A Systematic Review of Shared Sensor Networks

CLAUDIO M. DE FARIAS, Federal University of Rio de Janeiro
WEI LI, University of Sydney
FLÁVIA C. DELICATO, Federal University of Rio de Janeiro
LUCI PIRMEZ, Federal University of Rio de Janeiro
ALBERT Y. ZOMAYA, University of Sydney
PAULO F. PIRES, Federal University of Rio de Janeiro
JOSÉ N. DE SOUZA, Federal University of Ceará

While Wireless Sensor Networks (WSNs) have been traditionally tasked with single applications, in recent years we have witnessed the emergence of Shared Sensor Networks (SSNs) as integrated cyber-physical system infrastructures for a multitude of applications. Instead of assuming an application-specific network design, SSNs allow the underlying infrastructure to be shared among multiple applications that can potentially belong to different users. On one hand, a potential benefit of such a design approach is to increase the utilization of sensing and communication resources, whenever the underlying network infrastructure covers the same geographic area and the sensor nodes monitor the same physical variables of common interest for different applications. On the other hand, compared with the existing application-specific design, the SSNs approach poses several research challenges with regard to different aspects of WSNs. In this article, we present a systematic literature survey on SSNs. The main goal of the article is to provide the reader with the opportunity to understand what has been done and what remains as open issues in this field, as well as which are the pivotal factors of this evolutionary design and how this kind of design can be exploited by a wide range of WSN applications.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, shared sensor network, virtual sensor network, multiple applications, task scheduling, task allocation, information sharing, network virtualization, middleware, routing, security, applications

---

Dr. Wei Li's work is supported by Faculty of Engineering and IT Early Career Researcher scheme, The University of Sydney. Professor Zomaya's work is supported by an Australian Research Council Discovery Grant (DP130104591). This work was partly supported by the Brazilian funding agencies CNPq and FAPERJ. Flávia C. Delicato, Paulo F. Pires, Jose Neuman de Souza, and Luci Pirmez are CNPq Fellows.
Authors' addresses: C. M. de Farias, F. C. Delicato, L. Pirmez, and P. F. Pires, Av. Athos da Silveira Ramos, 274, Prédio do CCMN - Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21941-916, Brazil; emails: claudiofarias@nce.ufrj.br, fdelicato@gmail.com and flavia.delicato@pq.cnpq.br, luci.pirmez@gmail.com, paulo.pires@pq.cnpq.br and paulo.f.pires@gmail.com; W. Li (corresponding author) and A. Y. Zomaya, Centre for Distributed and High Performance Computing, School of Information Technologies, University of Sydney, NSW 2006, Australia; emails: liwei@it.usyd.edu.au, albert.zomaya@sydney.edu.au; J. N. de Souza, UFC-CC, Computer Science Department, Campus do Pici, Bloco 910, 60440-504 - Fortaleza - Ceara, Brazil; emails: neuman@ufc.br and neuman@ieee.org.

DOI: http://dx.doi.org/10.1145/2851510

ACM Reference Format:
Claudio M. de Farias, Wei Li, Flávia C. Delicato, Luci Pirmez, Albert Y. Zomaya, Paulo F. Pires, and José N. de Souza. 2016. A systematic review of shared sensor networks. ACM Comput. Surv. 48, 4, Article 51 (February 2016), 50 pages.
DOI: http://dx.doi.org/10.1145/2851510

1. INTRODUCTION

Wireless sensor networks (WSNs) are a key technology for the future and will shape the way people interact with the physical world. The sheer number of tiny, cheap and smart devices endowed with multiple on-board sensors, linked together through a wireless network and connected to the Internet will provide unprecedented opportunities for instrumenting and controlling homes, offices, buildings, cities, and the environment. Moreover, they will enable building a myriad of simple to sophisticated applications that provide value-added information and services for the end user, contributing to full realization of emergent paradigms such as the Internet of Things [Atzori et al. 2010] and Smart Spaces [Bose 2009].

Just as with many other advanced technologies, WSN's emergence was leveraged by military applications. The first network that has any resemblance to a modern WSN is the Sound Surveillance System (SOSUS), developed by the United States Military in the 1950s to detect and track Soviet submarines. This network used submerged acoustic sensors—hydrophones—distributed in the Atlantic and Pacific oceans [SiliconLabs 2013]. The system is still in use today, although serving more peaceful functions of monitoring undersea wildlife and volcanic activity [SiliconLabs 2013]. In 1980, the United States Defence Advanced Research Projects Agency (DARPA) launched the Distributed Sensor Network program (DSN) for formally investigating the challenges of implementing a wireless network of distributed sensors. With the birth of DSN and its inclusion in the academy through partner universities such as Carnegie Mellon University and the Massachusetts Institute of Technology Lincoln Labs, WSN technology soon found its way into academic and civil scientific research. Governments and universities eventually started using WSNs in applications such as habitat monitoring, forest fire detection, prevention of natural disasters, weather stations, and structural monitoring. As engineering students made their way into the major technology companies of the time such as IBM and Bell Labs, they began promoting the use of WSNs in heavy industrial applications, such as energy distribution, waste-water treatment, and factory automation.

However, most of the first generation WSN deployments follow an application-specific approach [Tobgay et al. 2011]. This happened because the focus of the research at the time was on solutions for increasing the functionality and performance of WSN systems, while factors such as cost, standardization, resource optimization, and scalability were neglected since they were not essential for the target applications. However, interest from the market in the use of WSNs in a wider range of applications started motivating investigations on how to minimize the deployment and maintenance costs of such networks as well as how to reduce the size of the sensors and the energy cost demanded by them. Such motivations have leveraged the research in the WSN field of the last decades, seeking new materials, transmission technologies, and communication protocols. It can be noticed that the research in WSN is multidisciplinary and requires the efforts of multiple fields of knowledge. Advances in semiconductor, telecommunications, and material science were the first to drive the widespread deployment of WSN on a large scale, catering to novel applications. From the advances in these fields, the focus of network-related research has turned to the design of algorithms and protocols to optimize the resource consumption of the WSN nodes in order to extend the operating lifetime of the system. Therefore, a myriad of studies have been published proposing

routing, location, topology control and medium access control algorithms, and power management mechanisms, all of them with the goal of optimizing network resources. These proposals enabled widening of the range of WSN applications and extending the network operational life, significantly expanding the potential use of these networks.

WSNs built at that time were designed for a single purpose, a single application. Specifically, each network node was programmed to collect and process a specific data type using custom protocols and data formats. Therefore, WSNs were traditionally considered as application-specific systems. The notion of application-specific specifically denotes a system serving a single application and users from the same authority, which is usually the owner of the infrastructure [Yick et al. 2008]. Within such systems, the resources of sensor nodes are all reserved for a single application. Sometimes, the redundancy is deliberately created in the node deployment just for ensuring the successful execution of the application long enough and meeting all the Quality of Service (QoS) requirements. Indeed, application-specific design is a sensible approach for short-term and small-scale deployment, and is also a good match for the "fit-for-purpose" rule [Gomez et al. 2008], which has been widely used in WSN applications development.

However, in many commercial and large-scale WSN applications, Return-On-Investment (ROI) is a crucial factor to ultimately determine the success of those deployments. On one hand, as is always the case in networks composed of resource-constrained devices, longer system lifetime is the desired goal, so that the need for hardware replacement is less frequent. In this sense, the ROI could benefit from longer system lifetime by using application-specific design. On the other hand, the utilization of sensor nodes plays an increasingly important role in WSN deployments. In application-specific design, each application is by default deployed into a dedicated WSN, even if some of the WSNs are used to monitor the same phenomenon from the same area. In other words, the application-specific WSN works somewhat like a tightly coupled system [Siberschatz and Galvin 1993], where hardware and software of the system are not only linked together, but are also dependent upon each other. Moreover, the increasing number (thousands) of sensor nodes in WSNs induces high costs of deployment and maintenance. Obviously, under such circumstance, the inefficient use of sensor nodes becomes counterproductive, and thus leads to low ROI. For these reasons, and also considering the possibility of a single sensor node to serve for different applications, the design trend of WSNs starts shifting from application-specific toward a promising new approach that enables multiple applications running on the same underlying infrastructure. Within such an environment, multiple applications share the resources from the same sensing and communication infrastructure in order to increase the overall system utilization, so that the system cost can be substantially reduced and the ROI is correspondingly increased. This type of WSN is often termed a Shared Sensor Network (SSN) [Efstratiou et al. 2010] or multipurpose wireless sensor network [Steffan et al. 2005]. For convenience of description, this kind of design is referred to as *shared sensor network* in the rest of this article.

The proposal of SSN opens new avenues for gaining better ROI by running multiple applications on the same system so as to utilize the resources of nodes more efficiently. An application can be performed on the whole infrastructure or on a selected subset of the physical network. Ideally, during the execution time, each application behaves just like running on its own dedicated network without causing any interference with others. Moreover, new operational models can be envisioned, where an already-deployed WSN acts as a common infrastructure to serve multiple applications dynamically uploaded by different owners. In such operational models, SSN design shows a clear separation between the ownerships of the applications and the underlying sensor network infrastructure, while no such clear separation is shown in the

application-specific design. This separation not only decouples the ownership of the infrastructure and applications from each other, but also enables the possibilities of nodes deployment, hardware maintenance, and application development to be managed by different authorities. With this feature, new business opportunities arise, such as to treat a deployed WSN as an open system and share its services with third parties on demand as long as the operation of existing applications is not jeopardized. The on-demand model has been widely used in cloud and utility computing. A cloud is a large-scale (ideally unlimited) set of user-friendly virtualized computing resources that can be dynamically reconfigured to serve a variable load, seeking optimum resource utilization [Khan et al. 2015]. Virtualization is a well-established concept for supporting multiple applications, since it allows the abstraction of actual physical computing resources into logical units, enabling their efficient usage by multiple independent users. In this sense, the SSN paradigm is fully aligned with the concept of WSNs virtualization [Santos et al. 2015], which is expected to provide a clean decoupling between services (provided to applications) and the underlying infrastructure.

Although SSN provides a number of benefits, it is still in its early stage and is not as well known as the application-specific design in the field. The main objective of our article is to provide the reader with the opportunity to understand the differences between SSN and the application-specific WSN, what has been done and what still remains open in SSN, as well as which are the pivotal factors of SSN and how SSN design can be applied to emergent WSN applications.

Our study is conducted as a Systematic Literature Review (SLR) based on the methodological framework previously introduced by Kitchenham et al. [2009], which provides a set of well-defined steps carried out in accordance with a predefined protocol. By means of a SLR, one can identify, evaluate, and interpret all available research relevant to a particular research question, or topic area, or phenomenon of interest, in a repeatable and impartial manner [Kitchenham 2004]. Therefore, systematic reviews provide a clear and consistent picture of the research question being posed instead of a number of smaller studies that may give contradictory answers to the same question. Considering the aforementioned goals of this article, we propose the following hypotheses to conduct our research:

- H1: A SSN design improves the utilization of the network nodes, thus contributing to increased ROI of the deployed infrastructure.
- H2: A SSN design promotes a clear separation between the ownerships of the applications and the underlying sensor network infrastructure.
- H3: A SSN design brings new business opportunities in the form of new applications that take advantage of the shared infrastructure.

These hypotheses not only provide us a rough understanding of the value of SSN, but also help us to define the research questions that are used to retrieve key findings from the surveyed literature (as described in Section 2).

The remainder of the article is organized as follows. Section 2 introduces the details of the systematic review procedure that we adopted and modified in our study. Section 3 describes a set of inherent properties of an SSN design and comparatively discusses such properties in the light of the application-specific WSN design. Section 4 describes a taxonomy that we proposed to organize the various levels of supporting mechanisms required in an SSN. Such taxonomy was derived by analyzing architectures and implementations found in the researched literature. From Sections 5 to 7, all the enabling technical details of the taxonomy proposed in Section 4 are thoroughly addressed. Section 8 presents a number of existing applications and ongoing projects to illustrate the benefits and potential of SSN design. At the end of this section, the security issue is

Table I. The Selected Sources Used in the Search Stage of this Review

Source	Type	URL
ACM Digital Library	Digital library	http://dl.acm.org/dl.cfm
IEEE Xplore	Digital library	http://ieeexplore.ieee.org/Xplore/home.jsp
ScienceDirect	Digital library	http://www.sciencedirect.com
SpringerLink	Digital library	http://link.springer.com/
ISI web of knowledge	Digital library	http://apps.webofknowledge.com
Wiley Inter Science	Digital library	http://onlinelibrary.wiley.com/advanced/search
CiteSeer	Digital library	http://citeseerx.ist.psu.edu
EI Engineering Village (includes Compendex, GEOBASE, GeoRef)	Digital library	http://www.engineeringvillage.com
Google Scholar	Search Engine	http://scholar.google.com

also addressed since it is an important application requirement in SSN systems. In Section 9, open issues of SSNs are discussed. Finally, Section 10 concludes our study.

2. SYSTEMATIC REVIEW PROCEDURE

In this section, we will explain the process adopted to perform our systematic review and how the method introduced in Kitchenham et al. [2009] was modified to explore the research field of WSNs so as to better fit the goals and objectives of our review. In addition to the guidelines proposed by Kitchenham et al. [2009], we have also used the systematic review conducted by Lillegraven and Wolden [2010] as a quick reference on how to perform the different stages of a rigorous review process.

2.1. Research Questions

As the first and the most critical step of all steps in a SLR, we turn the focus of our study into the following research questions:

RQ1: What are the major differences on the inherent properties of SSNs as compared to those of the application-specific WSNs?
RQ2: What techniques have been employed to enable running multiple applications on the same sensor network infrastructure?
RQ3: Which existing/ongoing/potential applications can benefit from the use of SSNs?

2.2. Search Process

In the first stage, we searched papers, research reports, books, dissertations, and documents for the review. To determine our search sources we only employed the relevant libraries with significant publications in the field of WSNs. To avoid the omission of valuable publications, Google Scholar was also used as a supplemental tool. The list of sources used in the searches encompasses the most well-known online digital libraries in the field of WSNs as shown in Table I.

After completing the selection of sources, we moved on to defining search terms as well as the procedure for searching papers in the online digital libraries. To create our search strings, we first selected multiple keywords from our previously defined research questions and then formed four groups of search terms as shown in Table II. Each group contains search terms that are either synonyms (different forms of the same word), or terms that have similar or related semantic meaning within the field. Each group aims at retrieving different sets of studies.

We defined one search string to search for studies related to each one of the research questions defined in this review. To search for studies related to RQ1, RQ2, and RQ3, we

Table II. The Search Terms Used in the Online Searches

	Group 1	Group 2	Group 3	Group 4
Term 1	Shared sensor network	User requirement	Layered services/supports	Application
Term 2	Multipurpose sensor network	Design objective	Node-level service/supports	Project
Term 3	Multiowner sensor network	QoS	Network-level service/supports	Testbed
Term 4	Multifunctional sensor network		System-level service/supports	Development
Term 5	Federated sensor network			

combined the terms of Group 1 with 2, Group 1 with 3, and Group 1 with 4, respectively. The effect of the search strings was that all studies, which included at least one of the terms in the first group and at least one term in the remainder groups, were retrieved. The general form of the search string for each RQ is shown as follows:

RQ1: (([G1, T1] ***OR*** [G1, T2] ***OR*** [G1, T3] ***OR*** [G1, T4] ***OR*** [G1, T5]) ***AND*** ([G2, T1] OR [G2, T2] ***OR*** [G2, T3]))
RQ2: (([G1, T1] ***OR*** [G1, T2] ***OR*** [G1, T3] ***OR*** [G1, T4] ***OR*** [G1, T5]) ***AND*** ([G3, T1] ***OR*** [G3, T2] ***OR*** [G3, T3] ***OR*** [G3, T4]))
RQ3: (([G1, T1] ***OR*** [G1, T2] ***OR*** [G1, T3] ***OR*** [G1, T4] ***OR*** [G1, T5]) ***AND*** ([G4, T1] ***OR*** [G4, T2] ***OR*** [G4, T3] ***OR*** [G4, T4]))

The literature searches for each research question were performed manually using the defined search strings listed in Table II within all the selected online digital libraries listed in Table I. We further set the following criteria for the search: (i) works are published in reliable computer science venues (peer-reviewed conference, peer-reviewed journal, or computer science/engineering organization); (ii) works are written in English; (iii) works are published during the period of Jan. 2000–May 2013. After the preceding process is completed, the extracted data was processed to draw out key themes as part of the synthesis stage of the review.

During the conduction of the SLR, at least two researchers independently evaluated whether the retrieved article was likely to be valuable to our targeted research questions. If the work was considered relevant, it would be saved and further processed in the later stages.

2.3. Inclusion Criteria

The purpose of this step is to progressively narrow down the number of articles found in the search stage to an appropriate collection of high-quality articles that is thematically relevant for answering the research questions. The selection criteria presented in this section involve inclusion criteria (stages one, two, and three). Then we used the following three stages for selecting the papers to be analyzed in the rest of the study.

—In stage one, we simply eliminated the articles that were found in the search phase based on the information provided in the abstract. Articles were only kept for further processing if the abstract text explicitly mentions that the main focus of the article is SSNs. For the papers with little information in the abstract, we temporarily kept such papers in the list to be processed in the next stage. It is important to note that, at this stage, we did not consider the quality of the papers.
—In stage two, we concentrated on the content of articles that had passed criteria used in stage one. For the papers including the search strings in the abstract, we briefly examined whether the search strings are the main focus of the papers. Papers in which the search strings represented a minor aspect of their work were eliminated. For those papers that did not provide enough information in the abstract, we further analyzed their content to determine whether they are related to our research

questions or not. Papers whose content matched our research questions were further processed in the next stage.
—In stage three, the remaining articles underwent a quality screening where we eliminated studies that did not meet the following quality criteria:

QC1: Is there a clear statement of the aims of the research?
QC2: Is the proposed system architecture/algorithm/protocol described in the work feasible (can it be applied to a real scenario)?
QC3: Are the simulations/experiments thoroughly analyzed and explained, and the results of tests strongly support the ideas presented in the work?

All the articles were accessed by researchers (Li, Farias, and Delicato) independently by answering yes/partly/no to whether each of the established criteria was met. After the assessment was completed, all disagreements were resolved for each paper and each criterion. Finally, we calculated a sum for each paper by giving one point for each "yes," 0.5 points for each "partly," and zero points for each "no." All papers that scored $P_{QC1} + P_{QC2} + P_{QC3} \geq 2.0$ points were accepted and included in the set of studies used in our review.

The next step of the SLR is the data collection and analyses/synthesis. Similar to Kitchenham et al. [2009], the data extracted from each study were (i) authors' information, including their names, institutions, and countries; (ii) source (journal or conference) and full reference; (iii) summary of the study including the main research questions and proposed answers; (iv) technical aspects related to the SSNs that was addressed in the work, including modeling, proposed solutions, and quality evaluation, and (v) findings and conclusions. For the sake of simplicity and focusing on the main objectives of this work, the detailed description of the data collection and synthesis is given in the Appendix. Interested readers are referred to the Appendix for further details about the conducted SLR. At the end of the search and filtering processes, 57 papers were selected to use later. After finishing the SLR, during the article's revision, another 31 relevant references were found and/or recommended by experts in the field in order to complement the SLR findings, thus resulting in a total of 88 papers, including references from 2014. All results and discussions presented in this article were derived from these 88 studies.

