp. 17, 40).

[Gut84] Antonin Guttman. "R-trees: a dynamic index structure for spatial searching". In: *SIGMOD '84: Proceedings of the ACM international conference on Management of data*. Boston, Massachusetts: ACM, 1984, pp. 47–57. ISBN: 0-89791-128-8. DOI: 10.1145/602259.602266 (cit. on pp. 22, 52, 78, 106).

[Har11] Peter Hartwell. "CeNSE: A central nervous system for the earth". In: *TTM 2011: IEEE Technology Time Machine Symposium on Technologies Beyond 2020*. 2011, pp. 1–1. DOI: 10.1109/TTM.2011.6005162 (cit. on p. 141).

[HASG07] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas. "A Survey of Application-Layer Multicast Protocols". In: *Communications Surveys & Tutorials, IEEE* 9.3 (Mar. 2007), pp. 58–74. ISSN: 1553-877X. DOI: 10.1109/COMST.2007.4317616 (cit. on pp. 18, 30, 44, 170).

[HFSK06] Michael P. Hunter, Richard M. Fujimoto, Wonho Suh, and Hoe Kyoung Kim. "An investigation of real-time dynamic data driven transportation simulation". In: *WSC '06: Proceedings of the 38th conference on Winter simulation*. Monterey, California: Winter Simulation Conference, 2006, pp. 1414–1421. ISBN: 1-4244-0501-7. URL: http://portal.acm.org/citation.cfm?id=1218112.1218369 (cit. on p. 71).

[HHB+03] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. "Range-free localization schemes for large scale sensor networks". In: *Proceedings of the 9th annual international conference on Mobile computing and networking*. MobiCom '03. San Diego, CA, USA: ACM, 2003, pp. 81–95. ISBN: 1-58113-753-2. DOI: 10.1145/938985.938995 (cit. on pp. 19, 51).

[Hu11] Xiaolin Hu. "Dynamic Data Driven Simulation". In: *SCS M&S Magazine* 2.1 (Jan. 2011), pp. 16–22. URL: http://www.scs.org/magazines/2011-01/index_file/Articles.htm (cit. on p. 71).

[HW05] Simon Haykin and Bernard Widrow, eds. *Least-Mean-Square Adaptive Filters*. John Wiley & Sons, Inc., 2005. DOI: 10.1002/0471461288 (cit. on p. 85).

[IBU+11] Kazushi Ikeda, Thilmee M. Baduge, Takaaki Umedu, Hirozumi Yamaguchi, and Teruo Higashino. "ALMware: A middleware for application layer multicast protocols". In: *Computer Communications* 34.14 (2011), pp. 1673–1684. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2011.03.004 (cit. on p. 45).

[IPS+09] Tudor B. Ionescu, Andreas Piater, Walter Scheuermann, Eckart Laurien, and Alexandru Iosup. "An Aspect-Oriented Approach for Disaster Prevention Simulation Workflows on Supercomputers, Clusters, and Grids". In: *DS-RT '09: Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 21–30. ISBN: 978-0-7695-3868-6. DOI: 10.1109/DS-RT.2009.35 (cit. on pp. 15, 34).

[IS09a] K. Iwanicki and M. van Steen. "On Hierarchical Routing in Wireless Sensor Networks". In: *IPSN '09: Proceedings of the 8th ACM/IEE international conference on Information Processing in Sensor Networks*. 2009, pp. 133–144. URL: http://www.few.vu.nl/~iwanicki/ (cit. on pp. 19, 53).

[IS09b] K. Iwanicki and M. van Steen. "Using Area Hierarchy for Multi-Resolution Storage and Search in Large Wireless Sensor Networks". In: *ICC '09: IEEE International Conference on Communications*. June 2009, pp. 1–6. DOI: 10.1109/ICC.2009.5199556 (cit. on pp. 16, 35, 53, 142).

[JCW04] Ankur Jain, Edward Y. Chang, and Yuan-Fang Wang. "Adaptive Stream Resource Management Using Kalman Filters". In: *SIGMOD '04: Proceedings of the ACM SIGMOD international conference on Management of data*. Paris, France: ACM, 2004, pp. 11–22. ISBN: 1-58113-859-8. DOI: 10.1145/1007568.1007573 (cit. on pp. 24, 107).

[JE07] Diane Jordan and John Evdemon. *Web Services Business Process Execution Language Version 2.0*. English. OASIS Web Services Business Process Execution Language (WSBPEL) TC, Apr. 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (cit. on p. 40).

[JGJ+00] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole Jr. "Overcast: Reliable Multicasting with an Overlay Network". In: *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*. San Diego, California: USENIX Association, 2000, pp. 14–14. URL: http://portal.acm.org/citation.cfm?id=1251243 (cit. on p. 45).

[JJE10] K. Jayaram, Chamikara Jayalath, and Patrick Eugster. "Parametric Subscriptions for Content-Based Publish/Subscribe Networks". In: *Middleware 2010*. Ed. by Indranil Gupta and Cecilia Mascolo. Vol. 6452. LNCS. Springer, 2010, pp. 128–147. ISBN: 978-3-642-16954-0. DOI: 10.1007/978-3-642-16955-7_7 (cit. on pp. 30, 171).

