A Distributed Coverage Node Scheduling Algorithm for Dense Wireless Sensor Networks

Daniel R. Matos[1], **Gabriel A. L. Paillard**[2], **Miguel F. de Castro**[1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici, bloco 910 – 60.440-900 – Fortaleza – CE – Brazil

[2]Instituto Universidade Virtual – Universidade Federal do Ceará (UFC)
Av. Humberto Monte, s/n, bloco 901, Campus do Pici
60.440-554 – Fortaleza – CE – Brazil

daniel@dc.ufc.br, gabriel@virtual.ufc.br, miguel@dc.ufc.br

Abstract. *Wireless Sensor Networks (WSNs) have remarkable application potentials: smart homes, environment monitoring, health care, and a myriad of other commercial areas. In general, WSNs operate under severe energy constraints: a typical sensor's battery is irreplaceable and its lifetime is limited. Moreover, those tiny sensor nodes usually have extremely limited physical capabilities as well. Consequently, sensor deployments with reasonable densities are fundamental in order to prolong the lifetime of the networks. Distributing sensors randomly can lead to a redundancy on some areas and this is desirable to overcome the failures of some sensors. In this work, we propose a distributed algorithm to schedule active sensors to both reduce the redundancy of data obtained by the network and prolong its lifetime. Experimental evaluation shows that our approach guarantees a coverage rate around* 99.9%.

1. Introduction

Advances in electronic devices with the introduction of new miniaturization techniques, lower power consumption and increasing processing power, is making the widespread use of devices featuring appealing power and wireless communication capabilities feasible. Wireless Sensor Networks make extensive use of this kind of devices for a variety of applications, such as environmental monitoring, agriculture, health monitoring, education, climate monitoring, and military uses, among others. In addition to its traditional applications, the so-called Internet of Things (IoT) poses a significant demand to WSN, since there is a set of technologies that provide connectivity at any moment and in all locations for IoT components [Atzori et al. 2010].

Every year, electronic devices gains more processor power. The computational capacity of a huge computer in the 60's can now be easily exceeded by a simple small calculator. Despite this evolution in the computational power, battery capacity did not evolve as quickly. Innovation in energy storage area has not kept pace with the growing demand of electronic devices, making it even more important to use the available computing resources efficiently. Moreover, even with the increase of available processing at an increasingly lower cost, it is still necessary to use cheaper devices with higher energy constraints, e.g., sensors to be embedded in the walls of a building to prevent its collapse or to assist in the search for survivors of disasters.

One of the most used techniques for prolonging network lifetime is coverage management, wherein coverage can be defined in terms of a percentage of the monitored space that is covered by the available sensors. This technique allows the turn off of the devices that are in the same sensor area, thereby reducing redundancy and, consequently, increasing the network's lifetime.

WSN usually assumes that the cost of sensors is low, such that a large amount of sensors can be used to monitor an area, specially when the sensors are randomly disposed to avoid black holes, i.e., no coverage areas. Therefore, coverage management turns sensors on and off in order to save resources, reducing redundancy in the gathered data. Solutions developed in this area must consider high level objectives like robustness, scalability and simplicity [Wang and Xiao 2006].

There are a lot of approaches in energy-efficient scheduling mechanisms, each of those considering different design assumptions, such as detection model, sensing area, transmission range, and different target applications [Wang and Xiao 2006]. This work focuses on a simple and elegant solution that can be used in virtually any WSN scenario that allows scheduling. We employed the Scheduling by Edge Reversal (SER) [Barbosa and Gafni 1989] to handle the active nodes in a WSN. The use of this algorithm, originally developed to distributed systems to enter critical region, gives rise to challenges that are unveiled and covered later in this paper. The main design goals of this work is to conceive a simple, scalable and robust solution.

The remainder of this article is organized as follows. Section 2 presents the state-of-the-art on WSN coverage. Scheduling by edge reversal is introduced in section 3. Our approach is detailed in section 4. Section 5 is devoted to present the simulations we designed for experimental evaluation and we analyze the results obtained. Finally, the conclusions of this work and further research perspectives are shown in section 6.

2. Coverage in Wireless Sensor Networks

The coverage in a WSN indicate how a target area is covered by the deployed sensors. There are some coverage models in the literature [Wang 2011]. Most of them use geometric relations to check whether a point is covered by a specific sensor. Boolean models have just one possible result for a given point and a target area: covered (TRUE) or not covered (FALSE). Such models are classified as boolean sector coverage models, that consider a circular sector, i.e., a boolean disk coverage model. In this case, each sensor has a radius and all points inside that circle are covered by that sensor.

