# Node discovery and replacement using mobile robot

Kalypso Magklara[1], Dimitrios Zorbas[1] and Tahiry Razafindralambo[1,2]

[1]Inria Lille - Nord Europe, France
[2]Inria Chile, Chile
{Kalypso.Magklara, Dimitrios.Zormpas, Tahiry.Razafindralambo}@inria.fr

**Abstract**

A critical problem of wireless sensor networks is the network lifetime, due to the device's limited battery lifetime. The nodes are randomly deployed in the field and the system has no previous knowledge of their position. To tackle this problem we use a mobile robot, that discovers the nodes around it and replaces the active nodes, whose energy is drained, by fully charged inactive nodes. In this paper we propose two localized algorithms, that can run on the robot and that decide, which nodes to replace. We simulate our algorithms and our findings show that all nodes that fail are replaced in a short period of time.

## 1 Introduction

Wireless sensor networks is a collection of a large number of small size, low-cost sensing devices organized into a cooperative network. A sensor network except from the nodes that collect data also consists of a sink (base station). The nodes use their communication range to send the collected data to the base station. This can be succeeded using either a multi-hop architecture or a Connected Dominating Set (CDS) network. The nodes can be deployed in various physical environments to monitor events in the area. Sensor networks can be used in various applications, such as military surveillance, environmental monitoring, health-care and agriculture.

One of the fundamental issues to be addressed in wireless sensor networks is the network lifetime. Due to the device's limited battery lifetime nodes fail, the connectivity is lost and coverage is degraded. This problem can be solved by exploiting the redundant nodes that exist in the terrain. Previous papers have proposed a few strategies that calculate when or if failed nodes should be replaced using mobile nodes or robots [1], [2] and some other algorithms that do not move any of the nodes but activate the ones needed to cover the environment in question [3]. However, most of them assume complete knowledge of the network or the position of redundant/inactive nodes.

We propose two localized solutions, that do not assume any previous knowledge of the network. This means that in an autonomous network, we need to find out the parameters of the nodes, namely the position of the nodes and their energy consumption, in order to replace them with new nodes. We assume that the robot can learn information on the location of the nodes only if they are within its detection range. The advantages of our solutions are summarized in the following:

1. previous knowledge of the network is not required, namely the position of the nodes or the energy they consume,
2. the robot has the capability to explore the area and detect both active and inactive nodes,
3. both solutions are localized and do not demand a central server to process the network's parameters and
4. we implement a realistic energy consumption model both for the nodes [14] and the robot, for the later specifically we rely on experimental values based on the wifibot [4].

The rest of the paper is organized as follows. In Section 2, we present the related work in the fields of node replacement and sensor network redeployment. In Section 3, we provide a description of our model, while in Section 4 we present our solutions. In Section 5 we evaluate our methods and we compare them. Finally, Section 6 concludes the paper.

## 2 Related work

Most of the research literature related to node replacement in wireless sensor networks deal with the problem by assuming they have previous knowledge of the nodes' position or the events' position or even the nodes' energy consumption [1], [5], [6], [7], [8], [9] and others require mobile sensors [2], [10].

In [1] three policies are proposed, each policy determines the importance of each failed node on the coverage and the lifetime of the network, by assigning a weight to each failed node. If the weight is greater than the policy threshold then the node is replaced, otherwise it is ignored. These policies try to maximize the lifetime and simultaneously minimize the redundant resources. Although to evaluate these weights knowledge of the network is needed and a central server to calculate them. In [5] knowledge of the events is needed and also the redundant nodes position is known since they are stored in the sink. The proposed scheme will schedule both the travels of the robot and the duty cycles of the sensors, the goal is to maintain the coverage while minimizing the traveled distance. However it only considers point coverage, so [6] was proposed, that considers area coverage. The nodes are grouped in sets and a staircase-based scheduling model is implemented, a fixed number of backup nodes are also deployed in sets. In this paper the sets are scheduled so that they will not all be exhausted at the same time, so that different sets can be reclaimed and replaced in different time. Still this scheme works under the assumption that active sensors consume energy at the same rate and there is no sensor failure, therefore in [7] an improvement of the staircase-cased scheme was proposed and