[JMJV] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. *The Peersim Simulator*. URL: http://peersim.sf.net (cit. on pp. 95, 130).

[JPD03] Manish Jain, Ravi S. Prasad, and Constantinos Dovrolis. *The TCP Bandwidth-Delay Product revisited: network buffering, cross traffic, and socket buffer auto-sizing*. Technical Report CERCS;GIT-CERCS-03-02. Georgia Institute of Technology, 2003. URL: http://hdl.handle.net/1853/5920 (cit. on p. 145).

[KCS05] Vibhore Kumar, B.F. Cooper, and K. Schwan. "Distributed Stream Management using Utility-Driven Self-Adaptive Middleware". In: *ICAC 2005: Second International Conference on Autonomic Computing*. 2005, pp. 3–14. DOI: 10.1109/ICAC.2005.24 (cit. on p. 47).

[KDHS10] Dimitrios Koutsonikolas, Saumitra M. Das, Y. Charlie Hu, and Ivan Stojmenovic. "Hierarchical geographic multicast routing for wireless sensor networks". In: *Wirel. Netw.* 16.2 (2010), pp. 449–466. ISSN: 1022-0038. DOI: 10.1007/s11276-008-0146-x (cit. on pp. 19, 51).

[KF02] Minseok Kwon and Sonia Fahmy. "Topology-aware overlay networks for group communication". In: *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. NOSSDAV '02. Miami, Florida, USA: ACM, 2002, pp. 127–136. ISBN: 1-58113-512-2. DOI: 10.1145/507670.507688 (cit. on p. 45).

[KHR02] Dina Katabi, Mark Handley, and Charlie Rohrs. "Congestion control for high bandwidth-delay product networks". In: *SIGCOMM Comput. Commun. Rev.* 32.4 (Aug. 2002), pp. 89–102. ISSN: 0146-4833. DOI: 10.1145/964725.633035 (cit. on p. 145).

[KK00] Brad Karp and H. T. Kung. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks". In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. Boston, Massachusetts, United States: ACM, 2000, pp. 243–254. ISBN: 1-58113-197-6. DOI: 10.1145/345910.345953 (cit. on pp. 19, 51).

[KS10] Jinu Kurian and Kamil Sarac. "A survey on the design, applications, and enhancements of application-layer overlay networks". In: *ACM Comput. Surv.* 43.1 (Dec. 2010), 5:1–5:34. ISSN: 0360-0300. DOI: 10.1145/1824795.1824800 (cit. on pp. 18, 30, 44, 170).

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. "Scientific workflow management and the Kepler system". In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), pp. 1039–1065. ISSN: 1532-0634. DOI: 10.1002/cpe.994 (cit. on p. 42).

[LB98] R. Lamprecht and D. Berlowitz. "Evaluation of diagnostic and prognostic flow fields over prealpine complex terrain by comparison of the Lagrangian prediction of concentrations with tracer measurements". In: *Atmospheric Environment* 32.7 (1998), pp. 1283–1300. ISSN: 1352-2310. DOI: 10.1016/S1352-2310(97)00189-1 (cit. on pp. 71, 110).

[LCLC04] Kam-Yiu Lam, Reynold Cheng, BiYu Liang, and Jo Chau. "Sensor Node Selection for Execution of Continuous Probabilistic Queries in Wireless Sensor Networks". In: *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*. New York, NY, USA: ACM, 2004, pp. 63–71. ISBN: 1-58113-934-9. DOI: 10.1145/1026799.1026811 (cit. on pp. 19, 54).

[LGS06] Ming Li, Deepak Ganesan, and Prashant Shenoy. "PRESTO: Feedback-driven Data Management in Sensor Networks". In: *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*. San Jose, CA: USENIX Association, 2006, pp. 23–23. URL: http://www.cs.umass.edu/~dganesan/papers/NSDI06_PRESTO.pdf (cit. on pp. 19, 23, 24, 55, 85, 87, 107).

[LHRM09] Andreas Lachenmann, Klaus Herrmann, Kurt Rothermel, and Pedro José Marrón. "On meeting lifetime goals and providing constant application quality". In: *ACM Trans. Sen. Netw.* 5 (4 Nov. 2009), 36:1–36:36. ISSN: 1550-4859. DOI: 10.1145/1614379.1614388 (cit. on pp. 15, 34).

[Lin06] W. Lindner. "Operator Allocation in Borealis with Integrated Sensor Network Query Processors". In: *MDM '06: 7th International Conference on Mobile Data Management*. May 2006, pp. 155–155. DOI: 10.1109/MDM.2006.120 (cit. on p. 48).

[LKNZ08] Liqian Luo, Aman Kansal, Suman Nath, and Feng Zhao. "Sharing and Exploring Sensor Streams over Geocentric Interfaces". In: *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. Irvine, California: ACM, 2008, pp. 1–10. ISBN: 978-1-60558-323-5. DOI: 10.1145/1463434.1463439 (cit. on pp. 19, 54).

[LLG13] Guohua Li, Jianzhong Li, and Hong Gao. "ε-Approximation to data streams in sensor networks". In: *IEEE INFOCOM*. 2013, pp. 1663–1671. DOI: 10.1109/INFCOM.2013.6566963 (cit. on pp. 19, 55).

