Assisted Navigation Algorithm for Wireless Sensor Actuator and Robot Networks

Franco Frattolillo

Department of Engineering, University of Sannio, Benevento, Italy

Abstract: Wireless Sensor, Actuator and Robot Networks (WSARNs) are made of mobile and static sensor nodes that interact to perform specific tasks, such as supporting assisted navigation for mobile robotic nodes that carry out requested operations in hostile environments, where the human presence is impracticable. In this regard, it is worth noting that assisted navigation algorithms have a highly dynamic nature, and are implemented by sensor nodes characterized by limited transmission power and lean autonomy in terms of computing and memory capacity. This paper presents an improved version of the assisted navigation algorithm based on the concept of "credit field". The main aim of the proposed algorithm is to reduce and balance the energy consumption among the static sensor nodes when running the algorithm to manage the presence of obstacles and adversary areas, thus extending the lifetime of WSARNs. The algorithm has been tested on a hybrid sensor network that employs Mica2 Motes as static sensor nodes and Lego Mindstorms robots integrated with a Stargate board developed by Crossbow as mobile robotic nodes.

Keywords**:** wireless sensor networks, wireless sensor actuator and robot networks, assisted navigation, energy consumption.

1. Introduction

Wireless Sensor Networks (WSNs) are collections of micro-devices, called sensor nodes, provided with wireless communication and sensing capability. Such nodes are tiny and low cost devices, which can be spatially distributed in a given area of interest for sensing or monitoring purposes. The main aim is the implementation of networks of collaborative nodes being able to gather information about different physical phenomena within unstructured, dynamic or even hostile environments for a wide spectrum of applications ranging from wildlife and habitat monitoring to health care or battlefield surveillance [1, 2, 3, 4, 5, 6].

Although sensor nodes are affected by limited communication and computing resources, the new trends characterizing WSNs are towards the development of fully-autonomous networks that can adapt to complex situations and react to unpredictable events within their coverage area. Such networks can augment their perceive-and-report capabilities by employing actuator nodes, so as to become perceive-and-react networks, also called Wireless Sensor and Actuator Networks (WSANs) [7, 8]. These capabilities enable WSANs to better control processes and events in complex applications, such as home automation, city lighting, traffic control, precision agriculture, etc.

A further improvement of WSAN consists in replacing the stationary actuators with "actor" nodes, such as mobile robotic nodes, in order to act upon the environment. In fact, the introduction of such nodes makes it possible to build the so-called Wireless Sensor, Actuator and Robot Networks (WSARNs), which can accomplish a lot of tasks besides actuating, such as autonomous nodes deployment or redeployment, batteries recharging, etc [9, 10].

In WSARNs, mobile robotic nodes can both assist the network in order to enhance its capabilities beyond its initial design goal based on the power of mobility, and exploit the network to carry out specific operations in hostile environments, where, for example, the human presence is impracticable [11]. In the latter instance, a relevant role of WSARNs is played in applications in which mobile robotic nodes, characterized by limited on-board resources or acting in constrained environments, have to be guided using the distributed sensing and computing capabilities of static sensor nodes. In such networks, when an event occurs in a particular location, static sensor nodes collaborate to create an optimal path that has to be followed by one or more mobile robotic nodes to reach the designated location. In this regard, it is worth noting that the collaboration among static sensor nodes is managed according specific algorithms that nodes have to run in order to assist the navigation of mobile robotic nodes across areas with dangerous zones that have to be avoided [12, 13].

The creation of navigation paths within WSARNs is a difficult task given both the highly dynamic nature of the applications that exploit them and the need for an effective use of the limited capabilities and resources of static sensor nodes. This paper presents an improved version of the assisted navigation algorithm based on the "credit field" concept [14]. The main aim of such a version is to calculate navigation paths for mobile robotic nodes in the presence of obstacles and to assist their navigation to the event location without causing excessive energy unbalances among the static sensor nodes running the credit field algorithm. Thus, the effectiveness of the whole WSARN can be maximally ensured, since it is possible to minimise the failures of the static sensor nodes that are close to obstacles or adversary areas, which are particularly stressed by the navigation algorithm. The proposed algorithm has been implemented on top of Agilla [15, 16, 17], a mobile agents based middleware purposely designed to fit in the limited capabilities of the Mica2 Mote platform used in the conducted tests. In particular, Mica2 Motes are employed as static sensor nodes whereas the Lego Mindstorms robots integrated with a Stargate board are employed as mobile robotic nodes.

The paper is organized as follows. Section 2 reports on the main related work. Section 3 describes the original navigation algorithm based on the concept of "credit field". Section 4 presents the improved version of the algorithm described in Section 3. In Section 5, the main details concerning the implementation of the improved version of the credit field algorithm are reported. Section 6 describes the main hardware

and software components needed to implement the proposed improved algorithm on a WSARN. Section 7 concludes the work.

2. Related Work

In the last few years many assisted navigation algorithms have been proposed in the literature. The most popular of them mainly follow two approaches, also called the "position-aware" approach and the "position-unaware" approach [18]. The former requires a WSARN to run a specific process to localize the static sensor nodes in global coordinates before executing the assisted navigation algorithm. The latter relies only on the topology of the WSARN, focusing on immediate neighbourhood of static sensor nodes to build the navigation strategies.

