Verteilte Abfragebearbeitung von Sensordaten

Dirk Bade
Verteilte Systeme und Informationssysteme
Department Informatik, Universität Hamburg
Vogt-Kölln-Strasse 30, 22527 Hamburg
Email: bade@informatik.uni-hamburg.de

Zusammenfassung—**Intensive Forschung und Entwicklung im Bereich von Sensornetzen haben Technologien für Hard- und Software ein inzwischen gebrauchstaugliches Niveau erreichen lassen. Trotzdem finden sich Sensornetze heutzutage selten im industriellen sowie privaten Einsatz wieder. Einer der Gründe hierfür sind die technischen Barrieren mit denen sich Nutzer von Sensordaten auseinandersetzen müssen. In dieser Arbeit wird daher eine Middleware-Infrastruktur vorgestellt, die Anbieter und Nutzer von Sensordaten zusammenbringen und eine transparente Abfrage von Sensordaten ermöglichen soll. Durch die Möglichkeit beider Seiten die Verarbeitung von Sensordaten innerhalb der Middleware durch Angabe eines Verarbeitungs-Workflows zu beeinflussen, können einerseits maßgeschneiderte Zugriffe auf Sensordaten gewährt werden. Andererseits trägt dieser Ansatz auch der Endgeräteheterogenität auf beiden Seiten Rechnung, da die Verarbeitungslast in die Middleware verschoben und Ressourcen der (mobilen) Endgeräte geschont werden können.**

I. Einführung

In den vergangenen Jahren führte die stetige Weiterentwicklung von Soft- und Hardware für Sensornetzwerke zu einem Entwicklungsstand, der nicht mehr nur vermehrt universitäre, sondern vor allem auch von der Industrie angetriebene Projekte entstehen ließ. Längst werden Sensoren zur Überwachung von Straßen, Ski- und Erdbebengebieten, intelligenten Häusern, Transportwegen etc. eingesetzt. Doch während sich Forschung und Entwicklung meist auf die intrinsischen Charakteristiken von Sensornetzwerken konzentrierten, bleibt häufig die Frage, was mit den Sensordaten nach ihrer Erhebung passiert, unbeantwortet. Einerseits fehlen geeignete Geschäftsmodelle und andererseits passende Softwareinfrastrukturen, die eine schnelle Umsetzung der Modelle erlauben.

In dieser Arbeit soll daher eine verteilte Middleware-Infrastruktur vorgestellt werden, welche Anbieter (z.B. unabhängige Sensornetzbetreiber) und Nutzer (z.B. Geschäftsanwendungen) von Sensordaten zusammenbringen soll. Anbieter stellen erhobene Sensordaten in nahezu Echtzeit dem System zur Verfügung und Nachfrager können ihr Interesse an bestimmten Daten im System hinterlegen und werden bei Eintreten korrespondierender Ereignisse entsprechend informiert. Das System selbst übernimmt hierbei die Aufgabe zwischen Anbietern und Nachfragenden sowie den von diesen verwendeten Technologien zu vermitteln. Auf diese Weise soll die herrschende Heterogenität verborgen und die vorgestellte Middleware auch für andere Kontextdaten (z.B. RFID, Nutzerkontext etc.) verwendet werden können.

Im folgenden Abschnitt II werden Anwendungsszenarien vorgestellt, anhand derer in Abschnitt III die Architektur und Funktionsweise der Middleware-Infrastruktur präsentiert wird. Abschnitt IV widmet sich dem aktuellen Stand der Implementierung bevor im letzten Abschnitt V eine Zusammenfassung gegeben wird.

II. Anwendungsszenario

Ein Handelsunternehmen hat eine Palette mit TV-Geräten gekauft und deren Verschiffung in einem intelligenten Container veranlasst. Dieser Container ist mit einer Reihe von Sensoren ausgestattet, welche in regelmäßigen Abständen Temperatur-, Feuchtigkeits- und Beschleunigungswerte protokollieren. Sobald der Container an seinem Bestimmungsort ankommt, sollen die Sensor-Logbücher des Containers am Eingang des Lagerhauses ausgelesen werden. Weisen die Sensormessungen daraufhin, dass die Fracht z.B. durch zu hohe Temperaturen oder Beschleunigungen beschädigt sein könnte, soll ein Mechaniker die TV-Geräte noch im Lagerhaus begutachten, bevor sie für die weitere Reise auf einen LKW verladen werden.

