Telecommun Syst (2016) 62:695–709
DOI 10.1007/s11235-015-0105-7

The self-configuration of nodes using RSSI in a dense wireless sensor network

Mohammad M. Abdellatif[1] · **José Manuel Oliveira**[2] · **Manuel Ricardo**[1]

Published online: 27 October 2015

Abstract Wireless sensor networks (WSNs) may be made of a large amount of small devices that are able to sense changes in the environment, and communicate these changes throughout the network. An example of a similar network is a photo voltaic (PV) power plant, where there is a sensor connected to each solar panel. The task of each sensor is to sense the output of the panel which is then sent to a central node for processing. As the network grows, it becomes impractical and even impossible to configure all these nodes manually. And so, the use of self-organization and auto-configuration algorithms becomes essential. In this paper, three algorithms are proposed that allow nodes in the network to automatically identify their closest neighbors, relative location in the network, and select which frequency channel to operate in. This is done using the value of the Received Signal Strength Indicator (RSSI) of the messages sent and received during the setup phase. The performance of these algorithms is tested by means of both simulations and testbed experiments. Results show that the error in the performance of the algorithms decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error. However, the value of the error is still acceptable even with a high number of nodes.

Keywords WSN · Multi-hop · RSSI · Self-organization · Auto-configuration

✉ Mohammad M. Abdellatif
mma@inescporto.pt

José Manuel Oliveira
jmo@inescporto.pt

Manuel Ricardo
mricardo@inescporto.pt

[1] INESC TEC and Faculty of Engineering, University of Porto, Campus da FEUP, Rua Dr. Roberto Frias, 4200 - 465 Porto, Portugal

[2] INESC TEC and Faculty of Economics, University of Porto, Campus da FEUP, Rua Dr. Roberto Frias, 4200 - 465 Porto, Portugal

1 Introduction

Wireless Sensor Networks (WSNs) are made of a large amount of small devices that are able to sense changes in the environment, and communicate these changes throughout the network [1]. WSN applications are used in smart-cities, environmental monitoring, distributed sensing in industrial plants, and health care [2].

Normally, low data rate is employed in WSNs. However, other challenging issues appear in the network. Challenges are mostly related to the reliability of the communication links and to the energy efficiency [3]. Also, because of the many-to-one feature of the communication in WSNs, wireless interferences and collisions, the huge sizes of these networks, as well as the scheduling of data transmissions becomes a challenging problem which needs to be carefully addressed. As the network grows, it becomes impractical and even impossible to configure all these nodes manually. And so, the use of self-organization and auto-configuration algorithms becomes essential.

This work aims to enable the complete self configuration of a dense wireless sensor network without any interactions from the human operators. The communications scenario is provided by the SELF-PVP project [4]. This project aims to increase the efficiency of a photo voltaic (PV) power plant. It assumes that 200,000 solar panels are deployed in a matrix like deployment spread over a large area (250 ha). The goal of the project is to build a self-organizing, truly distributed

computational network reflecting ambient intelligent, with the ability to sense and control the operation point of each panel in a PV power plant, in order to accomplish a global maximum power available at any environment condition of operation. Each solar panel will have a sensor/actuator that will sense local variables, communicate these values with other sensors and compute local errors with the goal of optimizing the overall performance of the panels' array.

In a PV power plant, panels are usually spread over a huge area. A WSN in this scenario can be viewed as a grid of sensors where each sensor is connected to a solar panel. Panels in the same column will have the same value of the current passing through them. In order to reach the optimum power delivery point, most of the communications will be among nodes in the same column. Based on this, we divide the network into small networks of columns with fixed sizes. Each column is to operate in a different frequency channel in order to reduce interference between adjacent columns. These networks can be considered as building blocks that can be replicated as many times as needed in order to cover the whole of the dense WSN. Additionally, because we are dealing with a power plant, we assume that nodes do not need to sleep in order to save energy as they can get all the energy they need from the panels.

In this paper, the aim is to allow each node in the network to automatically identify its closest neighbors as well as its relative location using the value of the Received Signal Strength Indicator (RSSI) of the messages sent back and forth among nodes during the setup phase of the network. This work is a continuity of the work already published in [5].

This study considers networks with different sizes, mainly by adding more columns to the network with each column having 9 nodes. The self-configuration of the network was divided into three algorithms, the neighbor identification, the relative location, and the frequency allocation algorithms. Their performance was evaluated using the Contiki COOJA simulator [6] which can be downloaded from [7].

The main contribution of this paper is the proposal and evaluation of the three algorithms which together can enable the complete self-configuration of a large network of sensors where all the sensors are identical and running the same code. Results show that the error in the network setup increases as the number of columns in the network increases. However, the value of the error is low even for high simulated number of columns. Additionally, experiments on real testbeds were performed in order to validate the performance of the algorithms in different scenarios.

The rest of the paper is organized as follows. Section 2 gives a background on the work related to this research work. Section 3 presents the topology proposed for the WSN, and the three self-configuration algorithms. Section 4 describes the simulation environment and different parameters tested in order to obtain the simulation results. These results are presented in Sect. 5. Section 6 describes a set of testbed experiments performed as well as the results obtained from running them. Finally, paper conclusions and ideas for future work are listed in Sect. 7.

2 Related work

Self-configuration in large WSNs has been a hot research topic in the last years. In [8], Patrawi et al. have proposed a sensor localization system based on either the Received Signal Strength (RSS), Quantized RSS, or the proximity measurements between the sensors. They have showed analytically that the standard deviation bound for proximity measurements was 48% worse than that for RSS measurements. Additionally, when the nodes are placed in a grid setup, proximity measurements had an average standard deviation bound about 57 % worse than that of RSS measurements. As for the K-level quantized RSS, they have shown that it performs as well as RSS even for low values of K. In our work, we use the value of the Received Signal Strength Indicator (RSSI) that is measured by the radio driver of the sensor node which is similar to the QRSS value mentioned in that paper, but with no quantization.

In [9], the authors investigated the use of radio frequency (RF) location systems for indoor domestic applications. They introduced the concept of RF location system for Integrated Indoor Location Using RSSI and Link Quality Indicator (LQI) provided by ZigBee module. They have shown different ideas to overcome the problems in the existing methods calculating the distance in indoor environments. They have presented a new method for reducing the error in the location identification due to interference within the infrastructure-based sensor network. The proposed method calculates the distance using LQI and RSSI predicted based on the previously measured values. The calculated distance corrects the error induced by interference. They have shown by experimental results that their method can reduce the average error around 25 %, and it is always better than the other existing interference avoidance algorithms. In our paper, we care more about the relative location of nodes with respect to the other nodes in the network.

