Journal of Computational Science 4 (2013) 199–208

Contents lists available at SciVerse ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Self-stabilizing algorithm for efficient topology control in Wireless Sensor Networks

Jalel Ben-Othman[a], Karim Bessaoud[b], Alain Bui[b,*], Laurence Pilard[b]

[a] L2TI Lab, University of Paris 13, 99 Avenue Jean-Baptiste Clément, F-93430 Villetaneuse, France
[b] PRiSM – CNRS, University of Versailles SQY, 45 Avenue des Etats-Unis, F-78035 Versailles Cedex, France

ARTICLE INFO

Article history:
Received 27 July 2011
Received in revised form 5 December 2011
Accepted 6 January 2012
Available online 1 February 2012

Keywords:
Wireless Sensor Networks
Self-stabilization
Minimum weighted connected dominating set
Routing

ABSTRACT

Wireless Sensor Networks lifetime mainly depends on energy saving efficiency. In this paper, we propose an energy-efficient self-stabilizing topology control protocol for WSN. We reduce the transmission power of each node so as to maintain network connectivity while saving maximum energy. Besides, we propose an approximation algorithm for minimum weighted connected dominating set that builds a virtual backbone formed by sensors with maximum energy. This backbone is used for efficient routing purpose. We proved the algorithm correctness and through our simulation results, we showed the efficiency of our proposed solution.

1. Introduction

A Wireless Sensor Network (WSN) consists of a large number of distributed sensors deployed in an ad hoc fashion that communicate by means of wireless technology and organize themselves in a multi-hop communication wireless network. A network of these devices collaborates for a common application such as environmental monitoring, smart spaces, medical systems or robotic exploration. Sensors are battery operated with a processor and a radio. They are characterized by limited resources in terms of memory, computation and energy. Sensors' battery is usually hard to recharge or even replace. Energy in WSN is thus a scarce and precious resource that should be saved.

Energy efficiency is an important design consideration for these networks. Energy in WSN is used for message treatment, transmission and reception. In addition, a large amount of energy is wasted due to the wireless shared-medium nature such as *idle listening*, *overhearing* and *collisions*.

The idle listening is the state where a node waits for a message. Since a node does not know when a message will arrive, it should keep its radio in receive mode whenever it is not transmitting. A huge energy is then consumed since nodes are in this mode most of the time. Moreover, this induces that each node keeps listening to all transmissions from its neighbors. As a result, each node overhears a lot of packets. This is a significant waste of energy, especially when node density is high and traffic load is heavy. Collisions are another aspect of the shared nature of wireless medium that increase energy wasting. When a collision occurs, the sender node should send its message again causing more receptions and thus more energy consumption.

In WSN literature, several solutions have been proposed to increase network lifetime. In this paper, we distinguish three main classes of solutions:

1 Energy-efficient routing [1–4]: This class of solutions aims to minimize the energy consumed by the end-to-end transmission of a message. The energy-efficient routing addresses energy minimization by reducing the amount of transmitted and received messages.
2 Sleep state [5–7]: This class is based on scheduling node activity by alternating node's states between sleeping and active state in order to minimize energy consumption while ensuring network connectivity and application functionalities. Hence, this class reduces energy consumption in the idle listening state.
3 Topology control solution [8–10]: In this class, sensors reduce their transmission range while maintaining network connectivity. The solution purpose is to minimize energy consumption due to transmission power, overhearing and collision.

* Corresponding author.
E-mail addresses: Jalel.Ben-Othman@univ-paris13.fr (J. Ben-Othman), Karim.Bessaoud@prism.uvsq.fr (K. Bessaoud), Alain.Bui@prism.uvsq.fr (A. Bui), Laurence.Pilard@prism.uvsq.fr (L. Pilard).

1877-7503/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.jocs.2012.01.003

In this paper, we propose a new solution based on the third class of solutions: the control of the network topology through the reduction of the transmission power. The transmission power is a critical parameter when addressing energy efficiency in WSN, since the energy consumed to send a message grows at least quadratically with the transmission range. Usually, all nodes in WSN use their maximum transmission power to send their messages even if the receiver is close to the sender. Therefore, some energy is wasted to send the message and also to receive it since all nodes in this huge range of transmission will receive the message. In this paper, we propose a distributed algorithm that reduces the transmission power of each node independently. The goal is to reduce as much as possible the transmission power while keeping the network connected. This leads to an energy-efficient topology and at the same time we obtain a topology that facilitates routing. Finally, our solution is self-stabilizing [11,12]. This is a very desirable property in WSN. Indeed, a self stabilizing algorithm is guaranteed to reach a correct behavior in a finite number of steps, regardless of its initial state. Then after any unexpected perturbation, the algorithm recovers itself without any outside intervention. This allows our solution to lead with classical events in WSN that are: failure, restart or addition of nodes in the network.