3. THE INHERENT PROPERTIES OF SSNS

In this section, we will discuss the findings of the conducted SLR that provided answers to the defined RQ1. The concept of coupling [Monares et al. 2014] will be used in our discussion. Such a concept provides a clear standpoint to analyze the differences in properties between SSNs and application-specific WSNs. This concept has been widely used to analyze software systems in different areas of Computer Science, such as computer programming [Stevens et al. 1974], distributed computing [Culler et al. 1997], networked embedded systems [Hughes et al. 2009], SOA (Service-Oriented Architecture) [Josuttis 2007], web services [Pautasso and Wilde 2009], and cloud computing [Zhang et al. 2010]. In general, coupling refers to the degree to which system components depend upon each other. In this context, tight and loose coupling are two attributes often used to describe computer systems [Yourdon and Constantine 1979]. Tight coupling is a property owned by two system components that are not only linked together, but are also dependent upon each other. Loose coupling is a property that describes two system components that could interact but are not dependent on each other to work properly. Determining the degree of coupling of a system requires exploring multiple properties at different levels. In the following, several properties of both SSNs and application-specific WSNs are selected from the papers retrieved in our SLR

and compared in order to help understanding of which kind of coupling can be found in both designs. By doing so, we can obtain a better understanding of the specific differences between SSN and application-specific WSN design, thus answering the defined RQ1. Please note that in the following, we have attempted to achieve a high degree of independence among the listed properties while we are enumerating them, but we do not claim that each of them is completely independent of each other.

3.1. Ownership

The ownership property concerns the relationship between the hardware owner and the hardware user. If these two parties are from the same authority or an agreement is established between the two stakeholders before the hardware deployment, the ownership is considered as having a tight coupling. This is because no third party is allowed to get involved in the system operation after the system deployment. Otherwise, the ownership is considered as having a loose coupling.

As mentioned earlier, the application-specific WSN aims to run a single application on top of the infrastructure during the whole system lifetime. In such designs, all the resources are reserved for the one application. Moreover, the application objectives as well as the application composition rarely change, and requests from other applications are unlikely to be served at runtime. These design principles make the application-specific WSN behave like a closed system. Therefore, it is obvious that the application-specific WSN behaves with a tight coupling in the aspect of ownership. This approach limits the possibility of outsourcing the hardware ownership to an external authority, as well as preventing third-party users from invoking services provided by the system. On the contrary, SSN must be built so that multiple applications are able to run on the same infrastructure, sharing the underlying hardware resources. More importantly, applications can be dynamically submitted to the system at runtime, regardless of whether any arrangement is established beforehand [Li et al. 2014a, 2014b]. With such design principles, the deployed system works like an open system offering a high degree of flexibility on hardware management [Hughes et al. 2009], system maintenance [Efstratiou et al. 2010], and application processing [Raicu et al. 2008]. Therefore, SSN can be perceived as having a loose coupling regarding the ownership property.

3.2. Platform Dependency

The property of platform dependency concerns how dependent the application is on the underlying infrastructure. If the design and implementation of application components are based on the prerequisite that they should use the same node hardware, Operating System (OS), and programming language, the resulting system is considered as tightly coupled to a specific technological platform. Otherwise, the system is considered as loosely coupled to the underlying platform.

In application-specific WSN, for reducing the complexity of the application development, the supported hardware is usually selected from the same manufacturer; all the nodes are preferably of the same model, run the same OS and communication protocols, and have identical sensing capabilities. This design principle implies that application-specific WSNs are normally homogeneous. In this sense, when two system components communicate with each other, no complex and expensive bridging solution is required. The application-specific WSN can thus be considered as tightly coupled in this aspect. However, this kind of system is usually bounded to specific requirements and restrictions posed by the platforms in which it is deployed, such as data rate, radio specifications, and radio frequency [Efstratiou 2010]. Moreover, building a WSN with homogenous nodes regarding their sensing capabilities constrains the types of applications that can use such networks. Finally, the strong coupling between applications and the underlying sensor platform prevents any kind of reuse of software artifacts

that constitute the application. If it is necessary to run the same application on a different type of node hardware, all code must be rewritten from scratch to accommodate the new programing language and the primitives provided by the new platform OS [Rodrigues et al. 2013].

On the other hand, SSNs are often heterogeneous, which means the system is composed of multiple types of sensor hardware that are manufactured by different vendors, or even use different OSs, and programming languages. To cope with such heterogeneity, SSNs must provide appropriate mechanisms [Flores-Cortés et al. 2007] that hide the hardware-specific details from end users and make the SSN operate as a homogeneous platform [Jayasumana et al. 2007] for each running application. In this sense, SSNs have loose coupling regarding platform dependency, which means (i) applications are not tied to the underlying sensor platform and (ii) a same infrastructure encompasses nodes with different hardware/software. This loose coupling along with the heterogeneity of SSN regarding the sensor platform promotes the reuse of software artifacts, but interoperability issues arise that are not easy to tackle. The interaction between nodes from different platforms and even between multiple networks, designed with different technologies and protocol stacks, is required in SSN design. Such interoperability issues are often addressed by the insertion of an intermediary software layer that can be implemented following different approaches, as we will discuss later on.

3.3. Code Modularization

Code modularization property concerns the relationship between the code at the application level and the code belonging to the underlying levels (communication protocols and OS). A tight coupling regarding this property occurs when these different pieces of code belonging to different levels are encapsulated in a single image deployed in the sensor node. A loose coupling denotes systems in which the code is developed in a modular fashion, with modules representing the application logic code separated from modules implementing underlying functions.

In general, the development of application-specific WSNs is carried out under the assumption that the particular application is the owner of the physical network and this application is the only one that uses the hardware infrastructure. Therefore, all the application requirements are known a priori and WSN applications are developed as monolithic code installed in the nodes before the network deployment in the target area, thus having a tight coupling regarding code modularization. This strong coupling leverages the customization of all the software layers in the network stack, and mostly aims at providing a high efficiency in terms of energy consumption. However, the design strategies for building code for application-specific WSN are often ad hoc and impose direct interaction of the application with the underlying embedded OS, or even with hardware components of sensor nodes. Although energy efficient, such an approach generates rigid systems that are difficult to maintain, update, and change, besides not promoting any type of reuse of software artifacts.

In the SSN, the presence of two user roles [Leontiadis et al. 2012; Farias et al. 2014] is generally considered, namely, the infrastructure owner and the application owner. The infrastructure owner is assumed to have full control over the hardware infrastructure, while the application owners are assumed to have basic knowledge of the geography of the monitoring area and the functionalities provided by the network. One major requirement [Delicato et al. 2013] for SSN design is to enable newly arrived applications to be performed on the shared hardware infrastructure without interrupting the operation of previously running applications. To address such requirements, it is first necessary to provide solutions allowing, at development time, the clear separation of the application code and the underlying layers of code. Thus, the code to be installed in the sensor nodes can be built as a set of cohesive modules/components,

with well-defined functions, instead of a monolithic piece of code encompassing multiple functions belonging to different abstraction levels. Second, it is necessary to break the tight coupling between the binary code and the physical hardware, built at compile and deployment times. However, both types of coupling (among the different layers of software to be deployed in a node and between the code and the underlying hardware) were adopted in application-specific WSN for the purpose of energy efficiency. Therefore, techniques used to break such couplings would allow the required flexibility and separation of concerns at the expense of lower energy efficiency. Such trade-off between flexibility/reusability/extensibility and energy efficiency is a challenge to be tackled in SSN design. Finally, it is important to isolate different applications in terms of the runtime environment inside the sensor node. Ultimately, code modularization in SSN aims at providing independent execution environments for each independently built application and making it operate in the same way as it would in an application-specific WSN.

3.4. Resource Sharing

The property of resource sharing [Yu et al. 2006] refers to the fact that the resources of a node can be used by different applications so that there is no need to replicate infrastructure to attend multiple applications. The property we address here regards whether or not the available node resources are completely devoted to a single application during the system lifetime. From the infrastructure owner point of view, when node resources are all dedicated to a single application, this system is then perceived as tightly coupled, since the resources and the application are inextricably bound together. Otherwise, if the system resources are expected to serve the needs of multiple applications, such a system is then perceived as loosely coupled regarding resource usage.

In application-specific WSNs, the idea of a single application utilizing the entire system induces that the bounding between the nodes and the application is fixed, that is, all the available resources are reserved for satisfying the needs of a single application. Therefore, the resource allocation can be determined as early as possible in the network operational lifecycle. Unless the underlying hardware is changed at runtime (e.g., due to node movement, node replacement) and the real-time resource lookup is actually needed, the resource allocation can be statically done at compile-time or deployment time [Bhattacharya et al. 2010; Wu et al. 2012]. With these characteristics, the application-specific WSN shows a tight coupling regarding the resource sharing property.

In SSN, resource allocation instead happens at runtime, and sometimes at the latest possible time. This approach addresses a problem that makes the static resource allocation inefficient in SSN. The problem lies in the fact that resource contention could happen when multiple applications are running simultaneously within the same system. SSN allows multiple applications on top of the same infrastructure, with all the available resources opened for applications' dynamic arrival and thus requiring runtime decisions about which application to execute at each time. When applications arrive in the SSN, they will be dynamically allocated to a set of selected sensor nodes for further processing according to different factors, such as the latest node status, user demands, and the priority of the applications [Bhattacharya et al. 2010; Wu et al. 2012; Li et al. 2013]. The allocation of node resources (sensing, computation, and communication) must not only meet the needs of simultaneously running multiple applications without causing interruption, but also comply with policies specified by different stakeholders. This further indicates that all the nodes have a chance to be used by any incoming application. This is a form of loose coupling, because the resource allocation between applications and nodes is no longer fixed and it is instead determined at runtime only when the requested resource becomes available. Dynamic resource allocation thus becomes an important requirement for SSNs, and it makes such network design

show a loose coupling with respect to the resource sharing property [Leontiadis et al. 2012].

3.5. Application Information Sharing

This property concerns whether the network design assumes that the intermediate data produced by sensors can be shared among different applications running within the same system. If no information sharing occurs among applications within a given WSN, this system can be perceived as tightly coupled since the participants of each information exchange are all from the same authority and they act as coalition members of the same application.

The major responsibility of an application-specific WSN is, in any of continuously, periodically, or an event-based fashion, to transmit the collected data back to a device with sufficient computing and storage resource for further processing. Data transmission has been widely recognized as one of the major energy consumers in WSNs. The in-network processing (e.g., data compression, data aggregation) is thus often used to reduce the size of the transmitting data to prolong the system lifetime. This imposes that all nodes between source and destination have to process the intermediate data with the same method or tool for the purpose of data encoding and decoding. By doing so, the messages are handled as a serialization of the same in-network processing technique in the system and are processed by the same method.

SSNs by default enable multiple applications to run simultaneously on the same system. Sharing application information [Le et al. 2009] offers a great potential to save substantial energy in SSN due to the fact that tasks from different applications could simultaneously require the same data (and at the same rate) provided by a single sensor. Motivated by such necessity, information sharing techniques are applied to SSN to achieve better energy conservation. Information sharing mechanisms used in SSN are generally designed as a cross-layer approach [Vijay et al. 2011], aiming to overcome the limitations of the layered protocol architecture by including more available information in a single message. However, the intermediate data of different applications might not adopt the same format and thus sharing information is dependent on the application format, which brings out interoperability issues. In order to enable information with different formats to be shared by applications, a commonly accepted format has to be adopted and all other formats need to be converted to it. This additional step of format conversion is considered as a major overhead that is not required in tightly coupled systems. However, this step is essential for loosely coupled systems, because it not only provides the design flexibility for application developers, but also allows an application to easily cooperate with others at runtime.

4. TAXONOMY OF SSNS

In this section, we start addressing RQ2, which aims at exploring the supporting mechanisms adopted in SSN in order to offer an environment to accommodate multiple applications running on top of a same infrastructure. The main objective of such mechanisms is to provide an execution environment that hides from the running applications the fact that they operate in a shared infrastructure. This allows application developers to build their own applications in the conventional way, assuming that they have full access to the resource of the nodes and hiding from them the inherent heterogeneity of nodes. In order to organize the several levels of SSN supporting mechanisms identified in the surveyed literature, we propose a taxonomy, described in this section.

Before giving the details of the selected literature regarding RQ2, we first examine the emerging challenges associated with an SSN design. These challenges can be classified into two levels: (i) low-level network design and (ii) high-level network design. The low-level network design focuses on abstracting hardware and allowing

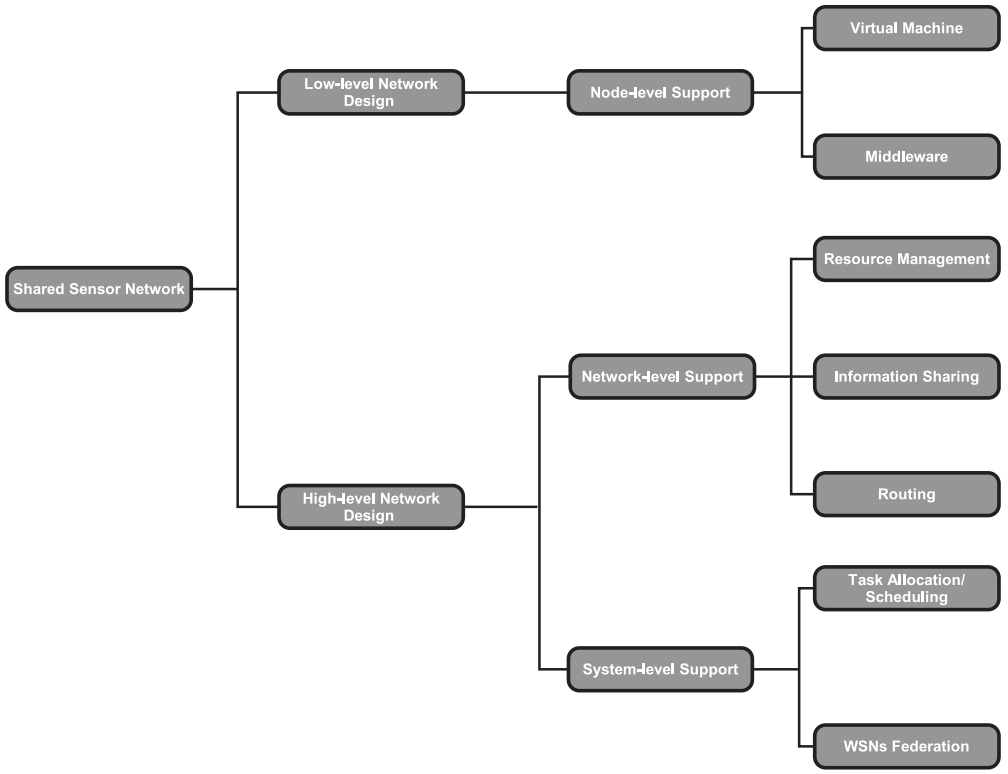

Fig. 1. A taxonomy of shared sensor networks.

flexible control of sensor nodes, while high-level network design focuses on enabling efficient collaboration among sensor nodes to complete the tasks of different applications without imposing additional workload on the application developers. The mechanisms employed in the low-level network design are mainly virtual machine and middleware. Both of them aim at offering a hardware abstraction layer on each sensor, which allows multiple applications to invoke the functions offered by the node without directly interacting with the OS primitives. The high-level network design can be further refined into two sublevels: network-level support and system-level support. The mechanisms employed at the network level are virtualization of sensor networks, information sharing, and routing. They mainly aim at providing the underpinning for handling volatile and dynamic resource requirements of different applications, as well as sharing common information between applications for achieving better energy conservation. Noticeable mechanisms employed at the system level are task allocation and task scheduling. They are both built on the top of the network-level supports and go further by seamlessly integrating the resources from nodes that might be controlled by different authorities as a single logic network. They are also responsible for deciding which nodes are used for performing a given application at runtime without causing any resource contention with the existing applications, while still meeting the various QoS requirements of each application. Besides that, the WSNs federation provides us another view from the system level on how to run multiple applications over WSNs.

Figure 1 depicts our proposed taxonomy for SSN, which was derived by fully analyzing the architectures and the implementations found in the studies retrieved in the SLR. The details of node-level support are discussed in Section 5. The details

of high-level network design, specifically network-level and system-level support, are then discussed in Sections 6 and 7, respectively. All three sections intend to provide the answers to RQ2 listed in Section 2.1 and also to discuss how each presented enabling mechanism supports one or more of the SSN properties described in Section 3.

5. NODE-LEVEL SUPPORT

The research issues of building SSN raised at the node level are mainly focused on two aspects: (i) how to handle node heterogeneity and (ii) how to create flexible execution environments for multiple applications inside a single sensor node. In the following subsections, we discuss how different approaches, namely, virtual machine and middleware, address these two aspects.

5.1. Virtual Machine Approaches

In an SSN, heterogeneous nodes can be deployed in the target area and used by multiple applications. As mentioned previously, one of the challenges at node level is how to hide the heterogeneity of sensor nodes so that application developers may use the same control interface/algorithms on SSN without requiring knowledge of the node-specific OSs, hardware platforms, or network protocols. For addressing this challenge, in-node virtualization [Ertin et al. 2006; Leontiadis et al. 2012] has been suggested as a possible approach to abstract various types of nodes equipped with different OSs and to expose the nodes' functionalities as services, relaying sensing data to upper layers, and providing common interfaces to application developers. Another important reason for using virtual machines in SSN is the small size of programs expressed in virtual machine byte code compared to native code.

The current in-node virtualization implementations in WSN mostly rely on interpreter-based virtual machines [Sugihara and Gupta 2008; Mottola and Picco 2011], which provide a common hardware abstraction of heterogeneous sensor nodes. Such virtual machines rely on the lightweight bytecode interpreters that reside inside each sensor node and offer fine-grained control over the applications execution. However, most designs follow an application-specific design and they are not suitable to be directly used in the SSN context. The motivation behind these implementations being designed as an application-specific virtual machine is the fact that, at an early stage, the general observation was that WSNs were mostly designed for running a specific application (fit for purpose). This is also reflected in the design of development environments like TinyOS [Hill et al. 2000], where the resulting application is a single binary file controlling all hardware components of the sensor node and operating directly on the node.

Another challenge at node level is how to support concurrent applications on a sensor node. To do so it is necessary to store code for applications, create and maintain a dedicated execution space for each application, and separately compile the code for each application to avoid variable sharing across applications. Enabling sensor nodes with dynamic reprogramming capabilities is one of the possible solutions to achieve such goals in SSN. The coexecution of multiple applications on each sensor node depends on whether the OSs allow the dynamic loading and execution of applications during runtime. Some OSs, including Contiki [Xu et al. 2011], Mantis [Rubio et al. 2007], SOS [Costa et al. 2009], and MoteRunner [Caracas et al. 2009] support features such as module and thread handling for dealing with dynamic applications more efficiently. However, many other WSN OSs, including the popular TinyOS, do not fully support such requirements. As an alternative solution to support TinyOS applications execution in SSN, one possibility is to deploy an in-node component that is capable of dynamically loading the binary codes from TinyOS applications at runtime, thus enabling on-demand deployment of multiple applications. This in-node component operates as a stand-alone process within each sensor node so as to fully control access

to the hardware resources of the node. In other words, the component shares access to the hardware through a cross-application hardware abstraction layer residing on each sensor node. The actual hardware access is performed transparently by using the virtual hardware components that are linked to different applications.

Yu et al. presented a significant extension of Maté [Levis and Culler 2002], called Melete [Yu et al. 2006] to support multiple concurrent applications. It enables reliable storage, creating and maintaining a dedicated execution space for each application, and separately compiling the code for each application to avoid variable sharing across applications so as to construct multiple application-specific environments within a single sensor. Considering the fact that, within highly resource-restricted hardware, the execution of multiple applications is interleaved, with a potentially high switching rate, Melete chooses to store the application code images in RAM rather than ROM or external flash, since the target hardware (TelosB node) has large enough RAM space to support up to five concurrent applications. To prevent node resource (e.g., RAM) overuse, Melete introduces a method, called group key, to limit the code images to be sent only to the nodes within the area of interest, instead of activating all available nodes within the network. Similar to the code images, Melete also reserves enough execution space for each application in RAM. To protect execution space, Melete prohibits variable sharing among running applications so that a single application interruption does not affect the execution of others. With the previously listed points, Melete creates a flexible execution environment for multiple applications to be simultaneously executed inside a single sensor node. However, the node heterogeneity issue is not explicitly addressed.

