Neighbors and Relative Location Identification Using RSSI in a Dense Wireless Sensor Network

Mohammad M. Abdellatif #1, José Manuel Oliveira *2, Manuel Ricardo #3

INESC TEC, Faculdade de Engenharia, Universidade do Porto
* INESC TEC, Faculdade de Economia, Universidade do Porto
Rua Dr. Roberto Frias, 378, 4200-465, Porto, Portugal
{1 mma, 2 jmo, 3 mricardo}@inescporto.pt

***Abstract*—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. An example of such network is a photo voltaic (PV) power plant, where there is a sensor connected to each solar panel. Because such a network covers a large area, the number of sensors can be very large. 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, two algorithms are proposed that can be used to allow each node in the network to automatically identify its closest neighbors as well as its relative location in the network using the value of the Received Signal Strength indicator (RSSI) of the messages sent back and forth during the setup phase. Results show that the error in neighbor identification 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.**

***Index Terms*—WSN, Multi-hop, RSSI, Self-organization, Auto-configuration.**

I. 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].

Even though low data rate is employed in WSNs, other challenging issues appear in the network. Challenges 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 hectares). The objective is to put the solar plant operating at an optimum power delivery point. 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.

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.

In a PV power plant, panels are 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. 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 must be done without any interference from the human operator of the network. After each node identifies its closest neighbors, different messages will be exchanged in order to identify nodes in the center of each column (column heads). These column heads will then exchange messages among themselves in order to find the central node of the network which will be referred to as (core network head). The core network head is different

978-1-4799-5258-8/14/$31.00 ©2014 IEEE

from the rest of the other nodes. It has the ability to decide if the setup was correct or not by confirming if it was selected by the other nodes as the central node or not. A setup is considered correct if the node selected as the central node is in fact the core network head as defined by the operator. Otherwise, the setup will be considered erroneous.

This study considers networks with different sizes, mainly by adding more columns to the network with each column having 9 nodes. The network self-configuration was divided into two algorithms, the Neighbor Identification and the Relative Location algorithms. Their performance was evaluated using the Contiki COOJA simulator [5] which can be downloaded from [6].

The main contribution of this paper is the proposal and evaluation of the two algorithms which together can enable the complete self-configuration of a large network of sensors without any interference from the human operator.

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.

The rest of the paper is organized as follows. Section II gives a background on the work related to this research work. Section III presents the topology proposed for the WSN, as well as the two self-configuration algorithms. Section IV describes the simulation environment and different parameters tested in order to obtain the simulation results. These results are presented in Section V. Finally, paper conclusions and ideas for future work are listed in Section VI.

II. Related Work

Self-configuration in large WSNs has been a hot research topic in the last years.

In [7], 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.

In [8], 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 environment. 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 [9], 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 [10], 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 should the network change. 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.

III. Proposed Topology and Self-Configuration Algorithms

A. 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 Figure 1. Such

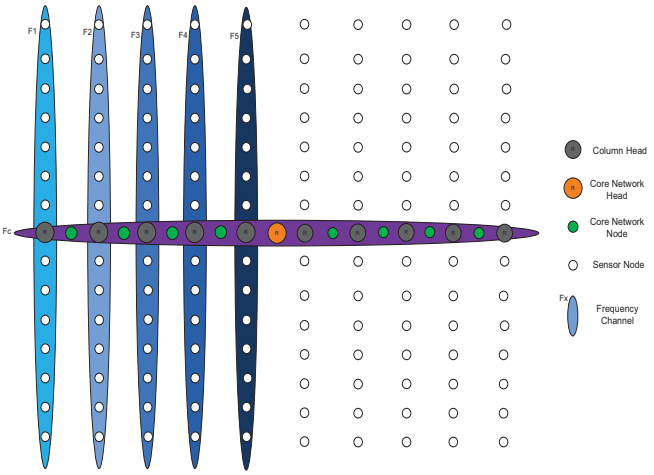

Fig. 1. Network topology.

setup is referred to as a strip-based deployment [11]. 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 meters between each consecutive sensor. While columns have a more wider distance between them in order to allow maintenance access.

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 (green) are placed between columns to enable inter-column communication, and when the need for inter-column communication arises, the respective column heads will have to switch to this core frequency and communicate through the core network. 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, two 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 who 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 (black node in Figure 1). These column heads will then have to decide which node should be the core network head (orange node in Figure 1). These two algorithms are described in more detail in the next two subsections.