The algorithm described in [12] follows the "position-aware" approach. It is based on the "potential field" computation protocol, which enables each static sensor node to compute its own potential value on the basis of the information received from the other nodes that have already computed their potential values. Such a value is an estimate of the node's vicinity to the goal location and, once calculated, has to be broadcasted by each static sensor node to the neighbours. This computation develops in parallel, and ends when the static sensor nodes that are nearest to the mobile robotic node have computed their potential values. Then, static sensor nodes can exploit their potential values to guide the mobile robotic node to the goal along a path that avoids dangerous areas.

The work presented in [19] analyses some algorithms based on the concept of "gradient". Furthermore, the proposed algorithm exploits the measurements of static sensor nodes to generate the optimal trajectories that mobile robotic sensors have to follow to reach an identified target.

The navigation protocol presented in [20] is based on the concept of geographic routing. It computes the path optimization by following the value gradient associated to the static sensor nodes.

The algorithm described in [13] follows the "position-unaware" approach, and is based on the concept of "probabilistic navigation". According to such an approach, each static sensor node computes the probability of the best direction that a mobile robotic node has to follow in order to reach the goal when it is in its vicinity. This enables the mobile robotic node to compute the direction to take in the proximity of each static sensor node. The computation of probabilities is developed by all the static sensor nodes, and enables the building of navigation paths. However, the mobile robotic node needs and maintains a transition model of the network, which has to be predetermined according to the deployment of the static sensor nodes.

Another relevant algorithm based on the "position-unaware" approach is described in [21]. The algorithm implements a method that enables mobile robotic nodes equipped with directional antennas to reach an identified target in an area covered by a sensor network. More precisely, when a static sensor node detects an event, it broadcasts a notification message throughout the entire sensor network so as to build a navigation tree rooted at that sensor node. This enables the mobile robotic node to follow a path on the navigation tree, from node to node, until it reaches the root of the tree.

3. Credit Field Navigation Algorithm

The algorithm proposed in this paper is based on the "credit field" concept [14]. The main idea is that the static sensor nodes placed in the proximity of an event have to start the computation of the navigational forces that will guide the mobile robotic node to the event location (see Figure 1).

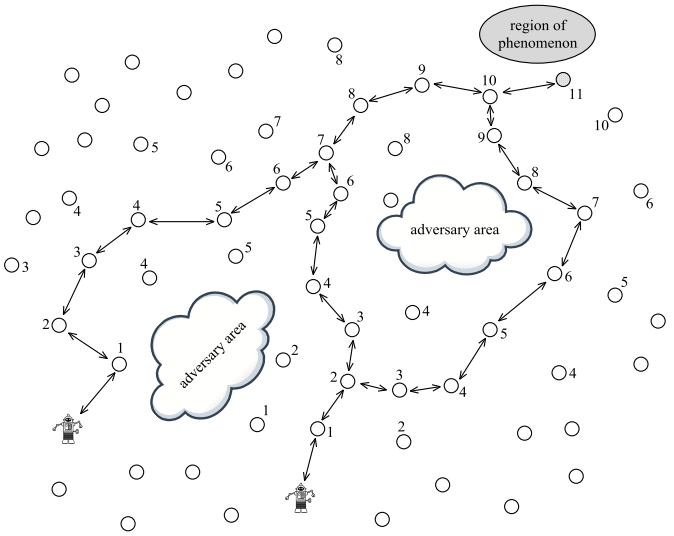

Figure 1. The credit field in the presence of adversary areas

More precisely, the static sensor nodes detecting an event identify the location that has to be reached by mobile robotic nodes, also called "region of phenomenon". They form a "cluster", and elect the static sensor node with the best event measure as their "cluster leader". Then, the leader broadcasts to its neighbours a request packet, which represents an alert signal concerning with the recognition of a phenomenon to monitor and propagates through the sensor network until it reaches one or more mobile robotic nodes.

Each mobile robotic node receiving the alert signal decides whether it can reply to the received request depending on an adopted decision criteria, which can take into account, for example, the initial location of the mobile robotic node with respect to the region of phenomenon or its power autonomy.

The mobile robotic node that can reach the region of phenomenon replies to the cluster leader by sending a packet that follows backward the same routing path of the request packet. This enables each static sensor node on the routing path to increase by one the number of hops required to the reply packet to reach the cluster leader. As a consequence, when the cluster leader receives the reply packet, it can generate its own "credit value" by setting it to the number of hops required to the reply packet to arrive. After that, the cluster leader can broadcast its credit value to its one-hop neighbours by sending an "advertisement packet".

When a static sensor node receives an advertisement packet with a given credit value, it can generate its own credit value by setting it to the received value minus one. Then, it is allowed to propagate this new credit value, but only if it lies on the routing path followed by the previous request/reply packets. During this flooding phase, if a static sensor node receives further advertisement packets and has already computed its own credit

value, it simply ignores them. In fact, such a procedure is followed recursively, hop-by-hop, until the selected mobile robotic node can compute its credit value and set it to 0.
When the mobile robotic node is selected, the procedure ends and the static sensor nodes appear to be arranged according to a credit hierarchy. In fact, such a hierarchy can be followed by the selected mobile robotic node to reach the region of phenomenon by choosing, at each hop, the node with the highest credit value.

4. Improved Algorithm

The static sensor nodes running the credit field navigation algorithm have to exchange many messages whenever the navigational forces, which will guide mobile robotic nodes to the event location, have to be computed. This causes a high energy consumption for the nodes due to the computing and communication overhead. Such a situation can be also worse if the navigation region is characterized by the presence of many dynamic obstacles or adversary areas. In this case, the static sensor nodes in proximity to obstacles and adversary areas have to preserve their energy, since their failure can prevent the algorithm from calculating a correct and secure path for mobile robotic nodes to avoid such obstacles and areas [22, 23, 24].