In [10], the authors have proposed what they called an Efficient Self-Organization Clustering Algorithm for Clustering (ESAC), which they based on the weighing parameters k-density, residual energy of the nodes and node mobility for cluster heads election. They showed that their algorithm enables the creation of low number of stable and balanced clusters while avoiding the problem of battery drainage of the cluster head. They have proven that this algorithm prolongs the lifetime of the network while reducing the broadcast overhead in the network. We also are interested in the operation of

the election of cluster head. However, we have used the relative location of the node to be the main factor in that process.

In [11], Borbash et al. presented an asynchronous and distributed algorithm for neighbor discovery. The proposed algorithm was shown to allow each node in the network to populate a neighbor list that may not be complete. Their algorithm is set to run at boot time and can be re-run whenever the network changes. In this algorithm, each node has its own time slotting and an initial estimate on how many neighbors it should have. In each time slot, nodes alternate between transmitting and receiving states with different probabilities and try to listen for messages sent from the other nodes. Each node runs independently of the other nodes and adds a node to its neighbors list as soon as it receives a message from it. The authors have analyzed the performance of their algorithm and shown that they can tune it to maximize the percentage of the neighbors discovered in a fixed running time. We are doing a similar job in this paper. However, we do not allocate specific time slots per node. We rely on the randomness in the message sending as well as a sufficiently large number of messages to be sent in order to assure that each node hears all of its neighbors. Additionally, we use RSSI to differentiate between one hop neighbors and further away nodes.

In [12], the authors presented a self-organizing control scheme for WSNs based on the gene regulatory networks (GRNs) model, a representation of genes/proteins and the interactions between them. The authors described the fundamentals of GRNs and discuss how that they were used to solve the problem of robustness in network design. As well as how GRNs were found to offer robustness and resilience to failures in WSNs when used to determine node location in the network. The authors proposed a GRN algorithm for self-organizing control of WSNs, which aims to automatically schedule node states according to the environmental changes and the application-specific requirements. They developed a theoretical model for a robust node scheduling design using GRN framework. This algorithm was simulated using MATLAB. These simulations lead them to conclude that the proposed scheme can guarantee delay requirement while achieving energy balancing among sensor nodes. In our work, we are not concerned with neither delay nor energy efficiency and so this algorithm does not suit our requirements.

In [13], Bajo et al. presented an open character platform that facilitates the integration and management of sensor networks as well as provide some services based on the information collected from these networks. They claim that their platform allows the addition of networks that were built using different existing or new technologies. It also allows the integration of the data in a cloud computing environment. They describe open systems as systems that exist in dynamic operative environments where new components are integrated or existing components leave the system continually and where even the operating conditions can change unpredictably. Their proposed platform is said in the paper to combine virtual organizations of multi-agent systems with information fusion algorithms, which provides great adaptability to changes in the controlled environment and a more efficient usage of the technologies available in that environment. They have also integrated the virtual organization within cloud computing environments, providing ubiquitous computation and communication mechanisms. They have shown by comparison with other related work that their proposed platform which has an open character, able to incorporate information fusion algorithms, and capable of integration with cloud technology built upon the base of social computation was never done before. The authors have proven the validity of their proposed architecture by applying it to a case study for indoor location where information is taken from different heterogeneous sensors. We are working with static networks were changes in the network doesn't happen suddenly or frequently. And even when changes happen, the system is not required to fix itself as soon as possible. However, it is a good idea to be done for future work.

3 Proposed topology and self-configuration algorithms

3.1 Network topology

As the network we are using for this study is based on a photo voltaic power plant where panels are spread over a huge area, we decided to simulate the network as a grid system of sensors where each sensor is connected to a solar panel and has constant distances between it and each of its neighboring panels. Additionally, since most of the communications will be done within each column, we decided to divide the whole network into small networks of columns. Each sensor reads the output voltage of its respective panel and sends it to a central node within the column. And since the changes in the light conditions are slow by nature, it is expected that the required traffic load is going to be low.

The proposed WSN topology is shown in Fig. 1. Such setup is referred to as a strip-based deployment [14]. It allows for full coverage of the area under consideration. In a real life scenario, nodes within a column are placed side by side with a distance of about 2 m between each consecutive sensor, while columns have a wider distance (4 m) between them in order to allow maintenance access. These distances were chosen from an already deployed solar power plant.

Additionally, in order to reduce the interference between columns, each column is to operate in a different frequency channel than the other columns. To enable inter-column communication, a core network will be created operating in a frequency different from all the columns. Core network nodes (solid circles in Fig. 1) are placed between columns to enable

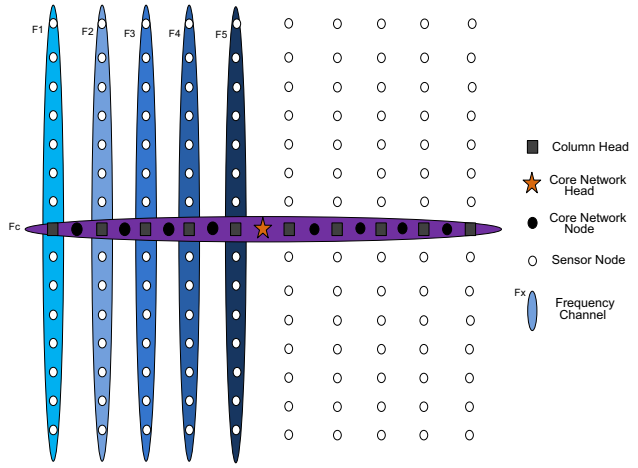

Fig. 1 Network topology

inter-column communication, and when the need for inter-column communication arises, the respective column heads of each column (squares in Fig. 1) will collect the information from their clients, switch to the core frequency, and then send this information through the core network to the core network head. A group of columns with their access network can be considered as a building block which can be replicated as the network grows. When more nodes are added to the network, it becomes impractical and even impossible to configure all these nodes manually. And so, the use of self-organization and auto-configuration algorithms becomes essential.

In this paper, three algorithms are proposed in order to perform the self-configuration of the nodes. Firstly, nodes will have to run the neighbor identification algorithm in order to allow each node to find its closest neighbors, i.e., neighbors that can be reached directly without multi-hopping. Secondly, the relative location algorithm will run in order to decide which node in a column is in its center and designate it a column head (squares in Fig. 1). These column heads will then have to decide which node should be the core network head (star in Fig. 1). And finally, the frequency allocation algorithm will run. It is used to assign a different frequency channel to each of the columns in order to reduce the interference between adjacent columns. These three algorithms are described in more detail in the next three subsections.

