Hardware and Software Platforms for Distributed Computing on Resource Constrained Devices

Gloria Martorella, Daniele Peri and Elena Toscano

Abstract The basic idea of distributed computing is that it is possible to solve a large problem by using the resources of various computing devices connected in a network. Each device interacts with each other in order to process a part of a problem, contributing to the achievement of a global solution. Wireless sensor networks (WSNs) are an example of distributed computing on low resources devices. WSNs encountered a considerable success in many application areas. Due to the constraints related to the small sensor nodes capabilities, distributed computing in WSNs allows to perform complex tasks in a collaborative way, reducing power consumption and increasing battery life. Many hardware platforms compose the ecosystem of WSNs and some lightweight operating systems have also been designed to ease application deployment, to ensure efficient resources management, and to decrease energy consumption. In this chapter we focus on distributed computing from several points of view emphasizing important aspects, ranging from hardware platforms to applications on resource constrained devices.

1 Introduction

Distributed computing in Wireless Sensor Networks (WSNs) represents an emerging scenario obtaining the attention of researchers. Performing individually a smaller task, sensor nodes of a WSN collaborate exchanging information, data or partial

G. Martorella · D. Peri (✉)
Dipartimento di Ingegneria Chimica Gestionale Informatica Meccanica, Università degli Studi di Palermo, Viale delle Scienze, Edificio 6, 90128 Palermo, Italy
e-mail: daniele.peri@unipa.it

E. Toscano
Dipartimento di Matematica e Informatica, Università degli Studi di Palermo, via Archirafi 34, 90123 Palermo, Italy
e-mail: elena.toscano@unipa.it

results to achieve a global goal. A WSN is composed of a variable number of autonomous sensor nodes deployed in the environment. Basically, a sensor node includes a processor unit, a radio transceiver for data transmission, small memory for data storage and sensor modules for data gathering. Moreover, a battery with a limited energy budget is often the only power source. Therefore, sensor nodes present several constraints in terms of storage, processing and energy. They are often deployed in a hostile environment that individuals cannot easily reach and consequently changing or charging batteries is unsuitable.

Due to the sensor's small memory and to the low processing capability, solving complex problems individually is often impossible and thus, some form of cooperation is required. Therefore, distributed computing in WSN allows to achieve complex targets by using cheap sensor nodes with a considerable cost saving and without a single point of failure. The advantages include higher performance, reliability, collaboration and scalability. In spite of this, distributed computing defines a scenario still full of challenges since resources management techniques and power saving strategies are required. Research is thus naturally focused on the creation of new platforms for specific application fields, and on the development of distributed computing techniques and algorithms improving the performances of WSNs.

Sensor nodes have earned the attention of the market with considerable success because of their low price and their potential in different fields. In addition to the traditional well known platforms, there are several low-cost hardware platforms and starter kits on the market, often including lightweight operating systems (OS) that allow anyone to create their own sensor network. In this chapter we provide an overview about distributed computing on low resources constrained devices such as sensor nodes of a WSN. Many chapters in literature discuss distributed computing but they focus just on one significant aspect such as algorithms, applications, challenges or issues. Unlike these papers, we cover several significant aspects of distributed computing ranging from low to high level, from hardware platforms to distributed applications, also describing the underlying algorithms and issues. We also discuss a typical distributed application showing how even a trivial problem may become complicated on resource-constrained hardware.

The remainder of this chapter is organized as follows. Section 2 presents some recent hardware platforms for WSN, while Sect. 3 describes some of the existing operating systems for sensor node's resources management. A common representation of sensor nodes is provided by sensor ontologies, briefly discussed in Sect. 4. Distributed computing algorithms and applications in WSN are surveyed in Sect. 5, while Sect. 6 provides a brief characterization of distributed synchronization as a real example of the issues and challenges related to distributed computing on low resources devices. Finally Sect. 7 concludes this chapter.