There are other network coverage models that consider other factors. The Attenuated Disk Model considers that coverage quality may decrease depending on the distance between the target point and the closest sensor. Therefore, this model defines an attenuation function that models such behavior. The Truncated Disk Model is similar to the Attenuated Disk Model, but imposes a limit to the distance from a sensor to a target. Hence, this model can be understood as a mix of Boolean Disk Coverage Model and Attenuated Disk Model. Moreover, there are also stochastic coverage models built on probabilistic frameworks.

According to [Wang 2011], the main design issues for coverage problems are: *coverage type, deployment method, coverage degree, coverage ratio, activity scheduling and network connectivity*. Briefly speaking, each one of these are:

- **Coverage Type** refers to what kind of target will be covered: *Point Coverage* considers the targets as discrete points inside a monitored area, hence the main goal is to cover all points; *Area Coverage* treats all points inside the monitored area equally, whereas the main objective is to cover the entire area; and *Barrier Coverage*, where the objective is to create a barrier and find a penetration path inside this barrier to the covered targets.
- **Deployment method** refers to how the sensors will be deployed in the target area. Sensors can be placed deterministically in specific points, or can be randomly placed, such as being dropped from an air-plane.
- **Coverage Degree** refers to how a point is covered by WSN. This characteristic shows how many sensors are covering a specific point, when a points is *k-covered*, there are *k* sensors covering that point.
- **Coverage Rate** is a value that indicates how much of target area is monitored (or how many points).
- **Node Scheduling** changes the states of the sensors to active or inactive. If a point is covered by more than one sensor, the scheduling method decides which sensors can be active and for which period time. This is the main area of this work.
- **Network Connectivity** guarantees that all pairs of nodes in the network can be reached, i.e., there is always a path between any pair of nodes.

2.1. Routing in WSN

The routing of collected information in a WSN poses a major challenge in this field. A typical WSN has hundreds or thousands of nodes such that it is not possible to keep a global addressing system, an agent responsible for routing or an address table maintenance. Sensors must act together, in a distributed way, in order to assure that sensed data will be within the reach of the base station.

There are several routing protocols for WSN in the literature, which can be divided according to their network structure in three types: plain routing, hierarchical routing and location based routing [Al-Karaki and Kamal 2004]. In plain routing, all sensors have the same rules and functions, these may belong to a particular routing path. In hierarchical routing, sensors are grouped into clusters and have different roles on the network, each cluster has a member exercising the role of a cluster head, a central node that receives all information from its components and sends to the base station. In location-based routing sensors, positions are available (the nodes are placed in a deterministic way or have some positioning system, like GPS) and this information is used for routing decisions.

3. Scheduling by Edge Reversal

Consider a neighborhood-constrained system composed of a set of *processing elements* (PEs) and a set of *atomic shared resources* represented by a connected directed graph $G = (V, E)$, where V is the set of PEs and E is the set of its directed edges (or arcs), stating the access topology (directed edges are henceforth referred to as arcs). The latter is defined in the following way: an arc exists between any two nodes *if and only if* the two corresponding PEs share at least one atomic resource. Scheduling by Edge Reversal (SER) works as follows: starting from any acyclic orientation ω on G, there is at least one *sink* node, i.e., a node such that all its arcs are directed to itself; all sink nodes are allowed to operate while other nodes remain idle.

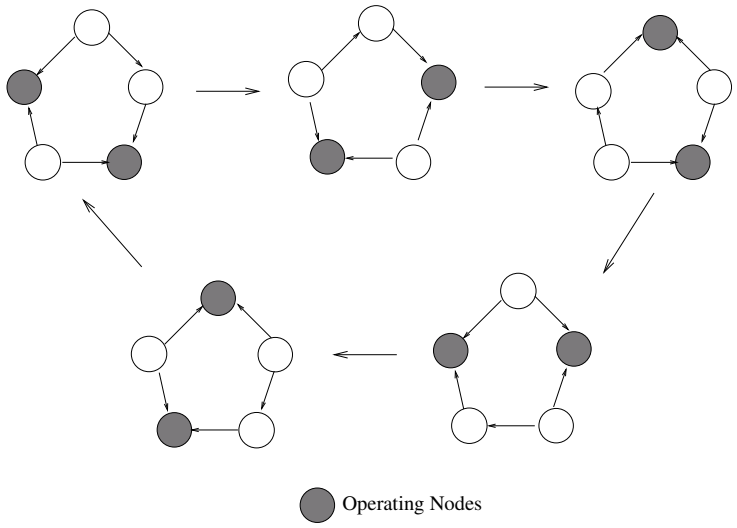

Figure 1. SER dynamics for the Dining Philosophers under heavy load

This obviously ensures mutual exclusion at any access made to shared resources by sink nodes. After operation, a sink node will reverse the orientation of its arcs, becoming a *source* and thus releasing the access to resources to its neighbors. A new acyclic orientation is defined and the whole process is then repeated for the new set of sinks. Let $\tilde{\omega} = g(\omega)$ denote this greedy operation. SER can be regarded as the endless repetition of the application of $g(\omega)$ upon G.