two other schemes, that minimize the hardware cost and the maintenance labor cost. The recharging time of the nodes is considered non-trivial. In [8] the novel combinatorial problem is introduced, one-commodity traveling salesman problem with selective pickup and delivery and also solved by applying the ant colony optimization meta-heuristic. The inactive nodes report their location to the base station, while the active ones report any adjacent sensing hole. Periodically a robot replaces the failed nodes are replaced by inactive ones. This algorithm tries to minimize the traveled distance and the visited nodes. In [9] a small number of robots to replace failed sensors is proposed, the goal is to minimize the motion and the messaging overhead. Three algorithms are studied, a central and two distributed. Every node knows its own location and needs to know which robot is its manager and the location of the manager.

On the other hand, in [2] a localized solution is proposed, however mobile sensors are required for it to work. This algorithm redeploys nodes from targets that are monitored by a large number of sensors and moves them to other targets that are sparsely covered. The nodes initially know only their own and the targets location. Each node covering a target detects and keeps in its memory other neighboring nodes, that cover only the same target. The nodes covering a target can communicate with the nodes covering 2-hop neighboring targets. A head node in its 2-hop neighbor is selected and that one decides which nodes will be moved towards the most sparsely covered neighboring target. Finally in [10] another localized algorithm is introduced that relocates sensors in a mobile sensor network. It is based on the Distance-Sensitive Node Discovery algorithm. All active nodes send their location to neighboring active nodes but also to redundant nodes. The last send reports to the closest active neighbor. When an active node fails the active neighbors try to find the nearest delegated redundant node.

# 3 System Description

Since the devices used in wireless sensor networks have a relatively short transmission range and the terrain size can be several thousands of $m^2$, the nodes are not always close to the sink and they cannot send their messages directly. Therefore, we need to configure the network in a way, so that it will be able to have the messages be relayed through intermediate nodes to reach the destination. Furthermore, using such an approach we also have a way to communicate with the robot even when it is far away from the base station. A multi-hop routing mechanism is required with reduced traffic during communication. Hence, we create a static sensor network and ensure connectivity by using a Connected Dominating Set (CDS).

The main parameters for communication in the energy model of the nodes are the following [12]:

1. the energy consumed by the transmitter $\alpha_{11}$,
2. the energy consumed by the receiver $\alpha_{12}$,
3. the energy for the power amplifier $\alpha_2$,
4. the energy consumed when inactive $\alpha_{idle}$ and

5. the energy consumed when sensing $\alpha_{sense}$

All the values are measured per bit. Assuming a $1/d^n$ path loss, the energy consumed is:

$$E_{tx} = (\alpha_{11} + \alpha_2 d^n)r, \tag{1}$$

$$E_{rx} = \alpha_{12} r, \tag{2}$$

and

$$E_{sensing} = \alpha_{sense} r, \tag{3}$$

where $E_{tx}$ is the energy to send r bits, $E_{rx}$ is the energy consumed to receive r bits, $d$ is the distance between the two devices that communicate with each other and $n$ is the path loss exponent, which depends on the distance.

The energy consumed by the robot per time unit can be distinguished in two cases. In the first case, when the robot is not moved, the energy consumption is constant. In the second case, when the robot is moved, the energy cost depends on the traveled distance.

$$E_{robot} = \begin{cases} \alpha_{idle} & \text{if } \upsilon = 0, \\ \alpha_{mv}\ dst & \text{if } \upsilon > 0. \end{cases} \tag{4}$$

where $dst$ is the distance the robot covered, $\alpha_{mv}$ is a constant value and $\upsilon$ is the speed of the robot. We also assume that there is only one available slot on the robot, so it can only carry one node at any time. We assume that the nodes know their own coordinates and send them to the robot when needed, but even if this information is not accurate enough, according to [13] the robot can navigate towards the nodes, then it can pickup and replace them without any knowledge. The received signal strength is used to determine the direction taken by the mobile robot and it is able to move to the destination, using a hop-by-hop approach.