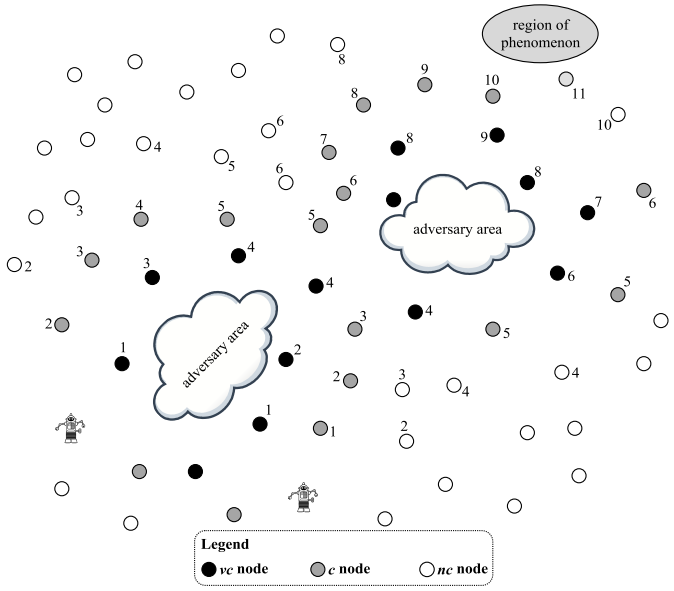

Figure 2. Classification of proximity

To take into account the situation described above, the improved version of the credit field navigation algorithm introduces a classification of the static sensor nodes based both on their distance from obstacles and on their battery power levels compared to those ones of their neighbours. In fact, based on such a classification, each static sensor node can decide to what degree it can participate in the message flooding needed to compute the navigational forces exploited by mobile robotic nodes during their navigation. The basic idea is that the early static sensor nodes to be involved in the message flooding have to be both far from obstacles and provided with a high level of battery power compared to that one of the neighbours. On the contrary, the static sensor nodes that are very close to obstacles and characterized by a low level of battery power should not be involved in the message flooding particularly when their neighbours are provided with high levels of battery power.

Table 1. Behaviour of a static sensor node classified as "very close" to an obstacle

Battery power level of the node	Battery power level of its neighbours		
	H	*M*	*L*
hl	*NRn*	*Nn*	*Sn*
ml	*NRn*	*NRn*	*Nn*
ll	*NRn*	*NRn*	*Nn*

More precisely, the improved version of the credit field navigation algorithm is based on three sub-algorithms, which have to be run by each static sensor node:

- the algorithm to classify the level of proximity of a static sensor node to an obstacle or adversay area, named *ProxA*;
- the algorithm to estimate and update the battery power level of a static sensor node, named *BattA*;
- the algorithm to estimate and update the battery power levels of the neighbours of a static sensor node, named *NeighA*.

A static sensor node runs *ProxA* whenever it directly detects an obstacle, which can be also represented by an adversary area or by a dangerous event for the navigation of mobile robotic nodes. In this case, the node is classified as "very close" (*vc*), and has to send this detection information to all its neighbours.
The static sensor nodes that cannot directly detect an obstacle or an adversary area but that can receive a detection information from a neighbour are classified as "close" (*c*) to an obstacle.
All the other static sensor nodes that cannot directly detect an obstacle and that do not receive any detection information are classified as "not close" (*nc*) to an obstacle.

Table 2. Behaviour of a static sensor node classified as "close" to an obstacle

Battery power level of the node	Battery power level of its neighbours		
	H	*M*	*L*
hl	*Nn*	*Sn*	*Sn*
ml	*Nn*	*Sn*	*Sn*
ll	*NRn*	*Nn*	*Nn*

BattA is run by each static sensor node of a WSARN and estimates the battery power level of the node as "high" (*hl*), "medium" (*ml*), and "low" (*ll*) depending on specific predefined thresholds. The estimate is made periodically or when a static sensor node takes part in the computation of the navigational forces for mobile robotic nodes. In fact, whenever the battery power level of a static sensor node changes, the new value of the level is sent to all the neighbours.
NeighA is run by a static sensor node upon the receipt of a message from a neighbour wanting to communicate the change of its own battery power level. The main aim is to derive a global information that summarizes the battery power levels of the neighbours. Therefore, if 50 percent of the neighbours have a high (*hl*) battery power level and the remaining neighbours have at least a medium (*ml*) battery power level, a static sensor node can consider its neighbours as highly charged (*H*). On the contrary, if 50 percent of the neighbours have a low (*ll*) battery

power level and the remaining neighbours have, at most, a medium (*ml*) battery power level, a static sensor node can consider its neighbours as little charged (*L*). In all other cases, a static sensor node considers its neighbours as averagely charged (*M*).

After the execution of *ProxA*, *BattA*, and *NeighA*, each static sensor node can decide how to behave when the credit field has to be computed to guide mobile robotic nodes to an event location. To this end, the behaviour of a static sensor node can be specified as "strategic" (*Sn*), "normal" (*Nn*), and "not relevant" (*NRn*) with respect to the computation of the credit field. More precisely, if a node is specified as *Sn*, it must participate in the computation of the credit field. On the contrary, if a node is denoted as *NRn*, it is not forced to take part in the computation of the credit field. Finally, if a node is specified as *Nn*, it may participate in the computation of the credit field with probability of 50 percent.