In diesem Szenario sind drei verschiedene Akteure beteiligt: 1) das Handelsunternehmen, 2) das Transportunternehmen und 3) das Lagerhaus. Das Handelsunternehmen hat die TV-Geräte eingekauft, Transport und Abwicklung in Auftrag gegeben und ist verantwortlich dafür, dass der letztendliche Käufer die Ware ordnungsgemäß zugestellt bekommt. Hierfür wurde ein Transportunternehmen beauftragt, welches den intelligenten Container stellt und für den sicheren Transport der Ware ins Lagerhaus verantwortlich ist. Das Lagerhaus letztlich soll die TV-Geräte vom Schiff auf einen LKW verladen und das Handelsunternehmen über aktuelle Vorkommnisse informieren. Zu diesem Zweck teilt das Handelsunternehmen dem Lagerhaus mit, dass es informiert werden möchte, sobald a) die Ware eintrifft und Sensorwerte keine Beschädigung vermuten lassen, b) die Ware eintrifft, aber möglicherweise beschädigt ist oder c) die Ware bis zu einem bestimmten Zeitpunkt nicht eingetroffen ist.

Bevor dieses Anwendungsbeispiel in Abschnitt III-A zur Erläuterung der präsentierten Middleware-Architektur wieder aufgegriffen wird, sollen im Folgenden kurz zwei weitere Anwendungsmöglichkeiten skizziert werden.

A. Weitere Anwendungsszenarien

Auf der Intensivstation eines Krankenhauses wird der Gesundheitszustand von Patienten durch eine Reihe von Sensoren überwacht. Um das Pflegepersonal zu entlasten, sollen

die Visiten-Intervalle vergrößert werden. Damit jedoch auch zwischenzeitlich schnell auf Änderungen eines Gesundheitszustands reagiert werden kann, werden individuell für jeden Patienten komplexe Ereignismuster für dessen Sensordaten definiert, die eine Verschlechterung des Krankheitsbildes andeuten. Wird ein solches Ereignismuster in den Sensorwerten erkannt, soll über verschiedene Kommunikationswege sofort ein Krankenpfleger benachrichtigt werden.

Ein weiteres Anwendungsszenario findet sich in der Hausautomation. Intelligente Häuser besitzen eine Vielzahl von Sensoren und Aktuatoren zur Wahrnehmung und Reaktion auf Vorkommnisse im Haus. Ein Server innerhalb des Hauses verarbeitet die Sensordaten und verfügt bereits 'ab Werk' über einige Standard-Reaktionsschemata. Der Besitzer eines intelligenten Hauses möchte die Funktionen dieses Hauses jedoch gerne weitergehend an sein Verhalten individuell anpassen. Hierfür soll jedes Familienmitglied eigene Ereignismuster und die zu erfolgende Reaktion des Hauses angeben können, allerdings sollen nicht alle Mitglieder der Familie die gleichen Rechte und Möglichkeiten bekommen.

Im nächsten Abschnitt III wird nun eine Architektur vorgestellt, welche die Realisierung der genannten Anwendungsszenarien ermöglichen soll. Abschnitt III-A erläutert nachfolgend die Funktionen in Bezug auf die Szenarien beispielhaft.

III. Middleware-Architektur

Der Ansatz beruht auf einer ereignisgesteuerten Architektur, deren logische Ebenen für die speziellen Anforderungen von Sensornetzen durch zwei weitere Ebenen ergänzt wurden: eine Ebene zur Vorverarbeitung der Daten und eine weitere zur nutzerspezifischen Nachbearbeitung (siehe Abbildung 1). Alle Ebenen der Architektur werden durch Softwareagenten realisiert, welche kooperativ an der Verarbeitung von Ereignissen und Nutzeranfragen beteiligt sind (weitere Details in [1]).