Assuming that G is finite, it is easy to see that eventually a set of acyclic orientations will be repeated defining a period of length P. This simple dynamics ensures that no deadlocks or starvation will ever occur since in every acyclic orientation there exists at least one sink, i.e., one node allowed to operate. In addition, it can be proved that in any period, every node operates exactly the same constant number of times (denoted M) [Barbosa and Gafni 1989].

SER is a fully distributed graph dynamics in which the sense of time is defined by its own operation, i.e., the synchronous behavior is equivalent to the case where every node in G takes an identical amount of time to operate and also an identical amount of time to reverse arcs. Another interesting observation to be made here is that any topology G will have its own set of possible SER dynamics [Barbosa and Gafni 1989].

As an example of SER's applicability, consider Dijkstra's paradigmatic Dining Philosophers problem under heavy load, i.e., in the case philosophers are either "hungry" or "eating" (no "thinking" state). Such system can be represented by a set $\{P_1, \ldots, P_N\}$ of N PEs, in which each PE shares a resource both with its previous PE and its subsequent PE. Thus, taking the original configuration where $N = 5$ and setting an acyclic orientation over the 5 nodes ring, the resulting SER dynamics where $P = 5$ and $M = 2$ is illustrated in Fig. 1.

4. Distributed Scheduling Proposal

The motivation for this work came across the evidence of the possibility of using the scheduling algorithm by reversal edges (SER) to scale the radio nodes in a wireless sensor network and model this solution to solve active nodes scheduling in a coverage manage-

ment problem. The Scheduling by Edge Reversal (SER) algorithm was the chosen one and this solution was used to schedule radios in WSN in order to maintain the coverage of the sensed area and reduce energy consumption. It is worth mentioning that the radio of a sensor can consume more energy receiving messages than transmitting, as we can observe in a data-sheet of Texas Instruments CC2420 radio [Texas 2013]. This algorithm made use of message exchange for coordinate active sensor scheduling, and turning off the radio is by itself a big challenge, because messages cannot be be received when the radio has been turned off.

4.1. Initializing

The first pass was mapping the SER algorithm to schedule radio sensors in a WSN. Our neighborhood-constrained system is the WSN itself. The *processing elements* are the sensors nodes in WSN. In a general way, this can be done as follows: if two sensors in a WSN are close enough, we can consider it redundant, so one of them can be turned off (so this is the clause to consider that the two PE's have an *atomic shared resource* in SER). Then, if two sensors are redundant, there will be an edge between them. After mapping redundant sensors and creating edges between them, the SER algorithm can be applied. In order to maintain the SER correctness, we assume that the nodes have a unique identifier and that the edges are initially set in direction to higher id sensors, as this is a requirement for avoid deadlocks in the use of the SER algorithm.

The first pass of our proposal is to identify all neighbors of a sensor. For this, all nodes send an initial *hello* message and wait for messages from its neighbors. With these incoming messages, the nodes can make an estimation of distances, based on the radio transmission. As this is not the focus of this work and we do not want to use euclidean distances (considering that all sensor knows its position), we used RSSI to estimate these distances. RSSI is not reliable to calculate distances [Heurtefeux and Valois 2012] [Benkic et al. 2008], as it can vary too much in greater distances. Here we just create an edge if nodes are close enough to have considerable redundancy in the sensed area, so the precision in this small distance is sufficient for this work.

In our algorithm we consider only the messages exchanged between sensors close enough, for this we use a reduced power to save energy in message exchange.

4.2. The Lonely Sensor Case

As SER was not originally intented to solve schedule in WSN, we identified a situation that can generate black holes in coverage, specially in network borders. We call these situations *lonely sensors case*. This particular case occurs when a node has only one neighbor and this neighbor, in turn, has more than a neighbor. Figure 2 shows an example. In (a), node 7 has all its edges directed for itself, becoming active. In (b) node 7 reverses its edges, making nodes 1 and 3 active. In (c) nodes 1 and 3 reverse their edges, enabling node 2, at this point node 1 could still remain active, because there are no neighbors active - node 7 will only become active when receiving the edge of 2 next round. This may cause a hole in the monitored area since nodes 1 and 7 will not be active.

In this case, the node that has only one neighbor could continue active after finishing its active time, once its neighbor gets the edge, but still have the edges of other neighbors, ie, the node being off will cause the appearance of a hole in the network that could be avoided if the application needs the largest possible coverage.