[LLL05] Bin Liu, Wang-Chien Lee, and Dik Lun Lee. "Supporting Complex Multi-Dimensional Queries in P2P Systems". In: *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*. June 2005, pp. 155–164. DOI: 10.1109/ICDCS.2005.75 (cit. on pp. 24, 106).

[LLS08] G.T. Lakshmanan, Ying Li, and R. Strom. "Placement Strategies for Internet-Scale Data Stream Systems". In: *Internet Computing, IEEE* 12.6 (Nov. 2008), pp. 50–60. ISSN: 1089-7801. DOI: 10.1109/MIC.2008.129 (cit. on p. 47).

[LM03] I. Lazaridis and S. Mehrotra. "Capturing Sensor-Generated Time Series with Quality Guarantees". In: *ICDE '03: Proceedings of the 19th International Conference on Data Engineering*. Mar. 2003, pp. 429–440. DOI: 10.1109/ICDE.2003.1260811 (cit. on p. 106).

[LMG+07] Andreas Lachenmann, Pedro José Marrón, Matthias Gauger, Daniel Minder, Olga Saukh, and Kurt Rothermel. "Removing the memory limitations of sensor networks with flash-based virtual memory". In: *SIGOPS Oper. Syst. Rev.* 41.3 (Mar. 2007), pp. 131–144. ISSN: 0163-5980. DOI: 10.1145/1272998.1273012 (cit. on pp. 19, 52).

[LMJM06] Guoli Li, Vinod Muthusamy, H.-Arno Jacobsen, and Serge Mankovski. "Decentralized Execution of Event-Driven Scientific Workflows". In: *SCW '06: IEEE Services Computing Workshops*. Sept. 2006, pp. 73–82. DOI: 10.1109/SCW.2006.10 (cit. on pp. 17, 41).

[LMMR07] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. "Meeting lifetime goals with energy levels". In: *Proceedings of the 5th international conference on Embedded networked sensor systems*. SenSys '07. Sydney, Australia: ACM, 2007, pp. 131–144. ISBN: 978-1-59593-763-6. DOI: 10.1145/1322263.1322277 (cit. on pp. 19, 50).

[LMP+05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. "Ambient Intelligence". In: Springer Berlin Heidelberg, 2005. Chap. TinyOS: An Operating System for Sensor Networks, pp. 115–148. DOI: 10.1007/3-540-27139-2_7 (cit. on pp. 19, 50).

[LR03] Koen Langendoen and Niels Reijers. "Distributed localization in wireless sensor networks: a quantitative comparison". In: *Computer Networks* 43.4 (2003). Wireless Sensor Networks, pp. 499–518. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(03)00356-6 (cit. on pp. 19, 51).

[LR99] Frank Leymann and Dieter Roller. *Production workflow: concepts and techniques*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. ISBN: 0-13-021753-0. URL: http://dl.acm.org/citation.cfm?id=317105 (cit. on pp. 17, 40).

[LZS04] Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. "Joint object placement and node dimensioning for Internet content distribution". In: *Information Processing Letters* 89.6 (2004), pp. 273–279. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2003.12.002 (cit. on pp. 18, 46).

[LZS05] Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. "On the optimization of storage capacity allocation for content distribution". In: *Computer Networks* 47.3 (2005), pp. 409–428. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2004.07.020 (cit. on p. 46).

[Mac95] Odile Macchi. *Adaptive processing: the least mean squares approach with applications in transmission*. Chichester [u.a.]: Wiley, 1995, XIX, 456 S. ISBN: 0-471-93403-8 (cit. on p. 85).

[Max] Maxmind. *GeoLite2*. URL: http://dev.maxmind.com/geoip/geoip2/geolite2/ (visited on 03/17/2014) (cit. on p. 155).

[MFHH03] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. "The Design of an Acquisitional Query Processor For Sensor Networks". In: *SIGMOD '03: Proceedings of the ACM SIGMOD international conference on Management of data*. San Diego, California: ACM, 2003, pp. 491–502. ISBN: 1-58113-634-X. DOI: 10.1145/872757.872817 (cit. on pp. 19, 50).

[MH00] George S. Moschytz and Markus Hofbauer. *Adaptive Filter*. Springer Berlin Heidelberg, 2000, p. 246. DOI: 10.1007/978-3-642-18250-1 (cit. on p. 86).

[MITLAB] Samuel Madden. *Intel Lab Data*. URL: http://db.lcs.mit.edu/labdata/labdata.html (visited on 03/17/2014) (cit. on pp. 77, 96).

[MLM+05] P.J. Marrón, A. Lachenmann, D. Minder, J. Hahner, R. Sauter, and K. Rothermel. "TinyCubus: a flexible and adaptive framework sensor networks". In: *Proceeedings of the Second European Workshop on Wireless Sensor Networks*. 2005, pp. 278–289. DOI: 10.1109/EWSN.2005.1462020 (cit. on pp. 19, 50).

[MRR06] Daniel Mölle, Stefan Richter, and Peter Rossmanith. "A Faster Algorithm for the Steiner Tree Problem". In: *STACS 2006*. Ed. by Bruno Durand and Wolfgang Thomas. Vol. 3884. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, pp. 561–570. ISBN: 978-3-540-32301-3. DOI: 10.1007/11672142_46 (cit. on p. 44).

