Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 740268, 12 pages
doi:10.1155/2012/740268

Research Article

A Survey of Wireless Sensor Network Abstraction for Application Development

Teemu Laukkarinen, Jukka Suhonen, and Marko Hännikäinen

Department of Computer Systems, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Correspondence should be addressed to Teemu Laukkarinen, teemu.laukkarinen@tut.fi

Received 25 June 2012; Revised 14 September 2012; Accepted 5 November 2012

Academic Editor: Arne Bröring

Wireless sensor network (WSN) application development is not an easy task due to its resource constrained nature and vast feature rich application space. Several abstractions are harnessed to ease out the difficult WSN application development. In this paper, three levels of abstractions are classified from the existing literature: node, network, and infrastructure abstractions. Since the node and network abstractions are already a well-studied area, the infrastructure abstraction is surveyed in detail to complete knowledge. Technology interoperability, service discovery, metadata support, and processing support are found as basic requirements for infrastructure abstraction. Problematic security and quality of service topics are discussed and the open research questions of ontology, service discovery, distributed processing, and performance metrics are defined. Finally, a distributed middleware design is presented as a possible solution for the key open research question: how to utilize capabilities of the abstracted technologies.

1. Introduction

A wireless sensor network (WSN) consists of even thousands of resource constrained devices (nodes), which form a distributed autonomous network [1]. Energy, computation, communication, and memory constrained WSNs must react to real world phenomena, process and fuse data, and eventually create new knowledge. This knowledge must be presented to an end-user or analyzed to create value added end-user services.

Getting data from a physical sensor to an end-user is not a simple task in WSN application development due to the resource constraints, complex protocols, and multiple levels of technologies involved in the delivery [2–4]. Therefore, different abstraction levels are needed to make application development easier. Three levels can be classified from the existing research work: node, network, and infrastructure abstractions.

The main contributions of this paper are the classification of the three abstraction levels, and a survey of the WSN infrastructure abstractions. The authors of this paper consider node and network abstractions well surveyed and defined area of the WSN research, but find a lack of definition of the infrastructure abstraction. The survey part presents the diverse field of the infrastructure abstraction and gathers a common set of requirements. In addition, we propose open research questions and present our design approach to meet some of those questions.

The paper is constructed as follows. Section 2 presents the three abstraction levels based on the existing publications. Section 3 presents the related work for this survey. Our motivation is given in Section 4. Section 5 presents the survey of infrastructure abstractions and Section 6 collects the properties of the surveyed proposals as requirements for the infrastructure abstraction. Section 7 discusses open research questions, which are derived from the survey. Our design proposals for the open questions are given in Section 8, and the paper is finally concluded in Section 9.

2. WSN Application Abstraction Levels

Three levels of abstraction for WSN applications can be classified from existing WSN research work: node, network, and infrastructure abstractions. Figure 1 positions these levels in the WSN infrastructure and each abstraction is described in detail in the following sections.

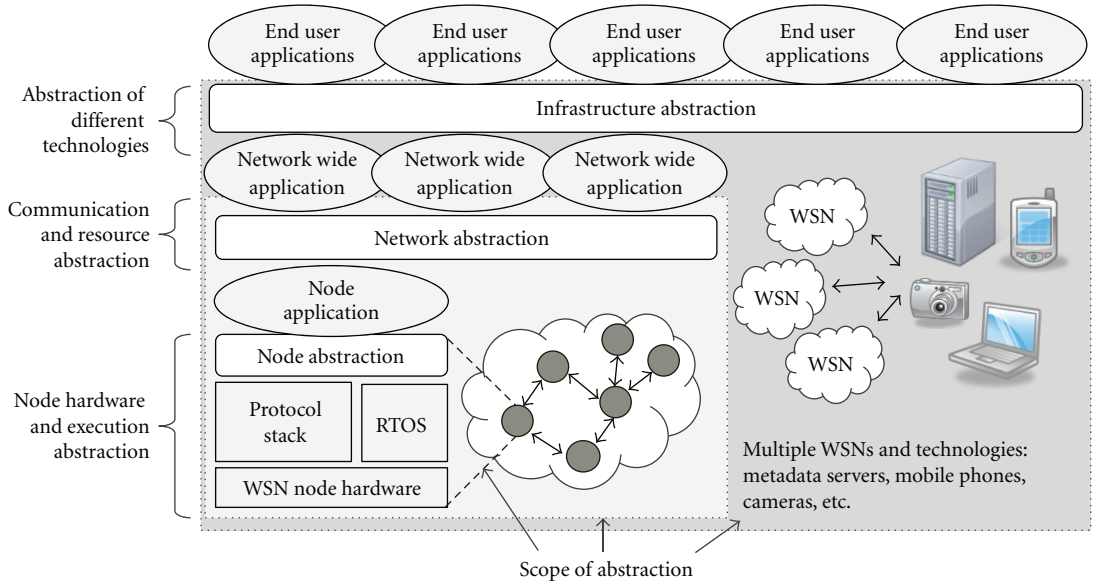

Figure 1: The abstraction levels of WSN application development.

2.1. Node Abstraction. The resource constrained embedded node hardware and communication protocols are abstracted with *node abstraction* that executes applications on each physical node [2, 3]. Embedded operating systems or virtual machines are often used approaches. For example, TinyOS [5], Contiki [6], and Maté [7] are well-known proposals of the node abstraction. The applications perform such actions as reading a sensor, processing measured data, sending data to interested parties when required, and even updating/distributing applications further in the network. The application development is typically conducted with C, NesC [8], or similar programming language, and the development is node specific.

2.2. Network Abstraction. The distributed node network is abstracted from data interested users with *network abstraction*. This level works in cooperation between distributed nodes and provides such services as data access through queries, and data processing in-network through aggregation and fusion [4, 9, 10]. Also, it can provide distribution services for node applications, such as sharing measurements through distributed memory abstraction. TinyDB [11], COUGAR [12], Agilla [13], and TinyLime [14] are often referred proposals of this abstraction level. The node and network abstractions have research history of over decade now, and several surveys have been published of them [2–4, 9, 10, 15, 16].

