Open access • Proceedings Article • DOI:10.1109/ICSNC.2007.63

Programming Approaches and Challenges for Wireless Sensor Networks

— Source link

Bartolomé Rubio, Manuel Díaz, José M. Troya

Published on: 25 Aug 2007 - International Conference on Systems and Networks Communications

Topics: Key distribution in wireless sensor networks, Wireless sensor network, Mobile wireless sensor network and Middleware (distributed applications)

Programming Approaches and Challenges for Wireless Sensor Networks [*]

Bartolomé Rubio, Manuel Díaz and José M. Troya
Dpto. Lenguajes y Ciencias de la Computación. Málaga University
29071 Málaga, SPAIN
(tolo,mdr,troya)@lcc.uma.es

Abstract

Wireless sensor networks (WSNs) constitute a new pervasive and ubiquitous technology. They have been successfully used in various application areas and in future computing environments, WSNs will play an increasingly important role. However, programming sensor networks and applications to be deployed in them is extremely challenging. It has traditionally been an error-prone task since it requires programming individual nodes, using low-level programming issues and interfacing with the hardware and the network. This aspect is currently changing as different high-level programming abstractions and middleware solutions are coming into the arena. Nevertheless, many research challenges are still open. This paper presents a survey of the current state-of-the-art in the field, establishing a classification and highlighting some likely research challenges and future directions.

1. Introduction

Due to a combination of recent technological advances in electronics, nanotechnology, wireless communications, computing, networking, and robotics, the development of Wireless Sensor Networks (WSNs) has been possible. They constitute a new form of distributed computing where sensors (tiny, low-cost and low-power nodes, colloquially referred to as "motes") deployed in the environment communicate wirelessly to gather and report information about physical phenomena [2]. WSNs offer numerous advantages over traditional systems, such as the large-scale flexible architecture (potentially hundreds or thousands of motes), high-resolution sensed data and application adaptive mechanisms. These unique characteristics make these systems very useful for a wide range of application areas, such as environmental monitoring, object and event detection, military surveillance, and precision agriculture [18].

[*]This work is supported by the EU funded project FP6 IST-5-033563 and the Spanish project TIN2005-09405-C02-01

However, these unique characteristics also make application development nontrivial. WSN programming has traditionally been an error-prone task since it requires programming individual nodes, using low-level programming issues and interfacing with the hardware and the network. The complexity of designing and implementing this kind of application makes the supply of higher-level abstractions of low-level functionality necessary in order to ease the application programmer task. Abstractions and middleware architectures, such as RPC, CORBA and Distributed Shared Memory, that have traditionally simplified and enabled the implementation of complex distributed systems, unfortunately cannot simply be applied to WSNs because they do not meet their special requirements [33] [43]:

- *Restricted resources*. Due to limited resources, the software components to be deployed in the motes should be lightweight. In addition, since it is anticipated that a WSN will execute multiple applications concurrently, it is very likely that performance requirements of all the running applications cannot be simultaneously satisfied. Therefore, it is necessary to provide mechanisms to optimize resource allocation and smartly trade the QoS of various applications against each other.
- *Network dynamics*. As an ad hoc network, a WSN may exhibit a highly dynamic topology due to mobility, communication failures or node failures. Programming paradigms and middleware should support the robust operation of WSNs despite these dynamics by adapting to the changing network environment.
- *Scale of deployments*. As stated before, a WSN is thought to consist of hundreds or thousands of nodes. In this sense, cluster-based architectures promote a more efficient use of resources in controlling large dynamic networks.
- *Data centric*. With the large population of sensor nodes, it may be impractical to pay attention to each individual node. Applications will focus on what data

is desired rather than on individual sensor nodes. For example, users would be more interested in querying which area(s) has(have) a temperature higher than $30°C$, or what the average temperature is in the south-east quadrant, rather than the temperature at sensor number 57.

- *Collection and processing of sensed data.* Most WSN applications involve nodes that contain redundant data and are located in a specific local region. This open up the possibility of in-network aggregation of data from different sources, eliminating redundancy and minimizing the number of transmissions to the sink. This saves considerable energy and resources, given that communication costs are much higher than computation costs.