2 Hardware Platforms

Distributed computing cannot be discussed separately from the specific underlying hardware implementation. In recent years many hardware platforms for WSN have been designed, tested and brought to market. These platforms present some common features since basically they include a microcontroller, a transceiver, low memory, power supply and expansion pins. These platforms differ in the on-board hardware components and in their actual use in several applications. Many available platforms have been used in several research projects such as the well-known MicaZ, Iris, Mica2 and Telos platforms. More recently, other hardware platforms are earning the market thanks to their low cost and high expandability through a large set of sensor boards.

In [22], the authors presents the commercially available **Atlas** sensor platform for the creation of pervasive smart spaces. It is composed of three basic layers: the Processing Layer, the Communication Layer, and the Device Connection Layer. Other add-ons provide additional capabilities to the node. The Processing Layer is based on Atmega 128L microcontroller including 128 KB Flash memory, 4 KB SRAM, 4 KB EEPROM. In addition, 64 KB of expanded SRAM is added. The Communication Layer is responsible of data transfer over the network. This layer can be based on wired 10BaseT Ethernet, Bluetooth, 802.11b WiFi and USB. The Connection Layer allows to connect up to 32 sensors and actuators to the platform. Atlas node is configured through an interface allowing the selection of the currently connected sensors and actuators. The authors of [6] describe Atlas and cases study based on it.

SHIMMER [7] is a platform designed for healthcare monitoring. It is based on Texas Instruments MSP430 microcontroller including 10 KB RAM, 48KB Flash, 8 ADC channels, USART and SPI connections. SHIMMER uses a CHIPCON CC2420 radio transceiver and Roving Networks RN-41 Class 2 Bluetooth module for communications. A MICRO SD Flash for additional storage is also on-board. Internal and external connectors allows the integration of other boards adding kynematics, ECG and GSR sensing capabilities, while a three axis accelerometer is already built-in.

Waspmote [41] is a general-purpose board to develop applications for WSNs. It is based on an Atmega1281 microcontroller running at 14 MHz, 8 KB SRAM, 4 KB of EEPROM, 128 KB of FLASH and a 2 GB microSD. A temperature sensor and an accelerometer, a radio socket, a SPI-UART socket, a solar socket and a battery socket are included. A 802.15.4/Zigbee module is used for communication with other nodes in the network and for node programming. Nodes can be programmed with Over The Air Programming (OTA) through 802.15.4 modules or 3G/GPRS/WiFi modules via FTP. Secure communication can also be implemented on Waspmotes by using AES 128, RSA 1024 and AES 256 encryption library provided implementations. Waspmote is well expandable with several modules for communication like RFID, Bluetooth for nodes discovery, radio expansion boards (for using two radios at the same time) and GPS. Sensor boards provide additional sensing capabilities to the node. Case studies affect smart agriculture projects, smart water and smart cities projects. Waspmote has also been used in the deployment of long distance WSNs [44].

Arduino is an open source general-purpose platform for developing a wide range of applications involving sensors, actuators and communication with other devices. The philosophy of Arduino is based on prototyping, tinkering and patching concepts. Arduino boards can be purchased pre-assembled or can be assembled manually. There are 14 commercially available Arduino versions. Typically an Arduino board includes an Atmel AVR8 microcontroller and it is expandable using other shields. Only Arduino Due includes an Atmel SAM3X8E ARM Cortex-M3 CPU. For example the Arduino Uno board [2] is based on Atmega328 running at 16 MHz including 32 KB Flash memory, 2 KB SRAM and 1 KB EEPROM. The board can be powered through either the USB connector or an external power supply. On board, digital IO pins, analog input and output pins are available. An integrated development environment (IDE) is freely available to developers. Thanks to this IDE, programs written in the Wiring programming language are easily cross-compiled on a host OS such as Windows, Macintosh and Linux, and deployed on Arduino boards through USB programming. Developers can add several Arduino kits including all the necessary for developing applications. In addition several shields, actuators and sensors are available for this platform. Several researches concerning WSN are based on this platform, while projects for beginners can be found in [4].