The network abstraction provides access to the WSN measurement data, but the measurement data itself is rarely sufficient for end-user application [4]: the measurement data can be combined with metadata (e.g., add physical location information, descriptive name, and place the measurement on a map), further processed, combined with data from other technologies, or archived for later study. Since the network abstraction and resource constrained nodes cannot provide all the required data and processing for the end-user application, an *infrastructure abstraction* is used to fulfill the gap. Further, infrastructure abstraction extracts end-user applications from the node and network abstractions. Without any infrastructure abstraction, tailored solutions would be needed for each technology utilized by the application.

2.3. Infrastructure Abstraction. Infrastructure abstraction proposals typically describe the requirements with the same terms as network abstractions, and both are typically referred as WSN middleware. However, the functional units are different: on network abstraction, *heterogeneous nodes in one network* are abstracted behind an interface. On infrastructure abstraction, multiple *heterogeneous sensor networks* are abstracted behind one interface.

The main purpose of the infrastructure abstraction is to separate end-user application from the heterogeneous sensor networks. The infrastructure abstraction is a relatively new research area, which has gained more attention lately. This is due to the wide application space: new end-user applications are tested, deployed, and evaluated using different technologies. An infrastructure abstraction makes end-user application development faster, easier, and WSN technology independent. Currently, there is a lack of surveys on infrastructure abstractions, and the requirements and the design space have not been defined comprehensively. This paper is targeted at contributing in filling this knowledge cap.

3. Related Surveys

Mohamed and Al-Jaroodi [17] survey and discuss WSNs as service oriented middleware (SOM). A SOM provides WSNs

as services for application developers. WSN derived requirements for SOMs are runtime for services, service discovery, heterogeneity abstraction, service configuration, service transparency, automated discovery and service change, interoperability between devices and systems, efficient handling of large volumes of data, security, and support for Quality of Service (QoS) [17]. However, Mohamed and Al-Jaroodi [17] state that very few of their surveyed work address even half of the presented requirements. We consider SOMs as a subset of infrastructure abstractions that mainly abstract network abstraction services from the application developer. For example, metadata or processing are not discussed in [17].

Dafei and Yu [18] survey sensor web proposals. By their definition, sensor web provides access, discovery, and interoperability of sensor services through WWW. This is one subset of infrastructure abstraction of the WSNs. For example, Yin et al. discuss context-awareness (referred to as processing in this paper) as an open issue in sensor webs. We consider that incorporating WWW as a part of the infrastructure abstraction is not a desired solution for every application, since connection to the Internet can be a security risk, for example, in factories and hospitals. Our survey concentrates on wider area of infrastructure abstractions.

Bröring et al. [19] discuss new generation sensor web enablement (SWE) of open geospatial consortium SWE (OGC). They present a layered stack for SWE, which consists of sensor layer, sensor web layer, and application layer. They identify four middleware classes (abstractions in this paper), which may overlap with each other and cross the three stack layers: sensor network management systems, sensor web infrastructure, centralized sensor web portals, and Internet of Things/Web of Things. Compared to our abstraction levels, their sensor network management systems class is the same as the node and the network abstraction levels. The rest of the classes belong to our infrastructure abstraction. Thus, Bröring et al. [19] present a more refined stack of the infrastructure abstraction, but this is done only for their definition of sensor webs. Further on, Bröring et al. [19] concentrate only to OGC SWE. Again, our survey concentrates on general infrastructure abstractions, and OGC SWE is one possible solution in this wide field.

4. Motivation

In our earlier work, we codeveloped WSN pilot applications with partners [20]: variety of interfaces were needed to integrate our WSN technology to the existing systems and tailoring was often needed for end-user application, be it weather web service, working conditions monitoring, factory automation system, or hospital personnel safety. These experiences motivated us to research abstractions for end-user application development to find proposals that remove tailoring and repeating work. We classified the abstractions to the presented three levels and studied their current status from existing surveys.

Although there are several proposals fitting to the infrastructure abstraction, only three related surveys were found, and they concentrated on one specific area of such abstractions. Also, related surveys suggested that infrastructure abstractions do not utilize processing capabilities of network abstractions, since it is not pointed by any of the three nor discussed. These findings motivated us to write this survey.

It is easy to envision several applications where a versatile infrastructure abstraction is a necessity. We describe a hospital use case as a motivating example: a fixed WSN is deployed for indoor positioning in a hospital. A patient can be equipped with a mobile device, such as a tablet, and another WSN that does physiological measurements, for example, patient temperature, electrocardiogram (ECG), and stress level. The tablet sends physiological and positioning data over WLAN or 3G to the infrastructure abstraction.

Patients can move around in the hospital, and they are tracked constantly. If the patient gets a seizure, an alarm of the location is sent to personnel. The ECG is stored for later study by a doctor. Further, the ECG is combined with location and indoor air condition information. These help the doctor to distinguish normal behavior from concerning situations. Finally, the patient can navigate in the hospital buildings with the help of the indoor positioning and the tablet. For example, if the patient is ordered to an X-ray and the tablet can lead him/her to the right place at the right time.

The presented hospital use case sets requirements for infrastructure abstractions: As patients check in and out, the infrastructure level processing must detect new patients and their measurement devices through *service discovery* in order to store, process, and monitor data related to the patients. *Technology interoperability* is an obvious necessity, since different technologies are used as data sources. *Data processing* is needed in several occasions: calculate position, combine position with patient data, detect abnormal ECG behavior, and so forth. Patient information and floor plans are examples of *metadata* required by the application.

5. Survey of the Infrastructure Abstraction

The material for this survey was collected using IEEE Xplorer and Google Scholar search engines. IEEE Xplorer returns 728 hits and Google Scholar approximately 15300 hits with *WSN* and *middleware* search words from year 2006 onwards. Only research papers abstracting WSNs for end-user application were considered for this survey. We selected a set of papers that emphasize the diversity of infrastructure abstractions and present comprehensive set of requirements. Further, we only concentrate on the high level designs and do not consider implementation specific details.