3.2 Neighbor identification algorithm

The neighbor identification algorithm is shown in Fig. 2. The aim of this algorithm is to allow each node in the network to find its closest neighbors, i.e., neighbors who can be reached directly without multi-hopping. The neighbor identification is achieved by measuring the value of the RSSI of the ping messages that are sent back and forth between nodes in the network after booting. Using this information, nodes calculate which neighbor(s) has the highest RSSI value. These

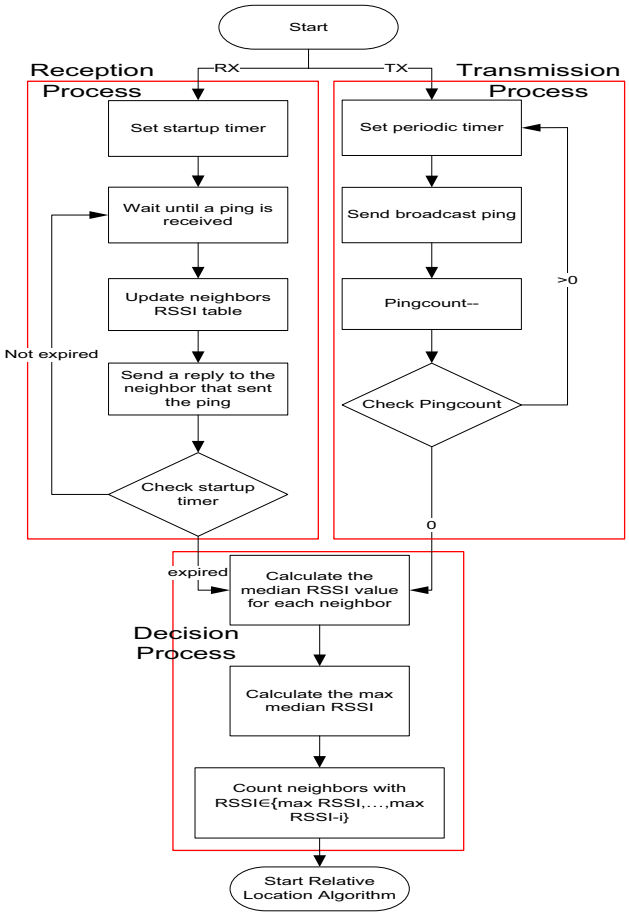

Fig. 2 Neighbor identification algorithm [5]

neighbors will be considered then as the closest neighbors in the network. For example, nodes on column extremities (white circles) should have only one close neighbor, while nodes inside a column (white circles) should have two close neighbors. Column heads (squares) should have three or four close neighbors depending on the column location. And, core network nodes (solid circles and star) should have two close neighbors.

The algorithm runs in each node after it boots and is divided into three processes:

- *Transmission process* Each node sets a periodic timer. Its duration is selected depending on the size of the network, i.e, the number of nodes in the network. Then, each node sends a broadcast ping message to its neighbors every time the timer expires with a small random delay added to help avoid collisions from other nodes' pings. This process is repeated until a certain number of ping messages are sent, which is predefined as a parameter of the algorithm and is referred to as the ping count.
- *Reception process* It runs simultaneously with the transmission process. A setup timer is set large enough to allow

the reception of all the ping messages sent from other nodes. To ensure that, this timer should be greater than the periodic timer multiplied by the ping count. While this timer has not expired, nodes listen for ping messages sent from neighbors, and update the neighbors table with the RSSI value of the received ping of the transmitting neighbor. Then, send a reply to that neighbor so it can also update its neighbors table. This last step can be skipped and rely only on the ping messages to update the RSSI in order to reduce traffic. However, it has been found that it helps with the decision. This process is repeated until the setup timer expires.

– *Decision process* Starting only after both Transmission and Reception processes have ended, each node begins its decision making process. It starts by calculating the median RSSI value for each neighbor in its neighbors table. Then it finds which neighbor has the maximum median RSSI. Nodes then have to find how many of their neighbors have a median RSSI that lies in a set containing max RSSI and max RSSI$-i$, where i is a positive integer. This will tell the node how many close neighbors it has. The value of i depends on the topology of the network. Initial tests has to be made in order to find the value of i that minimizes the error in the decision making. Then this value will be set in the algorithm for such a topology. After a decision is made, the relative location algorithm starts.

The algorithm uses the RSSI value of the received messages as calculated by the radio driver of the sensor node. It is a rounded value in dBm and it is mainly affected by the transmission power and the propagation medium. We have made sure that the transmission power remains constant throughout the simulation and the testbed experiments. We use the median RSSI in order to filter out any extreme values that may appear due to the noise or fading effects.

3.3 Relative location algorithm

The relative location algorithm is shown in Fig. 3. The objective of this algorithm is to enable each node to find its relative location in the network with respect to the other nodes and, based on this, decide its role in the network. After nodes have identified their closest neighbors from the output of the previous algorithm, they exchange messages among themselves in order to locate the central node of each column (the column head). Then, these column heads will also exchange different messages among themselves to try to find the central node of the network which will be referred to as the core network head. This algorithm begins only after the neighbor identification algorithm has ended. Each node checks the number of close neighbors it has. Based on this value, one of the following steps can be taken by the node:

– *Edge sensor node (one neighbor)* If the node has only one close neighbor, this can only mean from the topology that it is located on the end of a column. And so, it designates itself as an *edge sensor node*, i.e., a node at the edge of a column. Then it sends a column message to its neighbor with the count 1. A column message is a control message with a specific type and a count that will be used by nodes within the same column to elect the column head.
– *Intermediate sensor node (two neighbors)* If the node has two neighbors, it can either be a node inside a column or a node inside the core network. And so, it waits until it receives a message. If this message is a column message, it designates itself an *intermediate sensor node*, increases the column count received from the message, and then forwards the column message with the increased count to its other neighbor in the same column.
– *Edge column head (three neighbors)* If the node has three neighbors, it means from the topology that it can only be a node located at the edge of the core network and in the center of the edge column. And so, it designates itself as an *edge column head*, i.e., a column head at the edge of the network. Then, it waits until it receives two column messages from the two sides of its column. After receiving these two messages it confirms that it is a column head by checking that the two counts from the two column messages received are equal. Then it sends a row message to the third node with count 1. The row message is a control message with a specific type different than the column message and will be used to determine the core network head.
– *Intermediate column head (four neighbors)* If the node has four neighbors, it can only be a node that lies inside the core network and the center of a column. Based on this, it designates itself as an *intermediate column head*. Then, it waits until it receives two column messages from the two sides of its column, and uses this information to confirm that it is the column head as in the previous step. It then waits until it receives a row message. Then it forwards the row message to the fourth node with an increased count. Then, waits for another row message and forwards it to the other node with an increased count.
– *Intermediate core node/core network head (two neighbors)* If the node has two neighbors, and it receives a row message, it will increase the row count received from the message, and then forwards the message with the increased count to the other neighbor. It then waits for another row message and forwards it to its other neighbor with increased count.
After receiving two row messages, the node checks the row count from both the messages. If they are not equal, it designates itself as an *intermediate core node*. If the two counts are equal, it designates itself as the *core network head*.