Zolertia **Z1** [45] is a low-power general-purpose platform based on Texas Instruments MSP430F2617 microcontroller running at 16 MHz including 92 KB flash and 8 KB RAM. Z1 is equipped with a 2.4 GHz CC22420 transceiver, a TMP102 temperature sensor and a ADXL345 3 axis digital accelerometer, 52-pin expansion connector, 2 ports for connecting Phidget sensors, a micro-USB connector and a ceramic antenna. It can be powered via USB or through batteries. Z1 is expandable with several on-board sensors like a temperature-humidity sensor, a light sensor, a barometric sensor, and it can also be connected to a wide range of sensor modules producted by Phidget Inc. Z1 has be used in some researches and applications such as environmental and agriculture monitoring, healthcare and power consumption monitoring.

3 Operating Systems for Sensor Node Devices

In this section, we briefly discuss operating systems for sensor nodes. Since sensor devices present several constraints in terms of memory, processing and energy, operating systems for WSNs must be designed to manage resource efficiently, taking care of specific issues arising from use cases. For instance, in most applications batteries are often the only source of energy and changing or charging batteries is not suitable. Consequently, an OS for sensor devices must not only limit power waste, ensuring optimized energy consumption, but also be able to recover from power faults. Over the years, several lightweight operating systems for WSNs have been created. Although they all share the same goals, architectural and implementation choices differ. Architecture, programming model, programming language and energy saving support are the main varying factors. Several chapters in literature as [9, 16], survey operating systems for WSNs while [9, 34], provide a comparative study of them.

Some of these operating systems such as **TinyOs** [27] and **Contiki** [14] present an event-driven programming model where processes use the same stack in order to reduce memory requirements. Instead, a multithreaded model requires additional resources as each thread needs its own stack. A package for multithreading support in TinyOS is presented in [23]. For multithreading, Contiki uses a lightweight thread model called protothread. Protothreads share a stack and their implementation requires only 2 bytes. A comparison of TinyOs and Contiki can be found in [34]. Multithreading-based operating systems are **MantisOs** [5], **LiteOs** [8]. Scheduling management deals with resource allocation. For example, TinyOs provides a FIFO scheduling. Tasks are enqueued and executed until their completion without preemption by other tasks. A task gets preempted only if an event or interrupt occurs. MantisOs provides a priority-based round robin scheduling within each of five available priority levels. The first scheduled thread gets the highest priority and it is executed until it completes or its time slice expires. If any thread is ready for execution, the system goes into sleep mode. In Contiki, preemptive multithreading is implemented as a library that can be linked with applications. LiteOs implements both priority-based scheduling and round robin scheduling. Operating systems for WSNs can also differ in kernel architecture. TinyOs presents a component-based architecture where every component is specified in terms of used and offered interfaces, while Contiki's kernel is composed of a lightweight scheduler and a poll event handler supporting synchronous and asynchronous events. A Contiki system is divided into two parts: the core system and the loaded programs. The core system includes the kernel, libraries and the program loader. MantisOs system presents a layered architecture where each level provides services to the higher levels. The lower level is the hardware level. The second level includes the kernel and the scheduler, the communication level and the device drivers. Over this level, Mantis System API provides services to network stack, command services and user's threads. The architecture of LiteOs is composed of three main components: LiteShell, LiteFS, and the kernel on the node. LiteShell subsystem runs on a PC station and allows the user to interact directly to a sensor node through a Unix shell providing commands for file processing and device management. LiteFS subsystem mounts a sensor node to the root filesystem of the base station. **MansOs** [37] derives from LiteOs but it is designed to be easy portable to new platforms. For this purpose, it presents three levels providing different functionalities. As described in [37] the Hardware Presentation Level (HPL) is the hardware dependent level and thus it contains the specific code for the target hardware platform. The Hardware Abstraction Layer (HAL) contains the representation of the platforms in terms of platform constants, pin assignments, device assignments and function binding. The Hardware Independent Layer (HIL) provides user hardware-independent functions abstracting the lower levels. Operating systems for WSNs must also provide support to communication. For example, Contiki presents two communication stacks: the *UIP* stack for communication over Internet and *Rime* aimed at low power radio applications with addressing, broadcast and unicast communication. MantisOS handles the communication through a layered network stack implemented with user level threads. MantisOS *COMM* layer