Table 3. Behaviour of a static sensor node classified as "not close" to an obstacle

Battery power level of the node	Battery power level of its neighbours		
	H	*M*	*L*
hl	*Nn*	*Sn*	*Nn*
ml	*NRn*	*Nn*	*Nn*
ll	*NRn*	*Nn*	*Nn*

Table 1 summarizes how a static sensor node classified as "very close" to an obstacle has to behave depending on its battery power level and on the battery power level of its neighbours. Such a node is considered as "strategic" only when its battery power level is high and its neighbours have a low battery power level. In all other cases, it is scarcely involved in the computation of the credit field. This is because the node should save as much energy as possible in order both to extend its lifetime and to be useful in case the neighbours run out energy and cannot keep mobile robotic nodes away from the obstacle.

Static sensor nodes classified as "close" to an obstacle play a strategic role in the computation of the credit field (see Table 2). They have to participate frequently in such a computation, since they are required to keep mobile robotic nodes away from obstacles. In fact, they are allowed to consume more energy, since, if they run out of energy, they can be always supported by the neighbours classified as "very close" to an obstacle.

The static sensor nodes classified as "not close" to obstacles (see Table 3) are located in areas that do not require particular attention. They can be activated with a normal frequency, which has to be reduced only if the nodes are characterized by a low battery power level compared to that one of the neighbours.

5. Implementation

The improved version of the credit field algorithm has been implemented in Agilla, which is a middleware that makes it possible to program devices characterized by limited computing capabilities, such as static sensor nodes [15, 16, 17, 25].

Agilla, whose architectural model is shown in Figure 3, is based on a programming paradigm that employs "mobile agents" as basic programming units. Such units are small programs that can autonomously run on the static sensor nodes and communicate according to the Linda tuple space model [26]. They can also migrate or clone across the static sensor nodes of the network while maintaining their state. Moreover, Agilla provides each static sensor node with the list of its neighbours, and implements both local and remote tuple space operations.

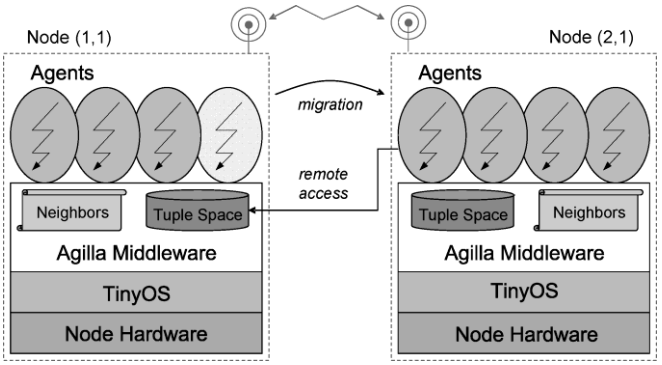

Figure 3. The Agilla model

The proposed implementation consists of the following main agents: *Detector*, *Searcher*, *Updater*, *Status*, *Adversary*, *Replier*, *Initializer*, and *Navigator*.

A *Detector* agent is run on each static sensor node of the network and is a "stationary" agent. It is created by a setup flooding, when the sensor network is deployed. Its main task consists in detecting events and in running the algorithm that elects the cluster leader node by exchanging messages in the tuple spaces of its neighbours. In this regard, when an event is detected, the Detector agent allocated on the cluster leader node takes charge of starting up an Updater agent, a Status agent, and a Searcher agent.

An *Updater* agent is created by a Detector agent on a cluster leader node. Its main task consists in running the *BattA* algorithm on the static sensor nodes of the network, thus classifying them as *hl*, *ml*, or *ll*. Therefore, once it has been created on a node, it clones itself on all its neighbours. Then, it estimates the battery power level of the static sensor node on which it runs, and periodically communicates it to all its neighbours.

A *Status* agent runs the *NeighA* algorithm on the static sensor nodes of the sensor network. It reads the battery power levels of its neighbours and classifies them as *H*, *M*, or *L*. Similarly to an Updater agent, it is created by a Detector agent on a cluster leader node. Then, it clones itself on all its neighbours, thus spreading throughout the sensor network.

A *Searcher* agent is created by a Detector on a cluster leader node, and starts the computing phase that searches a mobile robotic node which can reach the region of phenomenon. During such a phase, the first Searcher agent created on the cluster leader node clones itself on all its neighbours. In this way, it can reach the neighbours and check whether they are in one of the three possible conditions: the reached node is (1) a mobile robotic node; (2) a static sensor node; (3) a static sensor node inside an adversary area.

In the first case, the Searcher agent applies pre-defined criteria and decides whether the mobile robotic node can reach the region of phenomenon. In this case, the Searcher agent creates a Replier agent on the mobile robotic node, otherwise it stops its execution.

In the second case, the Searcher agent runs the *ProxA* algorithm to state the proximity level of the node to a possible obstacle or an adversary area. Therefore, if it detects an obstacle or an adversary area, it creates an Adversary agent, which takes charge of classifying proximity. Otherwise, if the Searcher agent does not detect any obstacle or adversary area and the proximity of the node has still not set, the Searcher agent directly sets it to *nc*.