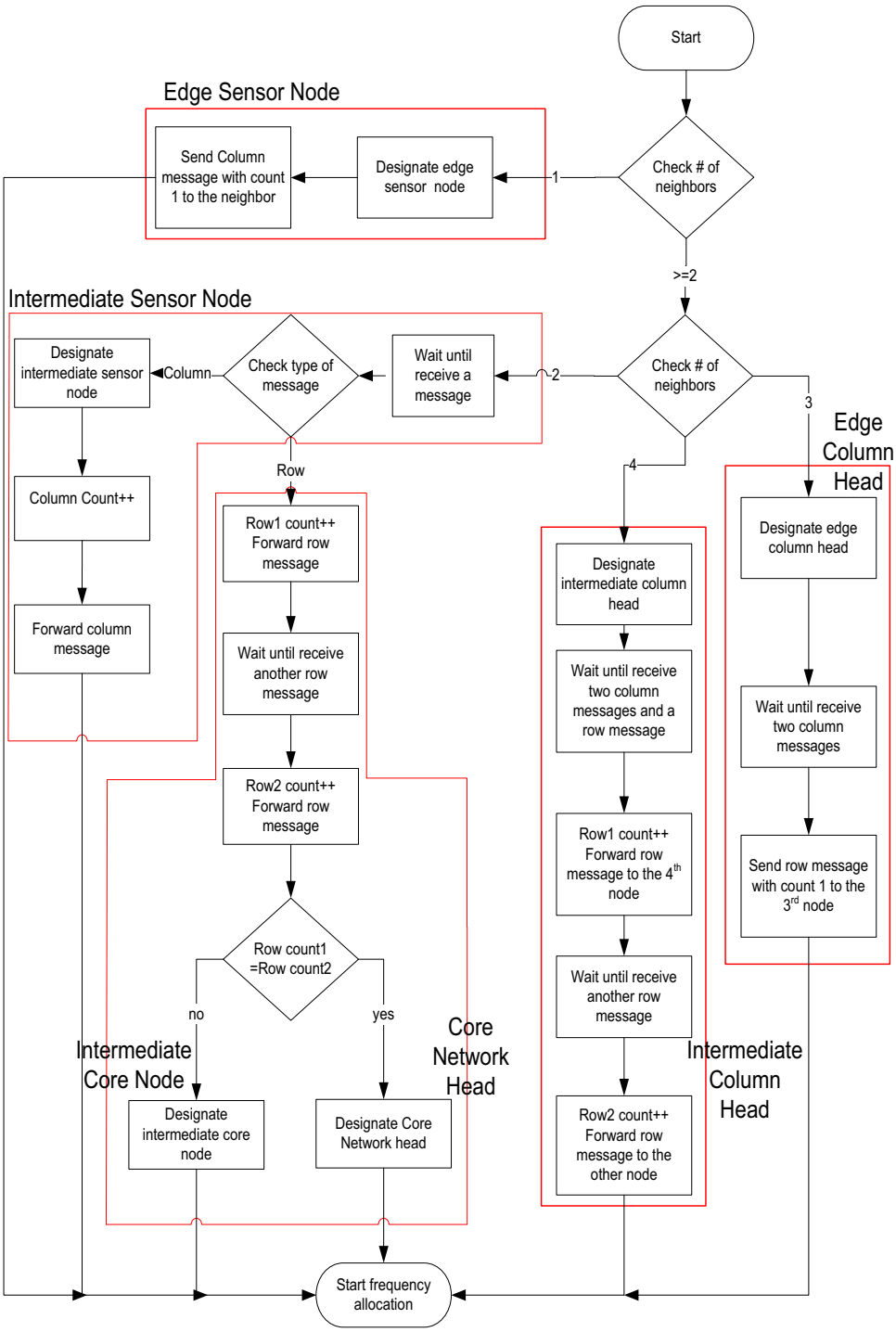

Fig. 3 Relative location algorithm [5]

The core network head then begins the frequency allocation algorithm.

3.4 Frequency allocation algorithm

The frequency allocation algorithm shown in Fig. 4 is used to assign different frequency channels to each of the different columns in the network. This is done in order to reduce the interference that will happen between nodes in different columns.

When the previous algorithm ends, each node will check its type "as decided by the Relative Location Algorithm". And according to that type, one of the following steps will be taken.