5.1. Open Geospatial Consortium Sensor Web Enablement. Open geospatial consortium (OGC) [21, 22] has proposed sensor web enablement (SWE), which is a set of XML specifications and interfaces for WSNs. OGC SWE has six specifications: sensor model language (SensorML) [23], observations and measurements (O&M) [24], sensor alert service (SAS) [25], sensor observation service (SOS) [26], sensor planning service (SPS) [27], and web notification service (WNS) [28].

SensorML is a set of models and XML schemas, which can be used for discovering services (including SensorML process models), tasking sensor services, processing observations (often measurements) with SensorML Process model, and chaining SensorML processes [21–23]. In addition, SensorML provides uniform data format for the OGC SWE services and contains metadata for processes. SensorML process has inputs, outputs, and parameters; a process without any inputs is a data source (e.g., a measurement device). O&M is a set of models and XML schema that describe the output information model for the sensor web applications [22, 24]. For example, an observation in O&M combines metadata, result, and sampling time in together. Now retired Transducer Markup Language (TML) defined a model for hardware characteristic of sensors and actuators, and a transportation method for sensor data [22, 29].

SOS is an interface to access observations with several parameters, such as temporal and geographical [22, 26]. It utilizes O&M for modeling sensor observations and SensorML for modeling sensors and sensor systems. SAS is an event stream processor and notification system [22, 25]. It continuously monitors data stream and creates events or alerts from pattern matching situations. SPS provides an interface to find out about available assets and possibility to execute tasks in the system [22, 27]. WNS is an interface for delivering notifications (events and alerts) to the user [22, 28]. Communication can be one-way, where notification is just delivered to the user, or two-way, where a response is expected from the user as well.

OGC SWE is an exhaustive but complete set of specifications for infrastructure abstraction. SensorML and O&M together resolve most of the abstraction problems. However, the OGC SWE does not distribute processing to the abstracted technologies; it only uses them as data providers. As mentioned earlier, not all WSN applications can be web services, although parts of the OGC SWE could be utilized without interaction in web: for example, SensorML and O&M could be used just as ontology. OGC SWE could be even considered too complex specification for some use-cases, for example, for resource constrained embedded devices performing simple interactions for actuating air conditioning according to temperature. Further, OGC SWE interoperability is restricted to sensor and actuator devices only and the metadata is mainly for describing physical features of the sensors and their measurements.

5.2. Bouillet et al. Bouillet et al. [30] present a middleware for creating sensor network applications that utilize data from many sources simultaneously. They reason it with application scenarios, which combine sensor data from temperature sensors to cameras and provide processed data for different end-users. The paper describes *Processing Elements* (PEs), which take data in as input streams, process data, and return result as one or more output streams. The middleware system discovers data source streams and connects them to correct PEs. Data sources and PEs are homogenized with Web Ontology Language (OWL) [31] ontologies. Further, PEs can be interconnected to refine processing more and to provide processed data for complex applications. Formal models are given for data sources and PEs. An algorithm is proposed, which can automatically composite applications from connecting PEs and data source streams according to a high level description of the application.

Bouillet et al. [30] proposal achieves technology interoperability with the OWL ontologies, if the connected technology can match the ontology format. The strongest part of the proposal is the processing: data from the sources can be refined and combined into infinity with connectible PEs. They state that in-network processing of data sources is transparent to the system, but their proposal does not seem to utilize this. The service discovery is limited to matching the input of a PE to data sources according to their semantic descriptions with the ontology: the temporal nature of the WSNs is not discussed, and therefore it is a question that can this proposal survive from, for example, disappearing data source. Metadata is not discussed in [30], but apparently metadata providers could be data sources and PEs could get their metadata through those.

5.3. Global Sensor Networks. Global Sensor Network (GSN) abstracts WSNs as a set of virtual sensors, which have one homogenized structure [32]. XML is used to define a virtual sensor, and each virtual sensor has one or multiple inputs consisting of any type of real sensors or other virtual sensors. SQL like language can be used to retrieve and process the data from the abstracted sensors through wrappers. A time model with count- and time-based windowing mechanism is presented to handle different application requirements for the temporal semantics. Metadata is used in these virtual sensors for service discovery and identification.

GSN is a complete proposal, which only lacks a definition of ontology. Instead, the data format is described as a part of the virtual sensor. This is a versatile approach, but the end-user application (or other virtual sensors) must understand all possible structures. If these are not defined globally, the application implementation can be cumbersome: for example, one virtual sensor could produce temperature as an integer and another as an double; the application should then figure out the differences. Further, GSN supports similar processing methods that network abstractions often provide, for example, averaging, but there is no indication that the virtual sensor abstraction would utilize these methods directly.

5.4. SenseWeb and SensorMap. SenseWeb collects wired, wireless and mobile sensors behind one application programming interface (API) to provide technology interoperability [33]. Coordinator forwards application requests for data, homogenizes data format, caches measurements to SenseDB, and provides service and resource discovery. Sensors are connected to the coordinator through tailored gateways, which uniform the access. DataHub gateway is provided as a reference gateway for those sensors that do not want to implement tailored one. A mobile proxy connects mobile sensors to SenseWeb and delivers measurements according to location when sensors are available. Data

transformers are used to process data. Coordinator indexes transformers and provides transformer service discovery for the applications.

SensorMap [34] is an application on top of SenseWeb. It provides tools to illustrate sensor data on a map. It consists of GeoDB, DataHub, Aggregator, and SensorMap GUI. GeoDB holds metadata for sensors, DataHub keeps track of connected sensors, Aggregator combines geographically nearby sensors, and SensorMap GUI presents measurements on a map according to queries. SensorMap allows technology interoperability and service discovery through the GeoDB and DataHub. Data processing is provided by Aggregator and SensorMap GUI. Aggregator resolves nearby and similar sensors from the provided metadata. SensorMap fuses results of queries together with a map to visualize the data. Query results are queried from Aggregator and DataHub.

SenseWeb and SensorMap provide a complete way to aggregate data and present it on web. SenseWeb even realizes node mobility, which is often overlooked by the other proposals. Since SenseWeb and SensorMap work tightly in the Internet and WWW, they raise the question of security and privacy. However, the Internet and WWW approach is not a general solution for the infrastructure abstraction: standalone WSN applications are required as well. SenseWeb and SensorMap only collect web published sensor data together, for example, actuator controlling is not possible.