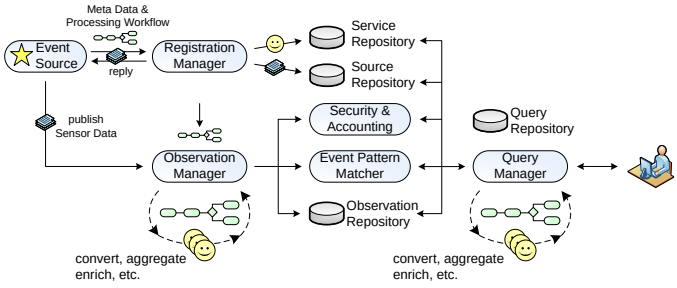

Abbildung 1. Middleware-Architektur

Die unterste Ebene stellen die *Ereignis-Produzenten*, also die Sensornetze, dar. Die in dieser Ebene erzeugten Ereignisse (primitive oder aggregierte Sensorwerte) werden vom Anbieter der Sensordaten an die Middleware übergeben. Diese Ebene ist nicht Teil der vorgestellten Architektur und es werden keine Annahmen oder Anforderungen an Entitäten dieser Ebene gestellt, sodass prinzipiell beliebige Ereignisquellen, neben Sensornetzen z.B. auch RFID-Leser, Mobiltelefone, Software-Sensoren etc., unterstützt werden.

In der zweiten Ebene bedienen spezifische Kommunikationsadapter (in der Abb. nicht dargestellt) unterschiedliche Ereignisquellen und bilden die Schnittstelle zu den durch Softwareagenten realisierten Middleware-Funktionen. Eine wesentliche Funktion bildet die Zugriffsverwaltung (*Registration Manager*), bei der sich ein Sensornetz initial registrieren muss. Bei der Registrierung werden eine Reihe von Meta-Daten über die Ereignisquelle sowie ein (optionaler) *Verarbeitungs-Workflow* verlangt. Ein Austausch von Schlüsseln und Zugriffs-Token dient der späteren gegenseiten Authentifizierung, Autorisierung und Abrechnung.

Als Teil der Meta-Daten kann eine Ereignisquelle ihre funktionalen Fähigkeiten angeben. Funktionale Fähigkeiten sind z.B. eine Anfragebearbeitung innerhalb des Sensornetzwerkes (falls Sensorknoten Messungen speichern und Nutzeranfragen selbst beantworten können) oder die Fähigkeit Instruktionen von der Middleware entgegenzunehmen (bspw. zur Adaption der Sampling-Rate, falls sich momentan kein Nutzer für die Daten des Sensornetzes registriert hat). Auf Basis der funktionalen Daten werden dedizierte Softwareagenten, welche für die Kommunikation mit dem Sensornetz zur Realisierung der Funktionen verantwortlich sind, erzeugt und in einem *Service Repository* angemeldet. Die weiteren Meta-Daten werden in einem Quellenverzeichnis (*Source Repository*), z.B. für die Abfrage durch Nutzer, hinterlegt.

Durch den erwähnten Verarbeitungs-Workflow hat eine Ereignisquelle die Möglichkeit, die von ihr bereitgestellten Sensordaten durch die Middleware in mehreren Schritten vorverarbeiten zu lassen. So können z.B. Filter-, Aggregations-, Gruppierungs- oder Übersetzungsfunktionen angegeben oder die Speicherung der Sensordaten in einer Datenbank (*Observation Repository)* veranlasst werden. Derartige Angaben erfolgen in Form eines BPEL-Prozesses, welcher durch die Middleware in eine Orchestrierung von Agenten übersetzt wird. Als Endpunkte für die Aktivitäten des Workflows können einerseits Standard-Endpunkte der Middleware, aber auch individuelle Endpunkte in Form einer beliebigen Agenten-ID angegeben werden. Somit ist es möglich die Funktionen der Middleware durch eigene, proprietäre Funktionen zu erweitern. Die Ausführung des BPEL-Workflows übernimmt der *Observation Manager*, welcher die Sensordaten an entsprechende Dienstagenten zur Ausführung der jeweiligen Prozessaktivität übergibt. Das Ergebnis der Vorverarbeitung dient als Eingabe für die nächste Ebene.

In dieser nächsten Ebene werden die vom Anbieter produzierten und von der Middleware vorverarbeiteten Ereignisse mit den Anfragen von Nutzern abgestimmt. In Abhängigkeit von den funktionalen Eigenschaften und des Verarbeitungs-Workflows der Ereignisquelle sowie den Anforderungen der Anfrage gibt es für die Abstimmung mehrere Möglichkeiten:

- Eine Laufzeitumgebung zur Verarbeitung komplexer Ereignisse (*Complex Event Processing Runtime*) kann aus einem Strom primitiver Ereignisse von den Ereignisquellen nutzerdefinierte komplexe Ereignisse, z.B. anhand von kausalen oder temporalen Beziehungen beliebiger Ereignisse, erzeugen (*Event Pattern Matching*).
- Das *Observation Repository*, eine Datenbank zur Speicherung von Sensordaten, kann abgefragt werden.
- Falls es die Ereignisquelle erlaubt, können spezielle Dienstagenten des *Service Repositories* Instruktionen an das Sensornetz schicken um eine Nutzeranfrage direkt

von der Ereignisquelle bearbeiten zu lassen.