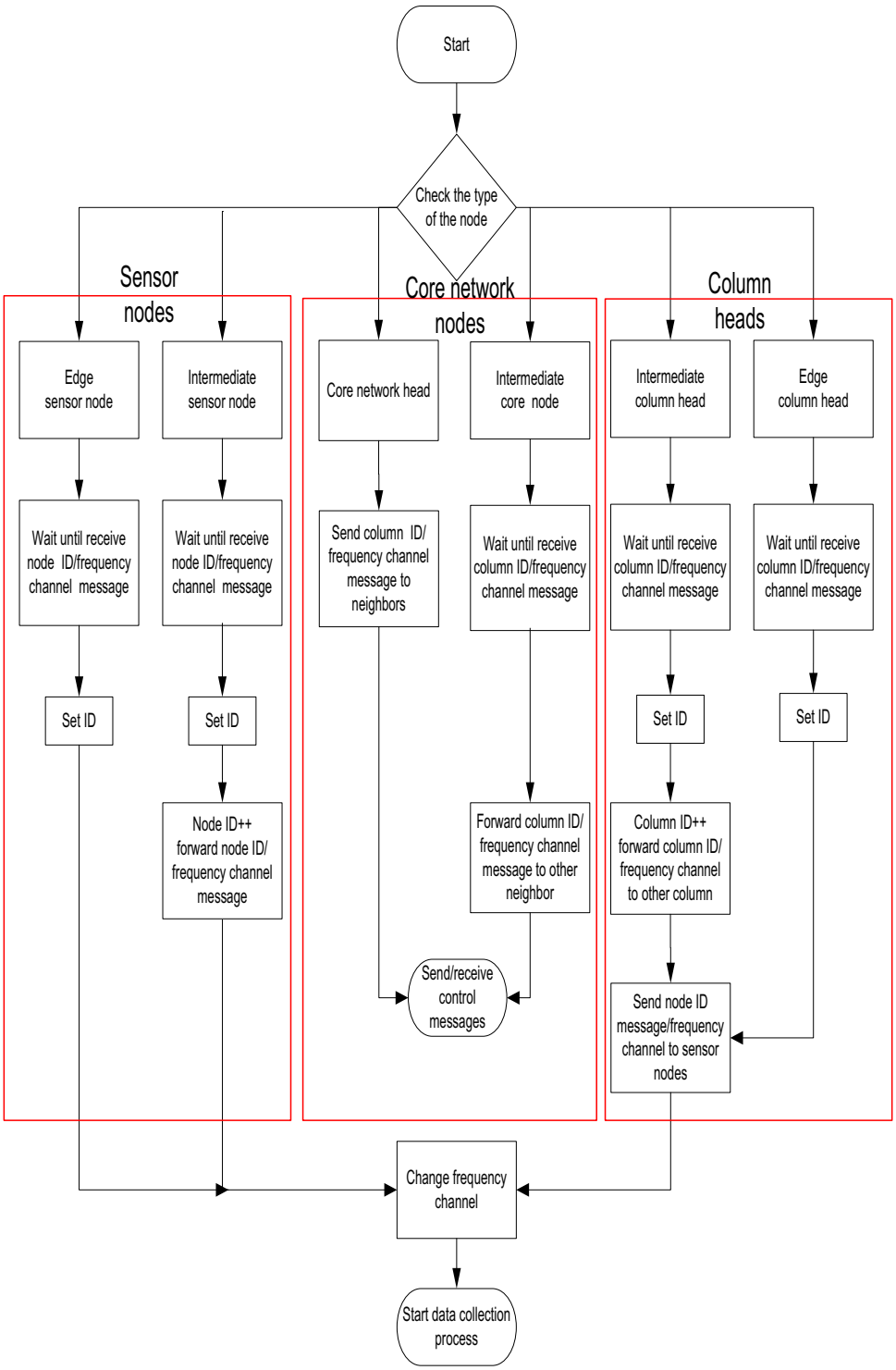

Fig. 4 Frequency allocation algorithm

– *Core network head* The core network head decides how many frequency channels are needed in the network according to the number of columns. And then, it sends massages to both sides of the core network. These messages will hold the frequency channel and the column ID of the first columns on both sides of the network around the core network head. It then can be used to send or receive any control messages between the operator and the sensor nodes if needed.
– *Intermediate core nodes* Intermediate core nodes receive the column ID / frequency channel messages and forward them to the other neighbor without changing them. These

messages will be used to set the column ID/frequency channel of the columns.
- *Intermediate column head* When an intermediate column head receives a column ID/frequency message, it sets its column ID, and then forwards the message with increased column ID and increased frequency channel to the other node in the core network. After a certain delay it sends a new message that holds the frequency channel of the column as well as the first node ID to sensor nodes in its column. And then switches its operating frequency to the value received.
- *Edge column head* An edge column head operates exactly as an intermediate column head but without forwarding the message to nodes in the core network.
- *Intermediate sensor nodes* Intermediate sensor nodes wait until they receive the node ID/frequency message. They set their node ID, and read the frequency channel from the message. Then, they increase the node ID count and forward the message to the other sensor node in the column. After a certain delay, they switch their operating frequency channel to the new value received from the message.
- *Edge sensor nodes* Edge sensor nodes wait until the node ID/frequency message is received. Then, they set their node ID, and after a certain delay, switch to the new operating frequency channel.

After all nodes have changed their operating frequency to the correct value, data collection processes can be started by the column heads [15,16]. Nodes in the core network are then free to monitor the network as well as send/receive any control messages from the network operator.

4 Simulation environment

Simulations were performed using ContikiOS-v2.6 [7] and emulated Sky motes [17]. The three proposed algorithms were simulated in Contiki's built-in simulator COOJA [6] using networks with 9 nodes per column. Networks of 1, 2, 4, and 6 columns were studied. We have selected 9 nodes per column in order to be symmetric around the central node and with a manageable column size. The same goes for the number of columns. This setup can be then replicated as the network grows. COOJA is more an emulator than a simulator, in which the same code that runs on the real motes is the code that runs in COOJA. And so, there is little difference between the performance inside and outside COOJA. In addition, it was validated by the use of a testbed in previous work published by the authors [16]. It was shown from the results that there is small difference in performance between running a code on real motes or inside the simulator. Other papers were found in literature such as [18,19], and [20] that also showed a similar conclusion.

Additionally, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) was used in the simulations as the medium access protocol with acknowledgment messages. IEEE 802.15.4 PHY 2.4 GHz [21] was used as the physical layer technology. The channel model used in the simulations is the Multi-path Ray-tracer Medium (MRM) provided by COOJA with free space distance loss, Additive White Gaussian Noise (AWGN), and low fading. We have chosen this model because in a real life scenario, both the nodes as well as the environment are static. And so, there will be low or no shadowing/fading to affect the data transmission, which is a reasonable assumption.

For the neighbor identification algorithm, parameters observed were mainly the number of ping messages that each node sends before making a decision as well as the periodic timer which is the duration between two consecutive pings. The number of ping messages to be sent tested was 10 and 15 messages/node in order to allow a good resolution of the RSSI value. Lower values were found to make the network to perform badly. Tests were performed with periodic timer as 1, 2, and 3 s.

As for the relative location algorithm, tests were performed in order to see if the nodes in the network will select the correct node as the core network head. Because losing the setup message (row or column message) is the main cause of error in the setup, the parameter analyzed was the number of maximum retransmissions of a packet before dropping it. Values observed were 3, 5, and 7 retransmissions. In Contiki-OS, the default value is 5. These values were chosen in order to test the limit of the system under different scenarios.

The frequency allocation algorithm was simulated using the optimum values that were observed from the previous two algorithms.

Simulations were repeated 200 times for each case with each run working until each node identifies its neighbors for the neighbor identification algorithm, until the core network head is selected for the relative location algorithm, and until every node in the network has selected its new operating frequency channel. The number of simulations was selected in order to get a good statistical analysis of the setup error. Transmission power was set to −25 dBm, which is the lowest level provided by the hardware. This value was selected in order to reduce the transmission range which in turns reduces the interference between the nodes. However, we have also simulated the system with different values of transmission power in order to evaluate its effect on the performance of the system.

5 Simulation results and analysis

In this section, we show the performance results of the three algorithms in networks obtained from the simulations per-

formed using the COOJA simulator. The parameter observed is the setup error. This parameter was selected to test if the algorithms have managed to self organize correctly or not. For the neighbor identification algorithm, the setup is counted as an error if any of the nodes makes a mistake identifying at least one of its neighbors. For the relative location algorithm, a test run is considered correct only if the node selected by the system is the same node defined by the operator, which lies in the center of the network. As for the frequency allocation algorithm, an error is counted if at least one of the nodes in the network has selected a frequency channel different than the one decided by the core network head for the column containing that node. We observed how many of the simulations ended up with the correct decision and which ended up with an error. The following three subsections show the results obtained for each of the algorithms. And the fourth section show the effect of the transmission power on the performance of the system as a whole.

5.1 Neighbor identification algorithm

Table 1 shows the error performance of the neighbor identification algorithm. As we can see from the table, for low values of the periodic timer, the error is high. This is due to the congestion that happens from the increased rate of the transmission as well as the increased loss from the collisions of the ping messages. We also can observe that increasing the number of ping messages helps in reducing the error dramatically. Increased number of messages leads to an increased resolution when calculating the RSSI and so, better decisions can be made. Additionally, as the number of columns increases, the error increases. This is also caused by the increased probability of collisions that comes from the increased number of messages being sent from the additional nodes in the system.

We can conclude that using 15 ping messages with 3 s intervals between them can lead to an almost negligible error, even for a network with 6 columns.

