An Overview of Centralised Middleware Components for Sensor Networks

Martijn Onderwater[1,2]

[1]Center for Mathematics and Computer Science (CWI), Stochastics Group, Science Park 123, 1098 XG Amsterdam, The Netherlands, m.onderwater@cwi.nl,

[2]VU University Amsterdam, Department of Mathematics, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands.

April 30, 2014

Abstract

Sensors are increasingly becoming part of our daily lives: motion detection, lighting control, environmental monitoring, and keeping track of energy consumption all rely on sensors. Combining data from this wide variety of sensors will result in new and innovative applications. However, access to these sensors – or the networks formed by them – is often provided via proprietary protocols and data formats, thereby obstructing the development of applications. To overcome such issues, middleware components have been employed to provide a universal interface to the sensor networks, hiding vendor-specific details from application developers. The scientific literature contains many descriptions of middleware components for sensor networks, with ideas from various fields of research. Recently, much attention in literature is aimed at what we, in this paper, define as 'centralised' middleware components. These components consider sensor networks that have no capacity – in terms of memory, data storage, and CPU power – to run middleware components (partially) on the sensor nodes. Often, viewed from the position of the middleware component, these sensor networks function as simple data providers for applications.

In this paper we introduce the term 'centralised' for such middleware components, guided by a literature review of existing middleware components for sensor networks. We describe their general architecture, give a description of a representative set of four centralised middleware components, and discuss advantages and disadvantages of these components.

Finally, we identify directions of further research that will impact centralised middleware systems in the near future.

Keywords: Sensor network middleware, Centralised middleware components, Sensor web enablement, Middleware categorisation, Sensor web, Web services, Sensor networks.

Biographical notes: Martijn Onderwater is a Ph.D. student at the Center for Mathematics and Computer Science (Amsterdam, The Netherlands), and at VU University Amsterdam (The Netherlands). He received a M.Sc. degree in Mathematics (2003) from the University of Leiden (The Netherlands), and a M.Sc. degree in Business Mathematics & Informatics (2010, Cum Laude) from VU University Amsterdam. He also has several years of commercial experience as a software engineer. His research focuses on sensor networks, and his interests include middleware components for sensor networks, dimensionality reduction, outlier detection, caching in sensor networks, Markov decision processes, and evolutionary algorithms.

1 Introduction

Sensor networks have been used for decades in a wide variety of applications, for instance habitat monitoring (Szewczyk et al. (2004)), the Smart Kindergarten (Srivastava et al. (2001)), health care (Baker et al. (2007)), and monitoring of infrastructural performance (Knobbe et al. (2010)). Sensors also play an increasing role in our daily lives: smart electricity meters provide real-time consumption information, washing machines contain fuzzy intelligence to dynamically determine the amount of needed water, lights turn on only when movement is detected nearby, and mobile phones contain a myriad of sensors as well.

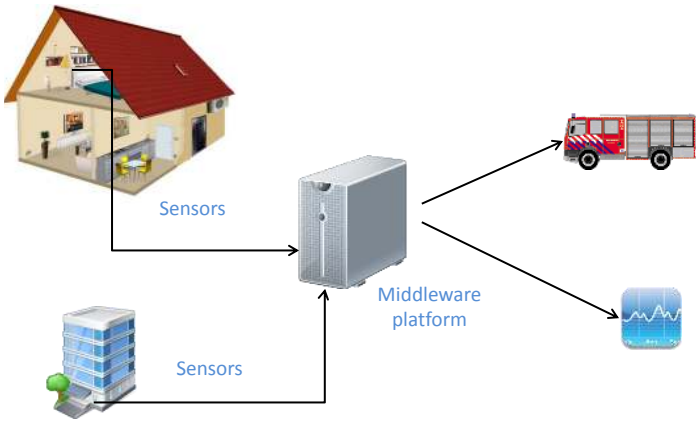

Figure 1: Illustration of the position of sensor middleware component

Sharing sensor data can be quite challenging in practice, because each sensor potentially has different capabilities, unique data representations, and proprietary access interfaces. How can we easily share sensor data, without troubling the user with such specifics? In traditional IT-systems, a *middleware component* is often used to provide a generic method for accessing resources, and the same approach is used in sensor networks. Figure 1 illustrates the position of a middleware platform in the context of sensor networks. The platform collects data from sensor networks located in, e.g., a house or an office. On the right side of the picture, the collected data is shared with 3rd party applications. For instance, an alarm can be sent to the fire department when a fire is suspected, or a weekly report on the kids' TV-time can be generated.

Hence, the middleware component is responsible for collecting data from the sensor networks, and for publishing their data for use by applications. In essence, a middleware component forms a bridge between the sensor network and applications relying on data from these networks. This broad description of the term 'middleware component' allows for a wide variety of approaches inspired by ideas from various fields of research and industry. Consequently, a large body of literature covering sensor network middleware components already exists, jointly containing more than 100 different components. These middleware components can be divided into two groups: components that (partly) run on the sensors in the network, and those that do not run on the sensors. Middleware components in the first group often require some software to be installed on each sensor in order to perform their tasks, thereby creating an intelligent sensor network. The second group views sensors as simple data providers and combines data from these sensors into intelligent applications.

In this paper we focus on components from the second group: those aimed at sensor networks that have no capacity – in terms of memory, data storage, and CPU power – to run middleware components (partially) on the sensor nodes. Recently, such components form a popular research topic and have high practical relevance. We will introduce the term *centralised* for these middleware components, and illustrate their importance using a literature review of existing middleware components. After an overview of their general architectural setup, we describe and discuss the most important centralised middleware components.

The next section contains a literature review of non-centralised middleware components, and discusses why centralised components have become popular in recent years. Section 3 starts with a general description of centralised middleware, then describes one of the standards for publishing sensor data on the web in Section 3.2, and continues with a description of the most well-known centralised middleware components in Section 3.3. We identify directions for future research in Section 3.4, and conclude in Section 4.

2 Middleware components in literature

Middleware platforms for sensor networks have received abundant attention in the scientific literature. Until a few years ago, the following categorisation of sensor network middleware was often used (adapted from Andreou et al. (2011); Hadim and Mohamed (2006a,b); Henricksen and Robinson (2006); Masri and Mammeri (2007)):

- *Database-inspired components*;
- *Virtual Machine-motivated components*;
- *Agent-based components*;
- *Application-driven components*;
- *Message-oriented components*.

We will use this categorisation as a guideline to give a high-level overview of sensor middleware components in the literature. For a more in-depth review we refer the readers to, e.g., the surveys by Wang et al. (2008); Molla and Ahamed (2006), or to the specialized surveys by Mohamed and Al-Jaroodi (2011) (on service-oriented middleware approaches) and by Sugihara and Gupta (2008) (on programming sensor networks).

Below, we will describe each of the categories, illustrate the architectural similarities within a category, and list several middleware components contained in the category. Table 1 contains the middleware components per category, with a timeframe, a reference to a publication, and a link to a website (if available).