The paper is organized as follows. After a description of related works in Section 2, we describe in Section 3 our solution through the design guidelines and the system model. The algorithm and the correctness proofs are presented in Section 4. Section 5 is dedicated to the experimental study. In Section 6, we give a discussion and finally, we conclude this paper and give some perspectives.

2. Related work

The issue of saving energy in WSN is particularly addressed by the research community. In this section, we focus on the approaches based on adjusting the transmission power.

To control the transmission range, two strategies are possible: reduce the transmission range of all the sensors to a common range or give different transmission range for each sensor. The use of a common range for all the sensors allows the network to be bi-directional. However, it is necessary to find the suitable range that allows keeping one connected component.

In [13], the authors show that the average range when using a variable transmission range is approximately half than when using the common transmission range. In addition, the results are obtained using uniform distribution of nodes, but this is not often the case in real networks. Finally [13] shows that in variable adjustment of the transmission range, the network capacity does not depend on the nodes number in the network, so the increase of the number of nodes does not affect the network capacity, unlike the case of the common transmission range.

In [8], an adjustment of the power transmission of nodes in an ad hoc network is performed to optimize the energy of the nodes while ensuring the connectivity or the bi-connectivity of the network. The authors proposed two algorithms CONNECT and BICONN-AUGMENT conceived for static networks. For mobile ad hoc networks, the authors designed two distributed heuristics LINT and LILT.

CONNECT and BICONN-AUGMENT are two centralized algorithms that minimize the maximal power used by the nodes of the network while maintaining the connectivity (resp. the bi-connectivity) of the network. Both algorithms merge different connected components to form a single connected (resp. bi-connected) component.

LINT and LILT heuristics are proposed to reduce the powers transmission of nodes while ensuring the connectivity (in LINT) and the bi-connectivity (in LILT) of the network. In LINT, each node uses three parameters related to the number of neighbors: the desired degree, the high threshold degree and the low threshold degree. Periodically, each node checks its degree and if it is not between the high and the low threshold, the node changes its energy level to reach the desired degree. LILT improves LINT by using the routing algorithm Link State Protocol to set the transmission power of some particular nodes to the maximum power to ensure the bi-connectivity of the network. The main inconvenience of LINT and LILT is the use of Link State protocol that needs a network global view.

In [9], the authors proposed a protocol (COMPOW) for power transmission control. In COMPOW, the network is represented by a non-directed graph where two nodes are neighbors if they are within the range of each other. In COMPOW, nodes can communicate using different transmission powers. Each node runs routing daemons, one for each power. Each routing daemon maintains its own routing table by exchanging control messages. By comparing the entries in different routing tables, each node can determine the smallest common power that guarantee the maximum number of connected nodes. More specifically, let $N(P_i)$ be the number of entries in the routing table corresponding to the power level P_i. Then the sufficient power level is set to the lowest power level P_i for which $N(P_i) = N(P_{max})$, where P_{max} is the maximum transmission power. COMPOW increases the traffic capacity of the network and the battery lifetime. However, COMPOW generates an important overhead due to the control messages generated by each daemon.

In [10], the authors propose a distributed algorithm called LMST for topology control by adjusting the transmission range based on the construction of spanning trees. Each node runs an algorithm to build a spanning tree and adjusts its transmission power to get only the one hop nodes in the tree. The authors proved several properties about LMST. First, LMST preserves the connectivity of the network. They also proved that the maximum number of neighbors for each node was bounded by 6 and that the topology could easily be transformed to leave only the bi-directional links. The major drawback of LMST is that it requires the construction of a spanning tree for each node, which induces a large overhead.

In [14], the authors propose a cone-based distributed topology control algorithm using direction information named $CBTC(\alpha)$. In this approach, each node increases its transmission power until it finds nodes that it can reach in every cone of angle α. The authors showed analytically that the connectivity of the network is preserved using $\alpha \leq 5/6$. In $CBTC(\alpha)$, there are three optimizations of the basic algorithm (1) the shrink-back operation, (2) asymmetric edge removal, and (3) pairwise edge removal, and proofs that improve performance while preserving connectivity. The algorithm can be extended easily to deal with dynamic reconfiguration in the presence of failures and mobility and asynchrony.