Table 1 Summary table of some operating systems for sensor nodes. They differ mainly in architecture, programming model and scheduling

	Architecture	Programming model	Scheduling	Language
TinyOs	Monolithic (component-based)	Event-driven	2 levels scheduling	NesC
Contiki	Modular	Event-driven and thread-based	2 levels scheduling hierarchy and preemptive multithreading	C
MantisOs	Layered architecture	Thread-based	Priority-based thread scheduling	C
LiteOs	Partioned into three parts: LiteShell, LiteFs and Kernel	Thread-based	Priotity-based and Round-robin	C

supports the MAC protocol. This layer provides a common interface for different communication device drivers and manages packet buffering.

These operating systems support distributed computing since they provide primitives needed by distributed applications. The choice of the operating system may in fact depend on the application requirements. Table 1 shows the main differences between some operating systems for sensor nodes.

4 Sensor Ontologies

Ontologies are used to represent formally the concepts of a domain through their properties and the relationships existing between them. Several ontologies for sensor nodes can be found in literature with the aim of providing a common representation of sensor nodes. These ontologies aim to describe characteristics, platforms, properties, sensing capabilities of a sensor node in a WSN. SensorML is a standard XML language for sensors definition. It is generic and therefore it supports a wide range of sensors. It provides archiving of sensor parameters, plug-n-play, support for tasking, observation, auto-configuration, sensor accuracy definition. CSIRO Sensor Ontology [11] is written in the Web Ontology Language (OWL) with the aim of representing sensors in a generic domain. A sensor is described in terms of its physical properties (platform, serial ID, weight, dimension) and its concrete implementation. The ontology also models the sensor's functionalities and its sensing capabilities (data flow, sensor's response, input, output). In WIreless Sensor Networks Ontology (WISNO) [3] a sensor node is described through its physical components (radio, memory, cpu and power supply) modeling both its properties and the existing relationship with other domain's concepts. OntoSensor is based on SensorML and it represents components like sensing units, radio and data acquisition board. In the

Semantic Sensor Network (SSN) ontology [10] a sensor is considered as a device able to detect changes or events from the environment to process them producing a result. The ontology can also represents the concept of single observation, platform, deployment, data and energy consumption. Therefore, distributed applications in WSNs can reach a certain degree of interoperability using ontologies. Ontologies allow to represent concept of a specific domain formally and thus heterogeneous applications can share information having the same representation of concepts and consequently a common base of knowledge.