5.5. Lamses. Lamses is a large-scale middleware proposal for ubiquitous sensor networks [35] with a complex architecture that concentrates on creating context-aware applications from sensor network data. Lamses provides a common interface for accessing abstracted WSNs technologies and managing applications. Lamses consists of a context aware engine, metainformation management, sensor network management, control and query management, and state management. The context-aware engine processes sensor data and events to create context-aware events. The data is handled and stored as XML packets, which are provided as queries for the application programs as well.

Lamses supports only context-aware processing, although it internally integrates data for the context-aware engine. It has a control and query management, which could deliver data or controlling commands to the abstracted sensor networks, but this is not discussed in the paper. The metadata describes only a limited set of attributes of the WSN devices mainly related to the sensing hardware and device identifiers.

5.6. SeNsIM. SeNsIM proposes an architectural and a data model for technology interoperability between sensing technologies [36]. It adapts existing technologies with wrappers and provides a mediator interface for end-user applications. The wrappers connect to the mediator, and the mediator uses an XML query interface for end-user applications. The data is unified by formatting it to an XML. SeNsIM does not provide any metadata or processing support. Casola et al. [36] implicate that processing of the abstracted technologies could be utilized, and that "the state of the sensor can be modified," which indicates that SeNsIM could deliver data to the abstracted technologies. However, these are not discussed clearly in the paper. The wrappers do service discovery on to the abstracted technologies, but it is not clarified how mediator shows this to the applications.

5.7. Smart-M3. Smart-M3 [37] is an interoperability platform for smart spaces. It allows small embedded devices to locally share semantic information. Any ontology can be used with it, and application developer can use ontology through Ontology API. Every device, or *Knowledge Processor* (KP), can store and retrieve information from the *Semantic Information Broker* (SIB). For example, a mobile phone can be used to control local sensor network actuators and home appliances through Smart-M3. KPs can only communicate through a SIB by inserting, querying, or subscribing/publishing the data into it. KPs can be mobile: they can join and leave to a SIB and they can discover the data of other KPs from it.

Smart-M3 is more a technology interoperation communication protocol than a complete infrastructure abstraction, but it does not propose any restrictions for the application development. However, it does require a common ontology for information storing. Without a standardized ontology, Smart-M3 will be only locally usable. For example, Smart-M3 implementation in home applications can use different ontology from Smart-M3 implementation in office applications. Switching between these locations with the same mobile device will require implementation of both ontologies (or use of both Ontology APIs) on that mobile device. Smart-M3 does not provide metadata or processing support. If these are required, they must be implemented on top of Smart-M3.

Smart-M3 has an unique approach compared to other surveyed proposals that each embedded device can directly interoperate through it: there is no need for end-user application or processing run-time in the infrastructure; the actuator device controlling the air condition can read by itself from the SIB what carbon dioxide and temperature sensors have reported and adjust the air according to those values.

6. Infrastructure Abstraction Requirements

The common features of the surveyed work are gathered in Table 1. These requirements are ruled by the end-user applications, and therefore are the requirements for the infrastructure abstraction. It should be noted that the basic paradigm of the infrastructure abstraction is to separate the end-user application from the abstracted technologies: the technologies behind the infrastructure abstraction can change without any need to modify existing end-user applications. The requirements are discussed in detail in the following and the references given in this Section are for example of the discussed topic.

6.1. Technology Interoperability. Heterogeneous WSN technologies must be homogenized for the end-user application. This is one of the major challenges with infrastructure

Table 1: Features of the surveyed infrastructure abstractions.

Abstraction	Data access method	Technology interoperability	Ontology	Service discovery	Metadata	Processing
OGC SWE	Publish/Subscribe, queried streams and historical values, events, and alerts	A set of XML specifications and interfaces for sensor interoperability	Ontology not mentioned, but SensorML describes uniform data structure and units, and O&M describes observation structure	SensorML models and XML schema can be mined for SensorML processes (includes connected sensors and actuators); SPS can be used to test asset availability for a task	SensorML contains process metadata and O&M contain observation metadata	Through SensorML processes and process chains; SAS for creating notifications from pattern matching data
Bouillet et al. [30]	Stream, but the accessing method is not described	Through the input semantics that data sources must adapt	OWL ontologies	Data sources and PE outputs that match to a PE input can be discovered	Not mentioned	PEs can be interconnected; an algorithm to combine PEs into an application
GNS	Query with streams and historical data	Abstracted technologies are described as virtual sensors	No ontology, but the virtual sensor describes the data structure freely	Virtual sensor can be discovered with their metadata descriptions	Each virtual sensor holds metadata descriptions	SQL like processing on the virtual sensors; virtual sensors can be sources to other virtual sensors
SenseWeb	Query with streams and historical data	Uniform API, with data and time abstraction	Not used, but the API homogenizes data format, but the method is not described	For sensors, according to sensor type or location, and for transformers	Not mentioned	Transformers can convert units, aggregate, fuse, and visualize data
SensorMap	Query with streams and historical data; location-based queries possible	Through SenseWeb DataHub	A lack of standard ontology discussed	Services can be discovered from GeoDB meta-data	GeoDB holds meta-data	Aggregate geographically close sensors and display on a map
Lamses	Events for context-awareness and queries for data retrieval	Describes a common interface for attaching WSNs into it	Not mentioned, but uses XML for homogenizing data	Not mentioned	An XML based meta-information database for complementary hardware information	Context-aware event and data processing; internally integrates data for context-aware events
SeNsIM	Query for real-time data and events	Wrappers connect technologies to a mediator and data is unified in an XML	Not mentioned, but an XML model is used for the data unifying	Wrappers discover sensors from abstracted technologies	Not mentioned.	Not mentioned clearly
Smart-M3	Publish/Subscribe and query/insert data	KPs can communicate through a SIB in a smart space without temporal connections	Any ontology can be used and there is an Ontology API for each used ontology	KPs can find different services through a SIB, if the semantics of their data match	Not mentioned	Not supported/possible

abstractions. Technology interoperability consists of three requirements: uniform data access, ontology, and technology feature homogenization.