Table 1 Error performance of the neighbor identification algorithm [5]

Number of ping messages	Number of columns	Periodic timer		
		1 s (%)	2 s (%)	3 s (%)
10	1	0	0	0
	2	2	0	0
	4	13	4	3
	6	37	23	16
15	1	0	0	0
	2	0	0	0
	4	1	1	0
	6	23	4.5	0.5

5.2 Relative location algorithm

The error performance of the relative location algorithm is shown in Table 2. This algorithm starts after the neighbor identification algorithm has ended. And so, we have selected the periodic timer as 3 s and 15 ping messages in order to achieve the lowest error. As we can see from the results, the setup error increases as the network grows. However, even large networks still have an acceptable value of error. Also, 5 MAC retransmissions is found to be the optimal value to be used, even for relatively large networks.

5.3 Frequency allocation algorithm

The error performance of the frequency allocation algorithm is shown in Table 3. This algorithm starts after both the previous algorithms have ended. And so, we have selected the periodic timer as 3 s, 15 ping messages, and 5 MAC retransmissions in order to achieve the lowest total error.
The error increases with the number of columns which is expected. However, since the number of messages needed to make a decision is fewer than in the case of the relative location algorithm, we can see that the frequency error is less than the location error even for a network with 6 columns.

5.4 Transmission power analysis

In this section, we present the results obtained from simulating the three algorithms with nodes using three values of transmission powers. These values are -25, -20, and -15 dBm. The other parameters of the system were set as the optimal parameters found in the previous subsections in order to reduce error from the other algorithms. Table 4 shows the total error of the system using three different values of transmission power.
As we can see from the results, increasing the transmission power leads to an increase in the total setup error. This is because increasing the transmission power causes an increase in the transmission range of the nodes which in turns leads to more collisions and to higher setup errors. The increase in the error can also be seen to correlate with the number of nodes in the system. Adding more nodes to the system will lead to more interference and in turn, to higher error.

We can conclude from the results that setting the transmission power to -25 dBm achieves the lowest setup error of the values we have tested. This shows that this value is enough to allow nodes in the system to communicate with their neighbors and collect enough information about them for the algorithms to operate without degrading the performance of the system. And so, we will continue using this value of transmission power throughout the rest of the paper.

Table 2 Error performance of the relative location algorithm [5]

Number of columns	Max MAC Retransmissions	Error			
		Total (%)	Cause of error		
			Neighbor identification (%)	Column head selection (%)	Core network head selection
1	3	1	0	1	N/A
	5	0	0	0	N/A
	7	1	0	1	N/A
2	3	1	0	1	0
	5	1	0	1	0
	7	1	0	1	0
4	3	5	0	1	4
	5	3	0	1	2
	7	8	0	3	5
6	3	18	1	9	8
	5	17	1	10	6
	7	23	1	14	8

Table 3 Error performance of the frequency allocation algorithm

Number of columns	Error			
	Total (%)	Cause of error		
		Neighbor identification (%)	Relative location (%)	Frequency allocation (%)
1	0	0	0	0
2	1	0	1	0
4	5	0	3	2
6	22.5	1	17	4.5

Table 4 Total error with different values of transmission power

Transmission power (dBm)	Number of columns	Total error (%)
−25	1	0
	2	1
	4	5
	6	22.5
−20	1	0
	2	1.5
	4	8.5
	6	25
−15	1	0
	2	2
	4	12
	6	29

6 Testbed experiments and results

Three testbed experiments were performed using Sky motes [17] with default settings in order to test the operation of the three algorithms. The three tests were selected with different number of motes and with different environments. Experiments were done 100 times each until each node has made a decision on its frequency channel. And as in the simulations, an error was counted if at least one of the nodes in the network has made the wrong decision in each of the three algorithms outcome. Parameters were selected as: the periodic timer as 3 s, 15 ping messages, and 5 MAC retransmissions in order to achieve the lowest error. The experiments are described in more details in the next subsections.

6.1 Indoor experiment

In this experiment, motes were placed on a wall 2 m above the ground in an indoor environment as shown in Fig. 5. Distance between nodes was set to 1 m. The indoor environment is a 10 m by 25 m open space office with concrete walls. Two tests were made, a 3-mote network and a 4-mote network. We have started with an experiment with a simple topology in order to confirm the operation of the algorithm on a small scale.

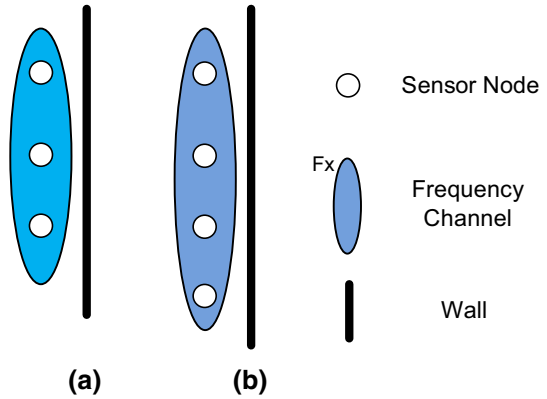

Fig. 5 Indoor experiment **a** 3-mote network, **b** 4-mote network

Table 5 Median RSSI in dBm per node in a 3-mote indoor network

Node	Neighbor	Number of received messages	Median RSSI (dBm)
1	2	16	**−29**
	3	16	−34
2	1	16	**−27**
	3	16	**−28**
3	1	16	**−27**
	2	16	−35

Tables 5 and 6 show the results obtained from the 3-mote and 4-mote networks respectively.

The bold entries in the tables show the neighbor(s) that were selected by the algorithm as the closest neighbors. They are the nodes with the maximum RSSI with respect to the node in question. As we can see from the table, the algorithm was able to decide correctly which of the nodes are closest to the node in question. The total error is shown in Table 14. Out of the 100 runs of the experiment, the percentage of error in the neighbor identification algorithm was found to be 10 % in the 3-mote network and 15 % in the 4-mote network. This is more than the simulation due to the interference from other devices in the office such as computers and laptops using WiFi which uses the same ISM band frequency as the testbed, as well as reflections from the wall and any moving objects like other researchers passing through the room. However, it is an acceptable value. Results from the two other algorithms are not applicable in this experiment due to the small size of the networks as the other two algorithms need more nodes in the column as well as more than one column in order to operate.