[MSTW04] S. Majithia, M. Shields, I. Taylor, and I. Wang. "Triana: a graphical Web service composition and execution toolkit". In: *IEEE International Conference on Web Services*. 2004, pp. 514–521. DOI: 10.1109/ICWS.2004.1314777 (cit. on pp. 17, 40, 41).

[MYSJ11] Vinod Muthusamy, Young Yoon, Mohammad Sadoghi, and Hans-Arno Jacobsen. "eQoSystem: supporting fluid distributed service-oriented workflows". In: *Proceedings of the 5th ACM international conference on Distributed event-based system*. DEBS '11. New York, New York, USA: ACM, 2011, pp. 381–382. ISBN: 978-1-4503-0423-8. DOI: 10.1145/2002259.2002320 (cit. on p. 41).

[NLZ06] S. Nath, J. Liu, and F. Zhao. "Challenges in Building a Portal for Sensors World-Wide". In: *WSW 2006: Workshop on World-Sensor-Web*. 2006 (cit. on pp. 19, 53).

[NOAA] National Severe Storms Laboratory. *Historical Weather Data Archives, Norman, Oklahoma*. URL: http://data.nssl.noaa.gov/ (visited on 10/13/2010) (cit. on pp. 77, 95).

[Obr98] K. Obraczka. "Multicast transport protocols: a survey and taxonomy". In: *Communications Magazine, IEEE* 36.1 (1998), pp. 94–102. ISSN: 0163-6804. DOI: 10.1109/35.649333 (cit. on p. 44).

[OJW03] Chris Olston, Jing Jiang, and Jennifer Widom. "Adaptive filters for continuous queries over distributed data streams". In: *SIGMOD '03: Proceedings of the ACM SIGMOD international conference on Management of data*. San Diego, California: ACM, 2003, pp. 563–574. ISBN: 1-58113-634-X. DOI: 10.1145/872757.872825 (cit. on pp. 24, 107, 137).

[OP05] Carlos A.S. Oliveira and Panos M. Pardalos. "A survey of combinatorial optimization problems in multicast routing". In: *Computers & Operations Research* 32.8 (2005), pp. 1953–1981. ISSN: 0305-0548. DOI: 10.1016/j.cor.2003.12.007 (cit. on p. 44).

[OP11] Carlos A. S. Oliveira and Panos M. Pardalos. *Mathematical Aspects of Network Routing Optimization*. Vol. 53. Springer, 2011. DOI: 10.1007/978-1-4614-0311-1 (cit. on pp. 44, 146).

[OS90] Y. Ohsawa and M. Sakauchi. "A new tree type data structure with homogeneous nodes suitable for a very large spatial database". In: *ICDE '90: Proceedings of the Sixth International Conference on Data Engineering*. May 1990, pp. 296–303. DOI: 10.1109/ICDE.1990.113481 (cit. on pp. 25, 110, 112).

[PAK+05] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero, R.L. Moses, and N.S. Correal. "Locating the nodes: cooperative localization in wireless sensor networks". In: *Signal Processing Magazine, IEEE* 22.4 (2005), pp. 54–69. ISSN: 1053-5888. DOI: 10.1109/MSP.2005.1458287 (cit. on pp. 19, 51).

[Pas12] Andrea Passarella. "A survey on content-centric technologies for the current Internet: CDN and P2P solutions". In: *Computer Communications* 35.1 (2012), pp. 1–32. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2011.10.005 (cit. on p. 46).

[PÇJ09] Olga Papaemmanouil, Uğur Çetintemel, and John Jannotti. "Supporting Generic Cost Models for Wide-Area Stream Processing". In: *ICDE '09: Proceedings of the 25th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1084–1095. ISBN: 978-0-7695-3545-6. DOI: 10.1109/ICDE.2009.11 (cit. on p. 49).

[PFS+12] Ingmar Poese, Benjamin Frank, Georgios Smaragdakis, Steve Uhlig, Anja Feldmann, and Bruce Maggs. "Enabling content-aware traffic engineering". In: *SIGCOMM Comput. Commun. Rev.* 42.5 (Sept. 2012), pp. 21–28. ISSN: 0146-4833. DOI: 10.1145/2378956.2378960 (cit. on p. 46).

[PIS08] A. Piater, T. B. Ionescu, and W. Scheuermann. "A Distributed Simulation Framework for Mission Critical Systems in Nuclear Engineering and Radiological Protection". In: *Int. J. of Computers, Communications & Control* III (2008), pp. 448–453. URL: http://journal.univagora.ro/index.php?page=article_details&id=324 (cit. on p. 41).

[PK00] G. J. Pottie and W. J. Kaiser. "Wireless integrated network sensors". In: *Communications of the ACM* 43.5 (2000), pp. 51–58. ISSN: 0001-0782. DOI: 10.1145/332833.332838 (cit. on pp. 19, 50).

[PLS+06] Peter Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh, and Margo Seltzer. "Network-Aware Operator Placement for Stream-Processing Systems". In: *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*. IEEE, 2006, p. 49. ISBN: 0-7695-2570-9. DOI: 10.1109/ICDE.2006.105 (cit. on pp. 30, 48, 137, 172).