In WSNs, time-critical applications must be completed in a timely manner according to the real-time network condition, rather than statically mapping a set of tasks to a specific physical node at design time. Melete has not considered the real-time requirement in their solutions. For addressing such issues, a new approach at node level, named as Embedded Virtual Machine (EVM) [Pajic et al. 2013] is proposed. Adopting a different approach to earlier works, EVM abstracts several connected physical nodes as a virtual component and this virtual component is represented to users as a single logical entity. By doing so, EVM allows users to use the same network control algorithms without knowing the underlying hardware details and topology changes. Unlike the one-to-one mapping (one virtual sensor to one physical sensor) solution, the goal of EVM is to maintain a set of functional invariants, such as a control law, and parafunctional invariants such as timeliness constraints, fault tolerance, and safety standards across a set of controllers given the spatiotemporal changes in the physical network.

For further addressing the heterogeneity issue of SSN at node level, Java is an attractive alternative for implementing virtual machines on sensor nodes due to its distinctive characteristic of platform independence, compared to other languages, such as C/C++. To enable the Java virtual machine running on severely resource-constrained devices rather than traditional computers, Simon et al. [2006] proposed a split virtual machine architecture, which allows code to be preprocessed and verified on a more powerful host machine, and then only the code necessary to execute the program needs to be sent to the devices. This helps to speed up the test and debugging phase and truly enables the reusability of code since the developed code only needs to be (re-)compiled on every target node regardless of its underlying hardware details.

Darjeeling [Brouwers et al. 2009] is perhaps the first well-known feasible open-source Java compatible virtual machine for most sensor nodes used in WSNs with limited memory resource (less than 10KB of RAM). The bytecode size is significantly reduced by means of an offline tool, called an infuser, which transforms the Java bytecode into a custom bytecode and performs static linking of groups of classes. By doing so, multiple applications within different infusers can run simultaneously. Since it is designed as

a virtual machine, Darjeeling cannot be used to update any native code such as device drivers or the core OS kernel. Instead, it acts as an OS process that runs along with the OS protothreads implementing the network stack communication protocol. The Darjeeling scheduler adopts a round-robin policy in which each thread is allowed to execute for up to a fixed number of bytecodes before releasing the CPU. Whenever a thread is suspended, Darjeeling waits for the next timer interrupt (possibly yielding resources) before resuming the execution of the next running thread.

SimpleRTJ [Haghighi 2013] is a simple real-time Java virtual machine implementation with small memory footprint for small embedded and consumer devices. It contains two main components, the SimpleRTJ virtual machine, which is in charge of interpreting the Java bytecode, and JavaOS, which is used to support multitasking Java applications, as well as to provide other Java features. This is necessary since the simpleRTJ virtual machine only supports single-tasking Java applications at runtime. With the multithreads feature provided by JavaOS, the tasks can be performed concurrently at nodes by the thread switching, and the selected scheduling algorithm (e.g., Round Robin) is used to control the switching operation. Moreover, each instance of SimpleRTJ can be built independently and be configured with various start-up configuration options, which allows the applications to be deployed uniformly in heterogeneous environments without reloading the virtual machine each time.

5.2. Middleware Approaches

As previously discussed, SSN design calls for mechanisms to deal with the heterogeneity of nodes, encompassing devices of multiple types of hardware that are manufactured by different vendors, and supporting different OS and programming languages. In such scenarios, heterogeneous sensor nodes will have to coexist in the same infrastructure and eventually collaborate to meet requirements of multiple applications. Therefore, interoperability becomes an issue.

In addition to the issues of interoperability and integration of heterogeneous devices, the development of applications for WSN has always been a challenge. First of all, WSN programming has traditionally been an error-prone task since it requires programming individual nodes, using low-level abstractions provided by the sensor OS and interfacing with the hardware and network protocols [Rubio et al. 2007]. Second, developers of WSN applications are usually experts in their knowledge field (civil engineering, biology, geology, etc.), not in networks. Therefore, there is a gap between the high-level requirements from WSN applications (the application-level knowledge), and the complexity of operations in the underlying network infrastructure (the network-level knowledge). A potential solution to bridge this gap and to deal with interoperability issues and other challenges posed by SSN application development and execution is to adopt a middleware platform. WSN middleware has recently been adopted to provide node-level abstractions and hide the heterogeneity of sensor nodes concerning node architectures, OS, and programming languages. SSN middleware can also be designed to allow seamless interoperability among various devices fabricated with different kinds of sensors, RFID (Radio Frequency Identification), and actuators in order to enable rapid application development and reliable operation. In this subsection, we summarize a few examples of existing middleware providing such support for SSN at the node level.

Impala [Liu and Martonosi 2003], as the middleware layer of ZebraNet Project [Juang et al. 2002], is one of the early inspiring works in the field, which was implemented on resource-constrained ZebraNet hardware nodes to efficiently handle scheduled operations as well as to ensure reliability and ease of upgrades for long-running applications. Compared to the aforementioned virtual machine solutions, Impala adopts a modular approach. In this approach, the running application is divided into several small

program modules, in a strategy that saves energy by allowing simple and lightweight software updates. This approach uses mobile agents or codes that are injected to the sensor network for collecting local data. The agents can also move from one node to another. One important feature of Impala is thus to provide dynamic programmability and rapid updates to the operation of sensor nodes, that is, the capability of allowing the application code running on nodes to be updated on the fly without service disruption. This feature is important in the context of SSN since it enables multiple applications to concurrently execute by loading code for different incoming applications (meaning applications that are not known at the design time, but instead are to be executed after the network deployment, in a dynamic way). However, issues regarding hardware heterogeneity and QoS are not addressed in this approach.

Agilla [Fok et al. 2009] is modular middleware based on mobile agents that is specifically tailored to address self-adaptive applications in WSNs. The authors focus on the need to provide autonomous behavior for sensor networks and argue that an approach based on software agents is particularly suitable for this purpose. In the mobile agent paradigm, a task-specific executable code traverses the relevant sources to acquire sensing data. Mobile agents can be used to reduce the communication cost in the network, by moving the processing function to the data rather than bringing the data to a central node. Agilla allows agents to move from one node to another using two instructions: clone and move. If an agent is cloned, a copy of it arrives and starts executing at the destination while the original one resumes on the original node. If an agent is moved, it will no longer exist on the original node after it arrives at the destination. Up to four agents can run on a single sensor node at the same time, thus supporting simultaneous execution of multiple applications on the network. Within each node, a tuple space and a neighbor list are also maintained. The tuple space is local and shared by the agents residing on the node. Special instructions are available to allow the agents to remotely access another node's tuple space. The neighbor list of a node contains the addresses of all the nodes located at one-hop distance from it. This enables a decoupled style of communication in which the sender and receiver are not required to agree on a shared memory address, or even coexist, for communication to occur. Using tuple spaces, agents can function autonomously and migrate freely while still being able to communicate among different devices. By exploiting agent mobility, the code related to an application can be installed only at relevant nodes and upon environment changes, it can autonomously migrate to nodes that best suit the application needs.

TinyLime [Curino et al. 2005] is a WSN middleware built as an extension of the Lime middleware [Murphy et al. 2001] for mobile ad hoc networks. It exploits location-dependent data collection and the data aggregation capabilities of sensor nodes to increase the efficiency of WSN operations in comparison to centralized approaches. TinyLime is based on the tuple space approach, which is an implementation of the shared memory paradigm used for distributed computing. It makes sensor data available through a tuple space interface, providing the illusion of shared memory between multiple applications and sensors. The tuple space model has several features that make it suitable for use in wireless network environments in general and in SSN in particular. First, only a small set of operations is needed to manipulate the tuple space, and therefore to enable distributed component interaction [Costa et al. 2009]. Second, in such models the coordination among processes is decoupled both in time and space, that is, tuples can be exchanged among producers and consumers without being simultaneously available, and without mutual knowledge of their identity or location. As previously mentioned, such decoupling is crucial in the SSN design. TinyLime has been implemented entirely in top of the original Lime and deployed using Crossbow MICA2 motes as the target platform.

Another WSN middleware approach that has been gradually extended to the SSN realm is service-oriented middleware [Papazoglou et al. 2007; Delicato et al. 2003]. Service-oriented middleware views the heterogeneous sensor nodes as a distributed system and offers a framework to represent the resources provided by the nodes (mainly sensing, but processing and storage as well) as services, with clear, unified, and accessible interfaces to end users and applications. TinySOA [Avilés-López and García-Macías 2009] is an example of such an approach that directly deploys the lightweight code units on top of the nodes' OS to allow programmers to access nodes from their applications by using a simple service-oriented Application Programming Interface (API) via the language of their choice. It provides mechanisms for WSN infrastructure registry and node discovery. Its architecture includes a gateway component that acts as a bridge between a WSN and external Internet applications. By using abstractions and components provided by TinySOA, web services can be easily employed to allow applications to access WSNs. Similar functionalities are also found in the works described in Delicato et al. [2003, 2005, 2010]. In Delicato et al. [2005], although at the time of publication of the work the prevailing trend of WSN designing was to adopt an application-specific approach, the authors had already glimpsed the need for future sensor networks to be composed of heterogeneous devices and assisting a wide range of applications, for different groups of users. To achieve this goal, they claimed that a new architectural approach was needed, in which the components were loosely coupled, having well-defined interfaces, and application-specific requirements were separated from the data dissemination functions. In order to achieve energy efficiency, applications should be able to change the network behavior dynamically, but these changes should be expressed in a powerful and flexible way, through a common protocol, preferably accepted as a ubiquitous standard. They proposed a service approach for the design of WSN middleware, in which services were defined in terms of the data provided by sensor nodes and the processing functions (for instance, a filtering program) to be executed on those data. Clients should access the sensor network by submitting queries to those services. Services were published and accessed by using Web services technology [Coyle 2002]. Their ultimate goal was to allow the design of networks to be independent from the applications that would use them, thus being fully aligned with the SSN paradigm.

Servilla [Seeger et al. 2014] is another example of service-oriented middleware for WSNs, which aims to use service-oriented computing to enable applications to be both platform-independent and efficient despite executing over a diverse and dynamic set of devices. The applications are decomposed as tasks in Servilla and these tasks act as service consumers. Services are invoked by tasks and perform platform-specific functions like sensing, actuating, and computations. Platform independence is thus achieved by providing a uniform instruction set across all platforms. The proposal also alleviates the need for application developers to tailor their applications for every potential WSN platform.

The work in Delicato et al. [2013] describes MARINE (MiddlewAre for Resource and mIssion oriented sensor Networks), WSN middleware based on a component model and built on two architectural patterns, namely, microkernel [Schmidt et al. 2000] and REST. MARINE was conceived under the basic premise that WSN infrastructures can be shared among different applications in different moments and thus addresses the inherent trade-off of optimizing the network for energy efficiency purposes while meeting requirements of multiple applications. The adoption of a component model along with microkernel pattern in MARINE promotes decoupling and modularization of system functionality in well-defined components, and facilitates the built-in functionality management of the middleware. The high modularity of the software allows identification of the specific functionality required by each application and loading in each node only the components that implement them. The idea is to have as much code as possible

previously stored in the node memory but only a minimum core of code components loaded at node initialization; the remainder components are loaded dynamically whenever the execution context and/or application requirements demand it. With its approach, besides facilitating the execution of new applications on demand, MARINE promotes energy efficiency by loading only the strictly necessary code for each execution context. Other relevant work is described in Hughes et al. [2009]. LooCI (Loosely coupled Component Infrastructure) [Hughes et al. 2009] adopts a loosely coupled, event-based binding model inspired by event-driven programming models, Service-Oriented Architectures, publish-subscribe interaction models, and pluggable networking support. The resulting architecture promotes a loose coupling between software components while facilitating advanced features in networked embedded systems.

In del Cid et al. [2010], the authors consider the inclusion of WSNs in business processes that take advantage of the integration of environmental data collected by sensors to meet multiple purposes of an organization. In such scenarios, the network infrastructure becomes a platform to provide (sensing) services for multiple concurrent distributed applications. The authors claim that in such scenarios, resource sharing is a key issue to reduce the deployment and administrative costs. To tackle this issue, in their middleware proposal they define and explore the concept of consequential contention over the resources of a network node. Such a concept refers to the fact that in WSN, the use of a service, for example, temperature sensing, is specified through a service request and encompasses more than simply accessing to the sensor unit: instead, it also requires processing, storing, and eventually transmitting the sensed data. Therefore, besides the direct contention over the sensing resources of a node, there is a set of consequential contentions for the indirect resources necessary to the completion of a sensing task. In this context, the authors propose a set of shareable, configurable components that minimize the required consequential resources, and a resource planner that effectively reserves these resources. Another key concept explored by the authors is the Quality of Data (QoD) required by applications. QoD refers to application-specific data quality properties [Basaran and Kang 2009], such as reliability (denoting the data accuracy) and resolution (denoting the granularity of data and its temporal and spatial qualities). The set of configurable components compose the core of the proposed distributed WSN management middleware and they may be concurrently used in several compositions, each with varying QoD parameters. By decoupling the components from their configurations (representing different application demands), the middleware allows the best usage of network resources, taking into account the consequential contentions over such resources. The middleware was implemented using Java ME CLDC1.1 configuration on the SunSPOT platform.

5.3. SSN Properties Addressed at the Node Level

By analyzing the discussed proposals in light of the previously mentioned inherent properties of SSN, we notice that the middleware and virtual machine approaches help to promote several of them, thus leveraging the SSN field. Ownership is a property in SSN denoting that the hardware owner and the hardware user can belong to different authorities. Middleware approaches provide runtime support for deployment and execution of services in SSN. The underlying sensor nodes are handled as a set of functionalities (or services, in service-oriented middleware) provided to support the client applications. With the appropriate mechanisms to deploy, load, and execute these services, different users can use the functions provided by the hardware owner in a loosely coupled way.

Platform dependency is another important property of SSN since the different hardware owners could use different sensor platforms in their infrastructure implementations, thus characterizing a high heterogeneity in the network composition.

Middleware and virtual machine approaches can be considered as key enabler technologies to tackle platform dependency in SSN. Middleware platforms provide unified interfaces to the end users, such as the discussed service-oriented middleware solutions [Avilés-López and García-Macías 2009; Delicato et al. 2010; Seeger et al. 2014]. Middleware solutions often not only model the nodes' functionalities and provide them via high-level interfaces to the users, but also address some other platform-related challenges for SSN and its users. One of these challenges is to provide the desired QoS support for different applications from the underlying heterogeneous nodes in a unified way. By hiding the hardware-level and OS-level differences among sensor nodes, data can be efficiently collected meeting application-defined parameters, such as QoD requirements, maximum delay, and others, from the best candidate nodes that can be manufactured from different vendors within the system. In turn, Virtual Machine (VM) proposals such as java-based implementations Darjeeling [Brouwers et al. 2009] and MoteRunner [Caracas et al. 2009] provide a platform-independent solution for SSN so that applications can be migrated freely from one node to another at runtime. The VM in these platforms provides applications with an independence from the heterogeneous underlying hardware by offering an execution environment for each application. There is no need for the applications to directly deal with the hardware. Such a feature ensures the applications are isolated from any hardware requirements/specificities.

Code modularization property is addressed by both middleware and VM approaches through their inherent mechanisms. Component-based middleware platforms, such as Delicato et al. [2013] and Hughes et al. [2009] allow building software as a set of components with well-defined and contractually specified interfaces. The granularity of a component can vary and applications can be constructed as a set of components of grain as thin as desired. Moreover, such middleware platforms provide advanced code modularization features such as support for dynamic reconfiguration, supporting the replacement of individual components within an application at runtime. Agent-based middleware proposals achieve the high modularization of code desired in SSN design considering both development and execution times. For instance, Agilla [Fok et al. 2009] promotes the rapid development of adaptive WSN applications by allowing developers to create and inject mobile agents that can migrate across the nodes performing application-specific tasks. In such approaches, the application code is implemented as a specialized agent and the agent execution environment provides the desired separation of code from different applications at runtime. VM approaches such as Squawk [Simon et al. 2006] favor code modularization by separating OS functionality from application functionality and allowing the replacement of complete application images at runtime without affecting the underlying layers of code.

Regarding the resource sharing property, a primary requirement to support resource sharing in a shared infrastructure is to decouple the applications from the network and expose network resources as a reusable set of functions or services. Middleware approaches naturally provide the software foundation for addressing such requirements. By representing all the basic functionalities provided by a WSN node as configurable and composable units that can be discovered, selected, and grouped together, middleware platforms facilitate the sharing of such functionalities among different applications, which can be built by composing such functional units. Component-based and service-oriented middleware provides the abstractions and mechanisms to compose applications both at development/deployment and at execution time. Mobile agents' middleware also provide the mechanisms to share the node resources, by representing applications as pieces of code that migrate to and execute in the most suitable nodes, where the suitability of a node changes according to the dynamics of network and environment. Finally, middleware platforms can also provide components that control, at

the node level, the concurrent access to the shared resources in order to optimize their usage and increase the overall network lifetime and potential profits.

Lastly, agent-based middleware can contribute to application information sharing in SSN. In addition to the control tasks commonly associated with them, software agents can also be used in sensor networks for information processing tasks. Such tasks include data fusion, inferences, and predictions. The sensors in a sensor network typically collect the raw data and process it to respond to user queries and build a consistent, accurate, and current view of the environment being monitored. To accomplish these tasks, the adoption of a centralized approach, where a single node aggregates raw data and performs all the processing is recognized as inefficient in terms of communication cost. Therefore, decentralized algorithms running with the support of agent-based middleware are promising approaches explored in the literature. In this context, research in the field of Multiagent Systems (MASs) introduced the concept of information agents [Rogers et al. 2008]. Information agents are in charge of locally and autonomously acquiring real-time sensor data to process it and obtain the relevant information required for operational decision making. The processing performed by such agents ranges from simple inferences based on rules to complex distributed data fusion operations. Although the conducted literature review has not recovered articles in this specific context, this is a promising approach to address the application information sharing property at the node level in SSN.

6. NETWORK-LEVEL SUPPORT

The research issues of building SSN at the network level are mainly related to three aspects: (i) how to allow newly developed applications to execute in the shared network infrastructure like a dedicated platform and without causing any disturbance to the already-deployed applications, (ii) how to share common information between applications for achieving better energy conservation, and (iii) how to successfully transmit the data to the desired destination with minimal cost via routing paths across multiple networks.

6.1. Virtual Sensor Network

When newly developed applications are deployed on SSN, they can be performed on any subset of nodes in the network. For some applications, there is the possibility of them being allocated on nodes that are physical neighbors, and thus being able to communicate with others directly. However, in the general case an application deployment can result in a number of clusters that are separated from each other. The intermediate data has to be transferred between these disconnected partitions in a multihop way rather than in a single-hop way. In addition, the communication of an application could be adversely affected by the communication from other existing applications. For addressing such challenges, an overlay mechanism is required to build a dedicated network for each application. Inspired by the work in Kabadayi et al. [2006], several existing works adopt Virtual Sensor Network (VSN) as the overlay mechanism for each application in order to better support concurrent applications execution on SSN. The concept of VSN is derived from network virtualization [Chowdhury and Boutaba 2009], which mainly focuses on how to enable a set of virtual networks to jointly share the resources of the same physical network, as well as to decouple infrastructure from services as in the traditional ISPs. This is one common way to enable VSN by constructing multiple logical WSNs over the same underlying infrastructure. In addition, a single VSN can be constructed over multiple WSNs that belong to different administrative entities. The primary goal of the VSN is to dynamically set up an overlay sensor network for each application so that it can treat the topology of the overlay sensor network as its actual network topology. Consequently, a dedicated platform for each application

is established at network level and application developers do not need to consider the potential disturbances from other applications both at design and execution time. Moreover, the members of a VSN can change over time according to the overall SSN status and incoming applications.

In Jayasumana et al. [2007], two real-world applications, (i) geographically overlapped sensing application and (ii) underground contaminant plume tracking, were chosen as examples to support the arguments of running multiple applications over a WSN. Instead of providing a practical implementation, the authors identified the main issues involved in the implementation of VSN and elaborated the possible functions to be provided in their proposed design. They classified the major functions of VSN into two closely related categories: VSN maintenance and node membership maintenance. The essential node membership maintenance functions include joining VSN and leaving VSN, where both functions are implemented in a distributed way and represented as decisions made by nodes themselves. After joining a VSN, the physical sensor node is identified as a current member and is able to process the assigned tasks of a specific application. VSN maintenance involves not just the existing members of the VSN, but also the nonexisting members. The nonexisting members serve as the supporting nodes for successfully implementing the VSN maintenance functions including broadcasting within the VSN, merging two VSNs, splitting VSNs, and deriving contour of boundaries. Besides that, these supporting nodes also act as relays between members of a VSN or different VSNs.