5 Distributed Algorithms and Applications in WSNs

Since sensor nodes present many constraints both in storage and processing, distributed computing can overcome these limitations increasing the performance of the entire WSN. In a distributed application, sensor nodes achieve complex tasks by a cooperative effort. Cooperation implies that sensor nodes need to communicate with each other to perform a given task. Radio communication, however, involves a considerable energy expenditure. Furthermore, managing routing tables for a large set of nodes is challenging. Therefore, researchers have designed several distributed routing protocols in order to increase sensor's battery lifetime by reducing the number of transmissions. Distributed applications are often based on clustering and grouping, on neighbors' discovery and localization, but also on synchronization and on data collection algorithms. Data-centric routing protocols aims to reduce transmissions between nodes removing the need of a node addressing mechanism using a query-based approach involving just a neighbor-to-neighbor communication. Flooding [20], gossiping [20] and SPIN [25] are few examples of data-centric routing protocols. Unlike data-centric algorithms, hierarchical routing protocols are characterized by clusters or groups of nodes and each cluster has a leader. Multi-hop communication is allowed within the same cluster or group, and communication between different clusters or groups is managed by the leader. Data fusion and aggregation are also performed to reduce the number of communications to the base station. Other hierarchical routing protocols are described in [36]. Location-based protocols focus on the idea that knowing the position of the nodes, a query can be addressed just to a limited region eliminating unnecessary transmissions. One significant advantage is that these protocols consider node mobility. Other protocols are based on network flow or Quality of Service (QoS) modeling. A survey on routing protocols for WSNs can be found in [1]. Multicast routing protocols could reduce the number of the messages exchanged, by transmitting simultaneously just to a subset of nodes. A multicast distributed routing algorithm is proposed in [18]. Data aggregation is also a fundamental task in WSNs because aggregation eliminates redundant transmissions saving the WSN's global energy. When sensory data are aggregated and then transmitted, collisions may occur during aggregation. A survey on data aggregation algorithms can be found in [33]. The authors of [33] also discuss the influence of the network topology and the routing protocol on data aggregation. In [43], the authors present a distributed scheduling algorithm to improve the energy management decreasing the time latency. The same chapter also proposes an adaptive version of the algorithm to deal with network topology changes. Detecting faults in WSNs makes it possible to identify occurred error situations in order to handle them properly. A distributed bayesian approach for fault detection is suggested in [30], where cooperation allows a node to infer the nodes source of incorrect data. Reducing the considerable energy consumption caused by channel listening improves node's lifetime. For this reason, several algorithms involving duty cycling techniques have been proposed. In this way, a node alternates sleep and wakeup times. Some of these protocols are based on sleep and wakeup time synchronization. Other protocols do not require synchronization being thus more flexible. The authors of [39] propose the PW-MAC algorithm based on prediction of receiver's wakeup. Distributed data storage overcomes problems related to node's low storage capability. Many approaches rely on the idea of a distributed database. In this context, the authors of [32] propose a cooperative middleware providing a distributed data storage for WSNs. For example, in Hood [42] the node shares its data only with some of its neighbors. In TinyPeds [19] the cluster head receives aggregated data from its cluster's members and stores them in a neighbor cluster head. In [21], the choose of the backup node is probabilistic in order to make the network more robust and efficient. WSNs often handle sensitive or critical data that must be protected from malicious attacks. Since many security algorithms rely on random number generator, the authors of [29] present a distributed algorithm for selecting a node as a number random generator using a leader election approach. The authors of [12] present a sample Ambient Intelligence (AmI) system for minimizing energy consumption in indoor environments according to users' preferences and needs. The system's middleware is distributed on sensory devices for low-level data gathering management, while higher level functionalities are implemented in a centralized manner. In [31] a WSN system including collaboration in the healthcare field is presented. The proposed system is composed of wearable and ambient sensors cooperating to monitor user's vital signs. Ambient sensors provide information about context and the information from wearable and ambient sensors are then fused. In [24] an intrusion detection system involving collaboration among sensors has been proposed. The sensor tracking is an important research aspect exploited by many application, like animal or person tracking. Other collaborative applications can involve data filtering techniques like Kalman filter to estimate the node position [35]. In [26] a system for environmental monitoring is presented. The authors used local storage and Constrained Application Protocol (CoAP) as communication protocol, concluding that a completely distributed algorithm decreases consumptions and increases reliability. In [40] a surveillance system relies on a peer to peer distributed sensor network infrastructure implementing collaborative on-line learning and target localization. The state of art about collaborative WSNs can be found in [28] while a survey about collaborative tracking is provided in [13].