Some additional requirements have also to be met in the growing and promising field of Wireless Sensor and Actor Networks (WSANs) [1], a variation of WSNs where the devices deployed in the environment are not only sensors able to sense environmental data, but also actors (also referred to as actuators in the literature) able to react by affecting the environment. Actors are resource rich nodes equipped with better processing capabilities, higher transmission power and longer battery life than sensors. WSANs have two major requirements:

- *Coordination.* Coordination mechanisms are needed for both sensor-actor and actor-actor interactions. In particular, sensor-actor coordination provides the transmission of sensed data from sensors to actors. After receiving sensed data, actors need to coordinate with each other in order to make decisions on the most appropriate way to perform the action.

- *Real-time.* On the other hand, depending on the application there may be a need to respond rapidly to sensor input. Moreover, the collected and delivered sensor data must still be valid at the time of acting. Therefore, the issue of real-time communication is very important in WSANs.

In the last few years, much work has targeted the development of programming support in the effort to meet all these requirements. Different high-level programming abstractions and middleware have appeared as promising solutions to address the challenges of this kind of system. This paper presents a survey of the current state-of-the-art in the field, establishing a classification covering a broad range of approaches and highlighting some open research challenges and future directions.

Some previous research has surveyed WSNs. Ian Akyildiz and his colleagues focused more on characteristics and challenges for WSNs and WSANs than on programming

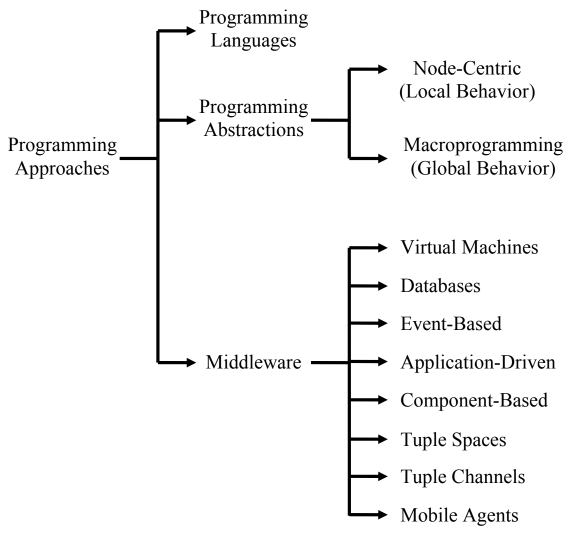

Figure 1. Classification of Programming Approaches for WSNs.

support in [2] and [1], respectively. Other surveys were focused on potential WSN applications [13] [18]. Others presented different routing protocols [24]. Finally, there are some surveys that particularly tackle the WSN programming issue [33] [23], but they target particularly middleware support. Our approach also considers programming abstraction proposals that have appeared in the literature. Moreover, this paper establishes a broader middleware support classification than the previous ones.

The rest of the paper is structured as follows. Section 2 establishes a classification for WSN programming approaches, describing the main features of representative work. In Section 3 some open research challenges and future directions for WSNs are highlighted. Finally, some conclusions are sketched in Section 4.

2. Programming Approaches for WSNs: A Classification

Figure 1 depicts the established taxonomy in order to classify the different programming approaches appeared for WSNs. In the following subsections the characteristics of each group are detailed and several proposals are identified.

2.1. Programming Languages

The first option we have to develop an application for WSNs is to directly use an existing programming language. The most popular programming languages for tiny embedded systems are C and nesC [17]. Currently, nesC is the most used in WSNs. It is a C-based programming language with a programming model that incorporates event-driven execution, a flexible concurrency model, and component-oriented application design. There is no dynamic memory allocation and the call-graph is fully known at compile-time. These restrictions make whole program analysis (for safety) and optimizations (for performance) significantly simpler and more accurate. The nesC component model and parameterized interfaces eliminate many needs for dynamic memory allocation and dynamic dispatch. In order to implement TinyOS [37], a simple but highly concurrent open source operating system, nesC has been used. Both have been adopted by a large number of WSN research groups. They are also used to implement the runtime support and infrastructures needed in most of the approaches discussed in the following sections.