2.1 Database-inspired components

This subclass of middleware views the sensor network as a distributed database and adapts existing querying techniques to the sensor network. Figure 2 illustrates a simple network with three sensor nodes, and a laptop interesting in collecting data from these nodes. Each node typically has a local database (DB) and a query engine for dealing with queries. Together, the storage and query processing facilities form the middleware component. As an example, suppose that the laptop issues a query to the network, requesting all measurements from sensors 2 and 3 at intervals of 1 second for the next 10 seconds. In SQL-like notation, this could be *SELECT * FROM sensors WHERE id IN (1,2) SAMPLE RATE 1s FOR 10s*. The laptop passes this query to node 1, which in turn

forwards it to nodes 2 and 3. There the query is executed and the results are returned.

This example illustrates three important aspects of middleware components in this category. First, issued queries should be routed to the correct nodes and thus the middleware should maintain some structured representation of the network. The second aspect concerns the need for a sensor-specific query language, which usually is a modified form of standard SQL. In the example above, the keywords "SAMPLE RATE" and "FOR" are extensions to conventional SQL. Thirdly, in conventional database systems the results of the queries are always immediately returned after it has been processed. In the context of sensor networks, however, queries can run and produce output continuously, resulting in a stream of data.

Middleware platforms in this category include TinyDB (Madden et al. (2005)), IRISNet (Gibbons et al. (2003)), SINA (Shen et al. (2001)), Cougar (Yao and Gehrke (2002)), DSWare (Li et al. (2004)), SNEE (Galpin et al. (2008)), and KSPOT Andreou et al. (2009).

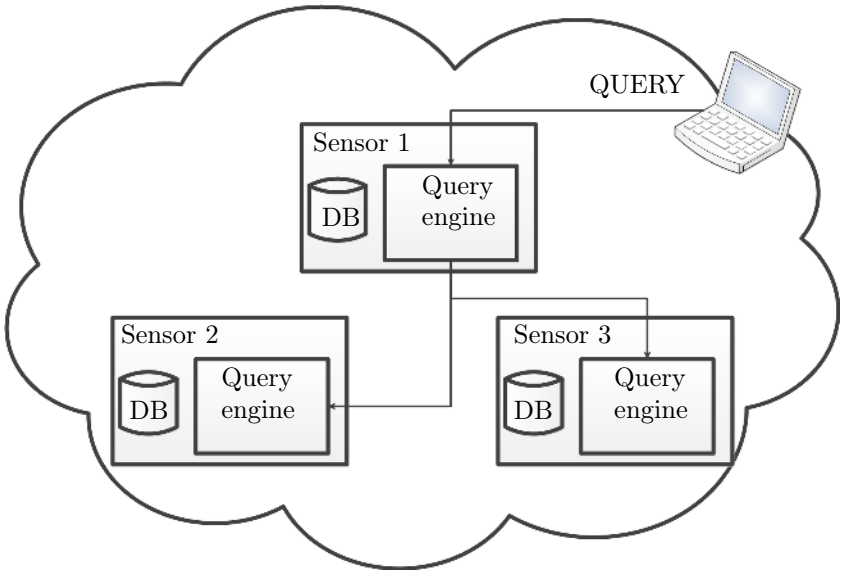

Figure 2: Simple sensor network where the nodes are viewed as parts of a distributed database. Each node comprises of a local database and query engine. The engine interprets and forwards queries to the correct nodes, and executes them to retrieve data from its local database

2.2 Virtual Machine-motivated components

A Virtual Machine (VM) is a platform-independent programming environment that hides details of the underlying operating system and hardware. Software developers can thus write programs in one language, and deploy it to any device running a virtual machine. Since sensors are based on a wide variety of software and hardware, using a virtual machine is also appropriate for sensor networks. When each sensor runs a virtual machine, then creating reusable programs for sensor networks becomes considerably easier.

Figure 3 illustrates the setup, containing a simple sensor network of three nodes, each running a virtual machine. The application on the laptop sends a program into the sensor network, where it arrives at Sensor 1. There, it is split into three parts (resulting in this case in three parts, one for each sensor) and forwarded to the correct nodes. Each node then executes the part it received, until the program ends.

Applications for sensor networks typically use data from multiple sensors, so a key feature of VM-motivated middleware is that it facilitates the splitting of programs. Additionally, it sends the parts of the program to the correct sensor nodes, so the middleware component also needs to maintain a structured view of the network. Finally, the middleware component must manage software updates across the sensor network.

Maté (Levis and Culler (2002)), MUSE (Rittle et al. (2005)), and Magnet (Liu et al. (2005)) are components in this category.

2.3 Agent-based components

Agents are small pieces of software that work together to achieve a predefined goal. Unlike conventional computer programs, agents are not activated by external commands but act autonomously based on a set of rules and on information from their environment. Additionally, agents are mobile and capable of moving from one environment to another.

The principles of agent-based systems have also been applied in the context of sensor networks. To facilitate the execution and migration of agents in sensor networks, middleware components in this category typically equip each node in the network with a special execution environment (EE). This is illustrated in Figure 4, which shows a sensor network with three nodes and two agents, one at the second sensor node, and one en-route from sensor 1 to sensor 3.

As with the VM-oriented frameworks, agent-based middleware components need

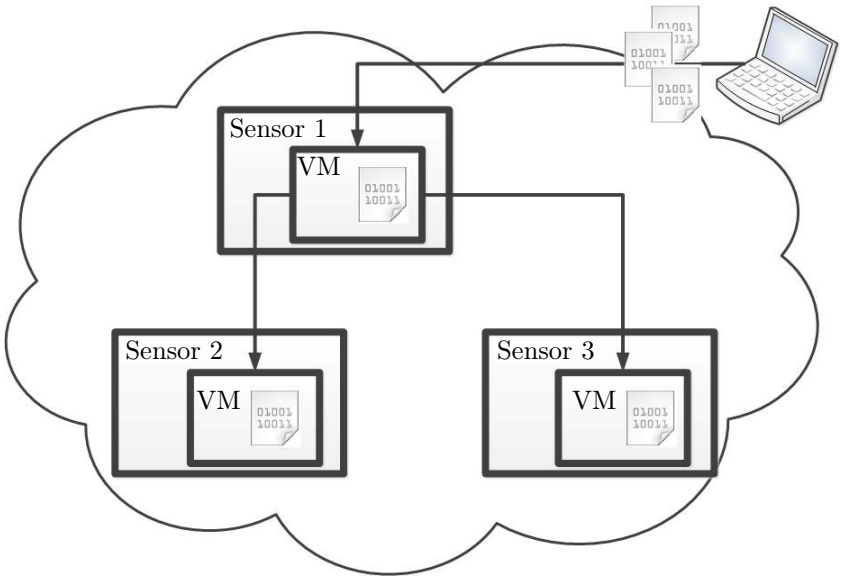

Figure 3: VM-based middleware components deploy a Virtual Machine on each node, and thus provide an execution environment for small pieces of code. In this example, a program is split into three parts and forwarded to the appropriate node. There, the program is executed until it is finished

a structured view of the network, require special skills from the software developers, and must facilitate the distribution of software updates.

Well-known agent-based approaches are IMPALA (Liu and Martonosi (2003)), AGILLA (Fok et al. (2009)), SWAP (Moodley and Simonis (2006)), and MAPS (Aiello (2009)).