6 An Example of Distributed Application: Synchronization

In this section, we aim to make the reader aware of distributed computing on resources constrained devices, such as sensor nodes of a WSN, showing how a trivial application, such as synchronization, becomes a complex goal on these platforms. Synchronization is a typical distributed application where sensors must align their own clock to share the same notion of time. Since sensors cooperate, having the same time axis is fundamental. Different microcontrollers must perform an action simultaneously or periodically together with the other nodes of the network. Even if the nodes are turned on at the same time, their clocks will drift increasingly because of the tolerances in clock generators. Therefore, synchronization poses some important questions such as how to estimate the frequency drift or how frequently sensor nodes must synchronize their clocks. In addition, to achieve common time knowledge, nodes cooperate through messages and consequently the transmission delay must also be considered. The drift and the propagation delay are thus important parameters to compute in order to perform synchronization with relative time measure. In this case, sensor nodes estimate their relative drift and the propagation delay adjusting their clocks accordingly. Since distributed synchronization relies on communication, a synchronization application must handle channel noise, unreliable packet delivery, packet losses, asymmetrical links between sender and receiver, and must implement some kind of fault tolerance. In literature, several synchronization protocols have been proposed. Some of them are based on hierarchical network topology or they use a probabilistic approach rather then a deterministic synchronization. Figure 1 shows the synchronization phase of the Timing-sync Protocol for Sensor Networks [17] occurring once the hierarchical topology in the network has been established. Synchronization is performed between pairs of nodes. In [15], the authors consider some evaluation metrics such as energy utilization, precision, lifetime, scope and availability, cost and size. A high energy conservation is provided by post-facto synchronization updating clocks only when it is strictly necessary. A synchronization application can be evaluated on the basis of the time the network remains synchronized before performing synchronization again, while cost and size indicate hardware costs. The authors of [38] evaluate the synchronization protocols on the basis of quantitative criteria such as precision, computational complexity, message complexity, convergence time, number of nodes and sleep mode support. In this context, they provide a comparison between some protocols based on these evaluation metrics. They also consider qualitative parameters for synchronization evaluation such as accuracy, scalability and overall complexity including storage requirements, communication overhead and fault tolerance. This brief discussion about synchronization shows that distributed computing applications on low resources devices are usually challenging. These applications must cope with the issues related to cooperation, coordination, scheduling but also with the typical constraints of WSNs such as limited hardware and computing capabilities, reduced energy and bandwidth availability, and latency.

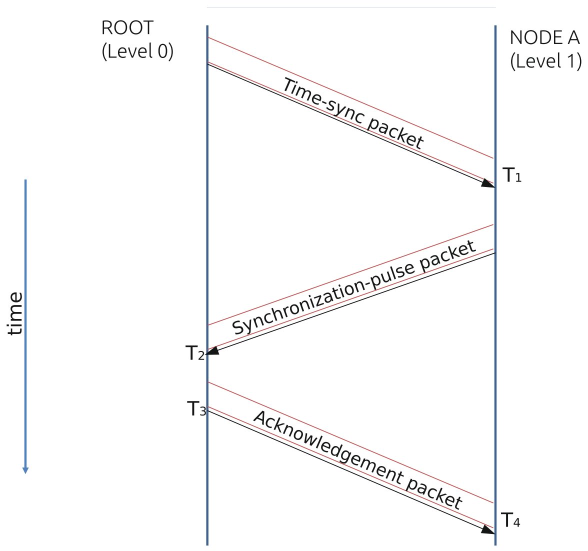

Fig. 1 Synchronization phase of the Timing-sync Protocol for Sensor Networks [17]. Once the root node has sent a time-sync packet to all of its neighbors, each neighbor of level 1 waits for a random time interval before sending a synchronization-pulse packet containing its level and its local time to the root. The root replies with its level, the time received from node A, its local time of arrival and its time of sending. Node A thus estimates the drift and the propagation delay and adjusts its clock accordingly. This process propagates through the network from nodes of level 1 to nodes of level 2 and so on

7 Conclusion

In this chapter, we provided an overview about a new emerging scenario concerning distributed computing on limited resource devices. We discussed it from several points of view, from recently appeared hardware platforms and operating systems to distributed algorithms and applications. We also provided a summary description of some existing ontologies for sensor nodes. We also briefly described a typical distributed application to show how even simple applications may be challenging in this scenario.

Acknowledgments This work has been partially supported by the PON R&C grant MI01_00091 funding the SeNSori project.