Yu et al. [2006] proposed a reference implementation of the previously mentioned principles for VSN. In their work, the selected sensor nodes form one logical group, called the associated group, which is dedicated to a single application execution. When the VSN is initialized, all the sensor nodes are set to be members of a default associated group for their whole lifetime. This default group code is stored on each node at all times so that it can be executed under various contexts for different applications. However, the nodes can also serve as members of other constructed groups at runtime. Each sensor node joins or leaves the dynamically formed associated groups in the VSN according to the grouping criteria, for example, the sensing data, the properties of the node, and information the node gathers from its neighbors. Once the VSN is formed, the tasks will be distributed on demand to the associated group members for further processing.

FRESNEL [Efstratiou 2010] is a recently launched SSN project focused on building a large-scale federated sensor network with different applications sharing the resources from the same underlying hardware infrastructure. SenShare [Leontiadis et al. 2012], as a part of the FRESNEL project, also presented an approach of constructing overlay sensor networks that are not only responsible for providing the most suitable members to perform an application, but also isolating the network traffic of this application from the traffic generated by other applications or the supporting mechanisms that are used to maintain the network overlay. To achieve the goal of traffic isolation, SenShare intentionally applies a 6-bytes-long application routing header to each application packet, but the entire network message is still formatted under the IEEE 802.15.4 standard. As mentioned earlier, a member of a VSN can be any sensor node belonging to the network. The member nodes form several clusters and they are normally isolated from each other. For constructing a VSN from these separated clusters as a single connected logic network, virtual links between the clusters need to be established with the help of nodes that are not performing tasks for the target application. Virtual links between clusters are incrementally generated by three consecutive steps, namely, (1) identify the nodes that are on the edges of a connected cluster, (2) discover optimum paths from the nodes selected in the previous step that connect the local cluster to other clusters, and (3) ensure all the clusters are connected together and can access the network's sink. The generation of virtual links utilizes the underlying Collection Tree

routing Protocol (CTP) [Coyle 2002] to discover routes for virtual connections between clusters. There are some other related projects, such as WebDust [Webdust], which also proposed similar solutions to address the same issue at the network level.

We noticed that VSN is a rapidly growing field and it aims to provide users with transparent access to the underlying infrastructure that usually comprises heterogeneous resources under multiple independent administrative entities. We believe in the near future SSN will certainly benefit from more sophisticated VSN developments.

6.2. Information Sharing

The next critical issue relevant to SSN at the network level is to achieve better energy efficiency by sharing information between applications. Energy efficiency is perhaps the most common and most important requirement for almost every WSN implementation. A wide range of energy conservation mechanisms and techniques has been successfully developed and used in WSNs for prolonging network lifetime. However, almost all the existing solutions are targeted on the application-specific WSN and they do not provide extra benefits on energy conservation when they are applied to SSN. In order to further prolong the network lifetime of SSNs, new energy conservation technologies and solutions are required to provide a decent level of energy efficiency by exploiting intrinsic features of a shared infrastructure and without imposing too much burden on application developers.

Task sharing is a kind of information sharing mechanism and is adopted as an energy conservation technique tailored to SSN based on a key observation that maximizing the sharing of common information among multiple applications can significantly reduce energy expenditure. A Multiapplication Task Sharing approach, called MATS, was proposed in Nirmalya et al. [2010] as a prototype implementation in SSN. The basic idea of such a solution is to construct an affiliation set by using a subset of tasks from a specific application while meeting the minimum quality requirement during the execution time. An affiliation set denotes the maximum resources requested (by resource we mean memory, processing, energy, communication capabilities, and other node features) to satisfy the application's minimum quality requirement. By applying this procedure to different applications, multiple affiliation sets are then generated. For each affiliation set, the dependencies with other affiliation sets could exist since they request the same resource at the same time. The total amount of the dependencies of each affiliation set indicates the task sharing across applications. The higher degree of an affiliation means a higher priority to be executed since it satisfies more concurrent applications due to a larger overlap of common tasks.

The work described in Farias et al. [2013] proposed a dynamic graph-based approach to address the task sharing issue in SSN in order to prevent common tasks belonging to different applications being performed multiple times. For example, if two targeted applications have a common task, the task is performed only once by the respective node(s) and the generated data will be shared for both applications. Unlike MATS, the common task identification in this work is based on application, not affiliation set. In addition, the identification is performed at runtime so as to efficiently handle the volatile and dynamic resource requirements from different applications. In this solution, the task allocation mainly relies on the remaining energy of sensor nodes. Similar to MATS, the common task set with more intensive sharing effect among applications is assigned with a higher execution priority. Li et al. proposed a practical task sharing solution in Li et al. [2012] by taking several factors, for example, application arrival time, data accuracy, and data freshness, into account. This task sharing solution is inspired by the idea of session persistence, which is a widely used job dispatching strategy to ensure that the same user is connected to the same server for the duration of the session. Unlike the two proposals introduced earlier, this approach defines a

user-configurable multiobjective function to determine the priorities of the common tasks, where users can dynamically adjust the weight of each parameter according to their own needs. Such an approach can help in maximizing the intersection of execution time of the tasks with the same resource requirement from different applications while ensuring the minimum data accuracy for all these tasks.

Finally, we noted that information fusion can be extended to SSN to achieve the same objective with proper modification. When a sensor node receives useful information from other nodes, the information could be effectively fused with the available local information to reduce the amount of data transmission. Thus, the energy consumption on wireless communication modules of sensor nodes can be significantly reduced, since these modules have been identified as the major energy consumers in a number of studies [Xiong and Svensson 2002; Nakamura et al. 2007; Li et al. 2013]. Information fusion approaches range from simple rules to model-based techniques with different goals on performance and robustness. Simple fusion rules are superior at robustness but suboptimal, while complicated fusion rules could be sensitive to the underlying infrastructure. More importantly, almost all the proposed information fusion algorithms are designed for application-specific WSNs without considering possible information fusion between different applications. Until fairly recently, at the time of conducting this systematic review, the first works to address this key issue are the ones described in Farias et al. [2012, 2014]. In Farias et al. [2012], the authors proposed a first simple rule-based information fusion algorithm called EMAF, tailored for SSNs. EMAF assigns different weights to the applications running on the SSNs and provides user-configurable parameters to indicate the importance of data to an application. However, limited by the features of simple rule-based information fusion algorithms, more sophisticated model-based information algorithms are expected to be migrated into SSNs and further improve the accuracy of information fusion. In Farias et al. [2014], the authors proposed three enhanced model-based fusion algorithms for SSNs by exploring application similarities on data thresholds. These methods are built on top of interval and probabilistic methods presented in de Aquino et al. [2007]. They were named as Enhanced Bayesian Inference (EBI), Enhanced Dempster-Shafer (EDS), and Enhanced Fault Tolerant Interval (EFTI). The authors demonstrated that the data range (range of values that a given application considers in its measurements) and the weight of an application (the relative priority assigned for each application) are crucial to the information fusion process of multiple applications. Since applications can have different degrees of importance, it is quite reasonable to assume that their data have different degrees of importance. If an application that is less relevant but has a large data range has the same degree of importance as the other existing applications, this could lead to an error, that is, a result that does not reflect the actual environment being monitored. By further considering the data semantics (by data semantics we mean a pattern that describes an application and enables interoperability and integration between applications) and the weight of each application, the enhanced fusion algorithms can return a result closer to reality than the traditional fusion algorithms in SSN.

6.3. Routing

The last but not least critical issue for constructing SSNs at the network level is routing. The conventional routing protocols developed for application-specific WSNs usually attempt to achieve routing efficiency by exploiting the application layer semantics [Rocha et al. 2012] and proposing an all-in-one solution that weaves the routing concern with other application layer concerns, such as data-centric and service-centric routing. They also tend to optimize routing performance for a specific communication pattern inspired by a specific class of WSN applications. In an SSN, where multiple applications run within the same network infrastructure, each application presents its own set of

requirements that must be dealt with and exploited by routing protocols in order to guarantee energy efficiency while forwarding data.

One of the possible approaches that tackle routing issues in SSNs is described in Eltarras and Eltoweissy [2010]. In the paper, the authors present Adaptive Multicriteria Routing (AMCR), a novel routing approach for large-scale shared Sensor-Actuator NETworks (SANETs). The proposed routing strategy takes into account the different traffic patterns, communication requirements, and QoS of the multiple applications. Furthermore, AMCR adapts to dynamic changes in these requirements and in the execution context of the applications. The main innovative contribution of AMCR design is that it provides a truly generic routing protocol capable of exploiting the message semantics and adapting itself to the observed application characteristics in order to support the efficient operation of shared platforms.

AMCR adopts a descriptive approach for addressing network nodes, which allows exploration of information that is relevant to applications. The authors argue that addressing schemes based on unique identifiers of nodes is not efficient in terms of scale and resource management in WSN. Such arguments have been advocated by many researchers [Krishnamachari et al. 2002a; Heidemann et al. 2001] since the beginning of research in the WSN field, who advocated the use of data-centric or attribute-based naming schemes. Such schemes identify nodes in a network based on their characteristics such as geographical location, sensing capabilities, and current context (e.g., residual energy). According to several studies [Krishnamachari et al. 2002a, 2002b; Heidemann et al. 2001], data-centric addressing is essential to promote data aggregation functions and in-network processing in general, thereby increasing the system efficiency.

Driven by such assumptions, the proponents of AMCR provide descriptive criteria-based addressing of resources, and allow applications to publish their own resource criteria. AMCR allows destination addresses to be specified as a qualitative reference to node capabilities, administrative settings, and/or application published criteria. It also provides seamless support for unicast/multicast/anycast/broadcast communication patterns. Applications only have to specify a descriptive predicate for the destination that may match one, some, or all nodes.

Another interesting work that takes advantage of application-level information sharing to optimize the routing network is Hefeida et al. [2011]. The authors argue that the interpretation of sensing information greatly depends on the context, and for efficient processing and communication, context awareness plays a major role in WSN. They claim that in the majority of existing WSN protocols, context awareness is exploited in a single dimension and is captured either at the application layer or at the routing layer using a single context parameter. Therefore, in their proposal they go a step further and propose a new WSN context model to efficiently capture multiple context parameters in multiple dimensions (i.e., context from/to different layers of the network stack). They also propose mechanisms to adapt the network behavior according to the current context while simultaneously balancing the network load. Their proposed model considers context parameters reflecting runtime application demands from a node, the current state of the node, as well as state and demands of neighboring nodes (thus leveraging the internodal context sharing). This paper also presents, as a case study incorporating the proposed context model, the Context-Aware Routing-MAC Scheme (CARMS). CARMS is able to perform multihop, multiflow, and multipacket transmissions in a single duty cycle. It considers two context parameters, namely, traffic load and routing information and utilizes the routing layer context of similar packets to be routed in order to set up multihop flows whenever required.

In addition to leveraging the use of application-level information to configure efficient routes, some works retrieved in our systematic review also take into account different

priorities for running applications, allowing higher-priority applications to use faster routes while promoting an application-aware load balancing. The work described in Shah and Szymanski [2012] presents a dynamic multipath routing protocol in which packets from different applications dynamically choose their paths toward the sink node or other nodes by taking into account the price to be paid for taking each path and their ability to pay. A mechanism is proposed in which the prices reflect congestion on routers (nodes along the path) and thus the waiting time for passing the router by a packet. The ability to pay is defined by the application priority and the packet waiting time. These prices increase as usage of shorter routes increases. As a result, low-priority applications tend to avoid paths with high prices. Instead, they go via low price routes, which may be longer but faster to pass by avoiding waiting time for traversing congested routers. This feature enables high-priority traffic to be routed quickly via short paths as their priority (often representing how important they are in the role of supporting the connectivity of other nodes to the network) enables them to pay high prices with little wait. Thus, the proposed approach segregates traffic flows of different applications and decreases congestion and delays for all applications. The authors further show that the proposed dynamic path allocation technique performs well both in normal and emergency situations in which the network is partially damaged.

6.4. SSN Properties Addressed at the Network Level

At the network level, the VSN approach benefits the ownership property providing a clear separation between infrastructure owners and users. In a VSN, sensor nodes that might belong to different authorities are viewed as a single entity at the virtual layer, and the access to their physical components and the provided data/resources is done through their virtual representations. Virtual sensors provide the user with a customized view of the underlying network resources built on demand and according to their requirements.

Regarding the platform dependency property, at the low-level design, in the VSN [Islam et al. 2012] approach, each VSN comprises a subset of the resources of the underlying physical network while it is logically isolated from others. Therefore, the benefit of the network virtualization is to make the underlying technology platform transparent when it comes to connecting distributed components from different applications. Accomplishing such a task is not trivial, especially when the underlying infrastructure is heterogeneous (composed of different types of devices). Regarding the application information sharing property, we found that data fusion techniques can potentially offer more advantages than simply sharing common information among applications. Data fusion techniques range from simple approaches such as the one presented in Li et al. [2012], where data is collected only once and each application uses it separately for resource optimization purposes, to more sophisticated ones such as the proposal presented in Rocha et al. [2012], where application-level information is used to group nodes in semantic clusters. The authors in Farias et al. [2012, 2014] go a step further. In their works, the authors state that the behavior of an application itself can affect the behavior of other coexisting applications. Through the exploration of common data, the authors concluded that the data range and the application weight (representing the relative priority assigned for each application running in an SSN) were extremely important in the fusion process. Since applications have different degrees of importance, it is intuitive to assume that their data also have different degrees of relevance. If an application with low relevance but with a wide data range has the same priority as other running applications, this may lead to an error, meaning the result will not properly mirror the current situation of the environment. In their approach, the authors claim that it is possible not only to share the sensing function, but also to share the decision making process over the data, among multiple applications.

The proposals described in Farias et al. [2012, 2014] were the only ones retrieved in the SLR addressing the application information sharing property in a comprehensive way. We believe this is still an open issue that calls for further work and presents a research opportunity, as discussed in Section 9.

Routing techniques can also benefit the application information sharing property. In systems allowing multiple applications to share intermediate data from multiple sources, nodes that are not the final destination of data process all messages that pass through them. This feature is already exploited in traditional WSN and the intermediary processing inside the nodes characterizes the so-called in-network processing. However, in the presence of multiple applications such features can be leveraged to promote further optimizations. In-network processing implies that the data passing through nodes *en route* to their destination can be explored by protocols from the different layers of the network stack for different purposes. In this context, the few proposals of routing protocols specifically tailored for SSN that were retrieved in our SLR take advantage of application-level information to establish the routes on the network. In addition to leveraging the use of application-level information to configure efficient routes, some retrieved works also take into account different priorities for running applications, allowing higher-priority applications using faster routes while promoting application-aware load balancing.

7. SYSTEM-LEVEL SUPPORT

At the system level, several WSNs belonging to different authorities are treated as a federated network and are regarded as a multiservice system. Two major concerns at this level are (i) how to create and organize the WSN federation and (ii) how to efficiently manage the resource usage of the nodes belonging to the several WSNs.

7.1. WSNs Federation

Federated WSNs are formed when multiple WSNs from different administrative entities are integrated as a single WSN, often through the Internet, and work as a loosely coordinated, large-scale working environment, that supports resource sharing throughout geographically and logically distinct spaces. Although each member WSN is independently administrated and maintained, the federation provides a consolidated view to users in terms of the way to use the resource from any member WSN and to follow the administration policy. The federation concept is not new in WSNs. The first noticeable need of requiring federation on WSNs is to manage the large-scale WSN experimentations over multiple testbeds for multiple reasons, such as hardware cost, space constraints, or lack of expertise. With such federation, experiments can thus simultaneously use resources from multiple testbeds, for applications ranging from regression testing, producer-consumer, parallel processing, to enterprise-edge cooperation. However, building a federated testbed is not trivial since the member testbeds are diverse in their designs, ranging from small-size indoor testbed to large-size outdoor testbed. Furthermore, there is an overlap in the functionality provided by each testbed member, such as multiple tools and techniques are available to describe experiments, multiple configurations to run experiments, and multiple ways to collect data. For solving these issues, building a federated WSN testbed entails a supplemental network and a software infrastructure over the stand-alone testbed facility. In general, there are two core issues to be solved in federated WSN testbeds. The first one is to design a resource management mechanism that is capable of providing efficient and flexible methods for resource description, discovery, and reservation. The second one is to provide a convenient and uniform way of tasking and utilizing the federation resources.

In federated testbeds, resource management aims at providing a framework for transparent resource usage in multiple WSNs that are testbed members. The

framework shall allow users to browse, discover, and reserve resources in a unified way in spite of the heterogeneity of the underlying infrastructures. A minimum requirement is that the exposition of resource information to users and its reservation procedure should be general enough, while specialization is still supported. KanseiGenie [Singh and Bacon 2009] clearly addresses this requirement by using the ORCA control framework that is investigated intensively in the GENI (Global Environment for Network Innovation) project [GENI 2014]. ORCA uses a resource lease contract abstraction in which resource providers advertise or delegate their resources to broker intermediaries, and users request resources from those brokers. More specifically, the ORCA-based resource management system used in KanseiGenie consists of three entities, namely, slice management, site authority, and broker. The slice management interacts with users and gets the resource request, forward it to the broker, and gets the lease for the resources. Once the lease is granted, the slice manager forwards it to the site authority to redeem the lease. The site authority keeps inventory of all the resources that are available for use. It delegates these resources to one or more brokers, which in turn lease the resources to users through the slice manager. The broker keeps track of the resources delegated by various site authorities. It receives the resource request from the slice manager and if the resource is available, it leases the resource to the slice manager. In WISEBED [Chatzigiannakis et al. 2010], resources are named using URN (Uniform Resource Names) in which each node in the federation is uniquely identified by the combination of the member site ID, the sensor network ID, and the node ID. Resource specifications about node capabilities and attributes are posted in a user-readable text. A special client-side implementation, called a federator, federates multiple WSN testbeds, exposing their features as a single virtual testbed server instance. As a result, the resource reservation service provided by each individual WSN testbed is aggregated to form a virtual single service.

The goal of uniform access to the resources of the federation is commonly achieved by web portal and associated services that hold a detailed and unified description of the functionality and characteristics of each testbed member. The Web portal of the KanseiGenie project consists of four components: (i) an Aggregate of Aggregate Manager (AAM) used by users to configure and run experiments, (ii) the Web Service Layer (WSL), (iii) the individual Component Managers (CMs), one for each device type, and (iv) the ORCA site authority module. With such tools, the users can easily automate tasks for resource specification creation, requesting, and subsequent experiment execution and download. Users could also gain more fine-grained control of their deployed experiments by directly programming them against the AAM web interfaces. The WISEBED project offers its APIs (implemented as web services) that range from services to manage the federation, such as creating WISEBED-compliant custom WSN testbeds, to services to run experiments by users. These APIs include SNAA APIs for authentication and authorization, RS APIs for reservation, WSN APIs for conducting, and finally controller APIs for collecting data from nodes.

There are several other implemented federated testbeds, such as SensLab and IoT-lab. However, they are not covered in this article since their focus is not on SSN. Readers are referred to Horneber and Hergenroder [2014] and Kim et al. [2015] for detailed surveys on this topic.