# 4 Node Replacement Algorithms

In this section we present the two localized solutions, the Reactive and the Proactive algorithm. In both solutions the robot decides which sensing node to pickup and which one to replace based on information that it receives dynamically from the nodes. The robot does not have any knowledge of the network, but it can detect both active and inactive nodes within its sensing range. On the other hand the nodes know their own coordinates. The algorithms never end but the network at some point runs out of inactive nodes.

## 4.1 Reactive Algorithm

Briefly, in the Reactive algorithm (see Algorithm 1) the nodes send an alarm when their energy falls below a threshold. If there are more than one pending alarms, the robot prioritizes them and decides which one to serve first. The robot stays idle, unless there is a node to serve. The energy threshold of each node depends on the respective consumption rate and the distance between the node and the base station. The messages that the nodes send to the robot contain the following information:

1. the coordinates of the node,

2. the remaining time until it fails

In more details, the algorithm works in rounds. Each round starts by updating the nodes energy according to the energy model described in Section 3. If an alarm has been sent by a node in the previous step or in a previous round, then the robot checks the priority of each outstanding alarm and decides which one is the most important. The weight of an alarm is based on three variables. More important are the nodes that have already failed, the more the time that have passed the higher the priority. If there are no failed nodes then it checks the values of the other two variables, namely the remaining uptime and the distance from the robot. In this case, priority is given to the node that is closer to the robot and has the smallest remaining time until it fails. Subsequently, there are three main cases.

In the first case where there is an alarm to be served, the robot checks if it has enough energy to move towards the node in question and then go back from that point to the base station plus the energy to travel the maximum pickup range. If the robot has enough energy then it sets the node as a target and moves towards it, otherwise it sets the base station as a target and moves towards the sink in order to recharge its battery. In the second case, the robot has no alarm to handle but has set the base station as a target, so it moves towards that direction. In the third case, the robot has absolutely no target. Here, as explained before, it will stay at the point where it last stopped at. At this point we can distinguish two sub-cases. Either the robot is at the base station or it is anywhere else in the terrain. If the first it does nothing, if the second then it has to check if it has enough energy to reach the base station.

When the robot is moving, it performs several tasks. It starts by detecting all active nodes and keeping the information it receives in its memory. It, also, detects any inactive node within its sensing range and keeps them in its memory as well. Next, for each detected inactive node, the robot computes the distance between itself and the node. After it finds the one that is closest to its current coordinates, if that distance is less than or equal to the predefined maximum pickup range, it sets that node as a temporary target. On the other hand, if the target of the robot is the base station, it is moving towards it. Otherwise if it is already there, it starts recharging. However, if there is a temporary target then it moves towards that direction, but if it is already there it picks up the inactive node. Finally, if an alarm was set as a target then the robot moves towards that node, but again if it is there it will try to replace it. The only reason not to succeed in the delivery is when it does not carry any nodes. At this point, it will check one more time the inactive nodes in its memory and set the one closest to itself as a temporary target, without taking into account the pickup range. The robot can have two targets, the main target which is an active node or sensing node that needs replacement and the temporary that is an inactive node, that will be delivered. A temporary target is defined only if the slot on the robot is empty and it is prioritized over the main target, because we need the inactive once we reach the failed node.