References

1. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad hoc Netw. **3**(3), 325–349 (2005)
2. Arduino uno http://arduino.cc/en/Main/arduinoBoardUno. (2013). Accessed 7 Sept 2013
3. Avancha, S., Patel, C., Joshi, A.: Ontology-driven adaptive sensor networks. In: MobiQuitous, Boston, pp. 194–202, 2004
4. Banzi, M.: Getting Started with Arduino. O'Reilly Media, Inc., Newton (2009)
5. Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., Han, R.: Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. Mob. Netw. Appl. **10**(4), 563–579 (2005)
6. Bose, R., King, J., El-Zabadani, H., Pickles, S., Helal, A.: Building plug-and-play smart homes using the atlas platform. In: Proceedings of the 4th International Conference on Smart Homes and Health Telematic (ICOST), Belfast, June 2006, citeseer(2006)
7. Burns, A., Greene, B.R., McGrath, M.J., O'Shea, T.J., Kuris, B., Ayer, S.M., Stroiescu, F., Cionca, V.: Shimmer-a wireless sensor platform for noninvasive biomedical research. IEEE J. Sens. **10**(9), 1527–1534 (2010)
8. Cao, Q., Abdelzaher, T., Stankovic, J., He, T.: The liteos operating system: Towards unix-like abstractions for wireless sensor networks. In: Proceedings of 7th International Conference on Information Processing in Sensor Networks (IPSN '08), pp. 233–244, April 2008
9. Chien, T.V., Chan, H.N., Huu, T.N.: A comparative study on operating system for wireless sensor networks. In: IEEE International Conference on Advanced Computer Science and Information System (ICACSIS'11), pp. 73–78, (2011)
10. Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., et al.: The ssn ontology of the w3c semantic sensor network incubator group. Web Semant. Sci. Serv. Agents on the World Wide Web **17**, 25–32 (2012)
11. Compton, M., Neuhaus, H., Taylor, K., Tran, K.N.: Reasoning about sensors and compositions. In: Proceedings of Semantic Sensor Network, pp. 33–48, (2009)
12. De Paola, A., Gaglio, S., Lo Re, G., Ortolani, M.: Sensor9k: a testbed for designing and experimenting with WSN-based ambient intelligence applications. Pervasive and Mob. Comput. **8**(3), 448–466 (2012)
13. Demigha, O., Hidouci, W.K., Ahmed, T.: On energy efficiency in collaborative target tracking in wireless sensor network: a review. IEEE Commun. Surv. Tutorials **15**(3), 1210–1222 (2013)
14. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462, (2004)
15. Elson, J., Römer, K.: Wireless sensor networks: a new regime for time synchronization. ACM SIGCOMM Comput. Commun. Rev. **33**(1), 149–154 (2003)
16. Farooq, M.O., Kunz, T.: Operating systems for wireless sensor networks: a survey. Sensors **11**(6), 5900–5930 (2011)
17. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In: Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys'03), pp. 138–149. ACM, New York (2003)
18. Gatani, L., Lo Re, G., Gaglio, S.: An efficient distributed algorithm for generating multicast distribution trees. In: International Conference on Parallel Processing Workshops (ICPP 2005 Workshops), pp. 477–484, (2005)
19. Girao, J., Westhoff, D., Mykletun, E., Araki, T.: TinyPEDS: tiny persistent encrypted data storage in asynchronous wireless sensor networks. Ad Hoc Netw. **5**(7), 1073–1089 (2007)
20. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18**(4), 319–349 (1988)
21. Jardak, C., Osipov, E., Mahonen, P.: Distributed information storage and collection for wsns. In: IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS'07), pp. 1–10, (2007)