7.2. Task Allocation and Task Scheduling

The primary concern at this level is how to collaboratively complete multiple WSN applications by properly allocating their component tasks to a set of sensor nodes that are potentially located at various WSNs, while still meeting their QoS requirements. For solving this issue, one popular approach adopted in SSN is task allocation or task scheduling [Zomaya 1996], which is a sophisticated technique routinely used in parallel

and distributed computing systems. The most important difference between task allocation and task scheduling is that the former often explicitly addresses the task dependencies and their execution sequences, while task allocation does not address these issues. In addition, one major difference between task scheduling in WSN and traditional scheduling algorithms is that the latter mainly focus on shortening the makespan (the elapsed time for a scheduling scheme to finish a group of tasks that form the critical path of the task graph). Instead, in WSNs the major concern is not only time, but also energy, since each sensor node has limited power supply and the system is expected to run as long as possible after deployment. Considering this vital requirement, the task scheduling problem becomes a multiobjective scheduling problem of which the cost function includes energy consumption, time, and other possible influencing factors. Existing approaches can be classified, according to the network topology, into two categories: task allocation/scheduling on a single WSN and task allocation/scheduling on multiple WSNs.

Within the task allocation/scheduling on a single WSN category, existing studies can be further classified into two subcategories: single-hop nonhierarchical WSN task allocation/scheduling and multihop nonhierarchical WSN task allocation/scheduling. For single-hop nonhierarchical WSNs, the task allocation and task scheduling problem has been addressed and well studied. Yu and Prasanna [Yu and Prasanna 2005] developed an Energy-Balanced Task Allocation (EBTA) algorithm to meet the deadline of a real-time application running on homogeneous sensor nodes connected via multiple wireless channels. This work considers the energy and time costs of both computation and communication activities since wireless communication is a major source of energy dissipation in WSNs. The authors introduce an idea to reduce the potential energy consumption for the communication activities over the same wireless channel. These activities need to be serialized so that runtime communication contentions can be avoided. In EBTA, communications over multiple wireless channels are modeled as additional linear constraints of an Integer Linear Programming (ILP) problem, and a heuristic algorithm with Dynamic Voltage Scaling (DVS) mechanism is presented. However, the authors did not utilize the broadcasting nature of wireless communication in EBTA effectively to save energy and time. Moreover, the adopted multiple wireless channel technique is not widely used in the real-world sensor nodes. The EcoMaps [Yuan et al. 2005] algorithm is proposed for the energy-constraint applications with no deadline requirements. It incorporates channel modeling, task mapping, communication and computation task scheduling, and retasking (sensor failure handling) algorithms. EcoMaps is an application-independent solution as it is based on the high-level application model that describes the task dependencies through Directed Acyclic Graphs (DAGs). A channel model is presented for single-hop wireless networks. Based on this channel model, communication scheduling is integrated as part of EcoMaps. In EcoMaps, communication and computation are jointly scheduled before the application starts being processed. In case of sensor failures, a retasking algorithm is executed to generate an alternative task scheduling scheme on the fly. The retasking algorithm is proved to meet energy consumption constraints if applied on an initially feasible solution. Since EcoMaps has no deadline guarantee for the applications, the authors presented their extension work RT-Maps in Tian et al. [2006], which can guarantee the real-time application deadline with minimum energy consumption. Compared to EcoMaps, the communication and computation tasks of RT-Maps are jointly scheduled in two phases: the Task Mapping and Scheduling Phase and DVS Phase. In the Task Mapping and Scheduling Phase, two low-complexity algorithms, CNPT and Min-Min algorithm, are both extended and implemented. The extended CNPT and Min-Min algorithms incorporate the proposed communication scheduling algorithm with the objective of minimizing energy consumption subject to deadline constraints, and broadcasting

capability is exploited to conserve energy consumption. The Dynamic Voltage Scaling technique is implemented in the DVS phase to further reduce energy consumptions. In Xie and Qin [2008], the authors presented a task scheduling algorithm called BEATA to solve the energy-delay dilemma that exists in heterogeneous WSNs. BEATA aims at minimizing the energy consumption while confining schedule lengths through task allocation. To achieve such a goal, for a given task, BEATA calculates the energy consumption and the finish time for all available nodes within the system. After this step, the BEATA algorithm sorts all of the nodes in the finish time in nondecreasing order. A new concept called Energy-Adaptive Window (EAW) is employed, which allows users to first select a number of nodes from all nodes as candidates for processing a given task. Then, BEATA only chooses the node whose energy saving is the most significant among the first selected candidate nodes. It is easy to understand that enlarging the EAW could result in more energy conservation, but a worse performance in task completion as the makespan time of application will accordingly increase. As such, users can readily adjust trade-offs between energy consumption and schedule length by fine-tuning the size of the EAW according to their actual needs.

The aforementioned techniques concentrated on information processing in a single-hop range, but in the real world WSNs often encompass a large number of nodes that are usually randomly deployed in an area of interest and form arbitrary topologies, thus favoring multihop communication. The single-hop nonhierarchical WSN scheduling algorithms cannot be directly applied to real-world scenarios. Therefore, proposals to tackle the issue of scheduling in multihop nonhierarchical WSNs have attracted researchers' attention recently. In Yuan and Ekici [2007], the authors proposed a multihop in-network processing algorithm called MTMS. In their algorithm, the nature of multihop communication is handled in two steps. First, they extended the representation of tasks from DAG to a Hyper-DAG by replacing a communication edge as a vertex and connecting this new vertex to the vertexes that it originally starts from and ends at. The cost of the communication vertex is equal to the communication load in the DAG. Second, all the sensors are assumed to connect to a virtual communication controller. In a specific time slot, the algorithm running on the controller determines whether or not processing a communication task in the system will cause interference with other communication tasks that are simultaneously being performed. If so, the algorithm will seek another time slot to process, otherwise, it allocates this time slot for the communication task to process. However, their communication scheduling algorithm only considers one-hop communication collision and some kinds of hidden communication collision (e.g., secondary interference) may occur. Furthermore, their model does not suitably address the communication concurrency; in most cases, the wireless communications in the selected sensors are performed sequentially.

All of the preceding works assume that the application is only performed inside a single WSN; they are not able to explore the potential cooperation among multiple WSNs in a SSNs design approach. In order to better utilize the resource of multiple WSNs and extend their lifetime, researchers proposed several task allocation/scheduling solutions tailored to the SSN scenario. In Bhattacharya et al. [2010], the authors presented Utility-based Multiapplication Allocation and Deployment Environment (UMADE), which is a task allocation system for distributing various applications in multiple WSNs based on QoM (Quality of Monitoring). QoM is a distributed quality metric for monitoring a physical phenomenon of interest that is based on the accuracy of measurements. Thus, the value of QoM depends on the monitoring performed by all nodes allocated to an application. Unlike traditional approaches that usually allocate nodes in the networks to a single application according to metrics such as the amount of resources used, delay, processing, and power consumption, UMADE dynamically allocates nodes to multiple applications according to the application's QoM. An inherent

property of an application is that sensing data belonging to sensors that are allocated to the same application are naturally correlated. As a consequence, the contribution of a sensor node for an application QoM is dependent on other sensor nodes allocated for the same application.

In Xu et al. [2010], the authors propose a greedy algorithm to schedule applications onto an SSN. Similarly to the work in Bhattacharya et al. [2010], the algorithm also performs the task allocation taking into account the QoM of the applications. The applications' QoM depends on the node to which the application was allocated. Therefore, it is important that the allocation algorithm seeks to optimize the allocation among multiple applications of an SSN so as to maximize the QoM. The work in Xu et al. [2010] uses the property called QoM submodularity. QoM submodularity regards the fact that the readings from the sensors of different nodes are often correlated. For example, once the temperature readings from different nodes in the same room are correlated with each other, the assignment of a new node in the room to perform monitoring of temperature does not produce a considerable QoM improvement. The work of Wu et al. [2012] is an extension of Xu et al. [2010] and presents a distributed game-theoretic approach to application allocation in SSNs. The authors transform the optimal application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring nodes. The authors prove that the network can converge to a pure strategy Nash equilibrium with an approximation bound of 1/2. The authors validated their results through simulations based on three real-world datasets (Intel dataset, DARPA dataset, and BWSN dataset) to demonstrate that their algorithm is competitive against a state-of-the-art centralized algorithm in terms of QoM. However, all of these approaches for task allocation are solutions that do not address the latency issue, which often occurs in practical WSN applications. To further address this issue, Li et al. [2013] proposed a heuristic called TPTS for enabling the task scheduling over multiple WSNs. TPTS is designed to find out a scheduling scheme that minimizes the overall energy consumption and balances the workload of the system while meeting the application deadline. However, TPTS considers that all sensor nodes contain the same sensing devices, which means each WSN within the system is homogeneous in terms of its capability. In the SSNs environment, the sensor nodes may contain many different types of sensing devices such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic, and radar, which are used to detect different events. Moreover, TPTS considers that each application is collaboratively processed by all WSNs within the system. In their recent work, Li et al. [2012] proposed a more general solution to the problem of task scheduling on multiple WSNs, called HTPTS, which circumvents the limitations observed in TPTS. HTPTS is intentionally designed as a multiobjective scheduling algorithm with user tunable parameters, for example, energy consumption, load balance, and data accuracy so as to elastically extend the system lifetime, while successfully completing time-constrained applications before their deadlines.

7.3. SSN Properties Addressed at the System Level

Regarding the resource sharing property of SSN, most solutions related to task allocation/scheduling operate in a centralized manner, where a single central point, often called a global scheduler, is responsible for collecting information about the available sensor nodes and processing incoming applications. Each application will be decomposed into a number of dependent/independent tasks at the global scheduler and directly distributed/assigned to the selected nodes. Centralized scheduling is an efficient mechanism for a small-size system because all information is available at the global scheduler [Farias et al. 2013; Xu et al. 2011], which can generate optimal scheduling schemes. This advantage is naturally inherited by the proposed

centralized SSN task allocation/scheduling approaches. However, the disadvantage is that the centralized scheduling approach is neither scalable nor fault tolerant, especially when the only global scheduler is suddenly disconnected from the network, which will cause the entire system to malfunction. In contrast to the centralized approach, the distributed approach [Wu et al. 2012] has more advantages regarding scalability and fault tolerance. In such design, the central point is only used as a pool to store all incoming applications. The task allocation/scheduling decision is collaboratively determined by the sensor nodes, which are also known as local schedulers. After the scheduling scheme is generated by the local scheduler, the selected nodes will retrieve the assigned task(s) from the task pool in either push or pull manner. To prevent serious task queueing in the task pool due to inappropriate scheduling, the distributed approach normally employs a task priority mechanism to allow high-priority tasks to be served before low-priority ones instead of a first-in-first-serve manner. Even with such a mechanism, in the distributed scheduling approach it is still hard to schedule tasks in an energy-efficient way due to the need of frequent communication between neighbor nodes during the decision time, especially when the number of neighbor nodes is large. In addition, the distributed scheduling scheme is weak at generating the optimal scheduling schemes for applications on behalf of the system due to local schedulers lacking global information. Consequently, the conducted SLR revealed that almost no SSN task scheduling/allocation approach is designed in a purely distributed manner. Instead, the hybrid scheduling approach, such as proposed in TPTS [Li et al. 2013] and HTPTS [Li et al. 2012], is considered as a promising solution for handling resource sharing in SSN at system level since it combines the advantages of both centralized and distributed approaches. However, no existing proposal fully addresses the challenges of performing efficient and fair resource sharing in SSN.

The works from the perspective of WSNs federation propose a similar idea to address the issue of resource sharing by providing a single access point for users to utilize the underlying heterogeneous hardware in a transparent way. Instead of allocating the resource to the tasks from different applications, they adopt a resource reservation approach. Users can request the necessary resources from the federated testbeds before the applications are deployed. During the resource reservation time, the allocated nodes form a dedicated platform for running a specific application. As long as the reservation time on the sensor nodes has no conflicts, multiple dedicated platforms can be built on the top of the federated testbeds for running different applications. By doing this, the ownership property is naturally addressed since the testbeds resource can be leased and reclaimed on demand. Moreover, testbeds could be built on different hardware and platforms. To work collaboratively, all the member testbeds have to follow the same API provided at the single access point to advertise their resource to users. The platform dependency is thus addressed at the testbed level. The users of the federated testbeds do not need to know any hardware details for running their applications.

8. APPLICATIONS OF SSNS

The early stage of WSNs development was motivated by the needs of military applications [Akyildiz et al. 2002]. However, nowadays these networks have been extensively applied to various civilian applications, many of them being easily found in our daily lives, including electric power transmission and distribution, health and medical applications, smart environments, environmental monitoring, security, structural monitoring, agricultural monitoring, and many more. This section presents a brief review of selected applications of SSNs found in our systematic review, in order to reveal the benefits and potentialities offered by such designs, thus providing answers for RQ3. We also present several works in Section 8.5 that concentrated on how to protect data in SSN, since security has been identified as an important application requirement

[Rodríguez-Molina et al. 2013]. The material is organized in five subsections devoted to summarizing applications focused on (1) smart buildings, (2) food transportation, (3) healthcare and medical applications, (4) smart grids, and finally (5) how to secure SSN applications' data.

8.1. Smart Buildings

A smart building [Meyer and Rakotonirainy 2003; Snoonian 2003] is an environment where a variety of sensors, actuators, and computational units work continuously and collaboratively to allow the occupants to customize the functionality of their living environment, such as homes and offices, industrial plants, and leisure environments. Smart building is a common application domain of SSN, and it consists of a dwelling that contains highly advanced automatic systems for constant monitoring and intelligent control of the building conditions, and providing appropriate services to the people staying in it according to their needs. Sensors and actuators deployed in the building can make the internal environmental conditions more comfortable and safer in several aspects. For example, the room temperature can be adapted to the owner's preferences and to the weather; the room lighting can be changed along with the time of the day; domestic incidents can be avoided with appropriate monitoring and alarm system; and energy can be saved by automatically switching off appliances when not needed.

Several works have realized the advantages of SSNs to implement smart building projects. Actually, smart buildings are one of the first real-world scenarios to motivate the design of shared networks, since they consist of an environment instrumented with sensors that generally belong to the same owner, and should ideally run various applications relevant to the building operation. In Bhattacharya et al. [2010], the authors implemented their design as an integrated environment, called UMADE. UMADE can simultaneously support five representative applications in the context of smart building, including temperature monitoring, humidity monitoring, air quality monitoring, light monitoring for lighting control, and acoustic signal monitoring for noise control. Each application periodically samples and transmits the processed sensing data to the sink node without causing any interruption, and then the sink node will send corresponding commands to the actuators to adjust the local condition. The aforementioned platform SenShare [Leontiadis et al. 2012] has also been successfully deployed by the authors in the office space of their research institution as a prototype demonstration of a smart office project for supporting two long-term primary applications and several short-term experimental applications. Two long-term primary applications are room environmental conditions and office occupancy, where the first one is responsible for recording temperature and humidity levels for all rooms, and the second one is responsible for monitoring employees' collective working time by recording how much time they spent at their desks. The short-term experimental applications are mainly focused on controlling the working mode of appliances, such as coffee machines, without disrupting preexisting applications. Several other smart home projects show similar research trends by exploring the use of SSNs to monitor the well-being of individuals in the home environment, including Aware Home [Kientz et al. 2008] from the Georgia Institute of Technology, PlaceLab [Intille et al. 2005] from the Massachusetts Institute of Technology, and Gator-Tech Smart House [Helal and Chen 2009] from the University of Florida.

The work described in Carr et al. [2013] proposes a Building Management Framework (BMF) for the rapid development and flexible management of pervasive building monitoring applications. BMF is a domain-specific framework based on WSANs (Wireless Sensor and Actuator Networks) for enabling proactive monitoring of spaces and control of devices/equipment. The aim of BMF is to overcome the drawbacks of existing frameworks by providing (i) flexible and efficient management of large sets of

cooperating networked sensors and actuators, (ii) abstractions for logical and physical node group organization to capture the morphology of buildings, (iii) intelligent sensing and actuation techniques, (iv) integration of heterogeneous WSANs, (v) flexible system programming at both low and high levels, and (vi) fast deployment of different applications through message exchange. BMF is organized in two processing layers: Low-Level Processing (LLP) and High-Level Processing (HLP). LLP resides at sensor nodes to provide sensor-based services, while HLP resides at the base stations to provide application-based services. The contribution of the framework is twofold. Through HLP, the BMF assists the application developer with an extensible set of OSGi Bundles [Bose 2009] that facilitate the management of a heterogeneous WSAN spanned buildings and the collection of data from the network. Through LLP, the BMF facilitates the deployment of heterogeneous nodes, management of network operations, and network maintenance.

8.2. Food Transportation

The freshness of perishable foods such as fruits of the season, fresh-cut vegetables, butcher's meats, and dairy products can drastically affect the quality of our lives. From the place of production to the consumption sites, foods could be transported as far as thousands of kilometers away and potentially exposed to the changing environment. During the transportation the conservation status needs to be carefully monitored to prevent food spoilage.

To achieve such a goal, an example was presented in Steffan et al. [2005] by using SSNs design. The perishable foods are placed into containers and a number of sensors are deployed in these containers for different purposes, for example, monitoring environmental conditions, detecting tampering or leakage of dangerous goods, and providing an RFID-based (Radio Frequency Identification) real-time inventory. The authors considered all the sensor nodes to form a SSNs that can support multiple applications simultaneously. The sensor nodes within the same container are treated as a logical sensor network mainly performing the environmental condition monitoring with different sampling rates, thresholds, and other settings for different foods. Some nodes within a container capable of communicating with the nodes located at other containers are thus treated as new logical sensor networks, called intercontainer networks. The intercontainer networks can be used for different purposes during the journey, such as detecting the containers that went overboard or were lost on the road or determining the position of a specific container. In addition, all the food containers possibly belong to different authorities so the collected information such as inventory or condition of foods is business-critical and should be confidential and only be available to the responsible personnel.

8.3. Health and Medical Applications

As the world population is aging, human healthcare becomes an increasingly prevalent issue worldwide. WSN techniques have long been applied to the healthcare domain as efficient and reliable tools for medical aids. In medical healthcare systems, sensor nodes can be placed on, near, or within the human body to provide continuous and uninterrupted reports on human vital signs, such as heart rate, blood pressure, blood oxygen, blood sugar, body temperature, and respiration. For this reason, this kind of WSN is generally referred to as Wireless Body Sensor Network (WBSN). WBSNs support various applications with different purposes, such as health monitoring, assisted living, elderly care, and medical care. According to Haque et al. [2014] these applications can be broadly classified into two categories: (i) assisted and (ii) controlled. Assisted applications aim at providing medical advice to users by simply acquiring their physiological data via sensor nodes without restricting independence in their normal living. With

individual medical requirements taken into account, it is possible to provide medical advice to users in real time, especially for those elderly people with less mobility or living alone. In addition to assisted applications, WBSNs can also be used in controlled environments that consist of hospital and intensive care units where the medical support is readily available and the level of observation is high and intense. Within such environments, medical workflows and patient safety can be highly improved.

Most WBSNs are special purpose designs with statically deployed applications, mainly for recording sensor traces. These applications are designed to be isolated from others, and in many cases the data acquired is not even accessible to the users. For example, an application that detects atrial fibrillation and an application that detects epileptic seizures cannot use the electrocardiogram (ECG) data collected from the same sensor. However, researchers have recently realized the benefits of applying SSN to WBSNs by providing capabilities for reuse and evolution of existing sensor networks and applications. These capabilities bring benefits such as reducing application development and maintenance cost, a higher degree of comfort for the user since the number of mounted devices remains limited (by reusing existing components and sensors), and more flexibility since new healthcare applications can be deployed into the same system without disturbing the old ones.

In Bui et al. [2012], the authors proposed a prototype body sensor platform that supports dynamic applications deployment, their concurrent running, data sharing between multiple applications, and can handle different sensors and their configurations. Two applications, namely, posture detection and activity monitoring, are simultaneously run on the demonstrated WBSNs platform, where the posture detection application is used to detect real-time events, such as standing, laying, and sitting; while the activity monitoring application is used to detect an activity event when the user is standing and doing an exercise with his arms. For each application, the corresponding sensors collect the relevant data to be kept separated in different data tables and the sensed data are then processed by two different algorithms. In order to reduce the algorithm complexity and accurately determine the user's behavior, the activity event application also subscribes to the posture events produced by the posture detection application. With the data from both applications, the activity monitoring application can finally determine user behavior in an efficient and accurate manner.