2.2. Programming Abstractions

By programming abstractions we mean high-level abstractions that, supported by a suitable programming model, compiler and runtime support, liberate the programmer from having to address the low-level WSN mechanisms such as messaging and routing protocols, data caches and neighbor lists. Basically, we can establish two main types of programming abstractions: Node-centric or local behavior approaches, which are centered on individual nodes, and macroprogramming or global behavior approaches, which focus on the behavior of a WSN as a whole.

Node-Centric (Local Behavior)

In node-centric programming, the programmer has to translate the global application behavior in terms of local actions on each node, and individually program the sensor nodes using the corresponding programming model. Proposals such as Hood [40], Abstract Regions [39], Logical Neighborhoods [30] and Virtual Nodes [9] belong to this class. We discuss Logical Neighborhoods as a representative of them. In a WSN, a node is able to exchange data directly only with the surrounding nodes located within its communication radius (physical neighborhood). This proposal introduces the abstraction of logical neighborhoods, whose span is not inherently limited by the physical communication range, but is controlled by the programmer using applicative and contextual information. Logical neighborhoods provide the programmer with a higher-level, application-defined notion of proximity. The span of a logical neighborhood is specified declaratively based on the characteristics of nodes, along with requirements about communication costs. For example, the programmer can establish a logical neighborhood for a node declaring that it will be formed by nodes hosting temperature sensors, that are currently showing a reading with a value higher than a given threshold and that are at a maximum of 2 hops away. The declarative language used is conceived to be a simple extension to existing WSN programming languages such as nesC.

Macroprogramming (Global Behavior)

This type introduces a new and completely different view on how to program WSNs. Macroprogramming involves programming the network as a whole, rather than writing software to drive individual nodes. The WSN global behavior is programmed at a high-level specification, enabling automatically generated nodal behaviors. This relieves application developers from having to deal with concerns at each network node. Regiment [31], Kairos [20] and ATaG [3] are significant contributions to this type of abstraction. We discuss ATaG as representative of them. The Abstract Task Graph (ATaG) seeks to raise the level of programming abstraction by allowing the architecture-independent specification of application behavior and transferring the responsibility of low level coordination, communication and optimization to an underlying runtime system, thereby allowing the application developer to focus on high-level behavioral aspects. To accomplish this, ATaG employs a data driven programming model and mixed imperative-declarative program specification for separation of concerns. Tasks are defined in terms of their input and output data objects. Availability of operands triggers task execution, subject to firing rules. The mixed imperative-declarative specification separates the ”when and where” of processing from the ”what”. The same program can be compiled for a different network size and topology by interpreting the declarative (”when and where”) part in the context of that network architecture, while the imperative (”what”) part remains unchanged.

2.3. Middleware

The main purpose of middleware is to support the development, maintenance, deployment and execution of applications, filling in the gap between the application layer and the hardware, operating system and network stack layers. In the case of a WSN, this includes mechanisms for formulating complex high-level sensing tasks, communicating them to the WSN, coordination of sensor nodes to split tasks and distribute them to the individual nodes, data fusion for merging the sensor readings into high-level result, and reporting it. Moreover, appropriate abstractions and mechanisms for dealing with the heterogeneity of sensor nodes should be provided [33]. We have established the following types of middleware which most of the proposals appeared fit into.

Virtual Machines

This category allows the developers to write applications in separate, small modules. The system injects and distributes the modules through the network using tailored algorithms, and therefore overall energy consumption and resource use are minimized. The Virtual Machine (VM) then interprets the modules. Solutions in this category include Maté [26], ASVM [27] and DAViM [29]. The most representative is probably Maté. It is a byte code interpreter that runs on TinyOS. It uses codes broken into capsules of 24 byte-long instructions, to the benefit of large programs, which are made up of multiple capsules that are easily injected into the network. Maté's key components are the VM, the network, the logger, the hardware and the boot/scheduler. Using a synchronous model that begins execution in response to an event such as a packet transmission or a time out, Maté avoids buffering and large storage. The synchronous model makes application-level programming simpler and far less prone to bugs than dealing with asynchronous event notifications.