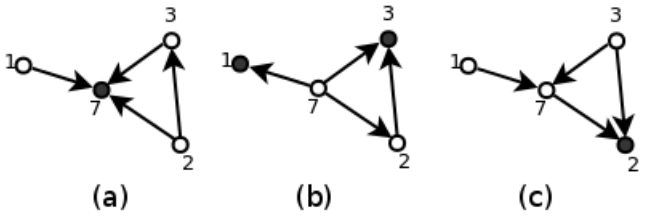

Figure 2. ***In the original SER when a node has only one neighbor, it may occur that both this node and its neighbor are disabled at the same time, which is not a problem in distributed systems, but could cause a hole in the coverage on WSN***

To prevent this situation from occurring, when all sensors identified their neighbors, they broadcast their neighbor list. In this way, all nodes know their neighbors and all neighbors of each one, making possible to identify when a node is in *alone sensor case*. When this happens, a sensor in *lonely sensor case* will not reverse this edge after the end of the round, and it will remain active until one of its neighbors reverts the mentioned edge. When a node is in this condition, it has all edges directed to itself, except the one of the lonely sensor, and it will take this one, finally scheduling that node to sleep.

The first difference in SER usage is in identifying lonely sensors, the inserted amendment aimed to maintain the node active up until its neighbor get its adjacent edges, which maintains the underlying acyclic graph (avoiding ”deadlocks”) and the alternation between nodes accessing their shared resources remains guaranteed (avoiding ”starvation”). The last point is the analysis of the residual energy of the sensors, which may be a delay in one or more cycles in the reversal of the edges and this can result in unequal access to shared resources but the correctness of the new algorithm remains.

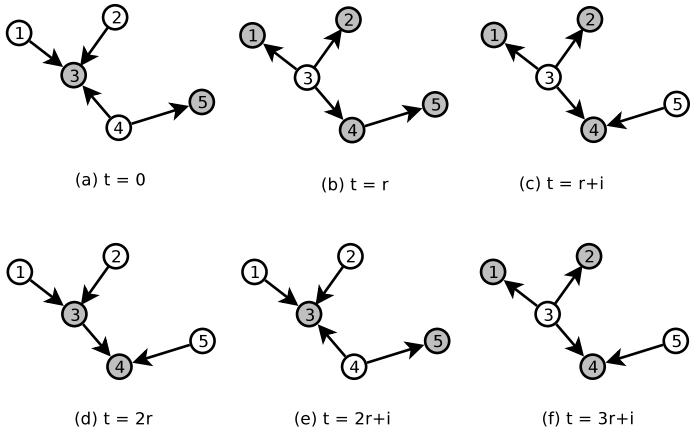

Figure 3. When there is a delay, as in (c) where the node 5 reversed its edge with delay i. In some rounds, the nodes would be again synchronized, because this time will be propagated to the rest of the network.

4.3. Synchronization

If a sensor tries to send a message to a neighbor that has a radio turned off, the message will get lost. In order to avoid that, one solution is the use of synchronized time in all nodes, which it is not always possible. We employ a scheme with the use of timers, counting round times. It works as follows: when a sensor inverts his edges, before putting

itself in a sleep state, a timer is started with a value of a round. When this timer expires, the sensor turns on its radio allowing it to receive messages from its neighbors again. After all incoming messages in that round have been received, it starts a new timer, but now the value of this timer is the duration of that round plus the elapsed time until the receiving of the last message from this round. In this way, we can guarantee that the sensor will be awake at the same moment of the first node, which ended the previous round, avoiding the loss of incoming messages from sensors using distinct timers. With this solution, a delay in a node clock would not be a problem, except for the waste of energy, provided that the delay is not in the order of magnitude of the time of a round, which could cause that the delayed node was considered dead. In some rounds the delay would be propagated to all nodes, as shown in Figure 3 which illustrates the synchronization scheme employed.

4.4. Energy Level and Edge Reversal

Another change in the original SER algorithm consists in the use of current energy level of a sensor to decide if it will or not invert its edges in current round. This strategy will balance the energy levels, allowing an energy consumption in a more elaborate way. Using this strategy creates a problem because of the expectation that the algorithm will reverse all edges in each round, and a sensor cannot be in an active state for an entire round waiting a message that will not come. In order to avoid this, when a round ends, an active sensor checks if this remaining energy is bigger than their neighbors energy level, if so, it will remain active for one more round. We added a field in all communications messages containing actual energy level of the sensors. When an active sensor checks its energy comparing with its neighbors, it will be checking the energy of the last round, but all neighbors remain in sleep state in this round, so not much energy change is expected. When a sensor decides not to invert, it needs to send a message announcing that it will not proceed to invert its edges in this round, so its neighbors can sleep for one more round.

When a sensor receives a message of this kind (not invert), it will check if there are still other edges to receive in this round, if not, it can go to sleep state and notify this to its neighbors. Using energy level to decide edge reversal can impact coverage causing black holes, because a sensor will break the edge reversal cycle. To solve this problem, when a sensor realizes that all its neighbors will be in a sleep state, it changes to an active state, avoiding the black hole creation. This is why a node needs to send a message when it is going to a sleep state, when the expected is to be in an active state.