2.4 Application-driven components

Whereas the middleware components in the previous categories were grouped by an architectural similarity, the components in this category have a common goal: optimising Quality of Service (QoS). There is no formal definition of QoS, so the various QoS aspects are typically application-specific. This is why this category is named 'application-driven', although perhaps 'QoS-driven' is a more appropriate name. An example of an application-driven middleware component is MILAN (Middleware Linking Applications and Networks, by Heinzelman et al. (2004)), which considers both the QoS requirements of an application and the QoS capabilities of the available sensors. These requirements and capabilities

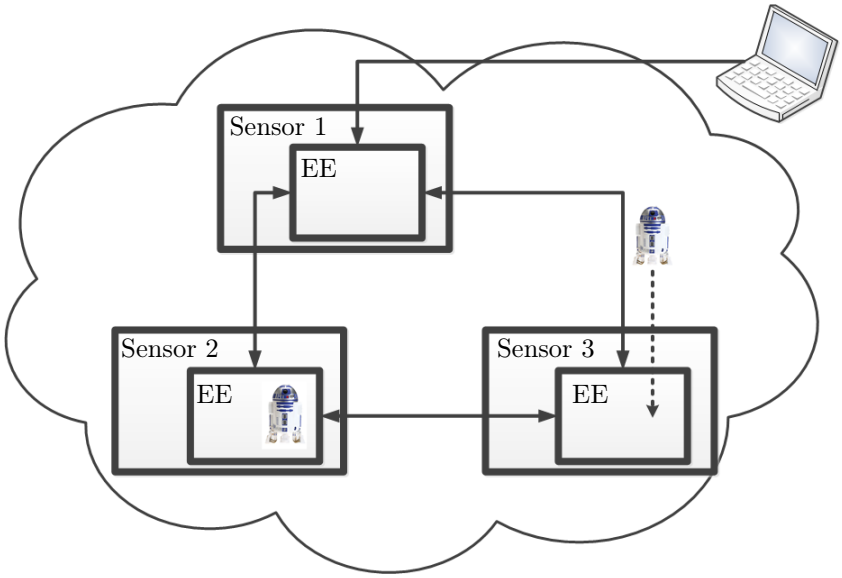

Figure 4: Agent-based middleware components equip each sensor with an execution environment (EE) allowing agents to, e.g., migrate to other sensors

are then matched to select the sensors involved in the application. In MiLAN, the QoS levels reflect uncertainty in sensor measurements, i.e., if faced with a choice between using a sensor with level 0.7 and a sensor with level 0.8, the latter is preferred due to its higher reliability.

Besides MiLAN, MidFusion (Alex et al. (2008)) and the adaptive middleware component (AMC) by Huebscher and McCann (2004) also belong this category.

2.5 Message-oriented components

In sensor networks, measurements often occur because of events that happen in the monitored environment. For instance, a high temperature measurement may result in the transmission of that measurement to all interested applications. In traditional IT-systems, the *publish/subscribe* mechanism is often used to provide such event-driven communication, and it has also been used in the context of sensor networks. In the publish/subscribe mechanism each sensor node broadcasts a standardized description of its capabilities across the network to interested applications. If an application is interested in the measurements of a sensor, it notifies this sensor that it wants to subscribe the sensors events. When a sensor detects an event, the resulting measurement is forwarded ('published') across

the network to all subscribing applications. Hence, applications are triggered if an event happens to which they have subscribed.

We use MIRES from Souto et al. (2005) as an example of a middleware component in this category, with Figure 5 illustrating the architecture of a sensor node as used by MIRES. At the heart of each node is the *Publish/subscribe service*, which provides communication between the services offered by the node and the rest of the sensor network. The *Node application* reads values from the sensors and notifies the Publish/subscribe service that new readings are available. Other local services (if any), for instance an *Aggregation Service*, can now do the processing that they require. After completion, the results are handed off to the *Routing* component, which publishes them to the sensor network. Full-featured user oriented applications can be constructed on MIRES by notifying the network which nodes and node applications are needed.

The event-based nature of many sensor networks makes the publish/subscribe mechanism a useful tool for middleware components in this category. Moreover, this mechanism fully decouples applications and sensor nodes, so that adding or removing applications and/or sensor nodes does not require global changes to the middleware component. Finally, the publish/subscribe mechanism hides vendor and hardware specific details about sensors from applications. Thereby, developing applications in networks with heterogeneous sensors becomes more convenient.

In this category AWARE (Ollero et al. (2007)), WMOS (Zhai et al. (2011)), and TINYMQ (Shi et al. (2011)) are also included.

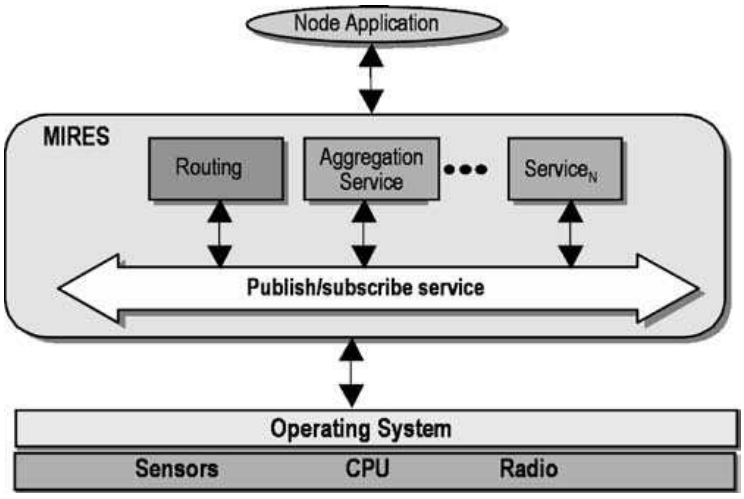

Figure 5: Architecture of one MIRES node, copied from Souto et al. (2005)

2.6 Discussion

Most of the middleware components in the categories from the previous sections have not been widely adopted in practice. One of the reasons for this is that the middleware components were motivated by increased technological possibilities of sensors (e.g., more memory, larger storage capacity, and better processors), but many of the sensors used in practice today are still resource-limited devices with no possibilities to run, e.g., a virtual machine.

A second reason is that industry standards such as Zigbee (2007) have received little attention in the literature for sensor network middleware. As the authors of Mottola and Picco (2011) put it: *Academic WSN research and ZigBee appear to intersect only seldom, if at all. [...] This progressively caused industry to lose interest in academic* WSN *research, as compliance with standards [...] is key to industry applications.*

Finally, one can question whether a sensor middleware component should extend into the sensor network. If it does, then the middleware component is faced with typical network-issues such as routing, and perhaps the responsibility for this should be assigned to the network, not to the middleware.

In recent years the scientific community picked up on these concerns and gradually moved towards a new generation of middleware components. These are the topic of this survey, and we elaborate on them in the next section.

3 Centralised middleware components

3.1 High-level outline

As explained in the previous section, the ideas underlying the approaches from the various categories have not been widely adopted in practice. Many real-world applications provide their services based on sensor measurements only, without requiring middleware components to extend into the sensor network. For these applications, sensor networks function primarily as simple data sources. Moreover, in recent years sensor-driven applications have started combining data from multiple existing sensor networks, and sensor middleware research has shifted in the same direction. Consequently, data collection from existing sensor networks, and data publishing for use by 3rd parties are now popular topics in sensor middleware research.