Databases

Examples of this category are Cougar [6], TinyDB [28] and SINA [34]. The first two proposals are based on pure database systems, which essentially provide a distributed database solution appropriate for resource-constrained sensor networks, focusing on efficient query routing and processing. SINA differs in that it uses an SQL-like query language for expressing queries, but also provides other functions which are outside the scope of traditional database systems. It provides support for scripting in such a way that sensor hardware access, communication and event handling can be managed. SINA uses an attribute-based naming scheme in order to facilitate the data-centric characteristics of sensor queries and it allows hierarchical clustering of sensor nodes in order to facilitate scalable operations within sensor networks.

Event-Based

Another approach to WSN middleware is based on the notion of events. Here, the application specifies interest in certain state changes of the real world (basic events). Upon detecting such an event, a sensor node sends a so-called event notification towards interested applications. The application can also specify certain patterns of events (compound events), such that the application is only notified if occurred events match these patterns. In [42], a reasonably sophisticated set of event operators for describing event patterns in sensor networks has been produced. A crucial limitation of this solution is the complexity that is necessarily involved in implementing it. In contrast, the Mires middleware [35] is a more pragmatic publish/subscribe solution that has been designed and implemented to run on TinyOS using nesC. It adopts a component-based programming model using active messages to implement its publish-subscribe-based communication infrastructure.

Application-Driven

This approach allows programmers to fine-tune the network on the basis of application requirements, that is, applications will dictate network operations management, providing a QoS advantage. However, the tight coupling with applications might result in specialized, not general purpose, middleware. The MiLAN middleware [22] is an example of this category. It takes an approach building on existing networking and service discovery protocols. Applications specify their sensing requirements to the middleware through a standard API, in terms of graphs describing sensor quality of service and state-based variable requirements. MiLAN's application-driven network management is well suited to application adaptation and it effectively tackles the challenges of openness and scalability.

Component-Based

Component-Based Software Engineering (CBSE) is a modern methodology that proposes software construction by plugging software components [21]. Based on component interoperability, this programming style allows the creation of more flexible and adaptable software. Recently, some middleware proposals based on CBSE are appearing in the field of WSNs. In [10], a reconfigurable component-based middleware for networked embedded systems, called RUNES, is introduced. This approach comprises two distinct layers: a foundation layer, called the middleware kernel, which is the runtime realization of a simple but well-defined software component model, and an on top layer of component frameworks that offer a configurable and extensible set of both middleware and application services. Other example is MWSAN [4], a real-time component-oriented middleware for WSANs, which provides a set of high level services for sensors and actors. Besides considering the real-time as a major aspect, it also takes into account issues such as the network configuration and the quality of service (QoS).

Tuple Spaces

The coordination needs in WSNs and WSANs have attracted the attention of the Coordination paradigm community [7]. More specifically, different coordination models and middleware based on the Linda abstract model [19] have appeared in the area of sensor networks. Linda can be considered the most representative coordination language. It is based on a shared memory model where data is represented by elementary data structures called tuples, and the memory is a multiset of tuples called a tuple space. Examples of this type of middleware are TinyLime [14] and TeenyLime [11]. In TinyLime, a new operational scenario is assumed, one that naturally provides contextual information, does not require multi-hop communication among sensors, and places reasonable computation and communication demands on the motes. Sensors are sparsely distributed in an environment, not necessarily able to communicate with each other, and a set of mobile base stations (laptops) move through the space accessing the data of nearby sensors. Each base station owns a tuple space and federated tuple spaces can be established in order to communicate and synchronize several base stations and some client hosts. TeenyLimne is an evolution of TinyLime to be applied in WSANs. As in TinyLime, the core abstraction is the transiently shared tuple space, but in this case the spaces are physically located on the sensors themselves.

Tuple Channels

An alternative to tuple spaces is the proposal based on the use of tuple channels to carry out communication and synchronization among the WSN nodes involved. Several advantages can be obtained from the use of channels with respect to shared memory models:

- *Architectural expressiveness*. Like messaging, using channels to express the communication carried out within a distributed system is architecturally much more expressive than using shared data spaces. With a shared data space, it is difficult to see which components exchange data with each other.