6.1.1. Uniform Data Access. There must be a method to access the data of the abstracted technologies. Data retrieval can be history exploration [32], continuous stream [30], or event based [21]. On history exploration, the application requests existing data within a time window from the abstraction. On continuous stream, data is delivered continuously to the application. On event based, data is delivered after certain event has occurred either continuously or as a one-time action. Events are often referred as alerts in many publications.

A query or publish/subscribe interface is typically used for accessing the data. On query interface, the application retrieves information with explicit queries [32]. On publish/subscribe interface [21, 37], the application subscribes for interesting data and the publisher delivers the data when it is available. Queries fit well for history exploration and continuous stream retrievals, whereas publish/subscribe fits for continuous stream and event retrievals.

6.1.2. Ontology for Data Format Homogenization. The main task for ontology is to remove heterogeneity between different data producing technologies for the same data type [33]. Ontology describes format, units, and ranges for the data. This simplifies end-user application development, for example, application can rely that temperature measurement is always in the same format and has a unit of Celsius. If data would be requested from different sources without ontology, end-user applications would have to parse and format the data from each technology separately for the final presentation. With a common ontology, the end-user application becomes technology independent: underlying technologies can be changed as long as they can produce data in the ontology format.

6.1.3. Technology Feature Homogenization. WSN technologies have several features, which should be homogenized. The list of such features could be exhaustive and in some case, transparency is a necessity. For example, end-user should know that the node controlling the power outlet of TV is that particular one he/she sees on the wall. Configuration, concept of time [32], and data delivery back to the abstracted technologies are common features that typically need homogenization at the infrastructure abstraction.

Resource constrained WSNs seldom have a real-time clock and/or the data packets are limited in size and cannot deliver the exact time of the observed phenomena. In some applications, it is vital to understand temporal connections between observed phenomenas [32], even if they are observed with different technologies. Therefore, the infrastructure abstraction should homogenize time format. This is an obvious part of the ontology.

WSNs and other supporting technologies are not only data providers, but are also consumers. There must be a method to deliver data to the abstracted technologies. For example, WSN can deliver measurement data, which is then used to control actuated device, such as air conditioning. Typically, this functionality is part of the data access. In addition to control tasks, there can be configuration tasks and data injection that abstraction should support.

6.2. Service Discovery. Not all services are continuously accessible on WSNs due to the possible node mobility, error-prone communication medium, hardware failures, or energy depletion. In addition, new services can be added to the WSN or the supporting technologies. For example, in web services such as SensorMap [34], third parties can add or remove data providers, which then appear as usable data sources for the already existing end-user applications. A service discovery method is needed to find the interesting services [32, 33].

The infrastructure abstraction must solve three main tasks in service discovery. First, networks may have their own service discovery methods, which must be homogenized to ease the end-user application development. Second, the service discovery must be general enough to expand discovery to those technologies/services that wither do not follow the existing WSN paradigm or are completely different from typical measurement services. Third, scalability and transparency are needed. For example, most applications are not interested about the sensor itself, only the measurement type. However, the accuracy of the sensor may be important on some applications and even the sensor product name and the manufacturer information may be needed. If these are used as service discovery parameters, the service discovery must scale to various levels of detail and be transparent when needed.

6.3. Metadata. Plain measurement data is not sufficient for the end-user applications but it must be completed with various kinds of metadata. Descriptions of location, hardware, measurement accuracy, measurement purpose, and so forth, are typical metadata for WSN measurements [35]. For example, location description could be "kitchen" or an identifier for a patient in patient monitoring. Even more complex metadata is a possible requirement. For example, if the measurement is made on a certain geographical location, the end-user might expect to see the measurement illustrated on a map, and the location must be updated, when the measurement is location information, or the node is mobile [34].

Metadata is such a wide topic that an infrastructure abstraction cannot specify all the possible metadata that an end-user application might require. However, the infrastructure abstraction should support at least the metadata that abstracted technologies typically provide. If there is heterogeneity in the available metadata between different abstracted technologies, the infrastructure abstraction should allow adding same metadata for all the abstracted technologies. For example, the infrastructure abstraction could require from the technology adapter that it completes required set of metadata to the measurements. Some of the metadata can be provided as a service as well. For example,

a map repository could be a service of the infrastructure abstraction.

6.4. Processing. A method to create and add processing services to the infrastructure abstraction produce at least two benefits. (i) Aggregated data can be reused between different end-user applications or between different instances of the end-user application. This will reduce data requests to the network, reduce data traffic, and make application development simpler, when aggregation services are needed (e.g., averaging or summing). (ii) Infrastructure has more resources for processing than resource constrained WSNs. Thus, it is possible to do more complex aggregation and processing on the infrastructure-level. The infrastructure abstraction can then provide more elegant services to the end-user application, if needed. For example, infrastructure-level processing could distinguish interesting situations from the measurement data and create an event or new measurement for the end-user applications.

The infrastructure abstraction should allow creating new processing services [30, 32]. In addition to the already presented service access requirements, processing services require an execution environment, a method to describe/create the processing services, and an injection method to add new processing services.

7. Discussion and Open Research Questions

The open research problems yield from the wide application space, which requires infrastructure abstraction to be versatile. Currently, many existing solutions are restricted to one application field or on web-based applications. Although each of them eases the application development, a general purpose infrastructure abstraction would yield easier adaptation, since developers can start to develop on the same abstraction without burdensome pre-examination of the WSN technologies. WSNs need standardizations and a commonly used abstraction in order to finally breakthrough. We consider that there is still research work and problems to be solved.

Security is discussed in only few of the surveyed papers. Mohamed and Al-Jaroodi [17] found this same issue with their survey, and debated about the importance of security in SOMs. They consider security via communication and operation safety and conclude that security is indeed an important topic. However, we consider that at the infrastructure level, communication or operation security is not a research problem as such, since the existing solutions are sufficient. For example, most infrastructure abstractions are traditional server or web software, which can rely on SSL, SSH, or other widely used schemes. The WSN has its own security mechanisms, which are not exposed to the upper levels. More prominent security topics are user access rights (e.g., who can access what data, and from where), processing provisioning (e.g., what is dropped, if the system processing capabilities saturate), or accounting (e.g., who has consumed data, and by how much).