[PS02] Hans Jürgen Prömel and Angelika Steger. *The Steiner tree problem: a tour through graphs, algorithms, and complexity*. 1. ed. Vieweg, 2002, VIII, 241 S. ISBN: 3-528-06762-4 (cit. on p. 146).

[PS06] G. Pierre and M. van Steen. "Globule: a collaborative content delivery network". In: *Communications Magazine, IEEE* 44.8 (Aug. 2006), pp. 127–133. ISSN: 0163-6804. DOI: 10.1109/MCOM.2006.1678120 (cit. on p. 46).

[PWCS02] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. "Distributing streaming media content using cooperative networking". In: *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. Miami, Florida, USA: ACM, 2002, pp. 177–186. ISBN: 1-58113-512-2. DOI: 10.1145/507670.507695 (cit. on p. 45).

[RBDP07] C. Reed, M. Botts, J. Davidson, and G. Percivall. "OGC ® Sensor Web Enablement: Overview and High Level Achhitecture". In: *IEEE Autotestcon*. 2007, pp. 372–380. DOI: 10.1109/AUTEST.2007.4374243 (cit. on p. 42).

[RBKK12] M. Reiter, U. Breitenbucher, O. Kopp, and D. Karastoyanova. "Quality of data driven simulation workflows". In: *IEEE 8th International Conference on E-Science*. 2012, pp. 1–8. DOI: 10.1109/eScience.2012.6404417 (cit. on pp. 17, 41).

[RCM+12] U. Raza, A. Camerra, A.L. Murphy, T. Palpanas, and G.P. Picco. "What does model-driven data acquisition really achieve in wireless sensor networks?" In: *PerCom 2012: IEEE International Conference on Pervasive Computing and Communications*. Mar. 2012, pp. 85–94. DOI: 10.1109/PerCom.2012.6199853 (cit. on pp. 19, 24, 55, 107).

[RCM10] Roque Rodríguez, Ana Cortés, and Tomás Margalef. "Towards policies for data insertion in dynamic data driven application systems: a case study sudden changes in wildland fire". In: *Procedia Computer Science* 1.1 (2010). ICCS 2010, pp. 1267–1276. ISSN: 1877-0509. DOI: 10.1016/j.procs.2010.04.141 (cit. on p. 71).

[RDR10a] S. Rizou, F. Dürr, and K. Rothermel. "Providing QoS Guarantees in Large-Scale Operator Networks". In: *HPCC 2010: 12th IEEE International Conference on High Performance Computing and Communications*. Sept. 2010, pp. 337–345. DOI: 10.1109/HPCC.2010.53 (cit. on p. 48).

[RDR10b] S. Rizou, F. Dürr, and K. Rothermel. "Solving the Multi-Operator Placement Problem in Large-Scale Operator Networks". In: *ICCCN 2010: Proceedings of 19th International Conference on Computer Communications and Networks*. Aug. 2010, pp. 1–6. DOI: 10.1109/ICCCN.2010.5560127 (cit. on pp. 48, 137).

[RDR11] S. Rizou, F. Dürr, and K. Rothermel. "Fulfilling end-to-end latency constraints in large-scale streaming environments". In: *IPCCC 2011: IEEE 30th International Performance Computing and Communications Conference*. Nov. 2011, pp. 1–8. DOI: 10.1109/PCCC.2011.6108086 (cit. on pp. 30, 48, 145, 172).

[RFC1112] S.E. Deering. *Host extensions for IP multicasting*. RFC 1112 (INTERNET STANDARD). Updated by RFC 2236. Internet Engineering Task Force, Aug. 1989. URL: http://www.ietf.org/rfc/rfc1112.txt (cit. on p. 44).

[RFC2236] W. Fenner. *Internet Group Management Protocol, Version 2*. RFC 2236 (Proposed Standard). Updated by RFC 3376. Internet Engineering Task Force, Nov. 1997. URL: http://www.ietf.org/rfc/rfc2236.txt (cit. on p. 44).

[RFC3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. *Internet Group Management Protocol, Version 3*. RFC 3376 (Proposed Standard). Updated by RFC 4604. Internet Engineering Task Force, Oct. 2002. URL: http://www.ietf.org/rfc/rfc3376.txt (cit. on p. 44).

[RFC4604] H. Holbrook, B. Cain, and B. Haberman. *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*. RFC 4604 (Proposed

Standard). Internet Engineering Task Force, Aug. 2006. URL: http://www.ietf.org/rfc/rfc4604.txt (cit. on p. 44).

[RFC791] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc791.txt (cit. on p. 44).

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. "A Scalable Content-Addressable Network". In: *SIGCOMM '01: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. San Diego, California, United States: ACM, 2001, pp. 161–172. ISBN: 1-58113-411-8. DOI: 10.1145/383059.383072 (cit. on p. 44).

[RGE02] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. "Topology-informed Internet replica placement". In: *Computer Communications* 25.4 (2002), pp. 384–392. ISSN: 0140-3664. DOI: 10.1016/S0140-3664(01)00410-8 (cit. on pp. 18, 46).