In $CBTC(\alpha)$, the authors do not investigate how to choose the initial power, nor investigate how to increase the power at each step. In addition, sensors have to compute at each time the transmission power required for Ack control messages.

3. Preliminaries

In this section, we define the model we use and our contribution. We also give the connected dominating set definition, with some related works.

3.1. System model

Let a wireless ad hoc network composed by a set of n wireless sensors nodes. Each sensor node uses an omni-directional coverage, so that a single transmission of a sensor is received by all the sensors that are in a disk centered on the sensor. This radius disk

is called the *transmission range*. We assume that all sensors have a maximum transmission range equal to R_{max}. Two sensors being at the transmission range of each other may communicate directly and are considered as neighbors, whereas two distant sensors can communicate through multi-hop wireless links using intermediary sensors to forward the message. One node in the network is the *sink*; this node is a powerful computer that can be considered as a leader. The *sink* follows the same communication rules as the other sensor nodes.

The wireless sensors and the *sink* form a graph $G=(V, E)$, where V is the set of nodes representing sensors and E is the set of edges representing the communication relation between the nodes. There is an edge between two nodes if and only if their Euclidean distance is at most R_{max}. The neighbors of a node u is noted $N(u)$. In the following, we assume that $G=(V, E)$ is a distinguished connected graph. We also assume that all wireless sensor nodes have distinct identifiers and can compute their energy consumed.

A node maintains a set of variables that make up the local state of the node. We use a message passing communication model, i.e. each node can send messages to all its neighbors and can receive messages from all of its neighbors. Each node executes a set of guarded rules. Each rule has the form <*Guard*> → <*Action*>, where <*Guard*> is a boolean condition over the node' local state or a message reception, and <*Action*> is a set of statements assigning new values to the variables of the node. The set of the local states of all nodes in the network union the set of all in transit messages is called a *configuration*.

A node is *enabled* if one of its guard is true. A daemon is a predicate over the execution. Our algorithm works under the *fair asynchronous daemon*: (i) to move from one configuration to the next one, the daemon selects a non-empty set of enabled nodes to execute one guarded rule and (ii) any node that is continuously enabled will eventually be chosen by the daemon.

A self-stabilizing system has the ability to recover in a finite time to a correct behavior after any transient failure, without external intervention.

Definition 1 *(Self-stabilization).* Let Γ be a set of all configurations. A system S is self-stabilizing with respect to Λ such that $\Lambda \subseteq \Gamma$ if and only if it satisfies the following two conditions: (Convergence) every execution of S contains at least a configuration in Λ, and (Closure) for any configuration $\lambda \in \Lambda$, any configuration γ that follows λ is also in Λ.

We assume that each sensor has the ability to compute the Euclidean distance between itself and any neighbor using for instance, the power with which it received the message from its neighbor.

3.2. Contribution

In this paper, we propose a self-stabilizing algorithm that builds a sub-graph of G, denoted $G^- = (V, E^-)$ with all nodes of G and $E^- \subseteq E$ obtained by reducing the transmission range of each sensor. The constructed graph G^- is bi-directional, connected and each node uses in G^- a transmission range smaller than or equal to the one used in G.

Since we reduce the transmission range of each node independently, we obtain a directed network. In such a network, the use of an acknowledgment mechanism is difficult to implement. Thus the only links we consider in G^- are the bi-directional links. G^- is then the graph obtained after the power reduction and after removing all uni-directional links.

The sub-graph G^- is obtained by building a connected dominating set. This set can then be used in order to route messages efficiently reducing the number of hops in the network.

3.3. Connected dominating set

In the following, we consider a graph $G=(V, E)$. $N(v)$ the set of v's neighbors.

Definition 2. A subset M of V is a dominating set (DS) iff $\forall u \in V$: $(u \in M \vee \exists v \in M, u \in N(v))$.

Definition 3. A subset M of V is a connected dominating set (CDS) iff M is a DS and M is a connected set.

Definition 4. Assume that each node u has a weight $w(u)$. Then a subset M of V is a minimum weighted dominating set (MWDS) if M is a DS with minimum total weight.

Definition 5. A subset M of V is a minimum weighted connected dominating set (MWCDS) if M is a CDS with minimum weight.

In the remainder of the paper, we use the terms backbone and MWCDS interchangeably.