Quality of Service (QoS) is typically a network level problem, and the topic is not discussed in the surveyed papers. Typically, there is no QoS support from the sensor network or it is already abstracted by the network abstraction, for example, TinyDB does not provide clear QoS support for the application. However, the infrastructure abstraction should take care of using the QoS support if such is available. The infrastructure abstraction should classify the application required data and demand better quality for the important data from the abstracted sensor networks to ensure the fastest and the most reliable data delivery.

QoS in the infrastructure abstraction is another open question. Infrastructure may need to receive and process significant amount of data, and still react to alarming conditions quickly. This problem can be seen as implementation or server infrastructure problem, but novel location, data, and/or user aware distribution approaches of the infrastructure may be needed.

For the ontology, the most demanding open task is to develop one that is accepted as a de facto standard. Currently, there is no such proposal. The problem is the versatility: ontology should support all the possible data sources that are required or will be introduced in future. This is not an easy task and will require novel approach or may well be impossible. Defining or modeling the WSN application space as well as possible is the ground work that should be made before such ontology can be produced.

Service discovery is incorporated on most of the surveyed proposals. Often papers only state that some feature of the proposal can be used for service discovery without any further analyzing the usability. Service discovery should gain more attention, since it is not a trivial task. The homogenization, generalization, scalability, and transparency features should be fulfilled.

Processing is supported by many of the surveyed abstractions, but there is lack of support for automated feedback loops in many proposals. WSNs are often used in controlling applications and if the infrastructure-level processing is versatile enough, the processing task can even make the controlling decision. This requires a feedback loop from the processing to the abstracted technologies. This should not be overlooked when designing an infrastructure abstraction.

Many network abstractions support in-network processing. The infrastructure abstraction should utilize these features, since there are clear benefits available from distributing the processing to the network where applicable. Krishnamachari et al. [38] found out with mathematical and simulation analysis of data aggregation models that in-network aggregation can produce 50%–80% energy savings in WSNs. Prakash et al. [39] simulated models with real hardware parameters and found out that simple in-network averaging can reduce energy consumption from 260–270 mJ to 60–70 mJ. Luo et al. [40] evaluated in-network aggregating queries for TinyDB [11] with TAG [41], and SKETCH [42]. They found out that in-network aggregation can reduce energy consumption, improve data quality, and/or reduce end-to-end packet loss rate. Currently, distributing processing in the abstracted networks is not implemented in any of the surveyed proposals.

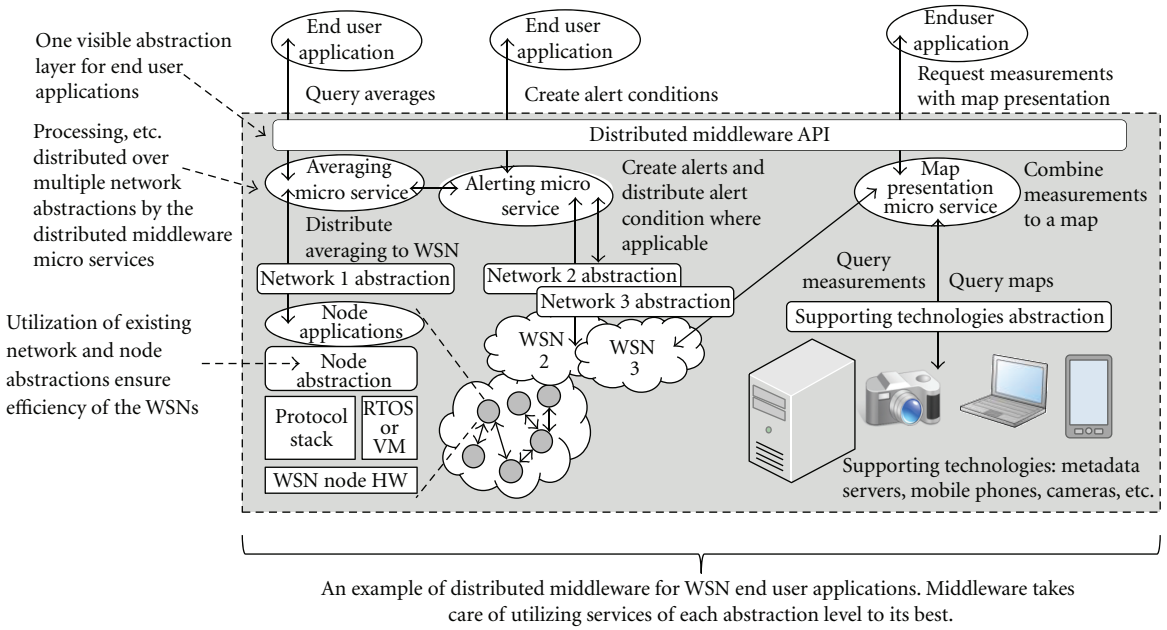

Figure 2: An example use case of distributed middleware design. Averaging micro service distributes the average calculation to the Network 1.

It should be noted that both Prakash et al. [39] and Luo et al. [40] found out energy consumption overhead of in-network processing as well, when the processing is complex or distributed over large areas of sparse network. Therefore, it should be carefully selected where the processing is executed: on the network or on the infrastructure. As far as the authors of this paper know, there is no existing proposal for algorithms or models to distinguish the best place for the processing to take place. This is an obvious open research problem. With careful distribution of the processing to the network, WSN performance can be improved energy and reliability wise.

From a scientific point of view, there are no metrics, benchmarks, or testing facilities for comparing infrastructure abstractions. These can be created, for example, the size of the data structures in different proposals could be compared using typical measurements, or the saturation point of the processing capabilities could be tested. As far as the authors know, such metrics have not been published or tested for infrastructure abstractions.