[RHKS01] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. "Networked Group Communication". In: Springer Berlin / Heidelberg, 2001. Chap. Application-Level Multicast Using Content-Addressable Networks, pp. 14–29. DOI: 10.1007/3-540-45546-9_2 (cit. on p. 44).

[RK08] Thomas Repantis and Vana Kalogeraki. "Replica Placement for High Availability in Distributed Stream Processing Systems". In: *DEBS '08: Proceedings of the second international conference on Distributed event-based systems*. Rome, Italy: ACM, 2008, pp. 181–192. ISBN: 978-1-60558-090-6. DOI: 10.1145/1385989.1386012 (cit. on p. 49).

[RKY+02] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. "GHT: A Geographic Hash Table for Data-Centric Storage". In: *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. Atlanta, Georgia, USA: ACM, 2002, pp. 78–87. ISBN: 1-58113-589-0. DOI: 10.1145/570738.570750 (cit. on pp. 19, 52).

[RLSS02] Ananth Rao, Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. *Flexible and Robust Large Scale Multicast Using i3*. Tech. rep. UCB/CSD-02-1187. EECS Department, University of California, Berkeley, June 2002. URL:

http://www.eecs.berkeley.edu/Pubs/TechRpts/2002/6196.html (cit. on p. 44).

[RM99] S. Ratnasamy and S. McCanne. "Scaling end-to-end multicast transports with a topologically-sensitive group formation protocol". In: *ICNP '99: Seventh International Conference on Network Protocols*. Oct. 1999, pp. 79–88. DOI: 10.1109/ICNP.1999.801918 (cit. on p. 45).

[RRS+11] Peter Reimann, Michael Reiter, Holger Schwarz, Dimka Karastoyanova, and Frank Leymann. "SIMPL - A Framework for Accessing External Data in Simulation Workflows". Englisch. In: *BTW 2011: Datenbanksysteme für Business, Technologie und Web, 14. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS)*. Ed. by Gesellschaft für Informatik (GI). Vol. 180. Series of the Gesellschaft für Informatik (GI). Lecture Notes in Informatics (LNI), Mar. 2011, pp. 534–553. ISBN: 978-3-88579-274-1. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2011-07&engl= (cit. on p. 42).

[RSMF88] DG Ross, IN Smith, PC Manins, and DG Fox. "Diagnostic Wind Field Modeling for Complex Terrain: Model Development and Testing". In: *Journal of Applied Meteorology* 27.7 (1988), pp. 785–796. DOI: 10.1175/1520-0450(1988)027<0785:DWFMFC>2.0.CO;2 (cit. on pp. 21, 71, 110).

[RZ00] Gabriel Robins and Alexander Zelikovsky. "Improved Steiner tree approximation in graphs". In: *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. SODA '00. San Francisco, California, United States: Society for Industrial and Applied Mathematics, 2000, pp. 770–779. ISBN: 0-89871-453-2. URL: http://dl.acm.org/citation.cfm?id=338219.338638 (cit. on p. 44).

[Sea00] Nelson L. Seaman. "Meteorological modeling for air-quality assessments". In: *Atmospheric Environment* 34.12–14 (2000), pp. 2231–2259. ISSN: 1352-2310. DOI: 10.1016/S1352-2310(99)00466-5 (cit. on pp. 21, 71).

[SimTech] Cluster of Excellence Simulation Technology. *Homepage*. URL: http://www.simtech.uni-stuttgart.de (visited on 03/17/2014) (cit. on pp. 19, 56).

[SKR11] B. Schilling, B. Koldehofe, and K. Rothermel. "Efficient and Distributed Rule Placement in Heavy Constraint-Driven Event Systems". In: *HPCC 2011: IEEE 13th International Conference on High Performance Computing*

and Communications. 2011, pp. 355–364. DOI: 10.1109/HPCC.2011.53 (cit. on pp. 49, 153).

[SL08] E.Y. Song and Kang Lee. "Understanding IEEE 1451 – Networked Smart Transducer Interface Standard". In: *Instrumentation & Measurement Magazine, IEEE* 11.2 (Apr. 2008), pp. 11–17. ISSN: 1094-6969. DOI: 10.1109/MIM.2008.4483728 (cit. on p. 50).

[SM10] Chia-Yen Shih and P.J. Marrón. "COLA: Complexity-Reduced Trilateration Approach for 3D Localization in Wireless Sensor Networks". In: *SENSORCOMM 2010: Fourth International Conference on Sensor Technologies and Applications*. 2010, pp. 24–32. DOI: 10.1109/SENSORCOMM.2010.11 (cit. on pp. 19, 51).

[SML+06] Olga Saukh, PedroJosé Marrón, Andreas Lachenmann, Matthias Gauger, Daniel Minder, and Kurt Rothermel. "Generic Routing Metric and Policies for WSNs". In: *Wireless Sensor Networks*. Ed. by Kay Römer, Holger Karl, and Friedemann Mattern. Vol. 3868. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 99–114. ISBN: 978-3-540-32158-3. DOI: 10.1007/11669463_10 (cit. on pp. 19, 51).

[SMW05] Utkarsh Srivastava, Kamesh Munagala, and Jennifer Widom. "Operator Placement for In-Network Stream Query Processing". In: *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. Baltimore, Maryland: ACM, 2005, pp. 250–258. ISBN: 1-59593-062-0. DOI: 10.1145/1065167.1065199 (cit. on pp. 30, 48, 172).