Most of works that address the construction of a CDS [15–18] try to reduce the size of the set (Minimum CDS). For our problem, we are more interested in reducing the consumed energy than the size of the CDS. Indeed the sensors in the CDS are responsible for routing, they are thus more affected to energy loss. Therefore, the sensors that consumed less energy are the one that should be chosen to form the CDS. This leads to a MWCDS construction.

In [19] the authors proves that the problem of MCDS, and therefore MWCDS, is NP-hard.

The problem of MWCDS construction has been studied in [20,21] and distributed algorithms have been proposed with constant approximation. However, these solutions are not self-stabilizing, and then are not suitable for dynamic networks.

In this paper, we propose a self-stabilizing algorithm to build a MWCDS approximation. Our algorithm of MWCDS is close to the algorithm in [17] with some differences since we used a stronger daemon, the fair asynchronous one and we consider the weight of the nodes. The energy consumed by the sensors is used as weights in the construction of MWCDS.

4. Algorithm description

In this paper, we propose a distributed self-stabilizing algorithm that reduces the transmission power of sensors while guaranteeing a topology to facilitate the routing. For that purpose, we first build a spanning tree such that the path in the tree between any node u and the root r is the path in the graph from u to r with the minimum weight. Each node stores the weight of its path to the root. Second, we use this weight to build a MWCDS that represents our backbone. Finally, each sensor node reduces its transmission range depending on whether or not it belongs to the backbone. A sensor that is not in the backbone, called a *dominated* node, selects its closest (in Euclidean distance) neighbor in the backbone to be its *dominator*. Then, it reduces its transmission range to the Euclidean distance between its dominator and itself. A sensor in the backbone reduces its transmission range in order to reach all of its neighbors in the backbone and all of its neighbors that it dominates.

Then, we obtain that all communication links between a dominated node and its dominator are bidirectional and all communication links in the backbone are also bi-directional. Fig. 1 illustrates the main steps of our solution.

The algorithm is self-stabilizing and depends on the energy consumed by the sensors. Since this energy varies over the time, there is a risk of ping-pong effect: the backbone can change all the time and therefore, the algorithm will never stabilize. To avoid that, instead of using the current energy consumed, we use a variable storing the energy consumed to build the backbone. Then we periodically

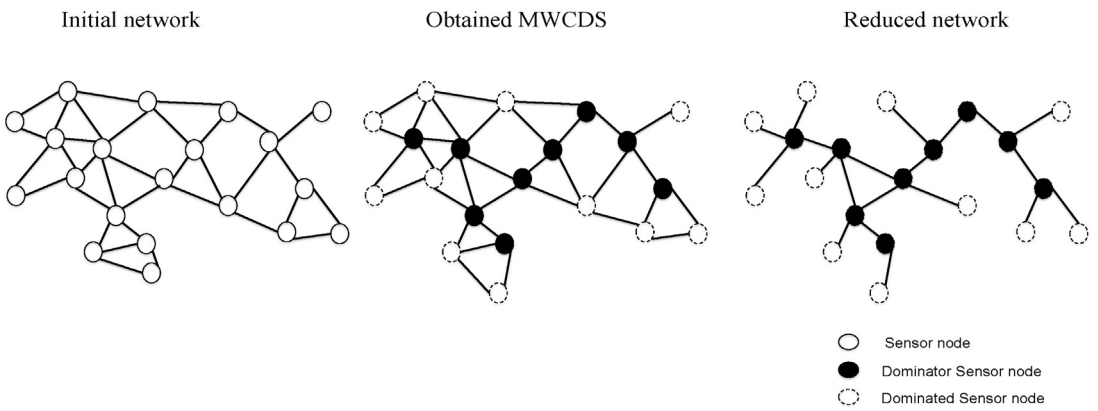


Fig. 1. Transmission range reduction process.

update this variable, using a period greater than the stabilizing time of our algorithm.

Our algorithm is divided into six main parts, each part is presented as a self-stabilizing algorithm. In each algorithm, the sensors can read the variables of previous algorithms, and read and write variables of the current one. This guarantees that if each part is self-stabilizing then the whole algorithm is self-stabilizing [22]. When many guards are simultaneously true, we select the guard the highest priority. The priority depends on the algorithms order (i.e. part 1 is the part with the highest priority, then part 2 and so on).