To illustrate the position of such middleware components, Figure 6 shows three sensor networks that are connected to a middleware component. This middle-

	Timeframe	Reference	Website
Database-inspired			
SINA	2001	Shen et al. (2001)	-
COUGAR	2000-2005	Yao and Gehrke (2002)	Cougar (2002)
IRISNET	2002-2005	Gibbons et al. (2003)	IRISNet (2003)
TINYDB	2002-2005	Madden et al. (2005)	TinyDB (2005)
DSWARE	2004	Li et al. (2004)	-
KSPOT	2007-2011	Andreou et al. (2009)	KSPOT (2009)
SNEE	2008-2009	Galpin et al. (2008)	SNEE (2008)
Virtual Machine-motivated			
MATÉ	2002-2005	Levis and Culler (2002)	Maté (2002)
MAGNET	2001-2005	Liu et al. (2005)	Magnet (2005)
MUSE	2005	Rittle et al. (2005)	-
Agent-based			
IMPALA	2002-2004	Liu and Martonosi (2003)	-
AGILLA	2004-2007	Fok et al. (2009)	Agilla (2009)
SWAP	2006	Moodley and Simonis (2006)	-
MAPS	2009-now	Aiello (2009)	MAPS (2009)
Application-driven			
MILAN	2002-2004	Heinzelman et al. (2004)	-
AMC	2004	Huebscher and McCann (2004)	-
MIDFUSION	2008	Alex et al. (2008)	-
Message-oriented			
MIRES	2005	Souto et al. (2005)	-
AWARE	2006-2009	Ollero et al. (2007)	AWARE (2001)
WMOS	2011	Zhai et al. (2011)	-
TINYMQ	2011	Shi et al. (2011)	-

Table 1: Overview of the middleware components per category. Each component has a reference to a paper describing the architecture, and a link to a website (if one exists)

ware component is responsible for collecting the data from these networks, and publishing it to 3rd party applications. Figure 6 contains two example applications: an indoor climate monitoring dashboard, and a building management application. Note that the middleware component forms a bridge between the applications and the sensor networks, hence we refer to this type of component as a '*centralised middleware component*'.

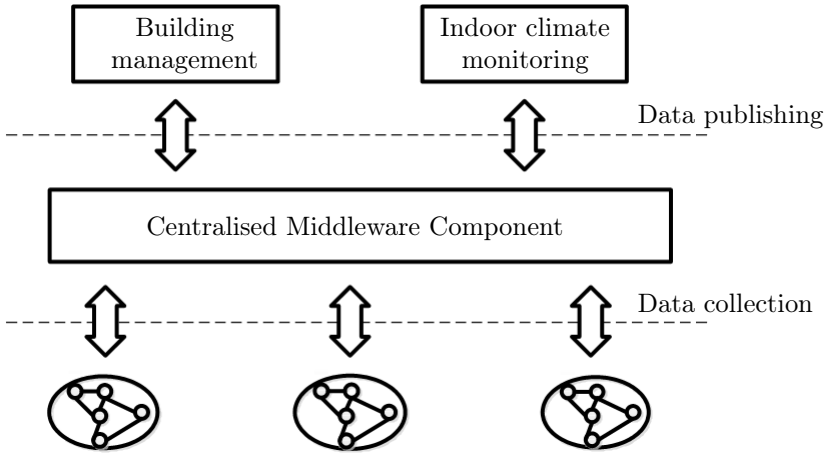

Figure 6: Position of the centralised middleware component with respect to the sensor network and applications

In the remainder of this section we show several examples of centralised middleware components from the scientific literature, and discuss their advantages and disadvantages. Before that, in the next section, we describe the leading industry standard for data publishing, as this is included in many of the middleware components.

3.2 Sensor Web Enablement initiative

Several years ago an international group of companies, government agencies and universities from the OpenGeospatial-Consortium (2010) (OGC) started the Sensor Web Enablement (SWE) initiative. This initiative aims to support the discovery and exchange of sensor information, as well as the tasking of sensor systems. It consists of standards covering the topics of modelling sensors and observations, and of interfaces for communicating with sensors. The standards and interfaces in SWE are defined as web services, and include the following specifications:

- *Observations & Measurements*: a schema for describing sensor observations and measurements.
- *Sensor Model Language*: an interface for describing sensor systems and their capabilities.
- *Sensor Observation Service (*SOS*)*: a web service to obtain observations and sensor and platform descriptions from one or more sensors.
- *Sensor Planning Service (*SPS*)*: provides users with a standard interface for setting their own data collection requests.
- *Sensor Alert Service (*SAS*)*: defines an interface for publishing and subscribing to sensor alerts.
- *Web Notification Service (*WNS*)*: handles the asynchronous message delivery to the subscribers of the SAS and SPS.

We illustrate the SWE specifications with a use-case from the SensorSA middleware by Bleier et al. (2010), one of the middleware components we describe in the following section. In this case study, a decision support system for marine risk management is created using SensorSA. The system monitors the quality of seawater in areas where people often swim, and also predicts this quality for the near future. Authorities use this information to close beaches with a high risk of contamination, thereby preventing sickness due to microbial pollution.

An abstract view of the application is shown in Figure 7. On the top, historical and online sensor data is imported into SensorSA and used to tune and update forecasting models. The import is done using the *Sensor Observation Service*, which provides methods for requesting, filtering, and retrieving sensor data and information. The forecasting in this application is done by a (application-specific) modelling service, which generates an alarm when a prediction indicates future bad water quality. This alarm is passed to the Alerting Service, an instance of the *Sensor Alert Service*. This service uses the publish/subscribe mechanism to allow applications to send and receive alarms, corresponding meta-data, or any other form of output. Sending the alarm is done using *Web Notification Service*, which provides the ability to send an alarm as, e.g., an email or a text message.

Despite SWE's popularity, several drawbacks of the standards have been identified in the scientific literature (from Bai et al. (2011); Moodley and Simonis (2006))

- There is no explicit ontological structure in the SWE framework.
- Security and privacy issues are not addressed.

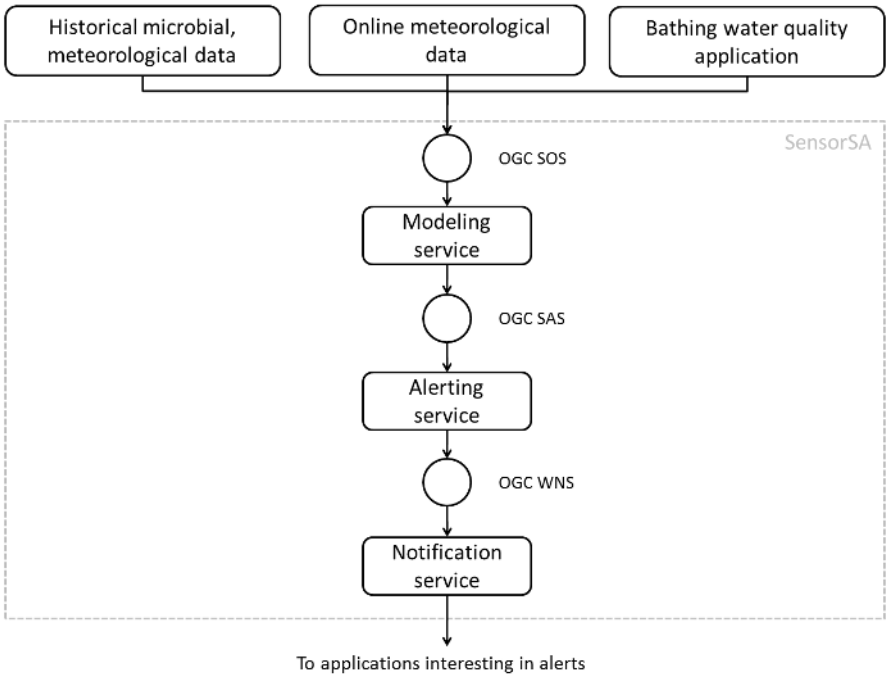

Figure 7: Abstract view of an application for monitoring the quality of seawater using the SensorSA architecture, adapted from Bleier et al. (2010)

- Conversion from a network-specific format to SWE standards requires detailed knowledge of both formats. Typically, off the shelve sensors do not provide their data in SWE form. We will return to this drawback in Section 3.4.
- There are no guidelines for the communication between services.
- Services are passive, so for example, a user can contact the SOS, but not vice versa.