In Haghighi [2013], the author introduces an agent-based middleware solution, called Sensomax, for WBSN applications. Sensomax exploits a multiagent communication model to support multiclustering network division, concurrent multiple application execution, and dynamically applying runtime modification for healthcare monitoring. For running multiple applications on WBSN, Sensomax abstracts the network into several groups, called clusters, in which one node acts as the cluster head and the remainder act as its members. Each cluster maintains a unique communication channel for interaction among its members. However, multiple applications can coexist in a single node while keeping their communication and execution domains isolated. Multiple clusters could thus overlap physically without interfering with each other's operations. The applications are characterized by their embedded requirements, which can be generally classified into query, data, time, and event conditions. Based on these requirements, every application can eventually fit into one or a combination of the four main categories of query-driven, data-driven, time-driven, and event-driven, respectively. These categories determine the operational paradigm in which the application needs to be executed. Sensomax further extends this mechanism by integrating the operational paradigms into the multiclustered network topology, in a way that each application, based on the number and type of embedded requirements, gets allocated to one or more cluster(s), and its requirements are deployed as stand-alone subtasks to the cluster members. Such a mechanism creates a centralized and collaborative execution

environment where the subtasks are executed independently in a decentralized way and only return their results to base station or respond to queries.

In Seeger et al. [2014], the authors propose an event-based middleware solution, called MyHealthAssistant, for managing body and ambient sensor networks for user-centric health monitoring. Employing event-based middleware in WBSN aims at providing efficient data processing to handle a large amount of data [Singh and Bacon 2009]. Examples of processing are real-time sleep analysis and intensive care. Unlike some existing solutions such as Lifeware [Rodríguez-Molina et al. 2013], SIXTH [Carr et al. 2013], and MobiSense [Waluyo et al. 2010] that focus on optimizing the performance of the network and the sensors, MyHealthAssistant aims at mediating among sensors from different manufacturers and multiple health-related applications running on a single mobile device. The core of MyHealthAssistant is an event-driven middleware, which is designed to aggregate and provide information from both body sensor network and ambient sensor networks. It is composed of three components, namely, sensor module, event composer, and message handler. The sensor module is used to handle the sensor communication and create a corresponding sensor event. The event composer is used to interpret incoming events, identify general situations on which the system needs to react, and create a corresponding derived event. Events are organized in a hierarchy. Besides that, the event composer also checks for inaccurate or invalid sensor data and emits events enriched with fidelity information. Therefore, the applications can only consider the sensor readings that reach a certain data fidelity threshold, and ignore those low-fidelity data, for example, heart rate reading with sensing artifacts or blood pressure readings taken after exertion or with a wrong arm posture during the measurement. The message handler is used to connect with the broadcast channels that connect all the sensors in the system, thus informing subscribed applications about new sensor readings. When an event of any type in the event hierarchy is available, the applications subscribed to such information will receive the latest reading from the sensors. One common reading can be shared among applications, thus avoiding redundant processing and potential communication interference. In addition, network management information such as low battery alarms or changes in network topology are propagated in the broadcast channels and structured in an event hierarchy as well.

8.4. Smart Grids Applications

The electricity grid is a collection of power plants and transmission and distribution facilities that are responsible for energy generation, power transmission, and electricity distribution processes and for delivering reliable electricity service to customers in real time [Erol-Kantarci and Mouftah 2011]. The techniques used in energy generation, power transmission, and electricity distribution are mostly mature and stable, whereas the upward trend in population along with the increasing consumer electronics market has substantially increased energy dependence worldwide. A smart grid is a modernized electrical grid that utilizes information and communication technology to gather and act on relevant information, such as information about the behavior of suppliers and consumers, in an automated manner in order to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity [EPRI 2008]. SSNs have a broad range of applications in the smart grid context and we will briefly discuss such applications in the following.

In the traditional power grid, energy generation facilities are generally monitored with wired sensors that are limited in amount and located only at a few critical places [Gungor et al. 2010]. One of the objectives of the smart grid is to strengthen the role of renewable energy sources in the energy generation cycle. These renewable energy generation facilities could be located in sparsely populated areas, and operate in harsh

and remote environments where continuous monitoring with low-cost sensors becomes necessary. SSNs offer an ideal solution for monitoring and control of the generation facilities in the smart grid, since multiple monitoring applications need to exchange data and work collaboratively to ensure the entire system is in good condition [Farias et al. 2012]. In the transmission and distribution segment of the power grid [Ishmanov et al. 2011], the components that need to be continuously monitored in near real time are the substations [Gungor et al. 2010], overhead power lines [Farias et al. 2012], and underground power lines [Galli et al. 2011]. A single equipment failure or breakdown may cause blackouts, and may even become a health and security threat to the citizens.

8.5. Securing SSN Applications

In order to ensure safety when applications are performed in SSN, several security mechanisms have been proposed recently. These mechanisms aim to prevent attackers from compromising data sent in critical applications such as smart grids and smart buildings since attacks on these applications could cause damage and endanger citizens.

Kumar et al. [2015] propose a secure and distributed data discovery and dissemination protocol called DiDrip (named after two well-known data dissemination protocols: (i) Dip [Levis et al. 2004] and (ii) Drip [Pajic et al. 2013]). DiDrip allows the network owners to authorize users with different privileges to simultaneously and directly command sensors to disseminate data items (a tuple containing a cryptographic key, a version, and sensed data) in the SSN. Moreover, as demonstrated by their theoretical analysis, DiDrip addresses the authenticity and integrity of the disseminated data in SSNs.

In Heidemann et al. [2001], the authors address the controlled usage of resources as a primary security requirement in case of sensor sharing. A distributed reference monitor is proposed as an enforcement mechanism for sensor sharing. The monitor is policy driven, which enables lightweight runtime control of accesses to resources. The strategy is basically making the monitor choose the tasks that will better fit in a node without depleting its resources. By doing so, the authors hope to avoid attacks that will deplete the nodes, making the network cease communication. The proposal is an extension of the work of Hefeida et al. [2011] and consists of an operational model that inserts controls in the network by instrumentation of local or remote interaction in the resource-rich backend (such as in a testbed), subsequently enforcing control at the nodes by using policy engines. The selective injection is achieved through aspect-oriented techniques [Haque et al. 2014]. The solution achieves local resource protection and protection of distributed event-based data flow.

Marien et al. [2011] presents SASHA: a proof-of-concept protocol that enables lightweight and secure deployment of multiple applications on heterogeneously owned sensor nodes. The protocol requires an Application Owner (AO) to request permission from all the Platform Owners (POs) on whose infrastructure he/she wishes to install an application. Each PO validates this request and, if approved, grants a token the AO can use to deploy his/her application. This token also contains parameters to limit the amount and frequency a component can send and the CPU time it can use.

In Tobgay et al. [2011], the authors propose a dynamic instruction detection system for SSNs called DISSN. DISSN gathers information about the environment (packet loss, interference, and system lifetime) as indicatives of attacks in the SSN. According to the information gathered, DISSN will choose a countermeasure, based on the QoS requirements of an application. As an example, a high rate of packet loss could be an indicative of Denial of Service, and a countermeasure to change the node state to idle.

9. OPEN ISSUES

As discussed earlier, SSNs present various challenges, which are not faced by conventional WSNs. Current research initiatives in SSNs have demonstrated promising results for some of these challenges; however, there are several open research issues that need to be further addressed. First of all, we would like to point out that energy efficiency is still a key concern and attracts the attention of most SSN researchers. Apart from that, in this section, we discuss possible promising and challenging issues identified during our study in the field of SSN. These issues are organized in different levels as presented in Section 4 and as shown in the following.

9.1. Node Level

This subsection presents the open research challenges in the SSN field at the node level.

9.1.1. Node-Level Virtualization. In current work, node-level virtualization is provided in multiple ways, such as part of the sensor OS, multithreaded OSs, and application-specific VMs, working on top of an OS to support the concurrent execution of multiple tasks from different applications. As the SSN trend moves forward, more efforts are required to virtualize the individual components of heterogeneous nodes and to allow them to be used by applications at the same time. As mentioned before, some solutions (e.g., Squawk VM) provide separation between the sensor OS and the user applications but other functions are still missing, like Over-The-Air programming, which can be used to load, unload, stop, and migrate applications without disturbing existing ones. One possible solution [Oliver et al. 2010] to tackle this issue is to design an abstraction layer that works on top of the sensor OS to provide application portability.

9.2. Network Level

This subsection presents the open research challenges in the SSN field at network level.

9.2.1. Network-Level Virtualization. As mentioned, overlay networks can provide an efficient solution to support multiple applications over a deployed WSN as they are robust and can work efficiently without changes in the underlying network. Some of the existing solutions are still in their infancy and do not fully consider the requirements of multiple applications utilizing the network nodes concurrently. In this context, one major challenge in SSN is how to provide multiple overlays over a topology-changeable WSN. To enable concurrent applications execution within a WSN, multiple overlays may need to coexist to prevent them from interacting with each other in a harmful way. One possible solution is the cluster-based approaches, which have traditionally been used in WSNs for improving routing, energy efficiency, and management. Managing clusters in a virtualized WSN is not trivial. However, cluster-based solutions can be quite useful in scenarios where a deployed WSN is used to monitor dynamic events. Another potential issue in network-level virtualization is how to quickly publish/discover the available resources. This is due to the fact that VSNs are created on demand and destroyed when no longer required, and the newly joined resources may only be available to some specific nodes, but not all of them. A mechanism that allows the resources to be quickly published/discovered on the fly is crucial to SSNs, especially in the emergent context of Internet of Things (IoT). A fully distributed, for example, P2P based, architecture that does not rely on any central controller to advertise the resources to the network could be a solution to this issue.

9.2.2. Information Fusion. So far, only a few preliminary studies of information fusion techniques tailored to SSN have been proposed. These works are based on probabilistic

methods, such as Dempster-Shafer and Bayesian Inference [Nakamura et al. 2007], to infer over the data shared by multiple applications. One problem about probabilistic methods is that they build on a static set of rules to process data, which does not suit the dynamic environment of SSN. When the new applications arrive at the network, the set of rules must be updated. In this context, future work emerges that consists in extending data fusion algorithms to use artificial intelligence techniques to deal with multiple applications. By using artificial intelligence techniques, such as fuzzy systems or neural networks, it is possible to dynamically adjust the applications weights and set of rules in the fusion algorithms and obtain more accurate results. Another possible future research direction is to design a framework to coordinate multiple fusion algorithms for addressing different applications in SSN. In such scenarios, the challenge is how to choose the appropriate fusion algorithm for each data type and/or data range. Nowadays, a (human) specialist usually makes the decision at design time. However, applications could have different goals and interests in how to fuse their data and these needs could be varied on the fly. When these situations happen, it is difficult for the specialist to consider all the possibilities and to determine the most appropriate fusion algorithm to use beforehand. Therefore, it is crucial to design a mechanism that is capable of autonomously deciding which one of all available fusion algorithms to employ to handle different requirements from multiple applications at runtime.

9.2.3. Routing. Even in the context of application-specific WSN, routing had always been a puzzling problem due to the unreliability of nodes and wireless links, the huge number of nodes and infrastructureless feature of the networks, among other factors. In SSNs, besides the aforementioned issues that affect routing, sharing the network between more than one application imposes new challenges and calls for novel solutions of routing. Such solutions must provide mechanisms allowing usage of various types of application-level information in routing decisions. Such information shall include static and dynamic data encompassing the location of nodes and events of interest, execution context of nodes and applications, QoS requirements and priorities between applications, among others. Therefore, an application-aware routing framework for SSNs is necessary. In addition to providing models for representing information of interest at a high abstraction level, and to addressing network nodes taking into account this information, such a framework should provide a means to make effective use of application-level information in all routing decisions, promoting efficient data and route sharing and keeping the load balancing between nodes, while meeting nonfunctional requirements of applications. Resource management and security functions could also be included in such frameworks. Finally, to accommodate the huge number of sensor nodes, such framework should preferably be endowed with self-configuration and autonomic behavior capabilities. We did not find any work that properly meets all these requirements in the researched literature.

9.3. System Level

This subsection presents the open research challenges in the SSN field at system level.

9.3.1. QoS Requirements. QoS is prevalent in WSNs since low-quality or stale data could ruin the application. The trade-off between energy consumption and various kinds of parameters of quality, for example, data accuracy, communication latency, and error rates, always needs to be considered in WSN applications. Compared to conventional WSNs, QoS requirements become a more critical issue in SSN since the system is required to address different needs from different applications simultaneously and these needs are sometimes contradictory. As a result, the support for QoS in SSN has to be offered so that the requirements of the most demanding and/or higher-priority applications are met, while trying to keep all the others within a reasonable degree of satisfaction. In the meantime, the SSN system should export interfaces to the applications/

users to allow the specification of the desired QoS parameters and a fine-grained tune of such parameters whenever it is necessary. A mechanism to translate and map high-level, application-defined QoS parameters to low-level network parameters as well as an underlying runtime mechanism that realizes precise control of quality is also required. This is because the ultimate objective of using sensor networks is not the sensing per se, but taking actions that are either implicitly or explicitly determined according to the result of sensing.

There are some works that present prototype designs for QoS-aware SSNs. In Li et al. [2014], the authors proposed an adaptive scheduling and feedback control based mechanism to allow users to control the data quality and exploit the energy-quality trade-off. It requires no prior knowledge of the system; users only need to configure the weight of each concerned QoS requirement for their applications. The adaptive scheduling approach is in charge of allocating the resources to applications based on detailed knowledge of application requirements, application characteristics, and network resource availability. After the allocation, the feedback control mechanism sends back the latest information to the sink for handling the newly arrival applications. In the VITRO project [Trakadas et al. 2013], the authors proposed a reference architecture that enables the realization of a scalable, flexible, adaptive, energy-efficient, and trust-aware SSN platform. In this QoS-aware SSN, sensors are deployed by potential (public or private) organizations and can then be used to feed information to applications deployed by other organizations. However, accommodating multiple services with different requirements (e.g., QoS and security) on a shared infrastructure, without predefined resource reservations, jeopardizes the experienced QoS acceptability. For this reason, there is a need to route and schedule solutions that support QoS differentiation per application. Among the routing protocols developed for supporting QoS differentiation, the RPL protocol (Routing Protocol for Low power and lossy networks) [Trakadas et al. 2013] has recently attracted attention for its capability of interoperating with IP networks. The RPL has recently been standardized by IETF and it provides a formal routing algebra framework for the proper composition of routing metrics allowing QoS differentiation per application. Although it was not specifically tailored to SSN, the protocol has the potential to be successfully applied to such networks in the context of providing the required QoS for multiple applications. The works discussed in this section address QoS using scheduling and routing solutions. However, a fully QoS-aware framework providing an integrated solution to tackle QoS at all levels of a SSN design is still lacking in the surveyed literature.

9.3.2. Task Allocation. Task allocation has been well studied in both WSNs and SSNs. However, most of the proposals assume that only one sink node exists in the system and it acts as the gateway between sensor nodes and external systems. Along with the increasing number of applications handled by SSN, the increasing workload assigned to the sink node may cause serious delays for the applications or malfunctions in the sink node. Consequently, multiple sink nodes can be considered as a solution for avoiding the preceding issues. In this case, applications can be submitted to SSN from different entry points. However, the existing task scheduling and allocation algorithms are less capable of handling this situation. To address this issue, we need to further adjust the existing algorithms to answer some questions, such as (i) how to determine the priority of the sink nodes in task allocation; (ii) how to ensure the allocation scheme of an application generated by a specific sink node does not interfere with the schemes generated by other sink nodes; and (iii) how to determine the execution sequence of the tasks from different applications when they are allocated to the same sensor node. It may also be necessary to include cooperation mechanisms between different sinks to allow the full completion of applications whose sensing interests are spread over a wide geographic area (encompassing multiple sinks).

9.3.3. Potential Synergy with Cloud Computing. The Cloud of Things (CoT) [Parwekar 2011] (previously known as sensor cloud [Hassan et al. 2009]) is a concept recently proposed as a promising paradigm of future IoT [Atzori et al. 2010], to orchestrate heterogeneous resources provided by a variety of objects around us with public/private clouds in order to create infrastructures capable of delivering new generation services. CoT not only retains all the benefits of IoT, but also is capable of integrating cutting-edge technologies, or even promoting new technologies to satisfy different global priority needs, for example, gas emission reduction, healthcare, environmental monitoring, to name a few. Two key components of IoT are sensing and actuation resources, which are mainly provided by wireless sensor/actuator networks. As suggested by current ICT trends, such networks are required to be properly integrated with the cloud environment. This implies that the heterogeneous sensing and actuation resources should not be exclusively used as simple endpoints for dedicated applications, but they need to be dealt with in the same way as computing and storage resources usually are in more traditional cloud stacks: abstracted, virtualized and grouped in the clouds.

A few works have recently attempted to address this challenge from the perspective of SSNs.

Dinam [Yong et al. 2012] is a three-tier architecture for providing in situ development, deployment, monitoring, and configuration of multiple WSN applications on the top of the same underlying hardware infrastructure. In accordance to the existing cloud computing stack, the authors proposed the one-to-one mapping Dinam cloud stack, namely, Dinam IaaS (Infrastructure as a Service), Dinam PaaS (Platform as a Service), and Dinam SaaS (Software as a Service). Dinam IaaS is the infrastructure services delivery layer of the proposed Dinam architecture and it provides fundamental computing resources in the context of WSNs. Dinam PaaS is the core component of the proposed architecture for enabling rapid development and deployment of WSN applications over the same hardware platform by hiding the actual underlying heterogeneous WSNs topology and excessive hardware implementation details from application developers and far-end users. The Dinam PaaS is also presented as a unified layer of networking, storage, data, and service interfaces to WSN applications hosted on the Dinam SaaS, which is a remote accessible layer for offering the available WSN-related services at the Dinam PaaS level to the users to construct their own applications.

TaaS (Things as a Service) [Distefano et al. 2012] is another preliminary layout design of CoT that aims at establishing a cloud-integrated IoT infrastructure for providing pervasive indexing and querying services on top of things. The ideas of SSNs are inherently used in the architectural modules of TaaS. The bottom layer of TaaS is called intraNode, operating at the node level, for abstracting away either embedded sensors installed on a smart device or stand-alone sensor nodes belonging to WSNs. The intraCloud/interNode level is the upper layer of intraNode, which is mainly used to handle communication and interaction issues between sensor nodes within the same cloud for generating relevant services. The interCloud level as the top level of the design is in charge of interoperability and federation among cloud peers.

In the aforementioned works, the authors devised interesting approaches for applying the ideas of SSNs design to CoT. Their schemes are more concentrated on integrating heterogeneous WSN resources to clouds and not on the subsequent shared sensor network design, thus many topics are still open for future research and will pose new challenges for the SSN field.

10. SUMMARY AND CONCLUSIONS

The increasing capabilities of WSNs open exciting possibilities for their applications in various fields. In order to reduce the costs of deployment and administration, a

noticeable design trend, the so-called shared sensor network, has been considered as a promising approach to support concurrent applications sharing the hardware resources from the same WSN, thus further increasing the utilization of the underlying infrastructure. In this systematic literature survey, we have provided a thorough search and study of the literature related to SSN and tried to cover the most important aspects of this new design approach.

We have formulated three hypotheses to conduct our research: (i) a SSN design improves the utilization of the network's nodes, thus contributing to increased ROI of the deployed infrastructure; (ii) a SSN design promotes a clear separation between the ownership of the applications and the underlying sensor network infrastructure; and (iii) a SSN design brings new business opportunities in the form of new applications that take advantage of shared infrastructure. All the obtained results arise from these fundamental hypotheses. They were used to define the research questions used to perform the literature review. The period considered in the searches carried out initially in our SLR was from 2000 to 2013. However, after these initial searches and completion of the stages of data extraction and analysis, new papers have been retrieved or included by recommendation of experts. These new papers were considered in the discussions presented in the survey. Therefore, the period included in this study extends to 2014, thus having a wider coverage. However, given the broad aspects encompassed in the WSN field, and the dynamism of this research field, our survey is still by no means complete. To better contextualize our findings, we presented a simple but useful taxonomy that classifies the works of SSNs into three levels, namely, node level, network level, and system level, based on their degrees of abstraction. We also investigated the benefits that can be obtained from running multiple applications in a SSNs infrastructure. The final goal of our survey was to shed light on the requirements and implications of a SSN design and also on existing open issues and future directions toward fully exploiting the potential of such an approach.