8. Design Proposals for Open Research Questions

We propose distributed middleware as a common naming convention for WSN abstraction that work on node, network, and infrastructure levels. Figure 2 presents how the distributed middleware fits to the WSN abstraction levels presented in Figure 1. We have concentrated to solve open problems in the service discovery, the distributed processing and the ontology. The key idea is to use small processing services that are simple enough to be distributed through the network abstraction, if it supports such action. If the distribution is not possible, the infrastructure handles the processing.

8.1. Tag-Based Service Discovery. For service discovery we propose categorized tags. For example, *location: Tampere* has a category *location* and a tag *Tampere*. An application can make a query of tags to discover existing and new services. The tag-based discovery is scalable, transparent, and future-proof, since it does not restrict the tags. For example, the service can have a tag that describes the sensor model *sensor: dallas*; *semiconductor*; *dm620*, and the same service can also complete tag search of *measurement*: *temperature*. Further, new technologies can always introduce tags that have not been used before, if they are needed with new applications. GSN has quite similar method to present metadata for discovering the virtual sensors.

Tags can be generated from the existing metadata, and they work as part of the metadata as well. For example, if the location of the sensor is known as coordinates, they can be converted to descriptive tags of the location. Further, tags can be easily generated from the data that sensor provides. If there is a temperature measurement value retrieved from the sensor, it should respond to tag query of *measurement*: *temperature*.

8.2. Distributed Processing. Distributed middleware design requires specifying interface between network and infrastructure abstractions, and creating a processing language that can be analyzed to find processing tasks that can be distributed to the network. We have started to design an XML specification, which describes the processing task called *micro service*. The typical in-network aggregating/processing operations are key components in describing these micro

```
<microservice>
  <descriptions><!- -defines what the service does and describes tags
                    and address, which can be used to find it- -></description>
  <input>
    <source name="src1"><!- -name defines name for this source in the processing part- ->
      <address><!- -this is direct access to the service, thus we already know it- -></address>
    </source>
    <source name="src2">
      <discovery><!- -Search for all outdoor temperature values of Tampere- ->
      <tag>location : tampere</tag>
      <tag>location : outdoor</tag>
      <tag>measurement : temperature</tag>
      </discovery>
    </source>
  </input>
  <output><!- -defines output(s) for this service- -></output>
  <process>
    <variable name="avg">
      <assign>
        <function type="average">
          <parameter>src1</parameter>
          <parameter>src2</parameter>
        </function>
      </assign>
    </variable>
    <if ><compare operator="greater"><valueOf name="avg"/>25.0 </compare>
    <then>
      <action type="increase"><destination name="dst"/>
    </then>
    </if>
    <if><compare operator="less"><valueOf name="avg"/>20.0</compare>
    <then>
      <action type="decrease"><destination name="dst"/>
    </then>
    </if>
  </process>
</microservice>
```

Algorithm 1: An example of the processing section of the micro service XML schema.

services. When the input technology supports used aggregation method, the infrastructure abstraction execution environment distributes processing through the network abstraction.

As an example, a pseudo-XML schema is proposed to describe a new micro service in Algorithm 1. First, the micro service is described. This includes the tags and address, which are used to find the micro service and to connect it. Second, the input sources are defined. The input can be either direct address to a service or a set of discovery tags. Third, the output is defined. The output is either direct connection to another micro service (e.g., for the feedback control loops) or a server that responds to the application queries. Finally, the processing of the micro service is described. The inputs are processed according to the processing schema and the results are driven to the output. The presented processing calculates average between two sources. If these sources can calculate average in-network, the processing is distributed through the network abstraction, for example, if the "src1" and "src2" are in the same network behind TinyDB network abstraction, the averaging could be executed by it. If there is no support or sources are from different networks, the infrastructure processing calculates the average.

8.3. Ontology. Instead of trying to create omnipotent ontology as a de facto standard, world could be squeezed into the mold that ontology provides. For example, world could be seen as measurements, where everything can be a measurement or can be measured. Expanding this vision to nodes and networks allows covering all kind of data in WSN applications with simple ontology structure. For example, [19] discusses integration of social networks as a part of sensor web as an open issue. As an example, Facebook type of social network could be expressed with the sensor centric ontology of WSN OpenAPI [43]: Facebook user's friends are a network, where every node is a friend and their status, birthday, pictures, and so forth, are measurements. Further,

a person can be a network of measurements, such as body temperature, first name, and social security number. A car is a network of sensors, and so forth.

9. Conclusions

This paper classified three levels of abstraction for WSNs: node, network, and infrastructure abstractions. A survey of existing infrastructure abstractions was presented and infrastructure abstraction requirements were defined. An infrastructure abstraction should provide technology interoperability by homogenizing data access, data format, and other technology features. Service discovery, metadata addition, and data processing are also required. As open research questions we discussed security and QoS and defined lack of de facto ontology, lack of focus on service discovery, lack of processing distribution to the network, and lack of benchmark metrics for infrastructure abstractions. We discussed a solution for the ontology problem and proposed design directions for the service discovery and distributed processing. As a future work, we will implement proposed designs and test them with simulations and real world deployments. Also, we will study benchmarking metrics for infrastructure abstractions.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] R. Sugihara and R. K. Gupta, "Programming models for sensor networks: a survey," *ACM Transactions on Sensor Networks*, vol. 4, no. 2, article no. 8, 2008.

[3] L. Mottola and G. P. Picco, "Programming wireless sensor networks: fundamental concepts and state of the art," *ACM Computing Surveys*, vol. 43, no. 3, article no. 19, 2011.

[4] K. Henricksen and R. Robinson, "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)*, pp. 60–65, New York, NY, USA, November 2006.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, 2000.

[6] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki—a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 455–462, Tampa, Fla, USA, November 2004.

[7] P. Levis and D. Culler, "Mate: a tiny virtual machine for sensor networks," *SIGPLAN Notices*, vol. 37, no. 10, pp. 85–95, 2002.

[8] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: a holistic approach to networked embedded systems," *SIGPLAN Notices*, vol. 38, no. 5, pp. 1–11, 2003.

[9] S. Hadim and N. Mohamed, "Middleware: Middleware challenges and approaches for wireless sensor networks," *IEEE Distributed Systems Online*, vol. 7, no. 3, pp. 1–23, 2006.