- Channels support *data streams in a natural and suitable way*. The application programmer does not have to deal with head and tail tuples as is necessary in a tuple space based approach to implement information streams. This is particularly important in information-flow applications, such as building WSNs.

- Channel interconnection provides great flexibility for the definition of *complex and dynamic interaction protocols*. Sensor data dissemination can be achieved elegantly, allowing for data redirection, data aggregation and redundant data elimination.

A representative of this category is TCMote [15]. This middleware is thought to support an operational setting based on a (hierarchical) architecture of sensor regions, each one governed by a leader with higher capabilities (power, memory, processing ability) than the rest of the region nodes (motes). A region leader host owns a tuple channel space, which stores tuple channels used to carried out communication and synchronization between region sensors and the leader in a single-hop way. A tuple channel is a FIFO structure that allows one-to-many and many-to-one communication of data structures, represented by tuples. Channel consumption behavior contributes to dealing with the data-centric characteristics of sensor queries. In addition, tuple channels can be dynamically interconnected through the use of predefined and user-defined connectors, providing great flexibility for the definition of different topologies. Recently, a new approach, called TC-WSANs [5], adapts and extends TCMote with real-time characteristics in order to satisfy this important requirement in WSAN systems.

Mobile Agents

In the traditional client/server-based computing architecture, data at multiple sources are transferred to a destination, whereas in the mobile agent based computing paradigm, a task-specific executable code traverses the relevant sources to gather data. Mobile agents can be used to reduce the communication cost, by moving the processing function to the data rather than bringing the data to a central node.

Recently, mobile agents have been proposed for efficient data dissemination in WSNs. Some proposals are Agilla [16], MAWSN [8] and actorNet [25]. We discuss the former as representative. It could also be included in the tuple space middleware category as agents coordinate through tuple spaces. Agilla facilitates the rapid deployment of adaptive applications in WSNs. It allows the programmer to create and inject mobile agents, which can migrate across the WSN performing application-specific tasks. Mobile agents can intelligently move or clone themselves to desired locations in response to changing conditions in the environment. Each node maintains a local tuple space, and different agents can coordinate through local or remote operations on these tuple spaces. This fluidity of code and state has the potential to transform a WSN into a shared, general-purpose computing platform capable of running several autonomous applications at a time.

3. Programming Challenges for WSNs

As discussed in the previous section, lots of approaches have targeted the development of programming support in the effort to meet the requirements of WSNs. However, many research challenges are still ahead faced. In this section we outline some of them and envision future directions. The three categories of programming approaches established before are affected by them.

3.1. Incorporation of Java

Sensor nodes are devices with scarce resources. For example, the well-known and most broadly used Crossbow Micaz/Mica2 family motes [12] have a 8 MHz Atmel ATMega128L 8-bit microprocessor, 4 KB of RAM and a flash memory with 640 KB (128 KB for program and 512 KB for user data). These constraints have influenced the characteristics that a programming language must have in order to be used in this kind of system, as was discussed in section 2.1.

However, more powerful motes are recently appearing. For example, the Imote2 from Crossbow has an Intel PXA271 32-bit XScale processor at 13–416 MHz, 256 KB of SRAM, 32 MB of SDRAM and 32 MB of flash memory. This may influence the arrival of other programming languages to the WSN area, providing it with new characteristics and advantages. In this sense, Java may be one of the most required languages (ease of development, security, dynamic capabilities for developer productivity, etc.). Squawk [36] is a first attempt to incorporate Java into WSNs. It is the Java VM for the Sun Small Programmable

Object Technologoy (Sun SPOT) wireless sensor/actuator device designed by Sun Labs. Squawk implements a full Java Platform, Micro Edition (Java ME) VM. We envision more approaches in the next few years.

3.2. Efficient Support for Programming Abstractions

As stated in section 2.2, programming abstractions liberate the programmer from having to address the low-level WSN mechanisms. However, these abstractions will only be useful if they are supported by a suitable compiler and runtime support, especially the necessary novel routing protocols to address the peculiarities of the established abstraction. In this sense, some work is being carried out as in the macroprogramming abstraction ATaG, where the node-centric abstraction Logical Neighborhood has been used as the target API of the compilation process, and as the underlying support to manage communication among the nodes [32]. Nevertheless, much work has to be done in order to achieve optimizations at the compilation phase and full end-to-end application development frameworks for programming WSNs using programming abstractions.