**Algorithm 1:** Reactive

```
require: N_0 ≠ ∅, TN = NULL, TC = NULL, ALARM = NULL, A_0 = NULL,
         I_0 = NULL, SLOTS = 0
foreach n ∈ N_0 do
    update n energy;
if ALARM ≠ ∅ then
    foreach a ∈ ALARM do
        check a priority;
    set TN = a, where a with highest priority;
if TN ≠ NULL then
    if robot's energy ≥ needed to go to TN and back to base station then
        move robot;
        update robot energy according to traveled distance;
else if TC = basestation then
    move robot;
    if robot NOT recharging then
        update robot energy according to traveled distance;
else
    if robot at base station and robot energy ≤ 1000 then
        recharge;
    else
        if robot's energy ≤ needed to go to base station then
            set TC = base station coordinates;
    if robot NOT recharging then
        update robot energy;
// Tasks performed while updating robot's position
detect active nodes;
insert nodes in A_0;
detect inactive nodes;
insert nodes in I_0;
if SLOTS = 0 and TC ≠ base station then
    foreach i ∈ I_0 do
        check distance from robot;
        keep i with minimum distance;
    if distance(i, robot) ≤ maximum pickup range then
        set TC = i;
if TC = base station then
    if robot is there then
        recharge;
    else
        move robot towards TC;
if TC ≠ ∅ then
    if robot is there then
        pickup;
    else
        move robot towards TC;
if TN ≠ ∅ then
    if robot is there then
        deliver;
    else
        move robot towards TN;
```

## 4.2 Proactive Algorithm

In a nutshell the Proactive algorithm (see Algorithm 2) does not have the nodes send any alarms and the robot never stops moving. It starts by moving randomly in the terrain and while discovering the active and inactive nodes around, it

checks their energy levels and it decides if it needs to take some action.

As in the Reactive algorithm every node has an energy threshold, which depends on the respective consumption rate and the distance between the node and the base station. The information that the nodes send to the robot once they are detected is the following:

1. the coordinates of the node,
2. the remaining time until it fails,
3. the node's current energy,
4. the node's threshold, and
5. the node's maximum consumption rate

More thoroughly, this algorithm likewise works in rounds and starts by updating all nodes' energy levels according to the aforementioned energy model. If no specific target exists or previously chosen coordinates, then a random set of coordinates is calculated. The robot checks if it has enough energy to go to its current destination and back to the base station. If yes it moves towards the coordinates, otherwise it sets the base station as its new target and moves on. The main idea behind the Proactive algorithm is to try to avoid failures, so the robot keeps moving and if it finds an inactive node being close to an active one, and having much more energy than the second, then it will swap them.

When the robot is moving, it performs again various tasks. At the beginning of every move, it detects the active nodes within its range and stores them in its memory. At the same time, it detects the inactive nodes storing them in a different list. It continues by distinguishing four cases. In the first one, the target is the base station and the robot has to move towards it, unless it has reached the sink, in which case it recharges. In the second case, a temporary target exists. If the robot is already at the inactive node then it will pick it up. If not it will move towards it. The third option corresponds to the target being an active node or sensing node. Once again if it has reached the node's coordinates it will try to deliver. If it is not carrying any nodes then as in the Reactive algorithm, the robot will search within its memory for the closest inactive and will go for it. If none of the previous cases are true, then the robot will check the list of detected active nodes and pick the one with the less remaining energy. It will compare that node's remaining energy against its threshold. If it is less than or equal, then the active node will be set as target. After this, it will search in the list of all inactive nodes for the one closest to the robot and will set it as a temporary target. However, if the energy of the active node is greater than its threshold and the robot is not carrying any inactive nodes, then it will search in the list of all inactive nodes but this time it will compare the energy between the chosen active and every detected inactive that is within its pickup range. If the energy of an inactive node is greater or equal to $k$ times the energy of the active node, where $k$ is a static integer, then it will set the active as main target and the inactive in question as temporary target.