[10] M. M. Wang, J. N. Cao, J. Li, and S. K. Dasi, "Middleware for wireless sensor networks: a survey," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 305–326, 2008.

[11] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.

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

[13] C. L. Fok, G. C. Roman, and C. Lu, "Mobile agent middleware for sensor networks: an application case study," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 382–387, April 2005.

[14] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco, "TinyLIME: Bridging mobile and sensor networks through middleware," in *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05)*, pp. 61–74, March 2005.

[15] M. M. Molla and S. I. Ahamed, "A survey of middleware for sensor network and challenges," in *Proceedings of the International Conference on Parallel Processing Workshops (ICPP '06)*, pp. 223–228, August 2006.

[16] W. Masri and Z. Mammen, "Middleware for wireless sensor networks: a comparative analysis," in *Proceedings of the IFIP International Conference on Network and Parallel Computing Workshops (NPC '07)*, pp. 349–356, September 2007.

[17] N. Mohamed and J. Al-Jaroodi, "Service-oriented middleware approaches for wireless sensor networks," in *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS-44 '10)*, pp. 1–9, January 2011.

[18] Y. Dafei and F. Yu, "From sensor net to sensor grid: a survey and taxonomy on Sensor Web," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '07)*, pp. 2935–2938, June 2007.

[19] A. Bröring, J. Echterhoff, S. Jirka et al., "New generation sensor web enablement," *Sensors*, vol. 11, no. 3, pp. 2652–2699, 2011.

[20] T. Laukkarinen, J. Suhonen, T. D. Hamalainen, and M. Hannikainen, "Pilot studies of wireless sensor networks: Practical experiences," in *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP '11)*, pp. 1–18, November 2011.

[21] M. Botts, G. Percivall, C. Reed, and J. Davidson, "Ogc sensor web enablement: overview and high level architecture," in *GeoSensor Networks*, S. Nittel, A. Labrinidis, and A. Stefanidis, Eds., vol. 4540 of *Lecture Notes in Computer Science*, pp. 175–190, Springer, Berlin, UK, 2008.

[22] I. Simonis, "Ogc sensor web enablement architecture, version 0.4.0," OGC 06-021r4, OGC Best Practice. Open Geospatial Consortium, 2008.

[23] M. Botts and A. Robin, "Ogc sensor model language (sensorml) implementation specification, version 1.0.0, OpenGIS Implementation Specification," OGC 07-000, Open Geospatial Consortium, 2007.

[24] S. Cox, "Observations and measurements—xml implementation, version 2.0, OpenGIS Implementation standard," OGC 10-025r1, Open Geospatial Consortium, 2011.

[25] I. Simonis, "Ogc sensor alert service candidate implementation specification, version 0.9," OGC 06-028r3, Candidate OpenGIS Implementation Specification Open Geospatial Consortium, 2006.

[26] A. Bröring, C. Stasch, and J. Echterhoff, "Ogc sensor observation service interface standard, version 2.0," OGC 12-006, OpenGIS Implementation Standard. Open Geospatial Consortium, 2012.

[27] I. Simonis and J. Echterhoff, "Ogc sensor planning service implementation standard, version 2.0," OGC 09-000,

OpenGIS Implementation Standard. Open Geospatial Consortium, 2011.

[28] I. Simonis and A. Wytzisk, "Web notification service, version 0.1.0," OGC 03-008r2, OpenGIS Discussion Paper. Open Geospatial Consortium, 2003.

[29] S. Havens, "Ogc transducer markup language (tml) implementation specification, version 1.0.0, retired," OGC 06-010r6, OpenGIS Implementation Specification. Open Geospatial Consortium, 2007.

[30] E. Bouillet, M. Feblowitz, Z. Liu, A. Ranganathan, A. Riabov, and F. Ye, "A semantics-based middleware for utilizing heterogeneous sensor networks," *Distributed Computing in Sensor Systems*, vol. 4549, pp. 174–188, 2007.

[31] W3C, "Owl web ontology language reference," http://www.w3.org/TR/owl-ref, 2012.

[32] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for data processing in large-scale interconnected sensor networks," in *Proceedings of the 8th International Conference on Mobile Data Management (MDM '07)*, pp. 198–205, May 2007.

[33] A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: an infrastructure for shared sensing," *IEEE Multimedia*, vol. 14, no. 4, pp. 8–13, 2007.

[34] S. Nath, J. Liu, and F. Zhao, "Challenges in building a portal for sensors world-wide," in *Proceedings of the 1st Workshop on WorldSensor-Web*, ACM, pp. 3–4, Boulder, 2006.

[35] Y. S. Jeong, E. H. Song, G. B. Chae, M. Hong, and D. S. Park, "Large-scale middleware for ubiquitous sensor networks," *IEEE Intelligent Systems*, vol. 25, no. 2, pp. 48–59, 2010.

[36] V. Casola, A. Gaglione, and A. Mazzeo, "A reference architecture for sensor networks integration and management," *GeoSensor Networks*, vol. 5659, pp. 158–168, 2009.

[37] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proceedings of the 15th IEEE Symposium on Computers and Communications (ISCC '10)*, pp. 1041–1046, June 2010.

[38] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, pp. 575–5578, 2002.

[39] G. L. Prakash, M. Thejaswini, S. H. Manjula, K. R. Venugopal, and L. M. Patnaik, "Energy efficient in-network data processing in sensor networks," *World Academy of Science, Engineering and Technology*, vol. 48, 2008.

[40] Q. Luo, H. Wu, W. Xue, and B. He, "Benchmarking in-network sensor query processing," Tech. Rep., The Hong Kong University of Science and Technology, 2005.

[41] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of the 5th symposium on Operating systems design and implementation, ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[42] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of the 20th International Conference on Data Engineering (ICDE '04)*, pp. 449–460, April 2004.

[43] J. Suhonen, O. Kivela, T. Laukkarinen, and M. Hannikainen, "Unified service access for wireless sensor networks," in *Proceedings of the 3rd International Workshop on Software Engineering for Sensor Network Applications (SESENA '12)*, pp. 49–495, June 2012.