Das Ergebnis dieser Abstimmung von Sensordaten und Nutzeranfragen kann - gleich welche Abstimmungsmethode in Abhängigkeit oben erwähnter Präferenzen gewählt wurde - auf Wunsch des Nutzers in einer letzten Ebene nachbearbeitet werden. Ziel dieser Nachbearbeitung ist die Erzeugung eines Anwendungsereignisses (*Application Level Event*), welches die für eine Anwendung relevanten Daten in einem entsprechenden Format enthält. Genau wie der Anbieter von Sensordaten einen Verarbeitungs-Workflow angeben kann, hat auch der Nutzer die Möglichkeit einen Teil der Nachbearbeitung von relevanten Ereignissen zu beeinflussen indem er z.B. weitere Filter-, Gruppierungs-, Aggregations- oder Translationsfunktionen angibt. Außerdem kann er, sobald er seine Anfrage zusammen mit dem Verarbeitungs-Workflow bei der Middleware registriert, angeben, zu welchem Endpunkt das Anwendungsereignis später gesendet werden soll. So ist es bspw. möglich das Anwendungsereignis entsprechend formatiert als SMS an ein Mobiltelefon verschicken zu lassen, obwohl die Registrierung über einen PC erfolgte.

Alle Verarbeitungsschritte, die Anbieter oder Nutzer in ihren jeweiligen Verarbeitungs-Workflows angeben, werden durch einen *Accounting Manager* protokolliert. So können Dienstleistungen der Middleware, aber auch die Bereitstellung von Sensordaten durch den Anbieter abgerechnet werden. Auf diese Weise ist es möglich nahezu beliebige Geschäftsmodelle zu realisieren, da durch die Middleware erbrachte Dienste bei Anbieter und Nutzer sowie durch den Anbieter produzierte Daten über den Nutzer abgerechnet werden können. Authentifizierungs- und Autorisierungsmaßnahmen bieten die Grundlage für entsprechende Abrechnungsmodelle.

A. Beispiel

Angenommen das Lagerhaus aus dem Anwendungsbeispiel in Abschnitt II betreibt eine solche Software-Infrastruktur und bietet seinen Klienten entsprechende Dienste an. Das Handelsunternehmen möchte, wie beschrieben, informiert werden, sobald eines von drei Ereignissen im Lagerhaus eingetreten ist, also sobald der Container eintrifft oder sich verspätet und falls Sensorwerte des intelligenten Containers eine Beschädigung der TV-Geräte vermuten lassen. Hierfür muss das Handelsunternehmen ein Ereignismuster definieren, welches vereinfacht folgendermaßen aussieht:

```
SELECT * FROM SmartContainerEvent
WHERE id BETWEEN x AND y AND
  ((overallacc > 5 OR humidity > 80 OR
    temperature > 50) OR
   [weitere Bedingungen])
```

Ein solches Muster, dargestellt in der SQL-ähnlichen *Esper Event Pattern Language* [2], wird in diesem Beispiel von der Complex Event Processing-Laufzeitumgebung verarbeitet, indem diese aus dem Strom primitiver Ereignisse korrespondierende komplexe Ereignisse zu identifizieren versucht. Die Variablen x und y stehen bspw. für RFIDs und identifizieren die TV-Geräte oder den Container, $overallacc$, $humidity$ und $temperature$ referenzieren Messungen der Container-Sensoren. Sobald das Muster in Übereinstimmung mit Ereignissen gebracht wurde, wird ein komplexes Ereignis erzeugt. Dieses Ereignis kann entweder direkt an das Handelunternehmen übermittelt werden, wobei dieses anschließend für die weitere Verarbeitung verantwortlich ist, oder noch in der Middleware weitergehend bearbeitet werden, um ein maßgeschneidertes Anwendungsereignis zu erzeugen.