Using the energy level as a decision before the reversal strategy of edges can affect the network coverage. We have in the left side of Figure 4 the inversion of edges without taking into account the energy level, at each iteration, the active nodes reverse its edges (simple use of scheduling by edge reversal). Note that in 1 - (d) is a repetition of the cycle which continues to occur continuously every three iterations. In 2 we have the inversion edge taking into account that we can not reverse its edges due to comparing the energy level with neighbors. 2 - (c) node 3 decided not reverse its edge by having more energy than neighbors. So during that round he was the one node to stay active. Coverage during the iteration 2 - (c) is reduced due to the reversal cycle break.

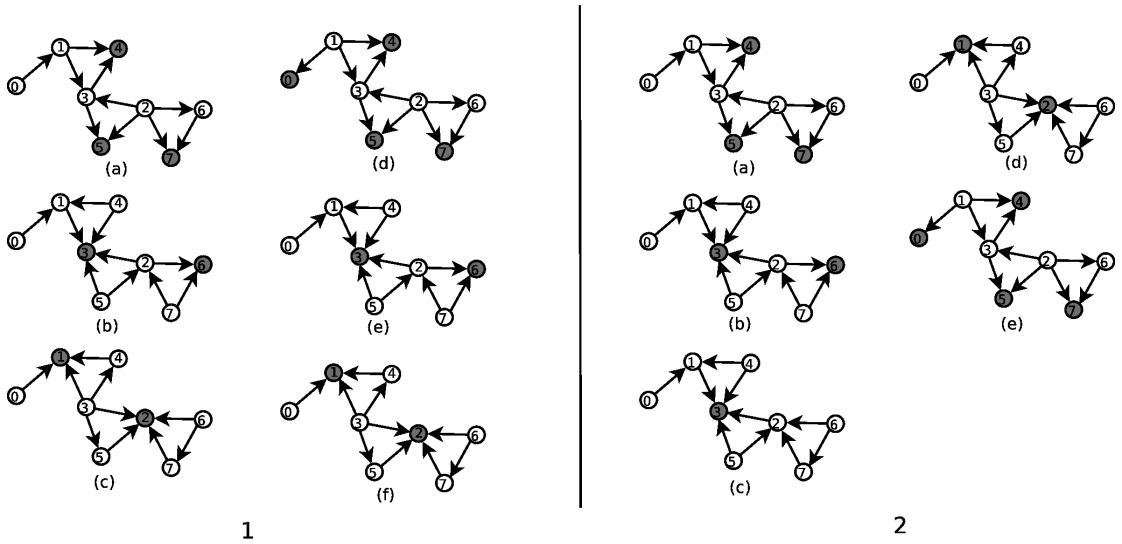

Figure 4. Inversion edges without taking into account the amount of energy (1) Inversion edges taking into account the amount of energy (2)

4.5. A Distributed Coverage Node Scheduling Algorithm for Dense Wireless Sensor Networks

The proposed solution is showed right below in high level pseudo code. Some details, related to the startup of the solution which includes the use of a probabilistic algorithm for the exchange of identities between neighboring nodes, have been omitted but they are detailed in [Paillard et al. 2004].

$Find\ neighborhood$
$Ajust\ Edges$; `Orient edges to higher ID nodes`
$Verify\ Edges$; `Check if all edges point to itself`
if $ActiveState$ **then**
  $StartTimer(EndRound\ :\ RoundTime)$;
else
  $Go\ to\ Sleep$
  $StartTimer(EndRound\ :\ RoundTime)$; `section 4.3`
end

Algorithm 1: Distributed Duty Sheculing

Algorithm 1, which is the main algorithm of the solution, starts by sending initial messages and waiting messages from its neighborhood. Shortly thereafter, another message send the neighborhood list, as stated in section 4.2. From there it checks if it has all the edges, getting active to the end of the round, or whether it should go to the inactive state, if it doesn't have all the edges. Fig 5 shows a diagram state for a better understanding of our entire approach.

Algorithm 2 is executed when one of the timers expires. At the end of a round, the node checks the level of energy in relation to its neighbors, reverses the edges and turns to an inactive state, or send a message annoucing that it will continue in the active state. When a node finishes a round in which it was in an inactive state, it turns on the radio and waits for messages from neighbors also starting the timer that will expire only if there are

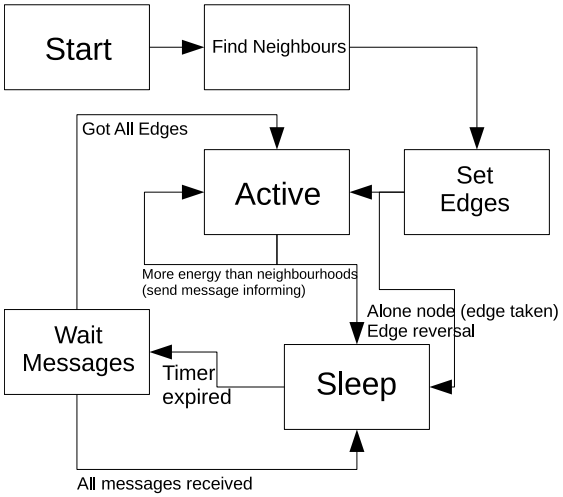

Figure 5. State Diagram

missing messages from any node at the end of the round - this will cause the removal of this node, considering him dead.