3.3. Reprogramming

In most applications, WSNs are deployed once and are intended to operate unattended for a long period of time. One of the key research challenges is to manage WSNs in such a way that they can be dynamically customized to various (unanticipated) circumstances. Since resource limitations prevent sensors from having an extensive set of services pre-installed, sensor software should be dynamically reconfigurable. In order to develop a practical and efficient reprogramming system, many challenges have to be addressed. Because of the characteristics of sensors and wireless communications, the following points should be considered:

- The time and space complexity of algorithms in reprogramming should be well matched to the capacity profile of a sensor node.
- Reprogramming requires the program code to be delivered in its entirety. However, wireless communication is unreliable due to possible signal collisions and interference. This makes reliable protocol designs more challenging.
- Scalability is crucial for large-scale sensor network deployment. Code dissemination and scope selection (any particular nodes in the network selected for reprogramming) will be conditioned by this issue.
- Reprogramming should be energy efficient. After being received over the air, the new program codes are usually stored in an external flash memory. A sensor node will then switch to the new program. Among computing, sensing and communication functions, the latter consumes a large portion of the energy compared to the other ones. In addition, the writing operation to the flash memory consumes four times more energy than sending a packet.

Several reprogramming systems have been designed in the past few years. In [38] various approaches are discussed, and their limitations established. There are lots of unresolved problems that need further investigation to make reprogramming highly usable and efficient. Code dissemination is a continuing focus of current research. Approaches such as Agilla, RUNES and DAViM, previously classified and discussed in section 2, have taken this issue into account. However, design trade-offs and impact factors have not been fully understood. There has been little research on scope selection, completion validation (no errors in a new received program) and code acquisition functions (initiated from targeted sensor nodes). In addition, for practical use, security measures in reprogramming need to be considered.

3.4. Heterogeneity

The presence of heterogeneity in a WSN is known to increase network reliability and lifetime. We can identify three common types of hardware heterogeneity: computational heterogeneity where some nodes have added computational power (e.g. Intel's Stargate and Imote2), link heterogeneity where some nodes have long-distance highly reliable communication links (e.g. 802.11 connectivity), and energy heterogeneity where nodes have unlimited energy resources (e.g. connection to a wall socket). Some work has been carried out to try to answer unexplored questions such as where, how many and what types of heterogeneous resources should be deployed to maximize benefit, mainly from low-level point of views such as MAC and routing protocols [41].

From the programming perspective, a key assumption of most of the programming solutions discussed in section 2 is that the nodes in a sensor network are resource constrained and homogeneous. Two exceptions could be the tuple channel based middleware TCMote and the component-based approach RUNES. In the former, nodes with high capabilities of power, memory and processing act as the leaders of regions/clusters of sensors. The application programmer is aware of this operational setting, and then s/he can take advantage of the present heterogeneity. But at the same time, s/he use the same high-level coordination model primitives to program both leaders and sensors. In the later, device heterogeneity is handled by exploiting different implementations of the middleware. Even if these implementations clearly differ in the underlying technology used, they nevertheless provide the same, simple, component-based programming environment to the application developers.

The homogeneity assumption is too restrictive in light of sensor network-based applications envisioned for the future. New programming solutions for sensor networks must be more generic and assume heterogeneous sensor hardware and diverse communication mechanisms.

3.5. WSAN challenges

As stated in section 1, Wireless Sensor and Actor Networks (WSANs) constitute a variation of WSNs where the devices deployed in the environment are not only sensors able to sense environmental data, but also actors (resource rich nodes) able to react by affecting the environment. In [1] different research challenges were established for WSANs. These challenges are mainly influenced by the two major requirements of these systems: coordination (needed for both sensor-actor and actor-actor interactions) and real-time. We might say that they are still open issues as less work has tried to tackle them. Recently, some approaches have proposed high-level constructs to ease the application programmer task and to address the challenges of WSANs. The aforementioned TeenyLime middleware and Virtual Nodes programming abstraction are two examples. However, neither of them directly deal with the real-time requirement of WSANs. This major requirement is tackled by MWSAN and TC-WSANs approaches, but more proposals are required in this growing and promising field of WSANs.