**Algorithm 2:** Proactive

```
require: N_0 ≠ ∅, TN = NULL, TC = NULL, A_0 = NULL, I_0 = NULL, SLOTS = 0
foreach n ∈ N_0 do
    update n energy;
if TN = ∅ then
    set random coordinates;
if robot's energy ≥ needed to go to TC and back to base station then
    update robot's position;
if robot not recharging then
    update robot energy;
// Tasks performed while updating robot's position
detect active nodes;
insert nodes in A_0;
detect inactive nodes;
insert nodes in I_0;
if TC = base station then
    if robot is there then
        recharge;
    else
        move robot;
else if TC ≠ ∅ then
    if robot is there then
        pickup;
    else
        move robot towards TC;
else if TN ≠ ∅ then
    if robot is there then
        deliver;
    else
        move robot towards TN;
else
    foreach a ∈ A_0 do
        if a remaining energy smaller than all then
            if a energy ≤ a threshold then
                set TC = A;
                foreach i ∈ I_0 do
                    check i with minimum distance from robot;
                    set TC = i;
            else
                if SLOTS = 0 then
                    foreach i ∈ I_0 do
                        if distance(i, a) ≤ maximum pickup range then
                            if energy i ≥ k * energy a then
                                set TC = i;
                                set TN = a;
```

# 5 Evaluation and discussion of the results

In this section, we start by presenting the main parameters of the simulations, we then simulate the proposed algorithms and finally we compute the robot's energy and the sensing nodes' down time. We assume there are no obstacles in the terrain. The base station is located in the middle of the left side of the terrain and it is the starting and recharging point of the robot. We use two programs written in Perl to create the terrain and produce 2-dimensional pictures of the network respectively and we have implemented our own simulation environment

written in Python for the two algorithms. The total number of nodes is 200 and we use two different terrain sizes, 20K $m^2$ and 40K $m^2$ respectively. We assess the algorithms using six scenarios, by running 30 instances per scenario with random node deployment for each terrain. While the total number of nodes is kept constant, the percentage of the nodes that act as sensing nodes is changed in each scenario and the values used are 20, 40, 60 and 80%. This means that in the first scenario (i.e. 20%) there are 40 sensing nodes out of the 200 nodes and from the remaining 160 some are inactive and some are active. The last ones constitute the CDS, while the sensing nodes constitute the leafs. In the second scenario (i.e. 40%) there are 80 sensing nodes and the remaining 120 are active and inactive and so on. In Figure 1 is depicted an example of a scenario, with 200 nodes in total, while 20 of them are sensing nodes. The base station is plotted as a big square, the sensing nodes as small circles, while the rest of the active nodes are drown as big circles and the inactive nodes as dots. The lines connecting the nodes represent the links between active and/or sensing nodes. The big circle with the base station in its center represents the communication range of the nodes.

The communication range of the nodes is $50m$ and their sensing range is $10m$. The battery of the nodes is initially equal to $20kJ$ (this is a typical energy capacity of one C-type or two AA-type batteries [14]). The values for the sensing node's energy model are $\alpha_{11} = 50nJ/bit$ when transmitting, $\alpha_{12} = 100nJ/bit$ when receiving, $\alpha_2 = 100pJ/bit$ for the power amplifier, $\alpha_{idle} = 10nJ/bit$ when idle and $\alpha_{sense} = 100nJ/bit$ when sensing. Regarding the path loss exponent $n$, its value ranges between 2 and 4, depending on the distance $d$. If $0 <= d < 10$ then $n = 2$, if $10 <= d < 20$ then $n = 2.5$, if $20 <= d < 30$ then $n = 3$, if $30 <= d < 40$ then $n = 3.5$ and if $40 <= d < 50$ then $n = 4$.

The robot on the other hand, when fully recharged has $388.8kJ$ and it needs 14400 seconds to fully recharge. The values used are $\alpha_{mv} = 24.836J/sec$ when it moves with $v = 0.9m/sec$ and $\alpha_{idle} = 8.568J/sec$[1]. Finally, the sensing range of the robot is $50m$ and the maximum pickup range is $100m$.