Apart from these we have two algorithms that operate when a node receives a message, verifying and executing the steps taken when receiving each type of message, such as whether the node will remain active or not [Matos 2013].

5. Results and analysis

Below are presented the obtained results by simulating the work proposed here and a discussion of results.

5.1. Simulator and parameters

The simulations were performed on the Castalia Simulator [Castalia 2013], version 3.2. The choice of the simulator was based mainly on the following facts: Castalia is WSN specific, open source and is widely utilized [Pediaditakis et al. 2010]. Castalia was developed on *Omnet++* framework [Varga 2001, Varga and Hornig 2008].

We employed an hierarchical architecture where cluster heads were arranged in a grid, for a total of 16 nodes arranged in $4\ x\ 4$ grid equally distributed inside a $100m\ x\ 100m$ simulation area. The routing to the nodes was defined statically following the shortest path to the sink (located at center). As routing is not our focus, we used this schema because of its ease to implement. Cluster Heads send one message when the simulation starts and each sensor associates a cluster head with more powerful signal. The application layer used just generates one package per second in each sensor, simulating sensor readings. The MAC utilized was TunnableMAC (a CSMA/CA implementation available in simulator).

Boolean disk model was used to calculate coverage rate in each round of the simulation (as a percent of total area), since Castalia does not provide a direct output of coverage and boolean disk is a model widely used in the literature.

We considered that an edge will exist between two sensors if the distance measured between them has a maximum of 6 meters. Measures on simulator with RSSI

```
begin
    switch ExpiredTimer do
        case EndRound
            if I don't have more energy than my neighbors then
                SendMsgReverseEdges() ;          Edge Reversal
                Turn off the radio;
                TimerStart(EndIdleTime : (RoundTime));
            else
                sends message that will not reverse
                SendMsgNotReverse();
                TimerStart(EndRoundWithTime : (RoundTime));
            end
        end
        case EndIdleTime
            Activates the radio again and waits for new
            messages
            Turn on the Radio;
            Activate neighbors check timer
            TimerStart CheckNodes with time : RoundTime;
        end
        case CheckNodes
            A neighbor doesn't send messages during a
            round find which one and remove
            LookneighborOff
        end
    endsw
end
```

Algorithm 2: Expired Timer

started showing inconsistency for distances higher than 8 meters, so, up to this distance, there are no problems in measurement estimation method utilized. We simulated a communication with -3dbm signal and obtained 24.84 meters of medium range (repeated 30 times). With this value we considered a sensing range of 12 meters (about half of communication range, as no measurement was less than 24 meters), what give about 62% of common area.

Sensors were randomly deployed and simulations were executed with the number of sensor ranging from 100 to 250 (we did not consider Cluster Heads), each being repeated 30 times. The total time of simulation was 500 seconds.

We executed three different configurations, as follows:

1. ECNS: Energy-Efficient and Coverage-specific Node Scheduling for Wireless Sensor Networks [Meng et al. 2010]. This solution was chosen because of its similarities in being a fully distributed, location aware and simple solution;
2. Hierarchical: No active sensor scheduling, all nodes send their data to Cluster Heads all time;
3. Our Proposal: Distributed Duty Scheduling Algorithm for Large Scale WSN's.

The solution proposed on this work, using adapted version of SER applied to WSN's.

5.2. Round time

To set how long will take a round, we made simulations starting from $5s$ to $75s$, corresponding to 1% to 15%, respectively, of the total amount of time execution, allowing at least six rounds to execute. We observed that with larger round times, more energy will be left at the end, but this is because some nodes have not yet been scheduled, so as $5s$ was the round time that resulted in a more balanced energy distribution between nodes, we chose this time to run the remaining simulations.

5.3. Coverage Rate Results

As the algorithm will schedule sensors, coverage rate is an important measure to verify the viability of our solution. Calculate the area covered by sensors in a WSN is not an easy job[Aziz et al. 2009, Huang and Tseng 2003] and Castalia simulator does not this output available, so we need to create an output in each round and calculate overlapping areas using boolean sensor model. As expected, with more dense networks, less impact on coverage rate will occur. Table 1 shows the results.