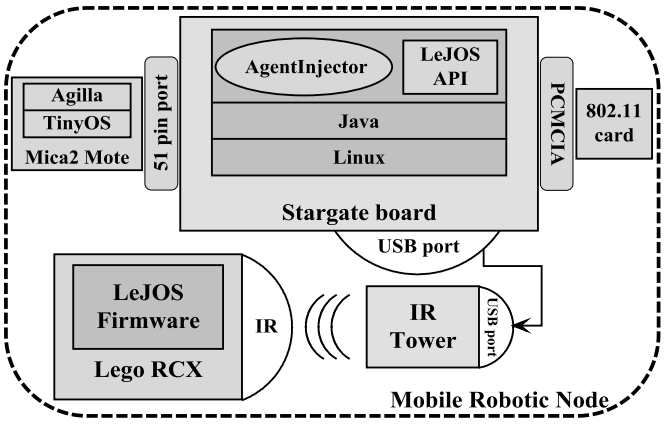

Figure 4. Architecture of a mobile robotic node

After the classification of the node proximity, the Searcher agent decides whether the node has to participate in the computation of the credit field. To take such a decision, it reads the tuples representing the classifications made by the Updater and Status agents, and implements the behaviour summarized by Table 1, Table 2, and Table 3 reported in Section 4. Therefore, if the Searcher agent decides to participate in the computation of the credit field, it simply clones itself on all its neighbours and propagates the computing phase to search an available mobile robotic node. Then, it creates a particular tuple in the local tuple space to prevent eventual other Searcher agents cloned on the node from starting another searching phase.

Finally, in the third case, the Searcher agent terminates without cloning itself, thus preventing the node from participating in the computation of the navigational forces.

An *Adversary* agent is created by a Searcher agent on a static sensor node whenever the node can directly detect an obstacle or an adversary area. In this case, the Adversary agent classifies the node as *vc* and clones itself on all the neighbours.

When an Adversary agent is cloned on a static sensor node, it starts the phase to detect an obstacle or an adversary area. If the detection fails, the Adversary agent sets the proximity of the node to *c* and stops cloning itself. Otherwise, the agent classifies the node as *vc*, and clones itself on all the neighbours. Finally, the Adversary agent, before stopping its execution, creates a particular tuple in the local tuple space to prevent eventual other Adversary agents cloned on the node from runnnig.

A *Replier* agent is created by a Searcher agent on a mobile robotic node that can reach the region of phenomenon. To this end, it can go backward to the cluster leader node to notify the availability of the mobile robotic node by exploiting the migration programming facility supported by Agilla. In fact, such a facility enables the Replier agent to follow backward, node-by-node, the same path previously followed by the Searcher agent, until the cluster leader node. This enables the Replier agent to track the followed path and count the number of hops in the path. In this way, the Replier agent can store a tuple in the local tuple space of the cluster leader node, which represents the credit value assigned to it. At this point, it can create an Initializer agent and stop its execution.

An *Initializer* agent takes charge of building the credit field. To this end, it clones itself to its neighbours, carrying the initial value of the credit decremented by one. However, such a cloning process goes on only on the static sensor nodes that lie on the path previously followed by the Replier agent. Moreover, when the Initializer agent reaches a new static sensor node, it first checks whether a credit value has been already assigned to the node: in this case, no further actions are done. Otherwise, it sets the credit value of the static sensor node to the carried value decremented by one. Finally, when the Initializer agent reaches the mobile robotic node, it starts a Navigator agent on it.

A *Navigator* agent is a "stationary" agent that can run only on a mobile robotic node. Its main task consists in driving such a node to the region of phenomenon by exploiting the navigational forces represented by the computed credit field. Therefore, the Navigator agent checks the credit values of the current neighbours and chooses the highest one. Then, it generates all the commands needed to guide the mobile robotic node towards the static sensor node that possesses the chosen credit value. This process is repeated whenever the chosen static sensor node is reached by the mobile robotic node, which can thus follow, node-by-node, the path determined by the credit field until the region of phenomenon.

6. Experimental Section

The proposed implementation of the improved version of the credit field algorithm has been tested on a WSARN consisting of 64 static sensor nodes and 2 mobile robotic nodes.

6.1 Main Network Hardware and Software Components

Static sensor nodes are Mica2 Motes, each provided with a radio platform based on the Atmel ATmega 128L and a low-power microcontroller able to run the TinyOS operating system loaded in the internal flash memory. TinyOS is a light, energy efficient operating system, which has been purposely developed to manage self-configuring WSNs. It enables static sensor nodes to concurrently run an application and communicate through a multi-channel radio transceiver. Moreover, it supports the Agilla middleware [15].

Mobile robotic nodes are made of a Stargate board connected to a Lego Mindstorms robot (see Figure 4). In particular, the Stargate board provides the Lego Mindstorms robot with the capacity to interact both with the static sensor nodes and with a PC acting as the "base station" of the WSARN. This enables the robot to receive the movement commands generated on the Stargate board by the Agilla agents through the infrared port (see Figure 5).

The Stargate board is an embedded system that can run a Linux kernel, and consists of a 400 MHz Intel Xscale processor, an Intel SA1111 StrongARM companion chip for I/O, a 32 MB Intel StrataFlash chip, and 64 MB SDRAM. It can be connected to a specific daughter card provided with Ethernet and USB ports, and with PCMCIA/CF slots, which can be used to get

wireless communication based on the IEEE 802.11 protocol. The Stargate board is not provided with a mote-compatible RF radio. Therefore, communication between the Stargate board and the static sensor nodes based on the Mica2 Mote platform is implemented by another Mica2 Mote, which can be connected to the Stargate expansion bus through the 51 pin ending port.