4 Conclusions

This paper has established a taxonomy in order to classify the programming approaches appeared in the research area of wireless sensor networks. Three main categories have been considered: programming languages, programming abstractions and middleware. Several proposals have been identified and classified, and representative approaches have been discussed and compared.

In spite of the significant effort carried out in this area, there are still a lot of research challenges to be tackled. We have also highlighted several important research challenges. Some preliminary work dealing with them has been identified and future directions have been outlined.

References

[1] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal*, 2(4):351–367, 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks Journal*, 38(4):393–422, 2002.

[3] A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner. The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems. In *Proceedings of the International Workshop on End-to-end Sense-and-respond Systems (EESR'05)*, pages 19–24, 2005.

[4] J. Barbarán, M. Díaz, I. Esteve, D. Garrido, L. Llopis, and B. Rubio. A Real-Time Component-Oriented Middleware for Wireless Sensor and Actor Networks. In *Proceedings of the IEEE International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07)*, pages 3–10, Vienna, Austria, April 2007. IEEE Computer Society Press.

[5] J. Barbarán, M. Díaz, I. Esteve, D. Garrido, L. Llopis, B. Rubio, and J. Troya. Programming Wireless Sensor and Actor Networks with TC-WSANs. In *Proceedings of the IEEE International Conference on Pervasive Services (ICPS'07)*, Istanbul, Turkey, July 2007. IEEE Computer Society Press.

[6] P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. In *Proceedings of the 2th International Conference on Mobile Data Management (MDM'01)*, pages 3–14. Springer. LNCS. vol. 1987, 2001.

[7] N. Carriero and D. Gelernter. Coordination languages and their significance. *Communications of the ACM*, 35(2):97–107, 1992.

[8] M. Chen, T. Kwon, Y. Yuan, and V. Leung. Mobile agent based wireless sensor networks. *Journal of Computers*, 1(1):14–21, 2006.

[9] P. Ciciriello, L. Mottola, and G. Picco. Building Virtual Sensors and Actuators over Logical Neighborhoods. In *Proceedings of the 1st International Workshop on Middleware for Wireless Sensor Networks (MidSens 2006), co-located with the 7th International Middleware Conference (Middleware'06)*, pages 19–24, Melbourne, Australia, November 2006. ACM Press.

[10] P. Costa, G. Coulson, C. Mascolo, L. Mottola, G. P. Picco, and S. Zachariadis. Reconfigurable component-based middleware for networked embedded systems. *International Journal of Wireless Information Networks*, 2007.

[11] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco. TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks. In *Proceedings of the 1st International Workshop on Middleware for Wireless Sensor Networks (MidSens 2006), co-located with the 7th International Middleware Conference (Middleware'06)*, pages 43–48, Melbourne, Australia, November 2006. ACM Press.

[12] Crossbow. http://www.xbow.com/.

[13] D. Culler, D. Estrin, and M. Srivastava. Sensor network applications. *IEEE Computer*, 8(37):50–78, 2004.

[14] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco. TinyLime: Bridging Mobile and Sensor Networks through Middleware. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pages 61–72, Kauai Island (Hawaii, USA), March 2005. IEEE Computer Society Press.

[15] M. Díaz, B. Rubio, and J. M. Troya. A Coordination Middleware for Wireless Sensor Networks. In *Proceedings of the*

IEEE International Conference on Sensor Networks (SENET 2005), pages 377–382, Montreal, Canada, August 2005. IEEE Computer Society Press.

[16] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid Development and flexible deployment of adaptative wireless sensor network applications. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 653–662, Columbus, Ohio, USA, June 2005. IEEE Computer Society Press.

[17] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI 2003)*, pages 1–11, San Diego, California, USA, June 2003. ACM Press.

[18] J. Gehrke and L. Liu. Sensor-network applications. *IEEE Internet Computing*, 10(2), 2006.

[19] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[20] R. Gummadi, O. Gnawali, and R. Govidan. Macro-Programming Wireless Sensor Networks using Kairos. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS'05)*, pages 126–140. Springer. LNCS vol. 3560, 2005.