Table 1. Coverage Rate

Number of Sensors	Coverage obtainned
100	90%
150	98%
200	99.75%
250	99.9%

As ECNS requires coverage rate as input, it was not directly compared here. We used the coverage rate computed by our algorithm as input for the ECNS to calculate energy levels.

5.4. Energy Consumption

Initial energy in each node was set to 90 Joules (about 1% of capacity of an AA battery). This value was chosen to reduce simulation time necessary until the first sensor dies. The energy results showed that our solution can save energy, every in low density simulations, but with lost of coverage, as seen in later section (90% coverage with 100 nodes). Fig 6 show results for 100 and 250 sensors respectively. ECNS starts saving energy with 250 nodes, but our proposal saved 20% more energy than ECNS and 35% compared to do not schedule sensors in this situation.

5.5. Energy Distribution

To verify as was the energy balance of our solution, we show the results of the spatial distribution of the energy level at the end of a simulation as we can observe in Fig 7. The hierarchical graph is virtually a flat, since all nodes were active all the time. On ECNs, as few of us have been disabled, there is also little energy gap, so we chose to override the results in the same image. More above you can see that there is a variation in energy distribution, since some nodes performed more rounds than others, but the depressions and ridges are small, showing a good balance in consumption between the nodes.

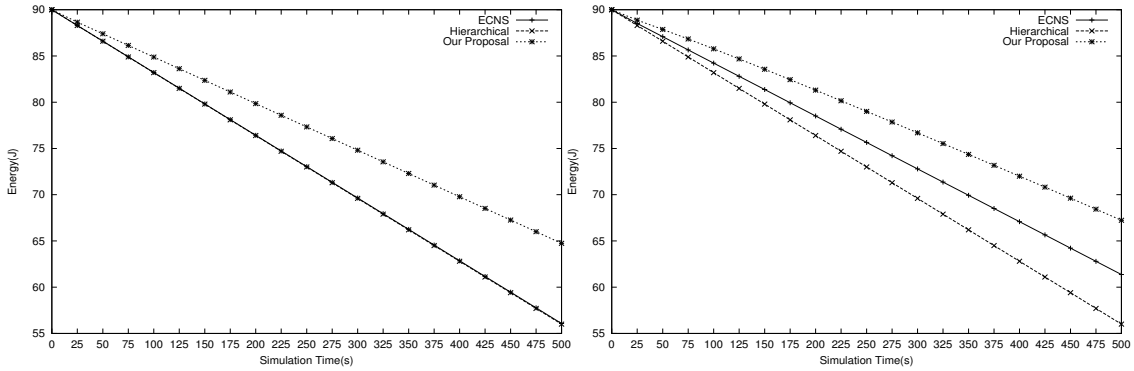

Figure 6. medium residual energy for 100 and 250 sensors

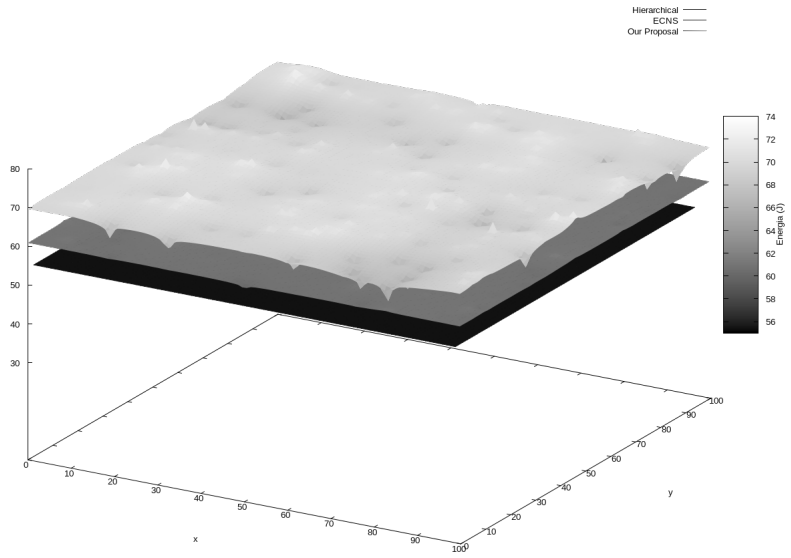

Figure 7. Spatial Energy distribution.

5.6. Network Lifetime

Network lifetime is the time an important feature in a WSN, since one of the main objectives is to extend the network lifetime as much as possible. The definition of Network lifetime may vary depending on the application since there are applications that work correctly only if they have data of all network sensors, can work with already other data from at least a proportion of functioning sensors. Network lifetime was defined here as the time until death of the first sensor (the battery of sensor was completely drained). Fig 8 shows the results in minutes (remembering that the amount of initial energy of the nodes is greatly reduced in our simulations, so the results in minutes).