Ein solches Anwendungsereignis beinhaltet nur die für eine Anwendung relevanten Daten in einem entsprechenden Datenformat. Eine Anwendung muss entsprechend nicht mehr etwaige Transformations-, Filterungs-, Aggregationsfunktionen etc. ausführen, sondern kann das Ereignis direkt weiterverarbeiten. Entsprechende Standardfunktionen müssen somit nicht von jeder Anwendung neu implementiert werden und entlasten die Verarbeitungsressourcen des Anbieters. Dies ist insbesondere für ressourcenarme, mobile Geräte von entscheidender Bedeutung. Der Inhaber des Handelsunternehmen könnte so zum Beispiel bei Eintreten eines entsprechenden Ereignisses direkt über sein Mobiltelefon informiert werden, dass die TV-Geräte während des Transports zu stark erschüttert wurden, und sofort einen Mechaniker rufen, der die Geräte noch im Lagerhaus begutachtet. Notwendige Vorraussetzung hierfür sind jedoch weitere Verarbeitungsschritte, die ein komplexes Ereignis in ein entsprechendes Anwendungsereignis überführen. Neben einem komplexen Ereignismuster kann das Handelsunternehmen also noch zusätzlich einen Verarbeitungs-Workflow vorgeben, der diese Überführung vollzieht. Ein einfaches Beispiel eines solchen Workflows ist in Abbildung 2 dargestellt.

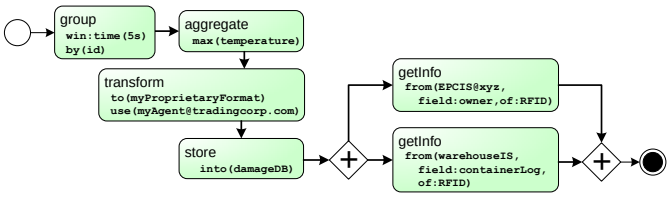

Abbildung 2. BPMN-Diagramm eines Verarbeitungs-Workflows

Dieser Beispiel-Workflow, welchen das Handelunternehmen zusammen mit obigem Ereignismuster bei der Middleware registrieren würde, beschreibt, wie die Middleware eine entsprechende komplexe Ereignisinstanz verarbeiten soll. Im gegebenen Beispiel sollen alle erzeugten komplexen Ereignisse zunächst in einem Zeitfenster von fünf Sekunden gruppiert und anschließend bzgl. der maximalen Temperatur aggregiert werden. Bevor im übernächsten Schritt das Resultat in einer Datenbank für Archivierungszwecke gespeichert werden soll, verlangt das Handelsunternehmen noch die Transformation des komplexen Ereignisses in ein proprietäres Format. Da das Lagerhaus einen entsprechenden Dienst nicht bietet, wird das Ereignis von der Middleware an einen externen Agenten des Handelsunternehmens geschickt. Erst nach dessen Transformation und Rückantwort wird das Ergebnis in der Datenbank gespeichert. Alle bisherigen Schritte mussten sequentiell ausgeführt werden, die nachfolgenden Schritte jedoch sind unabhängig und können parallel ausgeführt werden. In diesen Schritten werden zusätzliche Informationen über die mit

RFID-Chips versehenen TV-Geräte von einem externen *EPC Information Service* abgerufen und gleichzeitig das Sensor-Logbuch des intelligenten Containers an das komplexe Ereignis angehängt.

Ist der Verarbeitungs-Workflow erfolgreich abgearbeitet worden, werden die Resultate vom *Query Manager* als Anwendungsereignis zurück an das Handelsunternehmen oder einen anderen gewünschten Endpunkt gesendet.

IV. Implementierung

Die Implementierung basiert auf der *Jadex Event Stream Processing Architecture* [1] (JESPA), einer von uns entwickelten Architektur zur verteilten Verarbeitung von Ereignisströmen. Diese Architektur zeichnet sich u.a. durch ihre Flexibilität und Erweiterbarkeit aus, weshalb sie in großen Teilen den Kern der hier vorgestellten Middleware darstellt.