[21] G. Heineman and W. Councill. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley: Reading, MA, 2001.

[22] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *IEEE Network*, 1(18):6–114, 2004.

[23] K. Henricksen and R. Robinson. Middleware for Sensor Networks: State-of-the-Art and Future Directions. In *Proceedings of the 1st International Workshop on Middleware for Wireless Sensor Networks (MidSens 2006), co-located with the 7th International Middleware Conference (Middleware'06)*, Melbourne, Australia, November 2006. ACM Press.

[24] Q. Jiang and D. Manivannan. Routing Protocols for Sensor Networks. In *Proceedings of the 1st IEEE Consumer Communications and Networking Conference (CCNC'04)*, pages 93–98, Las Vegas, Nevada, USA, January 2004.

[25] Y. Kwon, S. Sundresh, K. Mechitov, and G. Agha. ActorNet: An Actor Platform for Wireless Sensor Networks. In *Proceedings of the IEEE International Joint Conference on Autonomous Agents and Multiagent Systems*, Hakodate, Japan, May 2006.

[26] P. Levis and D. Culler. 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'02)*, San Jose, CA, USA, October 2002.

[27] P. Levis, D. Gay, and D. Culler. Active Sensor Networks. In *Proceedings of the 2nh International Symposium on Networked Systems Design and Implementation (NSDI'05)*, pages 29–42, San Francisco, CA, USA, March 2005.

[28] S. Madden, M. Franklin, M. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 1(30):122–173, 2005.

[29] S. Michiels, W. Horré, W. Joosen, and P. Verbaeten. DAViM: a Dynamically Adaptable Virtual Machine for Sensor Networks. In *Proceedings of the 1st International Workshop on Middleware for Sensor Networks (MidSens'06), co-located with the 7th International Middleware Conference (Middleware'06)*, Melbourne, Australia, November 2006.

[30] L. Mottola and G. P. Picco. Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS '06)*, San Francisco, CA, USA, June 2006.

[31] R. Newton and M. Welsh. Regions Streams: Functional Macroprogramming for Sensor Networks. In *Proceedings of the 1st International Workshop on Data Management for Sensor Networks (DMSN'04)*, pages 78–87, 2004.

[32] A. Pathak, L. Mottola, A. Bakshi, V. Prasanna, and G. Picco. Expressing Sensor Network Interaction Patterns using Data-Driven Macroprogrammming. In *Proceedings of the 3rd International Conference on Software Engineering (ICSE'07)*, Minneapolis, USA, May 2007.

[33] K. Romer, O. Kastem, and F. Mattern. Middleware Challenges for Wireless Sensor Networks. *ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, 2(4):59–61, 2002.

[34] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, pages 52–59, August 2001.

[35] E. Souto, G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz. A Message-Oriented Middleware for Sensor Networks. In *Proceedings of the 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2004)*, pages 127–134, Toronto, Canada, October 2004.

[36] Squawk. http://research.sun.com/projects/squawk.

[37] TinyOS. http://www.tinyos.net/.

[38] Q. Wang, Y. Zhu, and L. Cheng. Reprogramming wireless sensor networks: Challenges and approaches. *IEEE Network Magazine*, pages 48–55, May/June 2006.

[39] M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *Proceedings of the 1st UNENIX-ACM Symposium on Networked System Design and Implementation (NSDI'04)*, pages 29–42, 2004.

[40] W. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: a Neighborhood abstraction for sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services (MobiSYS'04)*, pages 99–110, New York, NY, USA, 2004.

[41] M. Yarvism, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting Heterogeneity in Sensor Networks. In *Proceedings of the IEEE Infocom*, Miami, FL, USA, March 2005.

[42] E. Yoneki and J. Bacon. Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments. In *Proceedings of the IFIP International Conference on Cooperative Information Systems (CoopIS'05)*, pages 366–384. Springer. LNCS vol. 3760, 2005.

[43] Y. Yu, B. Krishnamachari, and V. Prasanna. Issues in designing middleware for wireless sensor networks. *IEEE Network Magazine*, January 2004.