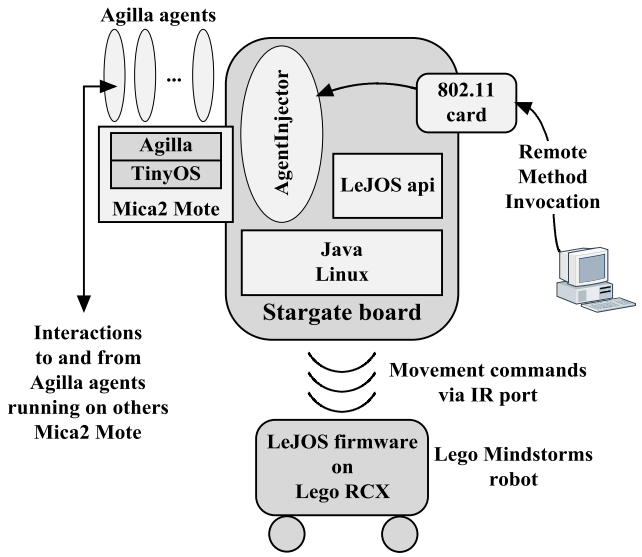

Figure 5. Interaction within a mobile robotic node

The Lego Mindstorms robot is managed by a programmable module, called RCX, whose core is a Hitachi H8/3292 microcontroller with 32 KB of external RAM. The RCX is provided with a simple firmware able to accept and execute only basic commands, received through the infrared port connected to the Stargate board, to control the motors and sensors installed on the robot. Therefore, in order to improve the capabilities of the RCX and to program it in a high level language, such as Java, the standard firmware has been replaced with the LeJOS firmware. In fact, this firmware is supplied with an extended API to fully control all the functionalities of the RCX, and supports a Java based programming on top of a "tiny" Java Virtual Machine", such as the TinyVM.

To set up the WSARN, Agilla has to dynamically load the agents described in Section 4 on the static sensor nodes. In this regard, agent loading is carried out by running a Java application, called *AgentInjector*, purposely customised for the Stargate board, that is a node with limited computing capabilities compared to a PC. This has required the installation of a "reduced" Java Runtime Environment on the Stargate board as well as the implementation of the remote access to the functionalities of the *AgentInjector* via the standard "Remote Method Invocation" software support. In this way, the mobile robotic nodes provided with Stargate board have been transformed into actual wireless "base stations" connected to a PC, which can thus act as a "remote console" able to dynamically program the WSARN.

6.2 Tests

To prove the effectiveness of the proposed navigation algorithm, two different WSARNs have been deployed. They are shown in Figure 1 and in Figure 8, respectively. The networks have been repeatedly activated to generate the navigational forces of credit field and to guide mobile robotic nodes to the region of phenomenon. The activations have been carried out always running both the original credit field algorithm and its improved version, in order to compare the different energy consumptions. In this regard, it worth noting that the WSARNs deployed in the tests are characterized by configurations that involve a different number of static sensor nodes in the navigation of mobile robotic nodes, and this entails a very different global energy consumption.

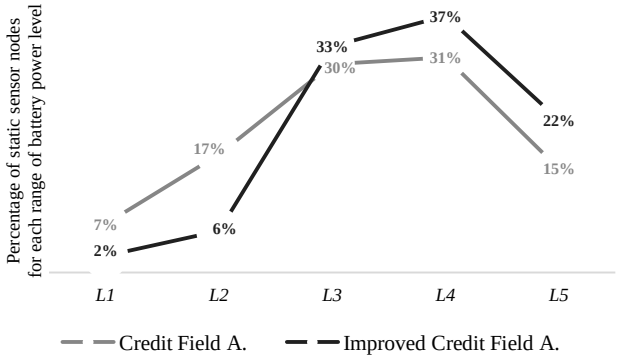

Figure 6. Energy consumption trend of the sensor network shown in Figure 1 after five executions of the navigation process

The former test has been conducted on the WSN depicted in Figure 1. In this network, most of the static sensor nodes are directly involved in the navigation process of the two mobile robotic nodes to the region of phenomenon. Such a process has been run 20 times in total, evenly divided in executions of the original and improved credit field algorithm.

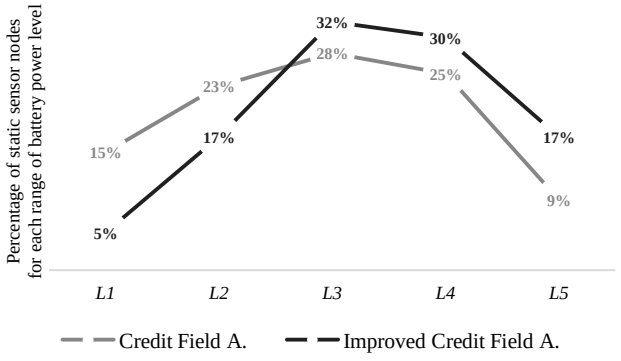

Figure 7. Energy consumption trend of the sensor network shown in Figure 1 after ten executions of the navigation process

Then, the battery power level of the static sensor nodes has been estimated and classified according to 5 ranges of values: *L1* includes nodes with a battery power level in the range from 0 to 20% of the maximum battery power level, denoted as *Lmax*; *L2* includes nodes with a battery power level in the range from 20% to 40% of *Lmax*; *L3* includes nodes with a battery power level in the range from 40% to 60% of *Lmax*; *L4* includes nodes with a battery power level in the range from 60% to 80% of *Lmax*; *L5* includes nodes with a battery power level in the range from 80% to *Lmax*. The estimates have been obtained after the fifth and

the tenth run of the navigation process of the mobile robotic nodes for each of the two algorithms, respectively. In both cases, they result in similar global energy consumptions, whose differences are within a range of 5%÷9%.

Figure 6 shows the distribution of the static sensor nodes in the network depicted in Figure 1 according to their battery power level after five executions of the navigation process of the two mobile robotic nodes. Figure shows that energy consumption is better distributed among the static sensor nodes when the improved version of the credit field algorithm is run. In fact, most of the nodes has an energy level equal to *L3*, *L4*, and *L5*, whereas only 2% of the nodes has a minimum energy level.