Despite these drawbacks, SWE is currently the main standard for publishing data from sensor networks.

3.3 Overview of components

In this section we describe four centralised middleware components from the scientific literature. First, we describe SensorSA, because it is a clear example

showing the use of SWE standards, and because it is well documented. Then we discuss SENSEWEB, to show that non-SWE web services can also play a valuable role. Next we present PULSENET, the most complete centralised middleware component available, featuring both SWE-based interfaces as well as various other industry standards for dealing with sensor data. Finally, we describe the middleware component LSM that utilizes the streaming nature of sensor data (one of the future research directions we identify in Section 3.4), and provides a way of publishing data without using web services.

3.3.1 SENSORSA

SENSORSA, short for Sensor Service Architecture, is a middleware component developed in the SANY (Sensors Anywhere) project. SANY *aims to improve the interoperability of in-situ sensors and sensor networks, allowing quick and cost-efficient reuse of data and services from currently incompatible sources*[...] (Bleier et al. (2010)). Its central role is illustrated in Figure 8, with in-situ sensors at the bottom, users at the top, and SENSORSA in the middle.

Its use of open standards from SWE makes SENSORSA an interesting component for companies seeking to include sensor data into their IT infrastructure. Several use cases are discussed in Bleier et al. (2010) and illustrate the use of SWE in practice. Additionally, SENSORSA uses several non-SWE interfaces from OGC for, e.g., visualisation.

SENSORSA also serves as an example of how a centralised approach simplifies security issues. Since the middleware component is not responsible for security on the sensor network (this networks is considered to be owned and managed by a 3rd party), it only needs to secure its own services. For this, SENSORSA relies on well-known security mechanisms for access control to service networks, see (Bleier et al., 2010, Chap. 5).

Furthermore, SENSORSA contains several data fusion algorithms for analysis along both the time-dimension and the space-dimension of sensor data. Together with a time series toolbox for analysing streaming sensor data, SENSORSA thus addresses two fields that we recognize as important future research directions in Section 3.4.

Despite the steps forward provided by SENSORSA, several of the drawbacks to SWE remain: there is no ontological structure, and services still seem to be passive. Moreover, there is no implementation of SENSORSA available for download, so a quick experiment with SENSORSA on an existing sensor network is not possible.

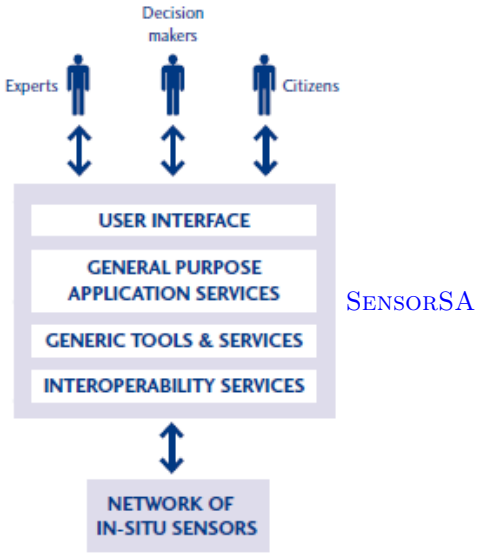

Figure 8: Illustration of the role of SensorSA, a centralised middleware component (adapted from Bleier et al. (2010))

3.3.2 SenseWeb

SenseWeb is a sensor middleware component from Microsoft Research, *designed to let multiple concurrent applications share sensing resources contributed by several entities in a flexible but uniform manner* (Kansal et al. (2007)). The key elements of SenseWeb are illustrated in Figure 9, and strongly resemble the high-level description of a centralised middleware component from Figure 6. The primary building block is the *Coordinator*, which collects data via the *Sense gateway* and publishes this data to *Applications* and *Transformers*. Sensors can be addressed through a *Sense gateway*, which provides a uniform interface for the rest of Senseweb, hiding any vendor-specific aspects. Transformers are components that process sensor data into other formats, for instance by calculating averages, or by creating figurative representations of data. In this way, transformers provide low-level elements that can be easily included in applications.

The Coordinator consists of two separate modules, the *SenseDB* and the *Tasking Module*. SenseDB provides load-balancing facilities by analysing requests for data to find overlap in their desired responses, and by using a cache for sensor data. Additionally, it is responsible for keeping track of the various sensors attached to the coordinator, and of their descriptions and capabilities. The Tasking Module determines which sensors are most suitable for answering a

query, taking into account, e.g., bandwidth, availability, and power levels. Thus, these two models together provide an intelligent mechanism for load balancing, which is the key distinguishing feature of SenseWeb.

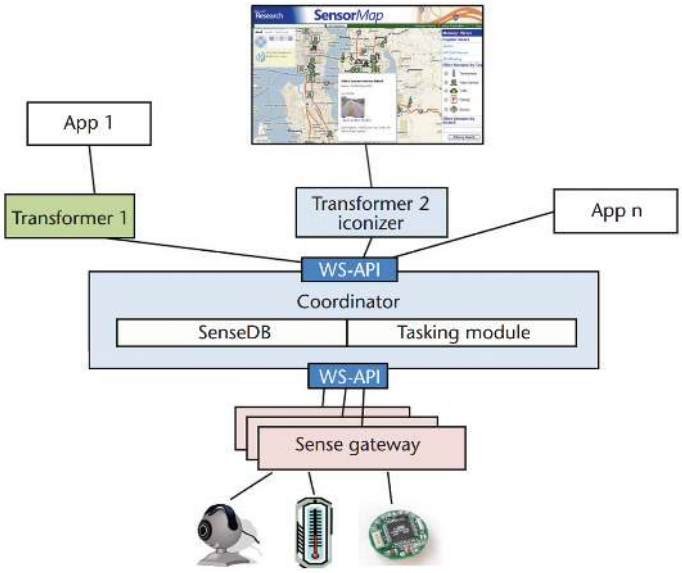

Figure 9: Overview of SenseWeb, showing how the coordinator mitigates between applications and sensors (adapted from Kansal et al. (2007))

3.3.3 PULSENet

PULSENet is a sensor web framework developed at the Northrop Grumman Corporation, with the objective to *provide a standards-based framework for the discovery, access, use and control of heterogeneous sensors, their metadata, and their observation data* (Fairgrieve et al. (2009)). It is based on the standards provided by SWE, supplemented by a wide variety of non-SWE standards for, e.g., describing public safety alerts, detailing military events, and visualisation (see Table 2 in Fairgrieve et al. (2009)). Sensor networks are connected to PULSENet via plugins, which hide vendor-specific interfaces and perform the translation to and from the SWE standards.