The six parts are as follows:

1 Updating the energies consumed by the sensors: that will be made thanks to a broadcast initiated by the leader.
2 Updating the state: each node sends all its local state to all of its neighbors.
3 Construction of a spanning tree based on the sensor energies: contrary to the classical method employed in [17], we build a spanning tree such that the path in the tree between any node u and the root r is the path with the minimum weight in the graph.
4 Construction of a MWDS: this algorithm selects sensors with the minimum weight locally.
5 Construction of a MWCDS: this algorithm selects the sensors which serve as connectors between those in MWDS.
6 Reduction of the transmission range: this algorithm aims to reduce the transmission range of sensors while ensuring the network connectivity.

4.1. Algorithm

4.1.1. Part 1: The energy consumed update

4.1.1.1. Variables.

- i: node identifier.
- $power_i(j), j \in N(i) \cup \{i\}$: The energy consumed by j.
- *timer*: The time before the next reconstruction of the MWCDS. All TIME_INTER_CDS the node will update the power which will lead to a reconstruction of MWCDS. This variable is only used by the leader.
- $last\_update_i$: contains the identifier of the last update message received by i.

4.1.1.2. Messages.

- update(*msgid*): This message is sent by i to indicate that it is necessary to update the consumed energy.

4.1.1.3. Functions.

- update_power(): This function allows to get the energy consumed by the node.

4.1.1.4. Algorithm. The *update* messages are initiated by the leader and are broadcasted to all the network. These messages are sent every TIME_INTER_CDS to update energy consumed by sensors, this will trigger a rebuilt of the tree, the MWDS and the MWCDS.

The rule GC1 is only executed by the leader. The rule GC2 is executed by all nodes but the leader.

GC1: $timer = 0 \rightarrow$
  $last\_update_i++$
  broadcast(update($last\_update_i$))
  $timer$:= TIME_INTER_CDS

GC2: In the reception of the update(msgid) message $\rightarrow$
  if($last\_update_i \neq msgid$) then {
    $power_i(i)$:= update_power()
    $last\_update_i := msgid$
    broadcast(update(msgid))
  }

4.1.2. Part 2: State updating

4.1.2.1. Messages.

- hello(sender, state): This message is periodically sent by the sender to broadcast its state.

4.1.2.2. Algorithm. *hello* messages are periodically sent by all nodes to maintain the coherence of neighbors state. These messages are sent with range R_{MAX} and contains all the local variables of the nodes.

GC3: In the reception of a message: hello(sender, state) $\rightarrow$
  $state_i$(sender):= state

4.1.3. Part 3: Tree construction

4.1.3.1. Variables.

- $f_i(j), j \in N(i) \cup \{i\}$: The identifier of the father of j in the spanning tree.
- $weight_i(j), j \in N(i) \cup \{i\}$: The weight of the path from the root to j in the spanning tree. The weight of the path is the sum of the power of each node in the path.

4.1.3.2. Algorithm. The rule GC5 is only executed by the leader. The rule GC4 is executed by all nodes but the leader.

GC4: $weight_i(i) \neq min\{weight_i(j) + power_i(i) | j \in N(i)\}$
or $f_i(i) \neq k | weight_i(k) = min\{weight_i(j) | j \in N(i)\} \rightarrow$
$weight_i(i) := min\{weight_i(j) + power_i(i) | j \in N(i)\}$
$f_i(i) := k | weight_i(k) = min\{weight_i(j) | j \in N(i)\}$

GC5: $weight_i(i) \neq 0 \rightarrow$
$weight_i(i) := 0$

4.1.4. Part 4: MWDS construction

4.1.4.1. Variables.

- $inMWDS_i(j), j \in N(i) \cup \{i\}$: is true iff $j \in MWDS$.

4.1.4.2. Algorithm. The rule GC8 is only executed by the leader. The rules GC6 and GC7 are executed by all nodes but the leader.

This rule allows a node to leave the MWDS.

GC6: $inMWDS_i(i) = true$ and
$\exists j \in N(i), [(weight_i(j) < weight_i(i)$ and $inMWDS_i(j) = true)$ or
$(weight_i(j) = weight_i(i)$ and $inMWDS_i(j) = true$ and $j < i)] \rightarrow$
$inMWDS_i(i) := false$

This rule allows a node to join the MWDS.

GC7: $inMWDS_i(i) = false$ and
$\forall j \in N(i), [(weight_i(j) > weight_i(i))$ or $(weight_i(j) = weight_i(i)$ and $i < j)$ or
$(inMWDS_i(j) = false)] \rightarrow$
$inMWDS_i(i) := true$

GC8: $(inMWDS_i(i) = false) \rightarrow$
$inMWDS_i(i) := true$

4.1.5. Part 5: MWCDS construction

4.1.5.1. Variables.

- $inMWCDS_i(j), j \in N(i) \cup \{i\}$: is true iff $j \in MWCDS$.