## 5.1 Simulation Results

In the above mentioned scenarios we assess the percentage of the average time of disconnection. For example in the first scenario, we calculate the average time during which the 40 sensing nodes can not send their data to the base station and then we measure the percentage of that time in the total simulation time. The disconnection can either be caused because the sensing node itself failed or because of the failure of an active node on the path between the sink and that sensing node. We also assess the average time needed by the robot to replace the nodes after they have failed, meaning that after we computed the time that passed from the second an active node or a sensing node failed until the second the robot replaced the given node. After that we calculate the average of all nodes and sensing nodes together and then again the percentage of that time in the total simulation time. Finally, another metric that we evaluate is the energy consumption of the robot during the simulation.

The results that are presented in Table 1 refer to the first set of simulations using the smaller terrain of 20K $m^2$. They show that concerning the time each

[1]These values were experimentally calculated in our lab

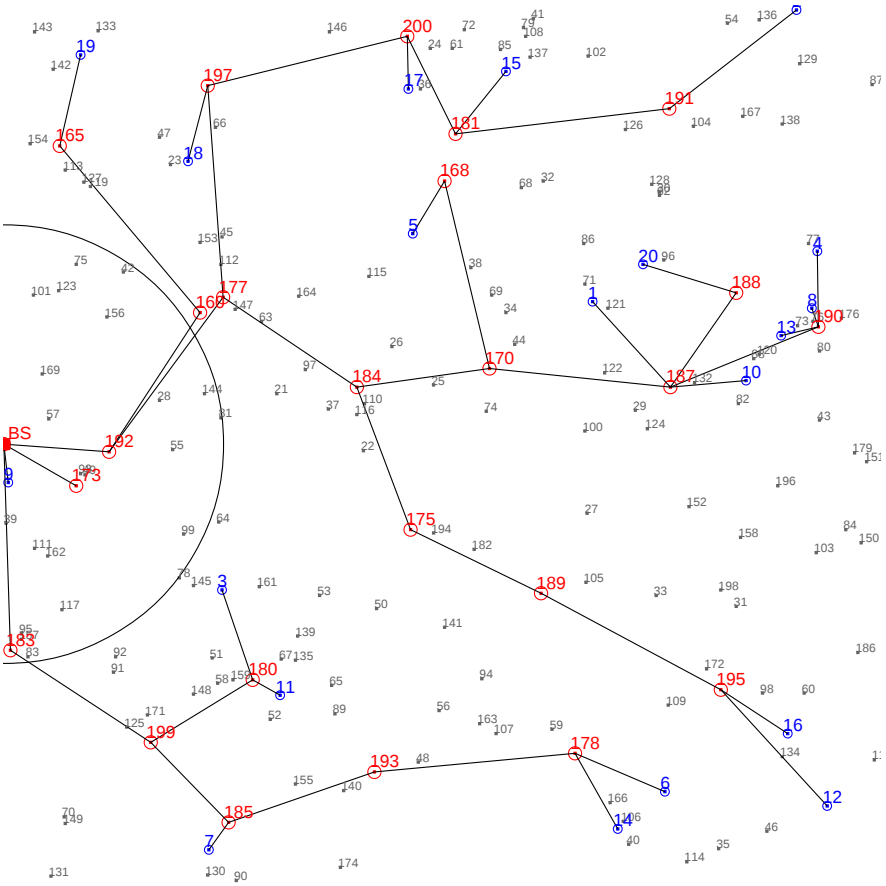


Figure 1: Terrain with 200 nodes from which the 10% are sensing nodes

point of interest in the terrain is not covered, the Reactive algorithm delivered better results than the Proactive. The Reactive algorithm takes advantage of the fact that the nodes are close to each other, since the network density is high, and reacts fast in occurring alarms. Therefore, it needs less time to replace a node. The Proactive algorithm performs well in the scenarios where there are few sensing nodes. In this case, the energy consumption in the network is low, and thus the robot has enough time to travel around the terrain and detect many inactive nodes before the nodes start to fail. Moreover, when the number of sensing nodes is too high, both approaches cannot perform as well as before, because there are less inactive nodes available for the replacements and the robot runs fast out of available nodes.