[SPJA13] Saket Sathe, ThanasisG. Papaioannou, Hoyoung Jeung, and Karl Aberer. "A Survey of Model-based Sensor Data Acquisition and Management". English. In: *Managing and Mining Sensor Data*. Ed. by Charu C. Aggarwal. Springer US, 2013, pp. 9–50. ISBN: 978-1-4614-6308-5. DOI: 10.1007/978-1-4614-6309-2_2 (cit. on p. 54).

[SPL+04] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. *Hourglass: An Infrastructure for Connecting Sensor Networks and Applications*. Tech. rep. TR-21-04. Harvard University, 2004 (cit. on pp. 76, 106, 137, 142).

[SR06] Silvia Santini and Kay Römer. "An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks". In: *INSS'06: Proceedings of the 3rd International Conference on Networked Sensing Systems*. 2006, pp. 29–36. URL: http://www.vs.inf.ethz.ch/publ/papers/santinis_inss2006.pdf (cit. on pp. 19, 23, 24, 55, 85, 107).

[TKK+10] Muhammad Tariq, Gerald Koch, Boris Koldehofe, Imran Khan, and Kurt Rothermel. "Dynamic Publish/Subscribe to Meet Subscriber-Defined Delay and Bandwidth Constraints". In: *Euro-Par 2010: Parallel Processing*. Ed. by Pasqua D'Ambra, Mario Guarracino, and Domenico Talia. Vol. 6271. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 458–470. DOI: 10.1007/978-3-642-15277-1_44 (cit. on pp. 27, 106, 138).

[TKK+11] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G. Koch, Imran Khan, and Kurt Rothermel. "Meeting subscriber-defined QoS constraints in publish/subscribe systems". In: *Concurrency and Computation: Practice and Experience* 23.17 (2011), pp. 2140–2153. ISSN: 1532-0634. DOI: 10.1002/cpe.1751 (cit. on p. 171).

[TKKR09] Muhammad Adnan Tariq, Boris Koldehofe, Gerald Georg Koch, and Kurt Rothermel. "Providing Probabilistic Latency Bounds for Dynamic Publish/Subscribe Systems". Englisch. In: *KiVS '09: Proceedings of the 16th ITG/GI Conference on Kommunikation in Verteilten Systemen*. Kassel, Germany: Springer, Jan. 2009. DOI: 10.1007/978-3-540-92666-5_13 (cit. on pp. 106, 138).

[TKKR12] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G. Koch, and Kurt Rothermel. "Distributed spectral cluster management: a method for building dynamic publish/subscribe systems". In: *DEBS '12: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. Berlin, Germany: ACM, 2012, pp. 213–224. ISBN: 978-1-4503-1315-5. DOI: 10.1145/2335484.2335508 (cit. on pp. 30, 171).

[TKR13] Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel. "Efficient Content-based Routing with Network Topology Inference". In: *DEBS '13: Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*. DEBS '13. Arlington, Texas, USA: ACM, 2013, pp. 51–62. ISBN: 978-1-4503-1758-0. DOI: 10.1145/2488222.2488262 (cit. on p. 171).

[TLH08] V.S. Tseng, K.W. Lin, and Ming-Hua Hsieh. "Energy Efficient Object Tracking in Sensor Networks by Mining Temporal Moving Patterns". In: *SUTC '08: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing*. June 2008, pp. 170–176. DOI: 10.1109/SUTC.2008.27 (cit. on p. 55).

[TM06] Daniela Tulone and Samuel Madden. "Wireless Sensor Networks". In: Springer Berlin / Heidelberg, 2006. Chap. PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks, pp. 21–37. DOI: 10.1007/11669463_5 (cit. on pp. 19, 24, 55, 107).

[TPS03] Yufei Tao, Dimitris Papadias, and Jimeng Sun. "The TPR*-Tree: An Optimized Spatio-Temporal Access Method For Predictive Queries". In: *VLDB '03: Proceedings of the 29th international conference on Very large data bases*. Berlin, Germany: VLDB Endowment, 2003, pp. 790–801. ISBN: 0-12-722442-4. URL: http://portal.acm.org/citation.cfm?id=1315451.1315519 (cit. on pp. 24, 106).

[Var10] Andras Varga. "OMNeT++". In: *Modeling and Tools for Network Simulation*. Ed. by Klaus Wehrle, Mesut Güneş, and James Gross. Springer, 2010, pp. 35–59. ISBN: 978-3-642-12330-6. DOI: 10.1007/978-3-642-12331-3_3 (cit. on p. 165).

[WB02] W. Wergen and M. Buchhold. "Datenassimilation für das Globalmodell GME". In: *Die neue Modellkette des DWD* I (2002), pp. 150–155. URL: http://www.met.fu-berlin.de/dmg/dmg_home/promet/27_34/27_3_4_Heft_lores.pdf#page=48 (cit. on p. 110).

[WB06] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, July 2006. URL: http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html (cit. on pp. 19, 24, 55, 107).