22. King, J., Bose, R., Yang, H.I., Pickles, S., Helal, A.: Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces. In: Proceedings of 31st IEEE Conference on Local Computer Networks, pp. 630–638, (2006)
23. Klues, K., Liang, C.J.M., Paek, J., Musaloiu-Elefteri, R., Levis, P., Terzis, A., Govindan, R.: Tosthreads: thread-safe and non-invasive preemption in tinyos. In: SenSys, vol. 9, pp. 127–140, (2009)
24. Krontiris, I., Benenson, Z., Giannetsos, T., Freiling, F.C., Dimitriou, T.: Cooperative intrusion detection in wireless sensor networks. In: Wireless Sensor Networks, pp. 263–278. Springer, Berlin (2009)
25. Kulik, J., Heinzelman, W., Balakrishnan, H.: Negotiation-based protocols for disseminating information in wireless sensor networks. Wireless Netw. **8**(2/3), 169–185 (2002)
26. Larios, D., Mora-Merchan, J., Personal, E., Barbancho, J., León, C.: Implementing a distributed wsn based on ipv6 for ambient monitoring. Int. J. Distrib. Sens. Netw. (2013)
27. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al.: Tinyos: an operating system for sensor networks. In: Ambient intelligence, pp. 115–148. Springer, Heidelberg (2005)
28. Li, W., Bao, J., Shen, W.: Collaborative wireless sensor networks: a survey. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC'11), pp. 2614–2619, (2011)
29. Lo Re, G., Milazzo, F., Ortolani, M.: Secure random number generation in wireless sensor networks. In: Proceedings of the 4th international conference on Security of Information and Networks, pp. 175–182, (2011)
30. Lo Re, G., Milazzo, F., Ortolani, M.: A distributed bayesian approach to fault detection in sensor networks. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM'12), pp. 634–639, (2012)
31. Nakamura, M., Nakamura, J., Lopez, G., Shuzo, M., Yamada, I.: Collaborative processing of wearable and ambient sensor system for blood pressure monitoring. Sensors **11**(7), 6760–6770 (2011)
32. Piotrowski, K., Langendoerfer, P., Peter, S.: tinydsm: a highly reliable cooperative data storage for wireless sensor networks. In: International Symposium on Collaborative Technologies and Systems (CTS'09), pp. 225–232, (2009)
33. Rajagopalan, R., Varshney, P.K.: Data aggregation techniques in sensor networks: a survey. IEEE Comm. Surv. Tutorials **8**, 48–63 (2006)
34. Reusing, T.: Comparison of operating systems tinyos and contiki. Sens. Nodes-Operation, Netw. Appli. (SN) **7** (2012)
35. Ribeiro, A., Giannakis, G.B., Roumeliotis, S.I.: Soi-kf: distributed kalman filtering with low-cost communications using the sign of innovations. IEEE Trans. Sig. Process. **54**(12), 4782–4795 (2006)
36. Singh, S.K., Singh, M., Singh, D.: A survey of energy-efficient hierarchical cluster-based routing in wireless sensor networks. Int. J. Adv. Networking and Appl. (IJANA) **2**(02), 570–580 (2010)
37. Strazdins, G., Elsts, A., Selavo, L.: Mansos: easy to use, portable and resource efficient operating system for networked embedded devices. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10), pp. 427–428, ACM (2010)
38. Sundararaman, B., Buy, U., Kshemkalyani, A.D.: Clock synchronization for wireless sensor networks: a survey. Ad Hoc Netw. **3**(3), 281–323 (2005)
39. Tang, L., Sun, Y., Gurewitz, O., Johnson, D.B.: Pw-mac: an energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In: INFOCOM, 2011 Proceedings IEEE, pp. 1305–1313 (2011)
40. Wang, X., Wang, S., Bi, D.: Distributed visual-target-surveillance system in wireless sensor networks. IEEE Trans. Syst. Man Cybern. B Cybern. **39**(5), 1134–1146 (2009)
41. Waspmote datasheet: Available online at http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf (2013). Accessed 7 Sept 2013
42. Whitehouse, K., Sharp, C., Brewer, E., Culler, D.: Hood: a neighborhood abstraction for sensor networks. In: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pp. 99–110. ACM (2004)

43. Yu, B., Li, J., Li, Y.: Distributed data aggregation scheduling in wireless sensor networks. In: IEEE INFOCOM 2009, pp. 2159–2167 (2009)
44. Zennaro, M., Bagula, A., Gascon, D., Noveleta, A.B.: Long distance wireless sensor networks: simulation vs reality. In: Proceedings of the 4th ACM Workshop on Networked Systems for Developing Regions, pp. 12:1–12:2. ACM (2010)
45. Zolertia z1 datasheet: Available online at http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf (2013). Accessed 7 Sept 2013