B. Neighbor Identification Algorithm

The Neighbor Identification Algorithm is shown in Figure 2. The aim of this algorithm is to allow each node in the

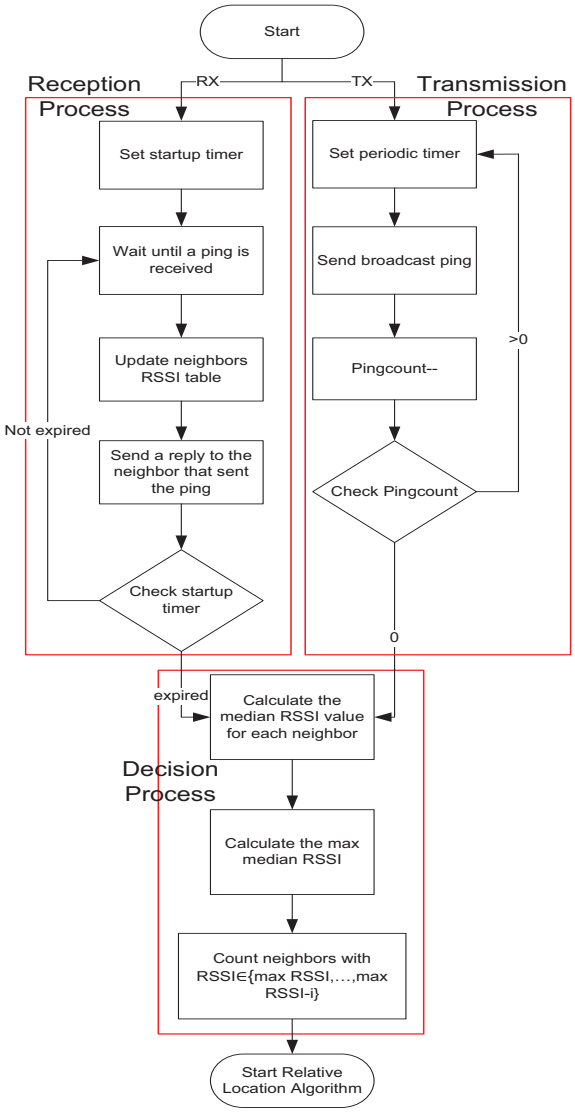

Fig. 2. Neighbor identification algorithm.

network to find its closest neighbors, i.e., neighbors who can be reached directly without multi-hopping. Nodes on each side of a column (white) should have only one close neighbor. While nodes inside a column (white) should have two close neighbors. Column heads (black) should have three or four close neighbors depending on the column location. And, core network nodes (green and orange) should have two close neighbors.

This 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 neighbors will be considered then as the closest neighbors in the network. 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. 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 contaning max RSSI and max RSSI-i, where i is a positive integer. This will tell the node how many close neighbors it has.
  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 quantized 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. We use the median RSSI in order to filter out any extreme values that may appear due to the noise or fading effects.

C. Relative Location Algorithm

The Relative Location Algorithm is shown in Figure 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:

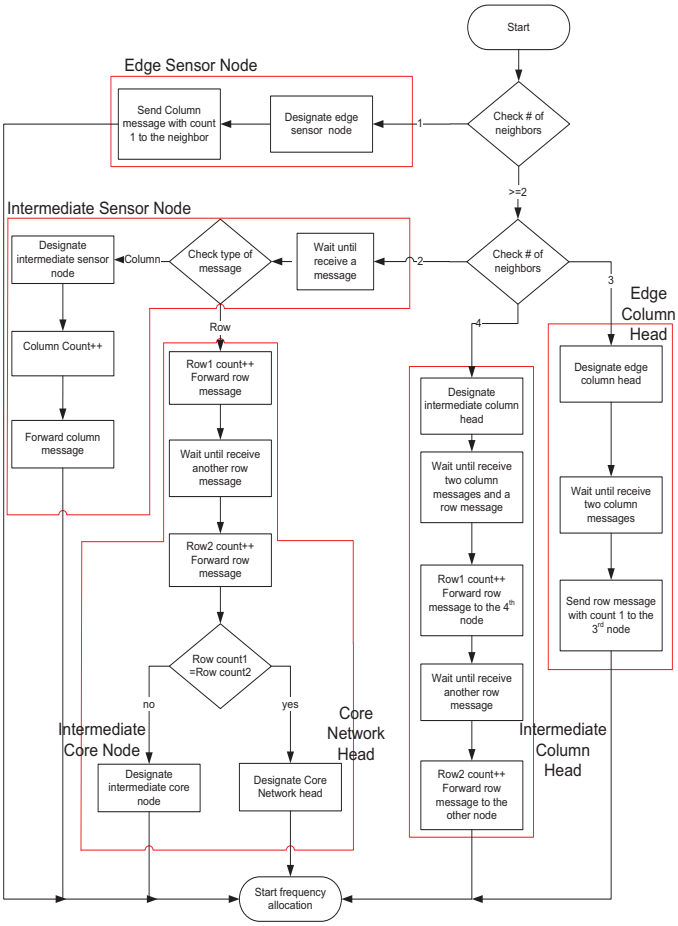

Fig. 3. Relative location algorithm.