Such considerations are confirmed by the results shown in Figure 7, where only 5% of the nodes have a low energy level after ten executions of the improved version of the credit field algorithm. In fact, these results appear to be relevant, since they prove that the proposed algorithm reduces the energy consumption particularly for the static sensor nodes that are more loaded by the navigation algorithm.

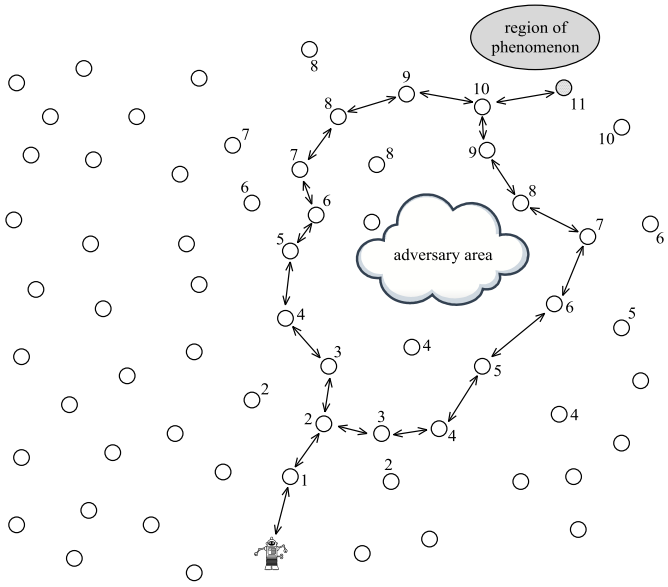

Figure 8. Credit field within a deployed sensor network

The latter test has been conducted on the WSN depicted in Figure 8. Differently from the previous test, in this network only a limited number of the static sensor nodes are directly involved in the navigation process of the mobile robotic node to the region of phenomenon, and this means that the global energy consumption is reduced with respect to the previous test, when the navigation process is run.

Also Figure 9 and Figure 10 show that the proposed improved algorithm can limit energy consumption for the static sensor nodes, thus promoting a more equal distribution of such a consumption particularly among the most loaded nodes. In fact, this is a relevant goal for WSARNs purposely deployed to guide mobile robotic nodes in hostile environments. In such networks, it is necessary to avoid getting static sensor nodes with a low battery power level, since these nodes could go out of service, thus preventing the detection of dynamic obstacles and adversary areas during the navigation of mobile robotic nodes.

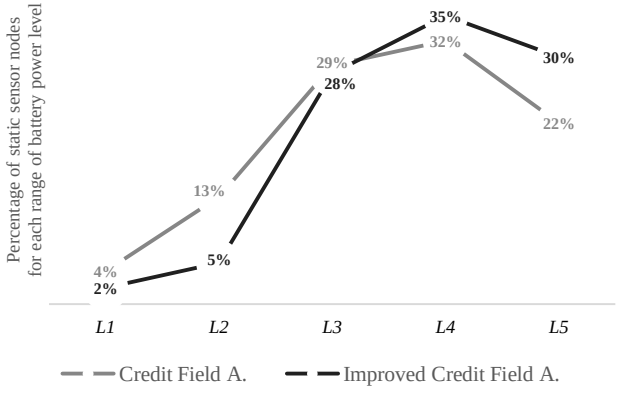

Figure 9. Energy consumption trend of the sensor network shown in Figure 8 after five executions of the navigation process

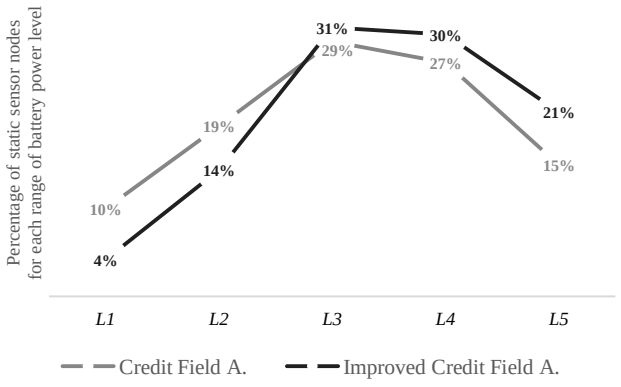

Figure 10. Energy consumption trend of the sensor network shown in Figure 8 after ten executions of the navigation process

7. Conclusions

WSARNs can be exploited to guide mobile robotic nodes to event locations. The literature presents a number of algorithms that can achieve such a goal. However, these algorithms, such as that one based on the credit field, do not adequately take into account the problem of energy consumption of the static sensor nodes that have to guide mobile robotic nodes. Such a problem appears to be relevant, since WSARNs are usually employed to guide robots within hostile environments, where human presence is impracticable. This means that, if a number of static sensor nodes fail because of a high energy consumption caused by the repeated execution of the navigation algorithm, the whole WSARN could fail to guide robots to the region of interest in the presence of dynamic obstacles and adversary areas. The proposed navigation algorithm just solves such a problem. It can reduce and balance energy consumption among the static sensor nodes that are close to obstacles and adversary areas. Thus, the algorithm can preserve just the nodes that appear to be strategic for the navigation of robots. In fact, the conducted tests reasonably prove such claims.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: A survey", Computer Networks, Vol. 38, No. 4, pp. 393-422, 2002.