The results that are presented in Table 2 refer to the second set of simulations using the terrain of 40K $m^2$. In this case, the network density is low and the distance between the nodes higher, forcing Reactive to travel longer distances when a new alarm appears and, thus, achieving a worse performance. Proactive starts moving from the first second and thus has already discovered a few inactive nodes for the first replacements. In this case, it seems that the trade off between the energy consumption of the robot and the reaction time in

Table 1: Average Disconnection and Replacement Time for the 4 Scenarios with Terrain 20K $m^2$

| No. of sensing nodes | Reactive | | | Proactive | | |
|---|---|---|---|---|---|---|
| | % of successful replacements | avg % disconnect time | avg % of replace time | % of successful replacements | avg % disconnect time | avg % of replace time |
| 40 | 100 | 15.5 | 3.45 | 100 | 18.55 | 14.05 |
| 80 | 100 | 26.31 | 2.77 | 100 | 19.46 | 11.18 |
| 120 | 100 | 11.46 | 2.89 | 100 | 36.01 | 6.05 |
| 160 | 100 | 54.4 | 11.77 | 100 | 69.26 | 15.89 |

Table 2: Average Disconnection and Replacement Time for the 4 Scenarios with Terrain 40K $m^2$

| No. of sensing nodes | Reactive | | | Proactive | | |
|---|---|---|---|---|---|---|
| | % of successful replacements | avg % disconnect time | avg % of replace time | % of successful replacements | avg % disconnect time | avg % of replace time |
| 40 | 100 | 27.93 | 4.20 | 100 | 28.81 | 12.33 |
| 80 | 100 | 36.41 | 4.24 | 100 | 36.19 | 6.68 |
| 120 | 100 | 57.83 | 11.88 | 100 | 55.17 | 9.92 |
| 160 | 100 | 81.63 | 41.60 | 100 | 85.06 | 39.37 |

case of a failure is better in Proactive algorithm.

In conclusion, we could say that in both terrains – dense and sparse – both algorithms are able to replace all the failed nodes in the field. The Reactive algorithm performs better than the Proactive when the density is low, while Proactive closes the gap in large terrain sizes. We need to stress that Proactive suffers from the long recharging period that occurs more frequently than in Reactive. A representative example is shown in Figure 2. In reference to the robot's energy in the Reactive algorithm the consumption is far slower than that in the Proactive, since in the first case the robot does not need to travel constantly during the simulation and by staying idle it conserves energy.

# 6 Conclusion and future work

In this paper we dealt with node replacement in wireless sensor networks in order to prolong the network's lifetime, while trying to minimize the time a sensing node's data is not able to reach the base station, the time it takes for the robot to replace an active node and the energy of the robot. The nodes are randomly deployed and we do not assume any previous knowledge of the network. To tackle this problem we presented two localized algorithms. In both of them the robot mainly receives information from the nodes that are within

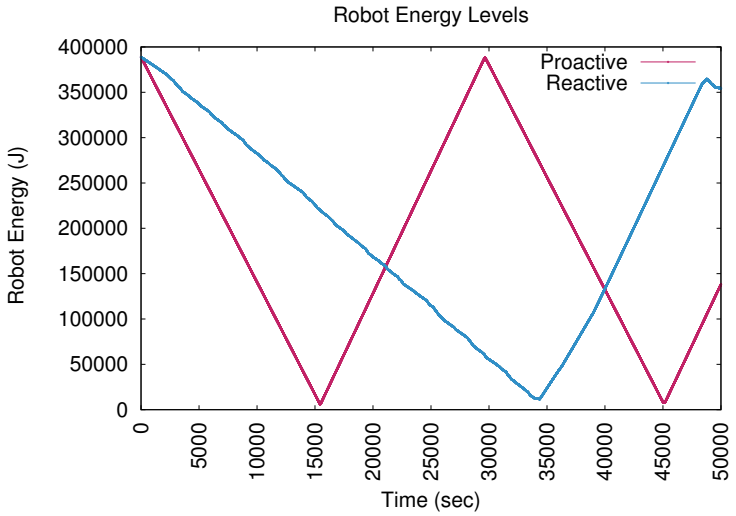


Figure 2: Robot Energy Consumption for second set (40K $m^2$), first scenario (20% of sensing nodes)