Wesentliche Teile des beschriebenen Konzeptes sind bereits umgesetzt. Auf unterster Ebene wurden bspw. Adapter zur Kommunikation mit Ereignisquellen implementiert. Ein Adapter für das *Nokia 6131 NFC*-Telefon [3] erlaubt die Verarbeitung von RFID-Daten, *SunSPOT*-Sensoren [4] liefern Helligkeits-, Feuchtigkeits- und Beschleunigungswerte, ein *Android*-Adapter [5] stellt aktuelle Aktivitäten eines Nutzers zur Verfügung und ein erster Prototyp-Adapter für die *MagicMap*-Software [6] erlaubt die Verarbeitung von Positionsschätzungen realer Objekte, welche MagicMap aus Signalstärkemessung diverser Funktechnologien errechnet.

Für die Dienstorchestrierung zur Vor- und Nachverarbeitung von Ereignissen in der Middleware sorgen spezielle Dienstagenten, welche mittels des *Jadex v2*-Agentensystems [7] implementiert wurden. Jadex unterstützt in der v2-Version nicht nur das Programmieren von 'intelligenten' *BDI*-Agenten (BDI=Belief, Desire, Intention) für komplexe Aufgaben, sondern auch das Erzeugen leichtgewichtiger Agenten, die einfache Dienstaufgaben übernehmen und für die dedizierte Verarbeitung einzelner Ereignisinstanzen verantwortlich sind.

Die *Esper*-Laufzeitumgebung [2] schließlich sorgt für die Identifizierung und Erzeugung komplexer Ereignisse basierend auf Strömen primitiver Ereignisse von Sensornetzen und liefert mit seiner *Event Pattern Language* die Basis zur Spezifikation von Ereignismustern für Nutzer von Sensordaten.

Die Implementierung konnte jedoch bisher nicht ausreichend evaluiert werden, da zur Umsetzung komplexer Szenarien noch wesentliche Bausteine der Architektur realisiert werden müssen.

V. Fazit

In dieser Arbeit wurde eine Middleware zur verteilten Abfragebearbeitung von Sensordaten vorgestellt. Ziel der Middleware ist es, Nutzer einen transparenten Zugriff auf relevante Sensordaten zu erlauben, indem die Heterogenität der Hard- und Software durch die Middleware verborgen wird. Der Zugriff auf Sensordaten kann über verschiedene Wege, entweder durch Identifikation komplexer Ereignismuster in Ereignisströmen, der Abfrage von Sensor-Historien aus Datenbanken oder der durch die Middleware vermittelten Kommunikation mit dem Sensornetzwerk selbst, erfolgen. Anbieter und Nutzer von Sensordaten haben die Möglichkeit durch Angabe von BPEL-basierten Verarbeitungs-Workflows die Vor- und Nachbearbeitung von Ereignissen durch die Middleware zu veranlassen, um so z.B. Ressourcen der Endgeräte (z.B. Sensoren, Mobiltelefone etc.) auf beiden Seiten der Verarbeitungskette zu entlasten.

In weiteren Schritten sollen bereits existierende Standards, z.B. *SensorML* [8] zur Beschreibung von Sensoren und deren Messungen, in das System integriert werden. Weiterhin sollen Gemeinsamkeit mit anderen Projekten, z.B. dem *COUGAR Sensor Database Project* [9] etc. und Möglichkeiten der Interoperabilität - sofern möglich - untersucht werden.

Literatur

[1] D. Bade, "Towards an extensible agent-based middleware for sensor networks and rfid systems," in *Inproceedings of the 3rd Int. Workshop on Agent Technology for Sensor Networks (ATSN-09)*, 2009.

[2] EsperTech, "Esper - event stream intelligence," esper.codehaus.org, 2009.

[3] Nokia, "Nokia 6131 nfc," http://www.nokia.de/A4420858, May 2009.

[4] Sun, "Sunspot," http://www.sunspotworld.com/, May 2009.

[5] Google, "Android," http://developer.android.com/, May 2009.

[6] S. Bruning, J. Zapotoczky, P. Ibach, and V. Stantchev, "Cooperative positioning with magicmap," March 2007, pp. 17–22.

[7] A. Pokahr and L. Braubach, "From a research to an industrial-strength agent platform: Jadex v2," in *Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, H.-G. F. Hans Robert Hansen, Dimitris Karagiannis, Ed. Österreichische Computer Gesellschaft, 2 2009, pp. 769–778.

[8] O. Open Geospatial Consortium, "Sensorml," http://www.opengeospatial.org/standards/sensorml, May 2009.

[9] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Rec.*, vol. 31, no. 3, pp. 9–18, 2002.