Regarding the formulated hypotheses, we have concluded that it is necessary to deal with different kinds of challenges imposed by the nature of a WSN on one hand and the requirements of multiple applications on the other hand. For handling multiple applications a SSN must be energy efficient. This is perhaps the most stringent requirement for almost all WSN applications, since sensor nodes are generally powered by nonrechargeable batteries, which are inevitably limited in energy capacity. As the number of applications increases, energy consumption also increases. A promising strategy to reduce energy consumption is through information sharing (hypothesis (i)). As discussed in Sections 5, 6, and 7, information sharing extends the benefits of collaboration to SSNs since some intermediate results can be directly reused among applications and the associated repeated operations. To design SSN is not a trivial task, especially when the underlying network is heterogeneous (composed of different types of devices). Within the heterogeneous sensor network, different communication technologies, such as Zigbee, Bluetooth, UWB, and WIFI are used and even different OSs could be installed on the sensor nodes. This requires proper mechanisms to hide the hardware-specific details from end users and provide them with a VSN that acts as a unique and dedicated platform to the application (see Section 5). Not only the infrastructure is heterogeneous but also the ways in which applications deal with data. Different applications have different degrees of importance and likewise their data should have different degrees of importance, as we discussed in Section 6 (hypothesis (ii)). Finally, as presented in Section 8, there are several scenarios that can benefit from a SSN design from home and health applications to smart grids (hypothesis (iii)), since they encompass several applications simultaneously.

APPENDIX: ADDITIONAL DETAILS AND SEARCH RESULTS OF THE CONDUCTED SLR

The goal of the data collection stage is to gather the necessary data to answer research questions in a credible way depending on the QoD. To ensure data quality, we further set the following criteria:

(1) works are published in reliable computer science venues (peer-reviewed conference, peer-reviewed journal, or computer science/engineering organization);
(2) the language for publication must be English; and
(3) the works are published during the period of 2000–2013.

After the preceding process is completed, the extracted data were processed to draw out key themes as part of the synthesis stage of the review. Similarly to Kitchenham et al. [2009], the data extracted from each study were as follows:

—the authors' information, including name of authors, their institution, and the country where it is situated;
—the source (journal or conference) and full reference;
—summary of the study including the main research questions and the answers;
—technical aspects related to the SSNs that was addressed in the work, including modeling, proposed solutions, and quality evaluation; and
—findings and conclusions.

Within this process, two researchers (Li and Farias) performed the data extraction and the rest checked the resulting data. In the event of disagreements, the data were discussed in detail until agreement was reached.

A total of 1,307 articles were identified as potentially relevant after searching the aforementioned databases. An additional 35 articles were identified through manual search via Google Scholar, increasing the total number of articles to 1,342. Before applying the inclusion and exclusion criteria presented in Section 2.3, duplicate findings were first removed from the preliminary result set; a total of 799 candidate articles remained. Then all abstracts of the remaining articles were downloaded for further processing. The responsible researchers (Li and Farias) reviewed all abstracts, applying the first stage rule of the inclusion/exclusion criteria; 473 candidate articles were excluded during abstract review, thus decreasing the total to 326. Full text documents were all retrieved for these 326 articles. During full text article review, 239 candidate articles were excluded, leaving a total of 87 articles for the quality screening. During quality screening, 26 articles were excluded, leaving a total of 61 articles for processing in the collection and analysis. At this stage, more restricted rules (mentioned earlier) were applied to the remaining articles. These led to another four articles being eliminated due to some conditions set for this survey only, for example, publication sources, language, and publication year, mentioned earlier. Eventually, there were 57 articles included in our survey studies.

After performing the SLR (after May of 2013), the authors found 31 papers that were included in this survey. These papers respect the same Quality Criteria (QC) used to select the papers used in the SLR. Therefore, the total amount of papers was 57+31, or 88 papers.

A flow chart of the search process is shown in Figure 2, formatted according to PRISMA statement guidelines [Moher et al. 2009].

As mentioned earlier, the final number of included articles in this survey was 88. Even though we did not directly answer the research questions at this stage, these questions are still used to classify the included articles. There are 22, 46, and 20 papers closely related to the defined research questions, respectively, and 17 out of 88 papers addressed more than one research question. As shown in Table III, for each question,

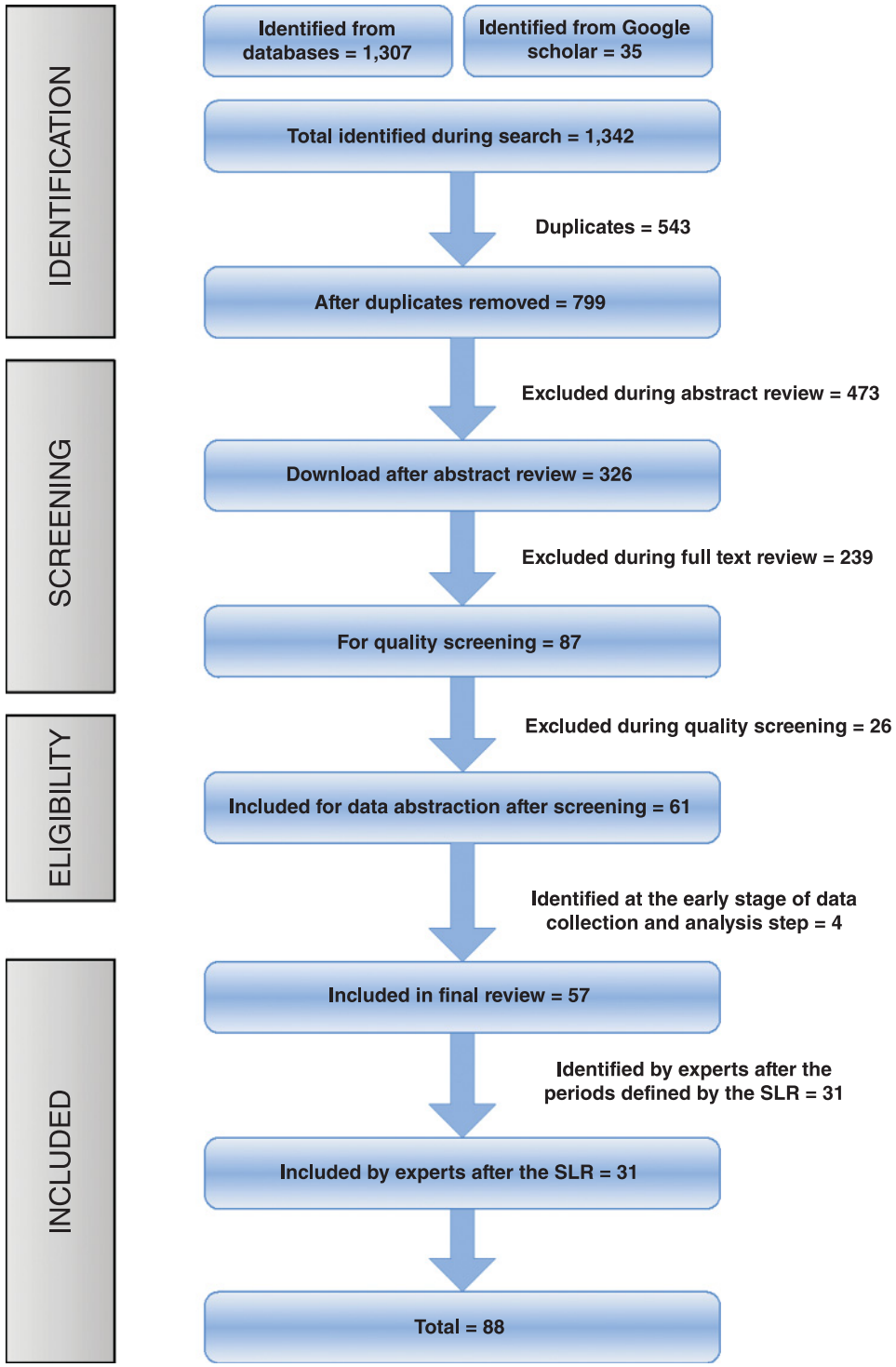

Fig. 2. Process flow for literature review.

Table III. Descriptive Analysis and Summary of Results for Included Articles

Research Questions	Region	Article Number	Article Type		Study Designs	Publication Periods		
			Conference	Journal		2000–2004	2005–2009	2010–2013
RQ1	Asia	2	0	2	Theoretical study	4	7	6
	Australasia	2	1	1	Theoretical study/technology trial			
	Euro	2	1	1	Theoretical study/technology trial			
	North America	10	5	5	Theoretical study/technology trial			
	South America	1	0	1	Theoretical study			
RQ2	Asia	1	0	1	Theoretical study	5	11	18
	Australasia	2	1	1	Theoretical study/technology trial			
	Euro	8	6	2	Theoretical study/technology trial			
	North America	18	14	4	Theoretical study/technology trial			
	South America	5	4	1	Theoretical study/technology trial			
RQ3	Australasia	1	1	0	Theoretical study	2	5	7
	Euro	4	2	2	Technology trial			
	North America	8	7	1	Technology trial			
	South America	1	1	0	Technology trial			

most articles retrieved from the SLR are from North America followed by European ones. Most of the included articles are from conferences rather than journals, but the articles targeted for RQ1 are the exception. In this category, the number of articles from conferences and journals are almost even. More than half included articles that are focused on both theoretical and technology studies, however, a few of them targeted only one aspect, especially for articles dealing with SSN applications. Eventually, from the publication period column shown in the table, we can identify that the emphasis of enabling SSN design on WSNs has shifted from the general possibilities and requirement discussions to detailed methodical studies and application developments. This information also confirms that SSN design has become more popular and attracted more researchers to the field of WSNs.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. 2002. Wireless sensor networks: A survey. *Computer Networks* 38, 4 (2002), 393–422. DOI:http://dx.doi.org/10.1016/S1389-1286(01)00302-4

Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805. DOI:http://dx.doi.org/10.1016/j.comnet.2010.05.010

Edgardo Avilés-López and J. Antonio García-Macías. 2009. TinySOA: A service-oriented architecture for wireless sensor networks. *SOCA* 3, 2 (2009), 99–108. DOI: http://dx.doi.org/10.1007/s11761-009-0043-x

Can Basaran and Kyoung-Don Kang. 2009. *Quality of Service in Wireless Sensor Networks*. Springer London.

Sangeeta Bhattacharya, Abusayeed Saifullah, Chenyang Lu, and Gruia-Catalin Roman. 2010. Multi-application deployment in shared sensor networks based on quality of monitoring. In *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'10)*. IEEE Computer Society, Washington, DC, 259–268. DOI: http://dx.doi.org/10.1109/rtas.2010.20

Raja Bose. 2009. Sensor networks motes, smart spaces, and beyond. *IEEE Pervasive Computing*, 8, 3 (2009), 84–90. 10.1109/MPRV.2009.55.

Niels Brouwers, Koen Langendoen, and Peter Corke. 2009. Darjeeling, a feature-rich VM for the resource poor. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 169–182. DOI: http://dx.doi.org/10.1145/1644038.1644056

Vinh T. Bui, Richard Verhoeven, and Johan J. Lukkien. 2012. A body sensor platform for concurrent applications. In *Proceedings of the 2012 2nd IEEE International Conference on Consumer Electronics*. 38–42. DOI: http://dx.doi.org/10.1109/ ICCE-Berlin.2012.6336468

A. Caracas, T. Kramp, M. Baentsch, M. Oestreicher, T. Eirich, and I. Romanov. 2009. Mote runner: A multi-language virtual machine for small embedded devices. In *Proceedings of the 2009 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM'09)*. IEEE Computer Society, Washington, DC, 117–125. DOI: http://dx.doi.org/10.1109/sensorcomm.2009.27

Dominic Carr, Michael J. O'Grady, Gregory M. P. O'Hare, and Rem Collier. 2013. *SIXTH: A Middleware for Supporting Ubiquitous Sensing in Personal Health Monitoring*. Springer, Berlin.

Ioannis Chatzigiannakis, Stefan Fischer, Christos Koninis, Georgios Mylonas, and Dennis Pfisterer. 2010. *WISEBED: An Open Large-Scale Wireless Sensor Network Testbed*. Springer, Berlin.

N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. 2009. Network virtualization: State of the art and research challenges. *IEEE Communications Magazine* 47, 7 (2009), 20–26. DOI: http://dx.doi.org/10.1109/mcom. 2009.5183468

Paolo Costa, Luca Mottola, Amy L. Murphy, and GianPietro Picco. 2009. *Tuple Space Middleware for Wireless Networks*. Springer, Berlin.

Frank P. Coyle. 2002. *Xml, Web Services, and the Data Revolution.* Addison-Wesley Longman Publishing Co., Inc.

David E. Culler, Anoop Gupta, and Jaswinder Pal Singh. 1997. *Parallel Computer Architecture: A Hardware / Software Approach*. Morgan Kaufmann Publishers, Inc.

Carlo Curino, Matteo Giani, Marco Giorgetta, Alessandro Giusti, Amy L. Murphy, and Gian Pietro Picco. 2005. Mobile data collection in sensor networks: The tinylime middleware. *Pervasive and Mobile Computing* 1, 4 (2005), 446–469. DOI: http://dx.doi.org/10.1016/j.pmcj.2005.08.003

Andre L. L. De Aquino, Carlos Figueiredo, Eduardo F. Nakamura, Antonio A. F. Loureiro, Antônio Otáivio Fernandes, and Claudionor J. N. Coelho Jr. 2007. On the use data reduction algorithms for real-time wireless sensor networks. In *12th IEEE Symposium on Computers and Communications (ISCC 2007)*. 583–588. 10.1109/ISCC.2007.4381527.

Pedro Javier del Cid, Sam Michiels, Wouter Joosen, and Danny Hughes. 2010. Middleware for resource sharing in multi-purpose wireless sensor networks. In *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)*, 1–8. 10.1109/NESEA.2010.5678061.

Flávia C. Delicato, Paulo F. Pires, Luci Pirmez, and Thais Batista. 2010. Wireless sensor networks as a service. In *Proceedings of the 2010 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'10)*. IEEE Computer Society, Washington, DC, 410–417. DOI: http://dx.doi.org/10.1109/ecbs.2010.54

Flávia C. Delicato, Jesús M. T. Portocarrero, José R. Silva, Paulo F. Pires, Rodrigo P. M. de Araújo, and Thais Batista. 2013. MARINE: MiddlewAre for resource and mIssion-oriented sensor NEtworks. *SIGMOBILE Mobile Computing and Communications Review* 17, 1 (2013), 40–54. 10.1145/2502935.2502944.

Flavia Coimbra Delicato, Paulo F. Pires, Luci Pirmez, and Luiz Fernando Carmo. 2005. A service approach for architecting application independent wireless sensor networks. *Cluster Computing* 8, 2–3 (2005), 211–221. DOI: http://dx.doi.org/10.1007/s10586-005-6186-4

Flavia Coimbra Delicato, Paulo F. Pires, Luci Pirmez, and Luiz Fernando Rust da Costa Carmo. 2003. A flexible web service based architecture for wireless sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCSW'03)*. IEEE Computer Society, Washington, DC, 730–735.

Salvatore Distefano, Giovanni Merlino, and Antonio Puliafito. 2012. Enabling the cloud of things. In *Proceedings of the 2012 6th International Conference on Innovative Mobile and Internet Services in*

Ubiquitous Computing (IMIS'12). IEEE Computer Society, Washington, DC, 858–863. DOI:http://dx.doi.org/10.1109/imis.2012.61

Christos Efstratiou. 2010. Challenges in supporting federation of sensor networks. In *NSF/FIRE Workshop on Federating Computing Resources*.

Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Jon Crowcroft. 2010. A shared sensor network infrastructure. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 367–368. DOI:http://dx.doi.org/10.1145/1869983.1870026

Ramy Eltarras and Mohamed Eltoweissy. 2010. Adaptive multi-criteria routing for shared sensor-actuator networks. In *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*. IEEE Communications Society, New York, NY, 1–6. DOI:http://dx.doi.org/10.1109/glocom.2010.5683555

EPRI. 2008. *The Green Grid: Energy Savings and Carbon Emissions Reductions Enabled by a Smart Grid*. White paper 1016905. EPRI, California.

Melike Erol-Kantarci and Hussein T. Mouftah. 2011. Wireless sensor networks for smart grid applications. In *Proceedings of the 2011 1st Saudi International Electronics, Communications and Photonics Conference (SIECPC 2011)*. 1–6. DOI:http://dx.doi.org/10.1109/siecpc.2011.5876687

Emre Ertin, Anish Arora, Rajiv Ramnath, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, Hui Cao, and Mikhail Nesterenko. 2006. *Kansei: A Testbed for Sensing at Scale*. ACM.

Claudio M. de Farias, Luci Pirmez, Flavia Coimbra Delicato, Igor L. Dos Santos, and Albert Y. Zomaya. 2012. Information fusion techniques applied to shared sensor and actuator networks. In *Proceedings of the 2012 37th IEEE Conference on Local Computer Networks (LCN)*. IEEE Computer Society, Washington, DC, 188–191. DOI:http://dx.doi.org/ 10.1109/ lcn.2012.6423603

Claudio M. de Farias, Wei Li, Jose N. de Souza, Luci Pirmez, Albert Y. Zomaya, and Flavia C. Delicato. 2013. A scheduling algorithm for shared sensor and actuator networks. In *Proceedings of the 27th 2013 International Conference on Information Networking (ICOIN 2013)*. 648–653. DOI:http://dx.doi.org/ 10.1109/icoin.2013.6496703

Claudio Farias, Luci Pirmez, Flávia Delicato, Luiz Carmo, Wei Li, Albert Y. Zomaya, and José N. de Souza. 2014. Multisensor data fusion in shared sensor and actuator networks. In *The 17th International Conference on Information Fusion (Fusion 2014)*.

Carlos A. Flores-Cortés, Gordon S. Blair, and Paul Grace. 2007. An adaptive middleware to overcome service discovery heterogeneity in mobile ad hoc environments. *IEEE Distributed Systems Online* 8, 7 (2007), 1–26. DOI:http://dx.doi.org/10.1109/mdso.2007.41

Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. 2009. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems* 4, 3 (2009), 1–26. DOI:http://dx.doi.org/10.1145/1552297.1552299

Stefano Galli, Anna Scaglione, and Zhifang Wang. 2011. For the grid and through the grid: The role of power line communications in the smart grid. *Proceedings of the IEEE* 99, 6 (2011), 998–1027. DOI:http://dx.doi.org/10.1109/jproc.2011.2109670

GENI. 2014. GENI: Global Environment for Network Innovation (2014). http://www.geni.net.

Mario Gomez, Alun Preece, Matthew P. Johnson, Geeth de Mel, Wamberto Vasconcelos, Christopher Gibson, Amotz Bar-Noy, Konrad Borowiecki, Thomas La Porta, Diego Pizzocaro, Hosam Rowaihy, Gavin Pearson, and Tien Pham. 2008. *An Ontology-Centric Approach to Sensor-Mission Assignment*. Springer, Berlin.