From a practical point of view, PULSENet has been tried and tested extensively. It is used, for instance, in the Defense and Intelligence domain, which contains sensors and platforms with many modalities, levels of complexity, data formats, and privacy issues. Other domains include the Ocean Science community, and Air Quality applications. This emphasizes the practical relevance of PULSENet, and of centralised middleware components in general.

The authors of Fairgrieve et al. (2009) also provide a list of best practices when dealing with SWE. These can be summarized as:

- *Apply the* SWE *standard only when needed.* Using SWE for data publishing typically means sacrificing some performance, due to SWE's complexities. So apply SWE only when a device has sufficient capacity to run web services and parse XML. Otherwise, consider using more low-level standards.
- *Keep it simple.* SWE is a large and flexible standard, offering both simple and complex data structures. Use the complex structures only if necessary to keep SWE overhead as low as possible.
- *Use [. . .] the* SWE *compliance tests.* OGC offers a compliance engine that allows 3rd parties to test their implementations of SWE standards. Passing the compliance test adds significant value to a SWE-enabled middleware component.
- *Avoid reinventing the wheel.* Several open source implementations for SWE webservices are available, and using them is advisable considering development time and software quality.

Unfortunately, the source code for PULSENet is unavailable from the corresponding website, so real-life experimentation with PULSENet is not possible. Also, we could not find a more detailed description of the PULSENet architecture than described in Fairgrieve et al. (2009).

3.3.4 LSM

LSM (Linked Stream Middleware), by Le-Phuoc et al. (2011, 2012) is a middleware framework from the field of Linked Stream Data. It aims to simplify the integration of sensor data with data from other sources by providing semantic descriptions for sensor sources and sensor data streams. As the name suggests, Linked Stream Data has two main properties: the data has mutual relationships (i.e. it is linked), and it is available via streams. The links are visible in Figure 10 in the Linked Data layer, and together form a complex structure of current and historical information. The data (both static and streaming) is collect in the Data Acquisition layer, and transformed to a Linked Data format via 'Wrappers' (which are similar to, e.g., the plugins in PULSENet). Access to the data for applications is provided by a query processor, using a query language for streaming data: CQELS. This query language is not a standard, but developed by LSM's author in Le-Phuoc et al. (2011).

A nice feature of LSM is that it uses W3C's Semantic Sensor Network ontology (described in Compton et al. (2012)), which also yields the relationships between

	Timeframe	Reference	Website
SensorSA	2006-2010	Bleier et al. (2010)	-
SenseWeb	2006-2010	Kansal et al. (2007)	SenseWeb (2007)
PULSENet	2009	Fairgrieve et al. (2009)	PULSENet (2009)
LSM	2011-2012	Le-Phuoc et al. (2011, 2012)	LSM (2011)

Table 2: Overview of four centralised middleware components, which together form a representative set of the centralised components

data points (for instance, they can be linked via their 'location' property). By using an ontology, standard query options become available via SPARQL (from Prud'Hommeaux and Seaborne (2008)). The query language CQELS is based on SPARQL and enables the expressiveness of an ontology for streaming data. A working demo of LSM is available online at `http://lsm.deri.ie/`.

Although the concept of Linked Stream Data is promising, it is relatively new in the context of sensor networks, and therefore untested in practice. More research and experiments are necessary, and will show, e.g., if the CQELS query language is applicable in a broad range of applications.

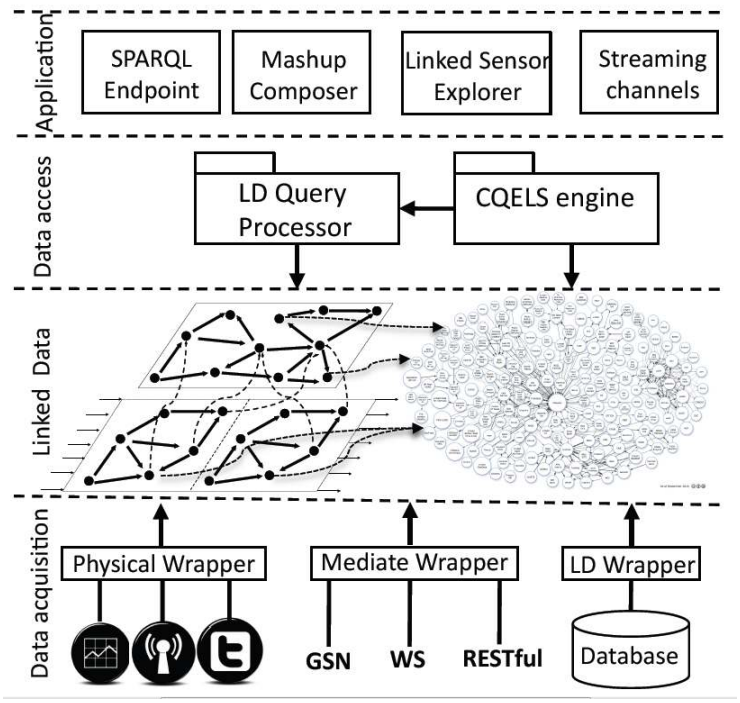

Figure 10: Overview of the elements of LSM, adapted from Le-Phuoc et al. (2011)

3.4 Outlook

The centralised middleware components in Section 3.3 all aim to collect data from sensor networks and to publish it for use by applications. With respect to data publishing, a huge step forward is provided by the SWE-initiative. Their standards are widely used in both industry and the scientific community, and the initiative is still vibrant and continually improving the standards. Readers interested in SWE-implementations can best start with the book describing SensorSA by Bleier et al. (2010) and the article on PULSENet (Fairgrieve et al. (2009)). These references contain general information on the standards, links to reusable components, and best practices.

Contrary to data publishing, data collection is still not based on standards. SWE attempts to solve this issue, but its overhead is often too large for sensors, which in practice seldom communicate in an SWE-based format. This leaves an 'interoperability gap' (identified by Walter and Nash (2009)), which requires conversion from sensor-specific formats to SWE or other (standards-based) formats. The frameworks described above convert collected data using gateways (SenseWeb), plugins (SensorSA, PULSENet), or wrappers (LSM). These are all synonyms for similar elements that bridge the interoperability gap, and at the time of writing there is no consensus yet on what is the best approach.

Looking forward, we expect that bridging the interoperability gap will be the natural next step in the development of sensor network middleware components. In this context the OpenMTC (2013) platform is worth mentioning. This platform is based on standards for Machine-to-Machine communication from the European Telecommunications Standards Institute, and is also suitable for use with sensor networks. In particular, OpenMTC's gateway (which bridges the interoperability gap) supports various popular access technologies (including Zigbee), and is aligned with industry standards. Additionally, OpenMTC can be deployed in the cloud, supports a RESTful architecture, and uses open APIs for data publishing.

Another next step in sensor network middleware research concerns the 'classic' issues of Quality of Service, Privacy, and Security. Much research on these topics already exists (see, e.g., Chen and Varshney (2004), Massaguer et al. (2009), and Ren et al. (2011); Zhou et al. (2008) respectively), but has been hampered by a lack of clear definitions in the context of sensor networks. As applications relying on sensors and sensor networks become ubiquitous, research on Quality of Service, Privacy, and Security will most likely regain momentum.