its range and uses that to it decides if and which node it should serve. The evaluation results showed that the Reactive algorithm performs better in high-density scenarios, while Proactive closes the gap when the density is low despite the fact that it has to recharge its battery more often. Our future work includes the implementation of a combination of the two algorithms, where the robot evaluates the possible alarm and traveled distance. Moreover, we investigate the use of more than one robot in the field and the use of robots with variable capacity in terms of how many nodes they can carry at the same time.

# References

[1] Parikh, S., Vokkarane, V.M., Liudong Xing, Kasilingam, D.: Node-Replacement Policies to Maintain Threshold-Coverage in Wireless Sensor Networks. In: Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on, pp. 760–765 (2007)

[2] Zorbas, D., Razafindralambo, T.: Wireless Sensor Network Redeployment under the Target Coverage Constraint. In: New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, pp. 2157–4952 (2012)

[3] Sakib, K., Tari, Z., Bertok, P.: Failed node replacement policies for maximising sensor network lifetime. In: Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on, pp. 1–6 (2011)

[4] Wifibots, http://www.wifibot.com/

[5] Tong, B., Wang, G., Zhang, W., Wang, G.: Node Reclamation and Replacement for Long-Lived Sensor Networks. In: Parallel and Distributed Systems, IEEE Transactions on, vol. 22, pp. 1550–1563 (2011)

[6] Tong, B., Li Z., Wang, G., Zhang W.: On-Demand Node Reclamation and Replacement for Guaranteed Area Coverage in Long-Lived Sensor Networks. In: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 148–166, Springer Berlin Heidelberg (2009)

[7] Tong, B., Li Z., Wang, G., Zhang W.: Towards Reliable Scheduling Schemes for Long-lived Replaceable Sensor Networks. In: INFOCOM, 2010 Proceedings IEEE, pp. 1–9 (2010)

[8] Falcon, R., Xu Li, Nayak, A., Stojmenovic, I.: The one-commodity traveling salesman problem with selective pickup and delivery: An ant colony approach. In: Evolutionary Computation (CEC), 2010 IEEE Congress on, pp. 1–8 (2010)

[9] Mei, Y., Xian, C., Das, S., Hu, Y. C., Lu, Y.-H.: Sensor replacement using mobile robots. In: Computer Communications, vol. 30, pp. 2615–2626, Elsevier Science Publishers B. V.(2007)

[10] Li, X., Santoro, N., Stojmenovic, I.: Mesh-Based Sensor Relocation for Coverage Maintenance in Mobile Sensor Networks. In: Lecture Notes in Computer Science, pp. 696–708, Springer Berlin Heidelberg (2007)

[11] Carle, J., Simplot-Ryl, D.: Energy-efficient area monitoring for sensor networks. In: Computer, vol. 37, pp. 40–46, IEEE (2004)

[12] Younis M., Youssef M., Arisha, K.: Energy-aware management for cluster-based sensor networks. In: Computer Networks, vol. 43, pp. 649–668, Elsevier (2003)

[13] Sheu, J., Hsieh, K., Cheng, P.: Design and implementation of mobile robot for nodes replacement in wireless sensor networks. In: Journal of Information Science and Engineering, vol. 24, pp. 393–410, Institute of Information Science, Academia Sinica (2008)

[14] Battery energy storage, http://www.allaboutbatteries.com/Energy-tables.html