6. Conclusions

As shown by the results, our solution provides good energy saving with low impact on coverage in dense networks and enhances network lifetime. If coverage required is not so high, the solution can be even used in low density networks. The ECNS is simple, based on message exchanges between neighbors during execution rounds, similar to the solution

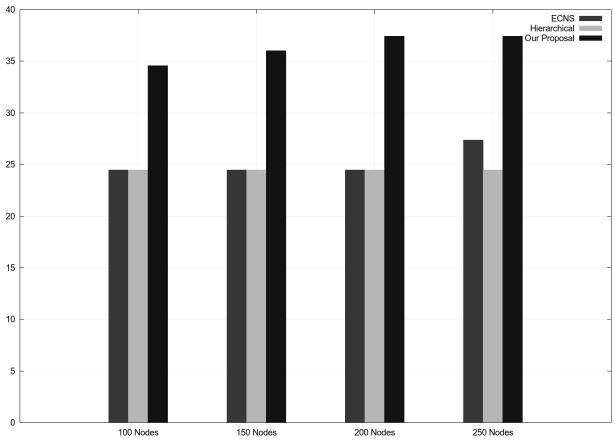

Figure 8. Medium time until first sensor die (in minutes).

proposed here. We have not found other specific solutions based on these same principles. ECNS was compared to two other solutions: LDAS and NRS regarding QoC. Our analysis focuses on energy conservation and the results matched the coverage obtained with the ECNS with much less active nodes.

ECNS starts showing good results only on high dense network, what is expected since it puts a node in inactive state if it has a high number of neighbors in order to guarantee coverage (the exact number depends on coverage rate required). We did not test with higher densities because we obtained 99.9% of coverage rate using 250 sensors.

This work can be easily adapted to heterogeneous networks (with different sensing radio) changing the calculation of distances, taking into account different radio.

The distance between sensors to enable an edge can have an important influence on results of solution. Considering higher distances will lead to more energy saving, but it will decrease coverage rate. Thus, this is application dependant.

As a future work, we could have an integrated solution with routing that can lead to more energy saving. Another future work will be to improve the algorithm to calculate the distances between nodes, as RSSI is not suitable in real world applications. It is possible to develop dynamic intervals between edge reversal, with higher or lower times, depending on network state.

References

Al-Karaki, J. and Kamal, A. (2004). Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *The International Journal of Computer and Telecommunications Networking*, 54(15):2787–2805.

Aziz, N. A. A., Aziz, K. A., and Ismail, W. Z. W. (2009). Coverage strategies for wireless sensor networks. 3(2):134 – 140.

Barbosa, V. C. and Gafni, E. (1989). Concurrency in heavily loaded neighborhood-constrained systems. *ACM Trans. Program. Lang. Syst.*, 11(4):562–584.

Benkic, K., Malajner, M., Planinsic, P., and Cucej, Z. (2008). Using rssi value for distance estimation in wireless sensor networks based on zigbee. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, pages 303–306.

Castalia (2013). Castalia wireless network simulator.

Heurtefeux, K. and Valois, F. (2012). Is rssi a good choice for localization in wireless sensor network? In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 732–739.

Huang, C.-F. and Tseng, Y.-C. (2003). The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, WSNA '03, pages 115–121, New York, NY, USA. ACM.

Matos, D. R. (2013). Um algoritmo distribuído para escalonamento de sensores em rssf. Master's thesis, Universidade Federal do Ceará, Departamento de Computação, Programa de Pós-Graduação em Computação, Fortaleza - Ceará.

Meng, F., Wang, H., Wei, G., and Fan, Z. (2010). Energy-efficient and coverage-specific node scheduling for wireless sensor networks. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, MSWIM '10, pages 368–375, New York, NY, USA. ACM.

Paillard, G., Djerourou, F., Lavault, C., and Ravelomanana, V. (2004). Assigning codes in a random wireless network. In Souza, J. N., Dini, P., and Lorenz, P., editors, *Telecommunications and Networking - ICT 2004*, volume 3124 of *Lecture Notes in Computer Science*, pages 348–353. Springer Berlin Heidelberg.

Pediaditakis, D., Tselishchev, Y., and Boulis, A. (2010). Performance and scalability evaluation of the castalia wireless sensor network simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 53:1–53:6, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Texas (2013). Cc2420(active) single-chip 2.4 ghz ieee 802.15.4 compliant and zigbee ready rf transceiver.

Varga, A. (2001). The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Wang, B. (2011). Coverage problems in sensor networks: A survey. *ACM Comput. Surv.*, 43(4):32:1–32:53.

Wang, L. and Xiao, Y. (2006). A survey of energy-efficient scheduling mechanisms in sensor networks. *Mob. Netw. Appl.*, 11(5):723–740.