A third promising research direction is formed by semantic specifications of sensors. Giving a semantic description of a sensor makes it clear what type a sensor is (e.g. a 'Temperature' sensor), what units its measurements are in (e.g. 'Degrees Celsius'), and how these measurements were obtained (e.g., 'Average of 10

measurements in the last 1 second'). Such properties of sensors become particularly important once sensor data is published for use by 3rd party applications, because they must understand exactly what the offered data represents. The papers by Compton et al. (2009, 2012) give a good overview of this topic.

Furthermore, as evidenced by LSM, techniques from the 'Data Streams' domain will become more popular. A large body of literature is available (see, e.g., Aggarwal (2006); Gama and Gaber (2007)) and is ready to be applied. Of particular interest are applications that combine streaming sensor measurements with static data from other sources, because centralised middleware components are in a unique position to collect and process both types of data.

Then, since sensor middleware components provide the ability to share sensor data, we foresee that applications will increasingly combine sensor data with other sources of data. This process is known as Data Fusion, and we think that techniques from this domain will boost the development of a new generation of innovative and intelligent sensor-related applications. Interested readers are referred to Khaleghi et al. (2013), which contains a review of the state-of-the-art in this domain.

Finally, we think it is important to determine a set of clearly defined criteria that allow centralised middleware components to be compared in an objective way. This will help keep track of the state-of-the-art, steer future research, and hopefully also prevent a proliferation of middleware components as we have witnessed in the past.

4 Conclusion

Recently, research in sensor network middleware components has shifted, resulting in a new type of middleware component for sensor networks. These components regard sensor networks as simple data providers, with no capacity to run (part of) the middleware component on the sensor nodes. They are aimed at collecting data from multiple existing sensor networks, and at publishing this data via (web-based) technologies. As such, they form the tie between sensor networks and applications, resulting in the name 'centralised' middleware component.

In this paper we started with a literature review of non-centralised middleware components, and explained several concerns that lead research to shift to centralised middleware components. Then, we gave a high-level outline of centralised middleware components, and provided descriptions of four such middleware components. Finally, we identified several 'next steps' for the development of sensor network middleware components: (1) bridging the interoperability gap;

(2) addressing the classical issues of Quality of Service, Privacy, and Security; (3) providing semantic specification of sensors; (4) employing procedures from the Data Streams domain; (5) applying Data Fusion techniques; (6) the identification of clearly defined criteria for comparing centralised middleware components.

From the discussions in this paper we conclude that research has indeed shifted and resulted in a new type of middleware component. These centralised components are highly relevant, both for industry and for the scientific community. Widely used standards for data publishing are available, and provide mature interfaces that enable the use of sensor networks in real-world applications. With research continuing in the identified directions, we foresee the development of increasingly complete sensor middleware components that form the tie between sensor networks and innovative applications.

Acknowledgements

This work was performed within the RRR Project (2011) (Realisation of Reliable and Secure Residential Sensor Platforms) of the Dutch program *IOP Generieke Communicatie*, number IGC1020, supported by the *Subsidieregeling Sterktes in Innovatie*.

References

Aggarwal, C. C. (2006). *Data Streams: Models and Algorithms (Advances in Database Systems)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Agilla (2009). http://mobilab.cse.wustl.edu/projects/agilla.

Aiello, F. (2009). MAPS: a mobile agent platform for Java Sun SPOTs. In *3rd International Workshop on Agent Technology for Sensor Networks (ATSN-09)*.

Alex, H., M. Kumar, and B. Shirazi (2008). MidFusion: an adaptive middleware for information fusion in sensor network applications. *Inf. Fusion 9*(3), 332–343.

Andreou, P., D. Zeinalipour-Yazti, G. Samaras, and P. Chrysanthis (2011). Towards a network-aware middleware for wireless sensor networks.

Andreou, P., D. Zeinalipour-Yazti, M. Vassiliadou, P. K. Chrysanthis, and G. Samaras (2009). KSpot: effectively monitoring the k most important events in a wireless sensor network. In *IEEE 25th International Conference on Data Engineering, 2009. ICDE '09*, pp. 1503–1506. IEEE.

AWARE (2001). http://grvc.us.es/aware/.

Bai, Q., S. Guru, D. Smith, Q. Liu, and A. Terhorst (2011). A multi-agent view of the sensor web. *Advances in Practical Multi-Agent Systems*, 435–444.

Baker, C. R., K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. D. Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E. S. Leland, T. Pering, and P. K. Wright (2007). Wireless sensor networks for home health care. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*, Volume 2 of *AINAW '07*, Washington, DC, USA, pp. 832–837. IEEE Computer Society.

Bleier, T., B. Bozic, R. Bumerl-Lexa, A. Da Costa, S. Costes, I. Iosifescu, O. Martin, S. Frysinger, D. Havlik, D. Hilbring, P. Jacques, M. Klopfer, S. Kunz, P. Kutschera, M. Lidstone, S. E. Middleton, Z. Roberts, Z. Sabeur, J. Schabauer, S. Schlobinski, T. Shu, I. Simonis, B. Stevenot, T. Usländer, K. Watson, and K. Wittamore (2010). *SANY - an open service architecture for sensor networks*. The SANY Consortium.

Chen, D. and P. K. Varshney (2004). QoS support in wireless sensor networks: A survey. In *Proc. of the 2004 International Conference on Wireless Networks (ICWN 2004), Las Vegas, Nevada, USA*.

Compton, M., P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web* (0).

Compton, M., C. Henson, H. Neuhaus, L. Lefort, and A. Sheth (2009). A survey of the semantic specification of sensors. In *2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference,*.

Cougar (2002). http://www.cs.cornell.edu/bigreddata/cougar.

Fairgrieve, S. M., J. A. Makuch, and S. R. Falke (2009). PULSENet^{TM}: an implementation of sensor web standards. In *Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on*, pp. 64–75. Northrop Grumman, CO.

Fok, C. L., G. C. Roman, and C. Lu (2009). Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems 4*.

Galpin, I., C. Y. Brenninkmeijer, F. Jabeen, A. A. Fernandes, and N. W. Paton (2008). An architecture for query optimization in sensor networks. In *IEEE*

24th International Conference on Data Engineering, 2008. ICDE 2008, pp. 1439–1441. IEEE.

Gama, J. and M. M. Gaber (2007). *Learning from Data Streams: Processing Techniques in Sensor Networks* (1 ed.).

Gibbons, P. B., B. Karp, Y. Ke, S. Nath, and S. Seshan (2003). IrisNet: an architecture for a worldwide sensor web. *IEEE Pervasive Computing 2*(4), 22–33.

Hadim, S. and N. Mohamed (2006a). Middleware for wireless sensor networks: A survey. pp. 1–7. IEEE.

Hadim, S. and N. Mohamed (2006b). Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online 7*.

Heinzelman, W. B., A. L. Murphy, H. S. Carvalho, and M. A. Perillo (2004). Middleware to support sensor network applications. *IEEE Network 18*(1), 6–14.

Henricksen, K. and R. Robinson (2006). A survey of middleware for sensor networks: state-of-the-art and future directions. In *Proceedings of the international workshop on Middleware for sensor networks*, MidSens '06, New York, NY, USA, pp. 60–65. ACM.