6.2 Panels experiment

In this experiment, 6 motes were connected panels in two columns of solar panels while a 7th mote was placed between

Table 6 Median RSSI in dBm per node in a 4-mote indoor network

Node	Neighbor	Number of received messages	Median RSSI (dBm)
1	2	16	**−27**
	3	16	−31
	4	16	−40
2	1	16	**−25**
	3	16	**−26**
	4	16	−30
3	1	16	−33
	2	16	**−27**
	4	16	**−28**
4	1	16	−42
	2	16	−31
	3	16	**−27**

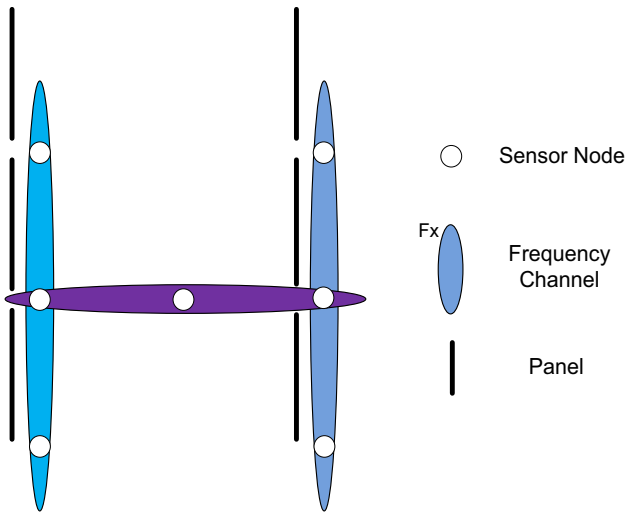

Fig. 6 Panels experiment

the columns as shown in Figs. 6 and 7. This can be viewed as a subset of the topology shown in Fig. 1. This experiment was performed on the roof of the INESC TEC building. Distance between motes on the same column was set to 2 m while the distance between the two columns was set to 4 m. This experiment was done to test the performance of the algorithm in the presence of an obstacle between the columns. We have placed the motes in the space between the consecutive panels in order to reduce the interference coming from the panels. Tables 7, 8, and 9 show the results obtained from the point of view of a node with one close neighbor, a node with two close neighbors, and a node with three close neighbors respectively. As it was stated in the previous experiment, the highlighted cells in the tables show the neighbor(s) that were selected by the algorithm as the closest neighbors. We have only shown results from the point of view of three nodes in order to conserve space. The total error is shown in Table 14. The

Fig. 7 Panels experiment (one column)

Table 7 Median RSSI in dBm per neighbor for a node with one close neighbor

Node	Neighbor	Number of received messages	Median RSSI (dBm)
1	2	16	**−27**
	3	16	−31
	4	16	−34
	5	1	−34
	6	2	−34
	7	16	−30

Table 8 Median RSSI in dBm per neighbor for a node with two close neighbors

Node	Neighbor	Number of received messages	Median RSSI (dBm)
7	1	16	−31
	2	16	**−27**
	3	16	−30
	4	2	−34
	5	16	**−26**
	6	4	−34

experiment was run 100 times, error in the three algorithms was found to be 17, 5, and 2 %, respectively. The increase in error when comparing with the previous experiment comes from the presence of the panels which increased the attenuation of the radio waves and caused some reflections that affected the RSSI calculation and the error in the transmission. However, the algorithms were found to still perform well 76 % of the times.

6.3 Outdoor experiment

The outdoor experiment was done with 11 motes. Motes were connected to the top of 1.5 m posts placed in an open field. Distance between nodes within the same column was set to 2 m while the distance between the two columns was 4 m. Additionally, transmission power was increased in order to get a comparable values of results to the previous experiments. The motes created 3 columns with 3 motes each and with two intermediate motes between the columns as shown in Figs. 8 and 9. This also can be viewed as a subset of the topology shown in Fig. 1. This experiment was done in order

Table 9 Median RSSI in dBm per neighbor for a node with three close neighbors

Node	Neighbor	Number of received messages	Median RSSI (dBm)
2	1	16	**−27**
	3	16	**−27**
	4	3	−34
	5	12	−34
	6	2	−34
	7	16	**−27**

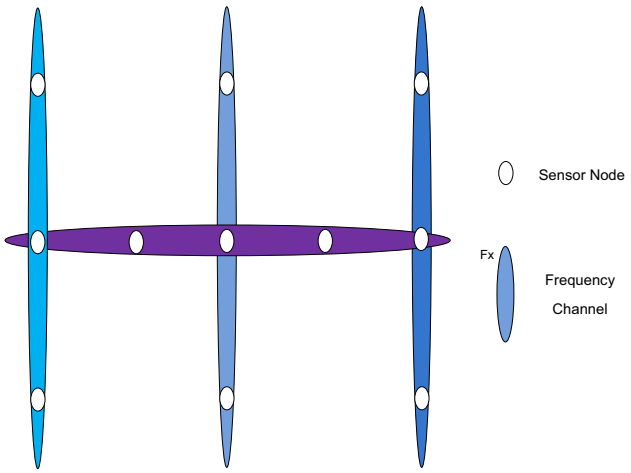

Fig. 8 Outdoor experiment

Fig. 9 Outdoor experiment (real testbed)

Table 10 Median RSSI in dBm per neighbor for a node with one close neighbor

Node	Neighbor	Number of received messages	Median RSSI (dBm)
1	2	16	**−26**
	3	16	−32
	4	16	−31
	5	16	−33
	6	9	−34
	7	2	−34
	8	2	−34
	9	2	−34
	10	16	−28
	11	2	−34

Table 11 Median RSSI in dBm per neighbor for a node with two close neighbors

Node	Neighbor	Number of received messages	Median RSSI (dBm)
11	1	2	−34
	2	2	−34
	3	1	−34
	4	16	−29
	5	16	**−26**
	6	16	−29
	7	16	−29
	8	16	**−25**
	9	16	−29
	10	16	−31

Table 12 Median RSSI in dBm per neighbor for a node with three close neighbors

Node	Neighbor	Number of received messages	Median RSSI (dBm)
2	1	16	**−25**
	3	16	**−26**
	4	15	−32
	5	16	−31
	6	16	−33
	7	1	−34
	8	2	−34
	9	1	−34
	10	16	**−25**
	11	4	−34

Table 13 Median RSSI in dBm per neighbor for a node with four close neighbors

Node	Neighbor	Number of received messages	Median RSSI (dBm)
5	1	16	−32
	2	16	−31
	3	15	−32
	4	16	**−26**
	6	16	**−26**
	7	16	−33
	8	16	−31
	9	16	−33
	10	16	**−25**
	11	16	**−26**

to test the performance of the algorithm in the absence of any interference either from the panels or from any moving objects.

The Tables 10, 11, 12, and 13 show the results obtained from the point of view of nodes with 1, 2, 3, and 4 close neighbors respectively.

From the 100 experiments, the error in the algorithms' performance was found to be 8, 2, and 1 %, respectively. These values are small because of the reduced interference from the lack of panels or moving objects in the open field. The total error is shown in Table 14.

7 Conclusions and future work

In this paper, we have proposed and implemented three algorithms that when used consecutively, allow nodes in a dense wireless sensor network to identify their neighbors, their relative location and select a different operating frequency channel than the other nodes within the network with no external interference from the human operator. The algorithms were implemented and tested using the Contiki-OS and its built-in simulator COOJA. Networks of 1, 2, 4, and 6 columns, each having 9 nodes were tested. Additionally, three testbed experiments were performed in different environments in order to test the performance of the algorithms. The study was done in order to measure the error percentage of the algorithms.