Vehbi C. Gungor, Bin Lu, and Gerhard P. Hancke. 2010. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics* 57, 10 (2010), 3557–3564. DOI:http://dx.doi.org/ 10.1109/tie.2009.2039455

Mo Haghighi. 2013. An end-to-end middleware solution with multiple concurrent applications support for wireless body area networks. In *2013 IEEE 6th International Workshop on Computational Intelligence and Applications (IWCIA)*. 201–206. 10.1109/IWCIA.2013.6624815

Shah Ahsanul Haque, Syed Mahfuzul Aziz, and Mustafizur Rahman. 2014. Review of cyber-physical system in healthcare. *International Journal of Distributed Sensor Networks* 2014 (2014), 20. DOI:10.1155/ 2014/217415

Mohammad Mehedi Hassan, Biao Song, and Eui-Nam Huh. 2009. A framework of sensor-cloud integration opportunities and challenges. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC'09)*. ACM, New York, NY, 618–626. DOI:http://dx.doi.org/10.1145/1516241.1516350

Mohamed Hefeida, Turkmen Canli, Ajay kshemkalyani, and Ashfaq Khokhar. 2011. Context modeling in collaborative sensor network applications. In *2011 International Conference on Collaboration Technologies and Systems (CTS)*. 274–279. DOI:10.1109/CTS.2011.5928698

John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. 2001. Building efficient wireless sensor networks with low-level naming. *SIGOPS Operating Systems Review* 35, 5 (2001), 146–159. DOI:10.1145/502059.502049

Sumi Helal and Chao Chen. 2009. The Gator Tech Smart House: Enabling technologies and lessons learned. In *Proceedings of the 3rd International Convention on Rehabilitation Engineering and Assistive Technology (i-CREATe'09)*. ACM, New York, NY, 1–4. DOI:http://dx.doi.org/10.1145/1592700.1592715

Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. 2000. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, 93–104. DOI:http://dx.doi.org/10.1145/378993.379006

Jens Horneber and Anton Hergenröder. 2014. A survey on testbeds and experimentation environments for wireless sensor networks. *IEEE Communications Surveys and Tutorials* 16, 4 (2014), 1820–1838. 10.1109/COMST.2014.2320051.

Danny Hughes, Klaas Thoelen, Wouter Horre, Nelson Matthys, Javier Del Cid, Sam Michiels, Christophe Huygens, and Wouter Joosen. 2009. *LooCI: A Loosely-Coupled Component Infrastructure for Networked Embedded Systems*. ACM.

Stephen S. Intille, Kent Larson, J. S. Beaudin, J. Nawyn, E. Munguia Tapia, and P. Kaushik. 2005. A living laboratory for the design and evaluation of ubiquitous computing technologies. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, NY, 1941–1944. DOI:http://dx.doi.org/10.1145/1056808.1057062

Farruh Ishmanov, Aamir Saeed Malik, and Sung Won Kim. 2011. Energy consumption balancing (ECB) issues and mechanisms in wireless sensor networks (WSNs): A comprehensive overview. *European Transactions on Telecommunications* 22, 4 (2011), 151–167. DOI:http://dx.doi.org/10.1002/ett.1466

Md. Motaharul Islam, Mohammad Mehedi Hassan, Ga-Won Lee, and Eui-Nam Huh. 2012. A survey on virtualization of wireless sensor networks. *Sensors* 12, 2 (2012), 2175–2207.

Anura P. Jayasumana, Qi Han, and Tissa H. Illangasekare. 2007. Virtual sensor networks—A resource efficient approach for concurrent applications. In *Proceedings of the International Conference on Information Technology (ITNG'07)*. IEEE Computer Society, Washington, DC, 111–115. DOI:http://dx.doi.org/10.1109/itng.2007.206

Nicolai Josuttis. 2007. *Soa in Practice*. O'Reilly.

Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. 2002. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, 96–107. DOI:http://dx.doi.org/10.1145/605397.605408

Sanem Kabadayi, Adam Pridgen, and Christine Julien. 2006. Virtual sensors: Abstracting data from physical sensors. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks (WOWMOM'06)*. IEEE Computer Society, Washington, DC, 587–592. DOI:http://dx.doi.org/10.1109/wowmom.2006.115

Imran Khan, Fatna Belqasmi, Roch Glitho, Noel Crespi, Monique Morrow, and Paul Polakos. 2015. Wireless sensor network virtualization: A survey. *IEEE Communications Surveys and Tutorials* PP, 99 (2015), 1–1. 10.1109/COMST.2015.2412971.

Julie A. Kientz, Shwetak N. Patel, Brian Jones, Ed Price, Elizabeth D. Mynatt, and Gregory D. Abowd. 2008. The Georgia tech aware home. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, NY, 3675–3680. DOI:http://dx.doi.org/10.1145/1358628.1358911

Hiecheol Kim, Won-Kee Hong, Joonhyuk Yoo, and Seong-eun Yoo. 2015. Experimental research testbeds for large-scale WSNs: A survey from the architectural perspective. *International Journal of Distributed Sensor Networks* 2015 (2015), 18. 10.1155/2015/630210.

Barbara Kitchenham. 2004. Procedures for performing systematic reviews. Keele, UK, Keele University 33, 2004 (2004), 1–26.

Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering—A systematic literature review. *Information and Software Technology* 51, 1 (2009), 7–15. DOI:http://dx.doi.org/10.1016/j.infsof.2008.09.009

Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. 2002a. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, 2002*. 575–578. 10.1109/ICDCSW.2002.1030829.

Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. 2002b. Modelling data-centric routing in wireless sensor networks. In *IEEE Infocom*. 39–44.

A. Senthil Kumar, S. Velmurugan, and E. Logashanmugam. 2015. A secure distributed data discovery and dissemination in wireless sensor networks. *International Journal of Engineering & Science Research* 5, 7 (July 2015), 708–713.

Thao Le, Timothy J. Norman, and Wamberto Vasconcelos. 2009. Agent-based sensor-mission assignment for tasks sharing assets. In *Proceedings of the 3rd International Workshop on Agent Technology for Sensor Networks*.

Ilias Leontiadis, Christos Efstratiou, Cecilia Mascolo, and Jon Crowcroft. 2012. SenShare: Transforming sensor networks into multi-application sensing infrastructures. In *Proceedings of the 9th European Conference on Wireless Sensor Networks*. Springer-Verlag, Berlin, 65–81. DOI:http://dx.doi.org/10.1007/978-3-642-28169-3_5

Philip Levis and David Culler. 2002. Maté: A tiny virtual machine for sensor networks. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*. ACM, New York, NY, 85–95. DOI:http://dx.doi.org/10.1145/605397.605407

Philip Levis, Neil Patel, David Culler, and Scott Shenker. 2004. *Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks*, USENIX Association, San Francisco, CA.

Wei Li, Flávia C. Delicato, Paulo F. Pires, Young Choon Lee, Albert Y. Zomaya, Claudio Miceli, and Luci Pirmez. 2014a. Efficient allocation of resources in multiple heterogeneous Wireless Sensor Networks. *Journal of Parallel and Distributed Computing* 74, 1 (2014), 1775–1788. http://dx.doi.org/10.1016/j.jpdc.2013.09.012.

Wei Li, Flavia C. Delicato, Paulo F. Pires, and Albert Y. Zomaya. 2012. Energy-efficient three-phase task scheduling heuristic for supporting distributed applications in cyber-physical systems. In *Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'12)*. ACM, New York, NY, 229–238. DOI:http://dx.doi.org/10.1145/2387238.2387278

Wei Li, Flávia C. Delicato, Paulo F. Pires, and Albert Y. Zomaya. 2014b. Energy-efficient task allocation with quality of service provisioning for concurrent applications in multi-functional wireless sensor network systems. *Concurrency and Computation: Practice and Experience* 26, 11 (2014), 1869–1888. 10.1002/cpe.3107.

Wei Li, Flavia C. Delicato, and Albert Y. Zomaya. 2013. Adaptive energy-efficient scheduling for hierarchical wireless sensor networks. *ACM Transactions on Sensor Networks* 9, 3 (2013), 1–34. DOI:http://dx.doi.org/10.1145/2480730.2480736

Terje Nesbakken Lillegraven and Arnt Christian Wolden. 2010. *Design of a Bayesian Recommender System for Tourists Presenting a Solution to the Cold-Start User Problem*. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Ting Liu and Margaret Martonosi. 2003. Impala: A middleware system for managing autonomic, parallel sensor systems. In *Proceedings of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, New York, NY, 107–118. DOI:http://dx.doi.org/10.1145/781498.781516

Jef Maerine, Sam Michiels, Christophe Huygens, and Wouter Joosen. 2011. SASHA: A distributed protocol for secure application deployment in shared ad-hoc wireless sensor networks. In *2011 IEEE 8th International Conference on Mobile Ad hoc and Sensor Systems (MASS)*. 43–48. DOI:10.1109/MASS.2011.16

Sven Meyer and Andry Rakotonirainy. 2003. A survey of research on context-aware homes. In *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003, Vol. 21*. Australian Computer Society, Inc., Darlinghurst, Australia, 159–168.

David Moher, Alessandro Liberati, Jennifer Tetzlaff, and Douglas G. Altman. 2009. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Annals of Internal Medicine* 151, 4 (2009), 264–269. DOI:http://dx.doi.org/10.7326/0003-4819-151-4-200908180-00135

Álvaro Monares, Sergio F. Ochoa, Valeria Herskovic, Rodrigo Santos, and José A. Pino. 2014. Modeling interactions in human-centric wireless sensor networks. In *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 661–666. 10.1109/CSCWD.2014.6846923.

Luca Mottola and Gian Pietro Picco. 2011. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys* 43, 3 (2011), 1–51. DOI:http://dx.doi.org/10.1145/1922649.1922656

Amy L. Murphy, Gian Pietro Picco, and Gruia-Catalin Roman. 2001. LIME: A middleware for physical and logical mobility. In *21st International Conference on Distributed Computing Systems*. 524–533. 10.1109/ICDSC.2001.918983.

Eduardo F. Nakamura, Antonio A. F. Loureiro, and Alejandro C. Frery. 2007. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys* 39, 3 (2007), 9. DOI:http://dx.doi.org/10.1145/1267070.1267073

Roy Nirmalya V. Rajamani, and C. Julien. 2010. Supporting multi-fidelity-aware concurrent applications in dynamic sensor networks. In *Proceedings of the 8th 2010 IEEE International Conference on Pervasive*

Computing and Communications Workshops (PERCOM Workshops). IEEE Computer Society, Washington, DC, 43–49. DOI:http://dx.doi.org/10.1109/percomw.2010.5470601

Ramon Serna Oliver, Ivan Shcherbakov, and Gerhard Fohler. 2010. *An Operating System Abstraction Layer for Portable Applications in Wireless Sensor Networks*, ACM.

Miroslav Pajic, Alexander Chernoguzov, and Rahul Mangharam. 2013. Robust architectures for embedded wireless network control and actuation. *ACM Transactions on Embedded Computing Systems* 11, 4 (2013), 1–24. 10.1145/2362336.2362349.

Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. 2007. Service-oriented computing: State of the art and research challenges. *Computer* 40, 11 (2007), 38–45. DOI:http://dx.doi.org/10.1109/mc.2007.400

Pritee Parwekar. 2011. From internet of things towards cloud of things. In *Proceedings of the 2nd 2011 International Conference on Computer and Communication Technology (ICCCT 2011)*. IEEE Computer Society, Washington, DC, 329–333. DOI:http://dx.doi.org/10.1109/iccct.2011.6075156

Cesare Pautasso and Erik Wilde. 2009. *Why is the Web Loosely Coupled?: A Multi-Faceted Metric for Service Design*, ACM.

Ioan Raicu, Zhao Zhang, Mike Wilde, Ian Foster, Pete Beckman, Kamil Iskra, and Ben Clifford. 2008. *Toward Loosely Coupled Programming on Petascale Systems*, IEEE Press.

Atslands R. Rocha, Luci Pirmez, Flávia C. Delicato, Érico Lemos, Igor Santos, Danielo G. Gomes, and José Neuman de Souza. 2012. WSNs clustering based on semantic neighborhood relationships. *Computer Networks* 56, 5 (2012), 1627–1645. DOI:http://dx.doi.org/10.1016/j.comnet.2012.01.014

Taniro Rodrigues, Thais Batista, Flávia C. Delicato, Paulo F. Pires, and Albert Y. Zomaya. 2013. Model-driven approach for building efficient wireless sensor and actuator network applications. In *2013 4th International Workshop on Software Engineering for Sensor Network Applications (SESENA)*. 43–48. 10.1109/SESENA.2013.6612265.

Jesús Rodríguez-Molina, José-Fernán Martínez, Pedro Castillejo, and Lourdes López. 2013. Combining wireless sensor networks and semantic middleware for an internet of things-based sportsman/woman monitoring application. *Sensors* 13, 2 (2013), 1787–1835.

Alex Rogers, Mike Osborne, Sarvapali D. Ramchurn, Stephen Roberts, and Nicholas R. Jennings. 2008. Information Agents for Pervasive Sensor Networks. In *6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008)*. 294–299. 10.1109/PERCOM.2008.22.

Bartolome Rubio, Manuel Diaz, and Jose M. Troya. 2007. Programming approaches and challenges for wireless sensor networks. In *2nd International Conference on Systems and Networks Communications (ICSNC 2007)*. 36–36. 10.1109/ICSNC.2007.63.

Igor L. Santos, Luci Pirmez, Flavia C. Delicato, Samee U. Khan, and Albert Y. Zomaya. 2015. Olympus: The cloud of sensors. *IEEE Cloud Computing* 2, 2 (2015), 48–56. 10.1109/MCC.2015.43.

Douglas C. Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. 2000. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, John Wiley & Sons, Inc.

Christian Seeger, Kristof Van Laerhoven, and Alejandro Buchmann. 2014. MyHealthAssistant: An event-driven middleware for multiple medical applications on a smartphone-mediated body sensor network. *IEEE Journal of Biomedical and Health Informatics* PP, 99 (2014), 752–760. 10.1109/JBHI.2014.2326604.

Syed Yousaf Shah and Boleslaw K. Szymanski. 2012. Dynamic multipath routing of multi-priority traffic in wireless sensor networks. In *Proceedings of the 2012 6th Annual Conference of International Technology Alliance*.

Abraham Siberschatz and Peter B. Galvin. 1993. *Operating System Concepts* (4th. ed.). Addison-Wesley Longman Publishing Co., Inc.

SiliconLabs. 2013. *The Evolution of Wireless Sensor Networks*.

Doug Simon, Cristina Cifuentes, Dave Cleal, John Daniels, and Derek White. 2006. Java^{TM} on the bare metal of wireless sensor devices: The Squawk java virtual machine. In *Proceedings of the 2nd International Conference on Virtual Execution Environments*. ACM, New York, NY, 78–88. DOI:http://dx.doi.org/10.1145/1134760.1134773

Jatinder Singh and Jean Bacon. 2009. *Event-Based Data Dissemination Control in Healthcare*. Springer, Berlin.

Deborah Snoonian. 2003. Control systems: Smart buildings. *IEEE Spectrum* 40, 8 (2003), 18–23. DOI:http://dx.doi.org/10.1109/mspec.2003.1222043

Jan Steffan, Ludger Fiege, Mariano Cilia, and Alejandro Buchmann. 2005. Towards multipurpose wireless sensor networks. In *Proceedings of the 2005 Systems Communications*. IEEE Computer Society, Washington, DC, 336–341. DOI:http://dx.doi.org/10.1109/icw.2005.77

W. P. Stevens, G. J. Myers, and L. L. Constantine. 1974. Structured design. *IBM Systems Journal* 13, 2 (1974), 115–139. 10.1147/sj.132.0115.

Ryo Sugihara and Rajesh K. Gupta. 2008. Programming models for sensor networks: A survey. *ACM Transactions on Sensor Networks* 4, 2 (2008), 1–29. DOI:http://dx.doi.org/10.1145/1340771.1340774

Yuan Tian, Jarupan Boangoat, Eylem Ekici, and Füsun Özgüner. 2006. Real-time task mapping and scheduling for collaborative in-network processing in DVS-enabled wireless sensor networks. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing (IPDPS'06)*. IEEE Computer Society, Washington, DC, 10–25. DOI:http://dx.doi.org/10.1109/ipdps.2006.1639264

Sonam Tobgay, Rasmus L. Olsen, and Ramjee Prasad. 2011. *Architecture for Running Multiple Applications on a Single Wireless Sensor Network: A Proposal*. Springer, Berlin.

Panagiotis Trakadas, Helen Leligou, Theodore Zahariadis, Panagiotis Karkazis, and Lambros Sarakis. 2013. *Managing QoS for Future Internet Applications over Virtual Sensor Networks*. Springer, Berlin.

Gayathri Vijay, Elyes Ben Ali Bdira, and Mohamed Ibnkahla. 2011. Cognition in wireless sensor networks: A perspective. *IEEE Sensors Journal*, 11, 3 (2011), 582–592. DOI:http://dx.doi.org/10.1109/jsen.2010.2052033

Agustinus Borgy Waluyo, Wee-Soon Yeoh, Isaac Pek, Yihan Yong, and Xiang Chen. 2010. MobiSense: Mobile body sensor network for ambulatory monitoring. *ACM Transactions on Embedded Computing Systems* 10, 1 (2010), 1–30. 10.1145/ 1814539.1814552.

Webdust. *Webdust Project*.

Chengjie Wu, You Xu, Yixin Chen, and Chenyang Lu. 2012. Submodular game for distributed application allocation in shared sensor networks. In *2012 Proceedings IEEE INFOCOM*. IEEE Computer Society, Washington, DC, 127–135. DOI:http://dx.doi.org/10.1109/infcom.2012.6195490.

Tao Xie and Xiao Qin. 2008. An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems. *IEEE Transactions on Computers* 57, 3 (2008), 329–343. DOI:http://dx.doi.org/10.1109/tc.2007.70809

N. Xiong and P. Svensson. 2002. Multi-sensor management for information fusion: Issues and approaches. *Information Fusion* 3, 2 (2002), 163–186. DOI:http://dx.doi.org/10.1016/S1566-2535(02)00055-6

Yi Xu, Sumi Helal, My Thai, and Mark Scmalz. 2011. *Optimizing Push/Pull Envelopes for Energy-Efficient Cloud-Sensor Systems*, ACM.

You Xu, Abusayeed Saifullah, Yixin Chen, Chenyang Lu, and Sangeeta Bhattacharya. 2010. Near optimal multi-application allocation in shared sensor networks. In *Proceedings of the 11th ACM International Symposium on Mobile ad hoc Networking and Computing*. ACM, New York, NY, 181–190. DOI:http://dx.doi.org/10.1145/1860093.1860118

Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. 2008. Wireless sensor network survey. *Computer Networks* 52, 12 (2008), 2292–2330. http://dx.doi.org/10.1016/j.comnet.2008.04.002.

Ding Yong, M. A. Neumann, D. Gordon, T. Riedel, T. Miyaki, M. Beigl, Zhang Wenzhu, and Zhang Lin. 2012. A Platform-as-a-Service for in-situ development of wireless sensor network applications. In *Proceedings of the 2012 9th International Conference on Networked Sensing Systems (INSS)*. IEEE Computer Society, Washington, DC, 1–8. DOI:http://dx.doi.org/10.1109/inss.2012.6240527

Edward Yourdon and Larry L. Constantine. 1979. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc.

Yang Yu and Viktor K. Prasanna. 2005. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *Mobile Networks and Applications* 10, 1–2 (2005), 115–131.

Yang Yu, Loren J. Rittle, Vartika Bhandari, and Jason B. LeBrun. 2006. Supporting concurrent applications in wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 139–152. DOI:http://dx.doi.org/10.1145/1182807.1182822

Tian Yuan and E. Ekici. 2007. Cross-layer collaborative in-network processing in multihop wireless sensor networks. *IEEE Transactions on Mobile Computing* 6, 3 (2007), 297–310. DOI:http://dx.doi.org/10.1109/tmc.2007.39

Tian Yuan, E. Ekici, and F. Ozguner. 2005. Energy-constrained task mapping and scheduling in wireless sensor networks. In *Proceedings of the 2005 IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*. IEEE Computer Society, Washington, DC, 211–218. DOI:http://dx.doi.org/10.1109/mahss.2005.1542802

Qi Zhang, Lu Cheng, and Raouf Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1 (2010), 7–18. 10.1007/s13174-010-0007-6.

Albert Y. Zomaya. 1996. *Parallel and Distributed Computing Handbook*, McGraw-Hill, Inc., New York, NY.

Received August 2013; revised September 2015; accepted November 2015