Huebscher, m. C. and J. A. McCann (2004). Adaptive middleware for context-aware applications in smart-homes. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, MPAC '04, New York, NY, USA, pp. 111–116. ACM.

IRISNet (2003). http://www.intel-iris.net.

Kansal, A., S. Nath, J. Liu, and F. Zhao (2007). SenseWeb: an infrastructure for shared sensing. *IEEE MultiMedia 14*, 8–13.

Khaleghi, B., A. Khamis, F. O. Karray, and S. N. Razavi (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion 14*(1), 28 – 44.

Knobbe, A., H. Blockeel, A. Koopman, T. Calders, B. Obladen, C. Bosma, H. Galenkamp, E. Koenders, and J. Kok (2010). InfraWatch: data management of large systems for monitoring infrastructural performance advances in intelligent data analysis IX. Volume 6065 of *Lecture Notes in Computer Science*, pp. 91–102. Berlin, Heidelberg: Springer Berlin / Heidelberg.

KSPOT (2009). http://www.cs.ucy.ac.cy/~panic/kspot/.

Le-Phuoc, D., M. Dao-Tran, J. X. Parreira, and M. Hauswirth (2011). A native and adaptive approach for unified processing of linked streams and linked data. In *The Semantic Web–ISWC 2011*, pp. 370–388. Springer.

Le-Phuoc, D., H. Q. Nguyen-Mau, J. X. Parreira, and M. Hauswirth (2012). A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web 16*, 42–51.

Le-Phuoc, D., H. N. Quoc, J. X. Parreira, and M. Hauswirth (2011). The linked sensor middleware–connecting the real world and the semantic web. *Proceedings of the Semantic Web Challenge*.

Levis, P. and D. Culler (2002). Maté: a tiny virtual machine for sensor networks. *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems 37*, 85–95.

Li, S., Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei (2004). Event detection services using data service middleware in distributed sensor networks. *Telecommunication Systems 26*(2), 351–368.

Liu, H., T. Roeder, K. Walsh, R. Barr, and E. G. Sirer (2005). Design and implementation of a single system image operating system for ad hoc networks. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, New York, NY, USA, pp. 149–162. ACM.

Liu, T. and M. Martonosi (2003). Impala: a middleware system for managing autonomic, parallel sensor systems. *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming 38*, 107–118.

LSM (2011). http://lsm.deri.ie/.

Madden, S. R., M. J. Franklin, J. M. Hellerstein, and W. Hong (2005). TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems 30*(1), 122–173.

Magnet (2005). http://www.cs.cornell.edu/people/egs/magnetos/.

MAPS (2009). http://maps.deis.unical.it.

Masri, W. and Z. Mammeri (2007). Middleware for wireless sensor networks: A comparative analysis. *Network and Parallel Computing Workshops, IFIP International Conference on 0*, 349–356.

Massaguer, D., B. Hore, M. Diallo, S. Mehrotra, and N. Venkatasubramanian (2009). Middleware for pervasive spaces: Balancing privacy and utility. In J. Bacon and B. Cooper (Eds.), *Middleware 2009*, Volume 5896 of *Lecture Notes in Computer Science*, pp. 247–267. Springer Berlin / Heidelberg.

Maté (2002). http://www.cs.berkeley.edu/~pal/mate-web.

Mohamed, N. and J. Al-Jaroodi (2011). Service-oriented middleware approaches for wireless sensor networks. In *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, HICSS '11, Washington, DC, USA, pp. 1–9. IEEE Computer Society.

Molla, M. M. and S. I. Ahamed (2006). A survey of middleware for sensor network and challenges. In *Proceedings of the 2006 International Conference Workshops on Parallel Processing*, Washington, DC, USA, pp. 223–228. IEEE Computer Society.

Moodley, D. and I. Simonis (2006). A new architecture for the sensor web: The SWAP framework. In *ISWC 2006: 5th International semantic web conference*.

Mottola, L. and G. Picco (2011). Middleware for wireless sensor networks: an outlook. *Journal of Internet Services and Applications*, 1–9.

Ollero, A., M. Bernard, M. La Civita, L. van Hoesel, P. J. Marron, J. Lepley, and E. de Andres (2007). AWARE: platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned AeRial vehiclEs. In *IEEE International Workshop on Safety, Security and Rescue Robotics, 2007. SSRR 2007*, pp. 1–6. IEEE.

OpenGeospatial-Consortium (2010). http://www.opengeospatial.org/ogc/markets-technologies/swe.

OpenMTC (2013). http://www.open-mtc.org/index.html.

Prud'Hommeaux, E. and A. Seaborne (2008). SPARQL query language for RDF. *W3C recommendation 15*.

PULSENet (2009). http://www.northropgrumman.com/Capabilities/PULSENet/Pages/default.aspx.

Ren, Y., V. Oleshchuk, F. Y. Li, and X. Ge (2011). Security in mobile wireless sensor networks – a survey. *Journal of Communications 6*(2).

Rittle, L. J., V. Vasudevan, N. Narasimhan, and C. Jia (2005). Muse: Middleware for using sensors effectively. In *INSS2005*.

RRR Project (2011). http://www.therrrproject.nl.

SenseWeb (2007). http://research.microsoft.com/en-us/projects/senseweb.

Shen, C. C., C. Srisathapornphat, and C. Jaikaeo (2001). Sensor information networking architecture and applications. *IEEE Personal Communications 8*(4), 52–59.

Shi, K., Z. Deng, and X. Qin (2011). TinyMQ: a content-based Publish/Subscribe middleware for wireless sensor networks. pp. 12–17.

SNEE (2008). http://snee.cs.manchester.ac.uk/welcome.html.

Souto, E., G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner (2005). Mires: a publish/subscribe middleware for sensor networks. *Personal Ubiquitous Comput. 10*(1), 37–44.

Srivastava, M., R. Muntz, and M. Potkonjak (2001). Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, New York, NY, USA, pp. 132–138. ACM.

Sugihara, R. and R. K. Gupta (2008). Programming models for sensor networks: A survey. *ACM Transactions on Sensor Networks 4*(2), 1–29.

Szewczyk, R., A. Mainwaring, J. Polastre, J. Anderson, and D. Culler (2004). An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, New York, NY, USA, pp. 214–226. ACM.

TinyDB (2005). http://telegraph.cs.berkeley.edu/tinydb.

Walter, K. and E. Nash (2009). Coupling wireless sensor networks and the sensor observation service-bridging the interoperability gap. In *12th AGILE International Conference on Geographic Information Science*, pp. 1–9.

Wang, M. M., J. N. Cao, J. Li, and S. K. Dasi (2008). Middleware for wireless sensor networks: A survey. *Journal of Computer Science and Technology 23*(3), 305–326.

Yao, Y. and J. Gehrke (2002). *The Cougar Approach to In-Network Query Processing in Sensor Networks.*

Zhai, L., C. Li, and L. Sun (2011). Research on the message-oriented middleware for wireless sensor networks. *Journal of Computers 6*(5).

Zhou, Y., Y. Fang, and Y. Zhang (2008). Securing wireless sensor networks: a survey. *IEEE Communications Surveys & Tutorials 10*(3), 6–28.

Zigbee (2007). http://www.zigbee.org/.