We were able to conclude that the algorithms allow nodes to configure themselves with no interference from the operator of the network. Results showed that the error in performance decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error greatly. However, the value of the error is still acceptable even for high number of simulated columns.

For future work, we will try to make a testbed big enough to test the scalability of the three algorithms and validate their

Table 14 Error performance of the algorithms in the testbed experiments

Testbed	Error			
		Cause of error		
	Total (%)	Neighbor identification (%)	Relative location (%)	Frequency allocation (%)
3-Mote network "indoor"	10	10	N/A	N/A
4-Mote network "indoor"	15	15	N/A	N/A
Panels	24	17	5	2
Outdoor	11	8	2	1

performance when comparing to the simulation. Self-healing is also a good topic to pursue; for example how does the network fix itself should a setup error happen. Also, adding the ability to detect a node failure during the operation phase seems to be a topic to study.

Acknowledgments This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT-Fundação para a Ciência e Tecnologia, Project Ref. CMU- PT / SIA / 0005 / 2009 and by the research Grant No. SFRH / BD / 68759 / 2010.

References

1. Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*, *52*, 2292–2330.
2. Dohler, M., Barthel, D., Maraninchi, R., Mounier, L., Aubert, S., Dugas, C., Buhrig, A., Paugnat, R., Renaudin, M., Duda, A., Heusse, M., & Valois, R. (2007) The ARESA project: Facilitating research, development and commercialization of WSNs. In *The 4th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks. SECON '07* (pp. 590–599). doi:10.1109/SAHCN.2007.4292871.
3. Bachir, A., Dohler, M., Watteyne, T., & Leung, K. (2010). MAC essentials for wireless sensor networks. *IEEE Communications Surveys and Tutorials*, *12*, 222–248. doi:10.1109/SURV.2010.020510.00058.
4. SELF-PVP. In http://www.cmuportugal.org/tiercontent.aspx?id=3374.
5. Abdellatif, M., Oliveira, J., & Ricardo, M. (2014). Neighbors and relative location identification using RSSI in a dense wireless sensor network. In *The 13th annual mediterranean ad hoc networking workshop (MED-HOC-NET)* (pp. 140–145). doi:10.1109/MedHocNet.2014.6849116.
6. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006). Cross-level sensor network simulation with COOJA. In *Proceedings of the 31st IEEE conference on local computer networks* (pp. 641–648). doi:10.1109/LCN.2006.322172.
7. ContikiOS. In http://www.contiki-os.org/.
8. Patwari, N. & Hero, A. O., Using proximity and quantized RSS for sensor localization in wireless networks. In *Proceedings of the 2nd ACM international conference on wireless sensor networks and applications (WSNA '03)*.
9. Halder, S. J. & Kim, W. (2012). A fusion approach of RSSI and LQI for indoor localization system using adaptive smoothers. *Journal of Computer Networks and Communications*, *2012*.
10. Lehsaini, M., Guyennet, H., & Feham, M. (2008). A novel cluster-based self-organization algorithm for wireless sensor networks. In *International symposium on collaborative technologies and systems, CTS* (Vol. 2008, pp. 19–26). doi:10.1109/CTS.2008.4543907.
11. Borbash, S. A., Ephremides, A., & McGlynn, M. J. (2007). An asynchronous neighbor discovery algorithm for wireless sensor networks. *Ad Hoc Networks*, *5*, 998–1016. doi:10.1016/j.adhoc.2006.04.006.
12. Byun, H., & Park, J. (2015). A gene regulatory network-inspired self-organizing control for wireless sensor networks. *International Journal of Distributed Sensor Networks*, *501*, 789434.
13. Bajo, J., De Paz, J. F., Villarrubia, G., Corchado, & J. M. (2015). Self-organizing architecture for information fusion in distributed sensor networks. *International Journal of Distributed Sensor Networks*, *2015*.
14. Wang, F., & Liu, J. (2011). Networked wireless sensor data collection: Issues, challenges, and approaches. *IEEE Communications Surveys and Tutorials*, *13*, 673–687. doi:10.1109/SURV.2011.060710.00066.
15. Abdellatif, M., Oliveira, J., Ricardo, M., & Steenkiste, P. (2012). Impact of data collecting techniques on the performance of a wireless sensor network. In *International symposium on wireless communication systems (ISWCS)* (pp. 111–115). doi:10.1109/ISWCS.2012.6328340.
16. Abdellatif, M., Oliveira, J., & Ricardo, M. (2013). The effect of data aggregation on the performance of a wireless sensor network employing a polling based data collecting technique. In *IFIP wireless days (WD)* (pp. 1–6). doi:10.1109/WD.2013.6686479.
17. Tmote Sky. In http://www.snm.ethz.ch/Projects/TmoteSky.
18. Dunkels, A., Gronvall, B., & Voigt, T. (2004). Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th annual ieee international conference on local computer networks (IEEE, 2004)* (pp. 455–462).
19. Sundani, H., Li, H., Devabhaktuni, V., Alam, M., & Bhattacharya, P. (2011). Wireless sensor network simulators a survey and comparisons. *International Journal of Computer Networks*, *2*, 249–265.
20. Nasseri, M., Al-Olimat, H., Alam, M., Kim, J., Green, R., & Cheng, W. (2015). Contiki cooja simulation for time bounded localization in wireless sensor network. In *2015 spring simulation multi-conference (SpringSim15)* (pp. 1–7).
21. IEEE 802.15.4. In http://www.ieee802.org/15/pub/TG4.html.

Mohammad M. Abdellatif received a B.Sc. in Electronics and Communications Engineering from Assiut University, Assiut, Egypt in 2004. An M.Sc. in Telecommunication Engineering from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in 2009. And a Ph.D. from the Faculty of Engineering, University of Porto, Porto, Portugal. He is also doing his research work with the Wireless Network Group at INESC TEC.

José Manuel Oliveira received his Ph.D. degree in engineering science from the University of Porto in 2005, his M.S. degree in electrical engineering and computer science from the University of Porto in 1996, and his B.S. degree in mathematics and computer science from the University of Porto in 1992. He began his academic career, in 1995, at the Mathematics and Informatics Group of Faculty of Economics, University of Porto, where he is currently an Assistant Professor. He is also a researcher at the Telecommunications and Multimedia Group of INESC TEC since 1992. His current research interests include the dynamic adaptation of multimedia services and telecommunications services provision on mobile environments.

Manuel Ricardo received a Licenciatura degree in 1988, a M.Sc. in 1992, and a Ph.D. in 2000, all in Electrical and Computer Engineering from the University of Porto, Portugal. He is currently an associate professor at the University of Porto, where he gives courses on wireless and computer networks. He also leads the Unit of Telecommunications and Multimedia of the INESC TEC research institute.