- **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 designate 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**.

The core network head then should begin the frequency allocation, which is currently under development.

IV. Simulation Environment

Simulations were performed using ContikiOS-v2.6 [6] and emulated Sky motes [12]. The two proposed algorithms were simulated in Contiki's built-in simulator, COOJA 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.

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 [13] 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 perform badly. Tests were performed with periodic timer as 1, 2, and 3 sec.

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 ContikiOS, the default value is 5. These values were chosen in order to test the limit of the system under different scenarios. Simulations were repeated 200 times for each case with each run working until each node identifies its neighbors for the Neighbor Identification Algorithm, and until the core network head is selected for the Relative Location Algorithm.

V. Results and Analysis

In this section, we show the results of the performance of the two algorithms in networks obtained from the simulations performed using the COOJA simulator. The parameter observed is the setup error. 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. As 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. We observed how many of the simulations ended up with the correct decision and which ended up with an error. The two following subsections show the results obtained for each of the algorithms.

A. Neighbor Identification Algorithm

Table I 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 also is caused by

TABLE I
ERROR PERFORMANCE OF THE NEIGHBOR IDENTIFICATION ALGORITHM.

Number of Ping Messages	Number of Columns	Periodic Timer		
		1 sec	2 sec	3sec
10	1	0%	0%	0%
	2	2%	0%	0%
	4	13%	4%	3%
	6	37%	23%	16%
15	1	0%	0%	0%
	2	2%	0%	0%
	4	1%	1%	0%
	6	23%	4.25%	0.5%

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 sec intervals between them can lead to an almost negligible error, even for a network with 6 columns.

B. Relative Location Algorithm

TABLE II
ERROR PERFORMANCE OF THE RELATIVE LOCATION ALGORITHM.

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%

The error performance of the Relative Location Algorithm is shown in Table II. This algorithm starts after the Neighbor Identification Algorithm has ended. And so, we have selected the periodic timer as 3 sec and 15 ping messages in order to achieve the least error. As we can see from the results, the setup error increases as the network grows. However, even large networks still has an acceptable value of error. Also, 5 MAC retransmissions seems to be the optimal value to be used, even for relatively large networks.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed and implemented two algorithms that when used consecutively, allow nodes in a dense wireless sensor network to identify their neighbors as well as their relative location 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. The study was done in order to measure the percentage of the error of the algorithms.
We were able to conclude that both the algorithms allow nodes to configure themselves with no interference from the operator of the network. Results showed that the error in neighbor identification 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 tackle the frequency allocation algorithm. This algorithm should allow the core network head to plan the frequency channel mapping that will allow nodes with a column to communicate among themselves with no interference from nodes in other columns.

ACKNOWLEDGMENT

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- P T / SIA / 0005 / 2009 and by the research grant number SFRH / BD / 68759 / 2010.

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," in *Computer Networks*, vol. 52, no. 12, 2008, pp. 2292 –2330.
[2] M. Dohler, D. Barthel, R. Maraninchi, L. Mounier, S. Aubert, C. Dugas, A. Buhrig, R. Paugnat, M. Renaudin, A. Duda, M. Heusse, and R. Valois, "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*, June 2007, pp. 590 –599.
[3] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 222 –248, 2010.
[4] "SELF-PVP," in *http://www.cmuportugal.org/tiercontent.aspx?id=3374*.
[5] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, November 2006, pp. 641 –648.
[6] "ContikiOS," in *http://www.contiki-os.org/*.
[7] N. Patwari and A. O. Hero, "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)*.
[8] S. J. Halder and W. Kim, "A fusion approach of RSSI and LQI for indoor localization system using adaptive smoothers." *Journal of Computer Networks and Communication, June 2012*.
[9] M. Lehsaini, H. Guyennet, and M. Feham, "A novel cluster-based self-organization algorithm for wireless sensor networks," in *International Symposium on Collaborative Technologies and Systems. CTS 2008.*, May 2008, pp. 19–26.
[10] S. A. Borbash, A. Ephremides, and M. J. McGlynn, "An asynchronous neighbor discovery algorithm for wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 998 – 1016, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870506000278
[11] F. Wang and J. Liu, "Networked wireless sensor data collection: Issues, challenges, and approaches," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 673 –687, 2011.
[12] "Tmote Sky," in *http://www.snm.ethz.ch/Projects/TmoteSky*.
[13] "IEEE 802.15.4," in *http://www.ieee802.org/15/pub/TG4.html*.