[2] W. W. Dargie, C. Poellabauer, "Fundamentals of wireless sensor networks: theory and practice", John Wiley & Sons, Hoboken, NJ, USA, 2010.

[3] K. Somoppa, K. Øvsthus, L. M. Kristensen, "An industrial perspective on wireless sensor networks – a survey of requirements, protocols, and challenges", IEEE Communications Surveys & Tutorials, Vol. 16, No. 3, pp. 1391-1412, 2014.

[4] A.-S. Khan Pathan, N. Badache, S. Moussaoui, "Strengths and Weakness of Prominent Data Dissemination techniques in Wireless Sensor Networks", International Journal of Communication Networks and Information Security, Vol. 5, No. 3, pp. 158-177, 2013.

[5] T. Arampatzis, J. Lygeros, S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks", IEEE International Symposium on Intelligent Control, Limassol, Cyprus, pp. 719-724, 2005.

[6] S. Tilak, N. B. Abu-Ghazaleh, W. Heinzelman, "A taxonomy of wireless micro-sensor network models", ACM Mobile Computing and Communications Review, Vol. 6, No. 2, pp. 28-36, 2002.

[7] A. Nayak, I. Stojmenovic, "Wireless sensor and actuator networks", John Wiley & Sons, Hoboken, NJ, USA, 2010.

[8] R, Verdone, D. Dardari, G. Mazzini, A. Conti, "Wireless sensor and actuator networks: technologies, analysis and design", Academic Press, Cambridge, MA, USA, 2010.

[9] I. F. Akyildiz, I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges", Ad Hoc Networks, Vol. 2, No. 4, pp. 351-367, 2004.

[10] T. Melodia, D. Pompili, V. C. Gungor, I. F. Akyildiz, "Communication and coordinationin wireless sensor and actor networks", IEEE Transactions on Mobile Computing, Vol. 6, No. 10, pp. 1116-1129, 2007.

[11] P. Gil, I. Maza, A. Ollero, P. Marrón, "Data centric middleware for the integration of wireless sensor networks and mobile robots", 7th Conference on Mobile Robots and Competitions, Albuferia, Portugal, pp. 1-6, 2007.

[12] Q. Li, M. De Rosa, D. Rus, "Distributed algorithms for guiding navigation across a sensor network", 9th ACM International Conference on Mobile Computing and Networking, San Diego, CA, USA, pp. 313-325, 2003.

[13] M. A. Batalin, G. S. Shukhatme, M. Hattig, "Mobile robot navigation using a sensor network", IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, pp. 636-641, 2004.

[14] A. Verma, H. Sawant, J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network", Pervasive and Mobile Computing, Vol. 2, No. 1, pp. 65-84, 2006.

[15] C. L. Fok, G. C. Roman, C. Lu, "Agilla: A Mobile Agent Middleware for Self-Adaptive Wireless Sensor Networks", ACM Transactions on Autonomous and Adaptive Systems, Vol. 4, No. 3, Article 16, 2009.

[16] G. Mahadevan, S. Nirmala, N. Pradeep, "Extended Architecture for Agilla Middleware to Reduce the Energy Efficiency for WSN", 4th International Conference on Emerging Research in Computing, Information, Communication and Applications, Bangalore, India, in "Emerging Research in Computing, Information, Communication and Applications", eds. N. R. Shetty, N. H. Prasad, N. Nalini, Springer Singapore, pp. 543-553, 2005.

[17] L. Corradetti, D. Gregori, S. Marchesani, L. Pomante, M. Santic, W. Tiberti, "A renovated mobile agents middleware for WSN porting of Agilla to the TinyOS 2.x platform", IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow, Bologna, Italy, pp. 1-5, 2016.

[18] N. Deshpande, E. Grant, T. C. Henderson, "Target localization and autonomous navigation using wireless sensor networks — a pseudogradient algorithm approach", IEEE Systems Journal, Vol. 8, No. 1, pp. 93-103, 2014.

[19] T. C. Henderson, E. Grant, "Gradient calculation in sensor networks", IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, Vol. 2, pp. 1792-1795, 2004.

[20] P. Corke, R. Peterson, D. Rus, "Localization and navigation assisted by networked cooperating sensors and robots", International Journal Robotic Research, Vol. 24, No. 9, pp. 771-786, 2005.

[21] J. R. Jiang, Y. L. Lai, F. C. Deng, "Mobile robot coordination and navigation with directional antennas in positionless wireless sensor networks", International Journal of Ad Hoc Ubiquitous Computing, Vol. 7, No. 4, pp. 272-280, 2011.

[22] T. Rault, A. Bouabdallah, Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey", Computer Networks, Vol. 67, pp. 104-122, 2014.

[23] W. Osamy, A. M. Khedr, "An algorithm for enhancing coverage and network lifetime in cluster-based Wireless Sensor Networks", International Journal of Communication Networks and Information Security, Vol. 10, No. 1, pp. 1-9, 2018.

[24] D. M. Omar, A. M. Khedr, D. P. Agrawal, "Optimized Clustering Protocol for Balancing Energy in Wireless Sensor Networks", International Journal of Communication Networks and Information Security, Vol. 9, No. 3, pp. 367-375, 2017.

[25] F. Frattolillo, N. Quarantiello, S. Ullo, "Implementing Assisted Navigation in Hybrid Sensor Networks", IEEE Geoscience and Remote Sensing Symposium, Barcelona, Spain, pp. 2909-2912, 2007.

[26] D. Gelernter, "Generative communication in Linda", ACM Transactions on Programing Languages and Systems, Vol. 7, No. 1, pp. 80-112, 1985.