[Wer02] W. Wergen. "Datenassimilation – ein Überblick". In: *Die neue Modellkette des DWD* I (2002), pp. 142–149. URL: http://www.met.fu-berlin.de/dmg/dmg_home/promet/27_34/27_3_4_Heft_lores.pdf#page=48 (cit. on p. 110).

[WFF+10] M.L. Williams, K.M. Fischer, J.T. Freymueller, B. Tikoff, A.M. Tréhu, et al. *Unlocking the Secrets of the North American Continent: An EarthScope Science Plan for 2010-2020*. Tech. rep. EarthScope, Feb. 2010. URL: http://www.earthscope.org/doc/reports/es_sci_plan.pdf (cit. on p. 141).

[WH60] Bernard Widrow and Marcian E Hoff. "Adaptive switching circuits." In: (1960). URL: http://www-isl.stanford.edu/people/widrow/papers/c1960adaptiveswitching.pdf (cit. on p. 85).

[WHF+13] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud". In: *Nucleic Acids Research* 41 (2013), pp. 557–561. DOI: 10.1093/nar/gkt328. eprint: http://nar.oxfordjournals.org/content/early/2013/05/02/nar.gkt328.full.pdf+html (cit. on pp. 17, 41).

[WHR10] H. Wolf, K. Herrmann, and K. Rothermel. "Robustness in context-aware mobile computing". In: *WiMob 2010: IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*. Oct. 2010, pp. 46–53. DOI: 10.1109/WIMOB.2010.5645026 (cit. on p. 40).

[WJZF10] Jiansheng Wei, Hong Jiang, Ke Zhou, and Dan Feng. "MAD2: A scalable high-throughput exact deduplication approach for network backup services". In: *MSST: IEEE 26th Symposium on Mass Storage Systems and Technologies*. May 2010, pp. 1–14. DOI: 10.1109/MSST.2010.5496987 (cit. on p. 170).

[WKSR04] M. Wolenetz, R. Kumar, J. Shin, and U. Ramachandran. "Middleware Guidelines for Future Sensor Networks". In: *Proceedings of the First Workshop on Broadband Advanced Sensor Networks*. Citeseer. 2004. URL: http://www.cercs.gatech.edu/tech-reports/tr2004/git-cercs-04-35.pdf (cit. on p. 34).

[WL09] Yiwei Wu and Yingshu Li. "Distributed Indexing and Data Dissemination in Large Scale Wireless Sensor Networks". In: *ICCCN 2009: Proceedings of the 18th International Conference on Computer Communications and*

Networks. Aug. 2009, pp. 1–6. DOI: 10.1109/ICCCN.2009.5235382 (cit. on pp. 19, 35, 53).

[WSS05] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. "Meridian: a lightweight network location service without virtual coordinates". In: *SIGCOMM Comput. Commun. Rev.* 35.4 (Aug. 2005), pp. 85–96. ISSN: 0146-4833. DOI: 10.1145/1090191.1080103 (cit. on p. 155).

[WWG+05] Yansen Wang, Chatt Williamson, Dennis Garvey, Sam Chang, and James Cogan. "Application of a Multigrid Method to a Mass-Consistent Diagnostic Wind Model". In: *Journal of Applied Meteorology* 44.7 (2005), pp. 1078–1089. ISSN: 08948763. DOI: 10.1175/JAM2262.1 (cit. on pp. 21, 71).

[XWL04] Yingqi Xu, J. Winter, and Wang-Chien Lee. "Dual Prediction-based Reporting for Object Tracking Sensor Networks". In: *MOBIQUITOUS 2004: The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. Aug. 2004, pp. 154–163. DOI: 10.1109/MOBIQ.2004.1331722 (cit. on p. 55).

[YLE04] C.K. Yeo, B.S. Lee, and M.H. Er. "A survey of application level multicast techniques". In: *Comp. Comm.* 27.15 (2004), pp. 1547–1568. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2004.04.003 (cit. on pp. 18, 30, 44, 142, 170).

[ZHS+04] B.Y. Zhao, Ling Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. "Tapestry: A Resilient Global-Scale Overlay for Service Deployment". In: *IEEE Journal on Selected Areas in Communications* 22.1 (Jan. 2004), pp. 41–53. ISSN: 0733-8716. DOI: 10.1109/JSAC.2003.818784 (cit. on p. 45).

[ZOTW06] Yongluan Zhou, BengChin Ooi, Kian-Lee Tan, and Ji Wu. "Efficient Dynamic Operator Placement in a Locally Distributed Continuous Query System". In: *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*. Ed. by Robert Meersman and Zahir Tari. Vol. 4275. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 54–71. ISBN: 978-3-540-48287-1. DOI: 10.1007/11914853_5 (cit. on p. 49).

[ZWJ+06] Beichuan Zhang, Wenjie Wang, Sugih Jamin, Daniel Massey, and Lixia Zhang. "Universal IP multicast delivery". In: *Computer Networks* 50.6 (2006). Overlay Distribution Structures and their Applications, pp. 781–806. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2005.07.016 (cit. on p. 44).

[ZZJ+01] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination". In: *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*. Port Jefferson, New York, United States: ACM, 2001, pp. 11–20. ISBN: 1-58113-370-7. DOI: 10.1145/378344.378347 (cit. on pp. 30, 44, 170).