4.1.5.2. Algorithm. This rule allows a node to join the MWCDS

GC9: $(inMWCDS_i(i) = false$ and $inMWDS_i(i) = true)$ or
$\exists j \in N(i), (f_i(j) = i$ and $inMWDS_i(j) = true) \rightarrow$
$inMWCDS_i(i) := true$

This rule allows a node to leave the MWCDS.

GC10: $inMWCDS_i(i) = true$ and $MWDS_i(i) = false$ and
$\forall j \in N(i), (f_i(j) \neq i$ or $inMWDS_i(j) = false) \rightarrow$
$inMWCDS_i(i) := false$

4.1.6. Part 6: Transmission range reduction

4.1.6.1. Variables.

- $range_i$: The transmission range of node i.
- $dom_i(j), j \in N(i)$: Contains the identifier of the closest dominant node if the node is dominated.
- $dist_i(j), j \in N(i)$: The Euclidean distance between i and j.

4.1.6.2. Algorithm. This rule is executed by sensors that are not in the MWCDS.

GC11: $inMWCDS_i(i) = false$ and
$[dom_i(i) \neq k | k \in N(i)$ and $inMWCDS_i(k) = true$ and
$dist_i(k) = min\{dist_i(j) | j \in N(i)$ and $inMWCDS_i(j) = true\}$ or
$range_i \neq dist_i(dom_i(i))] \rightarrow$
$dom_i(i) := k | (k \in N(i)$ and $inMWCDS_i(k) = true$ and
$dist_i(k) = min\{dist_i(j) | j \in N(i)$ and $inMWCDS_i(j) = true\})$
$range_i := dist_i(dom_i(i))$

This rule is executed by sensors that are in the MWCDS.

GC12: $(inMWCDS_i(i) = true$ and
$range_i \neq max\{dist_i(j) | j \in N(i)$ and
$(inMWCDS_i(j) = true$ or $dom_i(j) = i)\} \rightarrow$
$range_i := max\{dist_i(j) | j \in N(i)$ and
$(inMWCDS_i(j) = true$ or $dom_i(j) = i)\}$

4.2. Proof

Theorem 1. *The algorithm of part 3 is self-stabilizing for the construction of a spanning tree with minimum weighted path from the root to any node.*

Proof. The spanning tree algorithm (part 3) is based on the algorithm presented in [22]. In our algorithm, the weight is used instead of the hop-distance. According to [22], this algorithm is self-stabilizing and builds a spanning tree with minimum weighted path from the root to any node. □

Theorem 2. *The algorithm of part 4 is self-stabilizing for the construction of a DS and eventually no guard of this algorithm will be evaluated to true anymore.*

Proof. We are going to build two sets of nodes: X and Y. X is the set of nodes u such that $inMWDS_u(u) = true$ and no guard of u' algorithm will be evaluated to true anymore. Y is the set of nodes u such that $inMWDS_u(u) = false$ and no guard of u' algorithm will be evaluated to true anymore. We start with $X = Y = \emptyset$.

1) Eventually, the leader joins DS (*i.e.* it sets its variable $inMWDS_{leader}(leader)$ to true) executing *GC*8 and all its guards will then remain false forever. Thus, we add the leader in X.
2) We then consider the neighbors of the leader. Let u be such a neighbor.
   - If $inMWDS_u(u) = false$ then u cannot execute any rule. In particular, the *GC*7 guard is false because the leader that is a neighbor of u has a weight smaller than the weight of u. Thus all guards are false and will remain so.
   - If $inMWDS_u(u) = true$ then u can execute *GC*6: the leader is a neighbor of u and has a weight smaller than the weight of u. Thus the guard of *GC*6 is true and will remain so until u executes this rule. Thus u will eventually leave DS. According to the previous point, from here all guards of u are false and will remain so.

   Thus, when all neighbors leader have left DS, we add them in Y (this will necessarily happen).
3) Let $u \in V \setminus \{X, Y\}$ be the node with the minimum $(weight, id)$ value among all nodes in $V \setminus \{X, Y\}$.

   Note that the only u' neighbors than can have a $(weight, id)$ value smaller than u's $(weight, id)$ value are nodes in $X \cup Y$. Until now, all nodes in X have all their neighbors in Y. Then the only u' neighbors than can have a $(weight, id)$ value smaller than u's $(weight, id)$ value are nodes in Y. However, nodes in Y will never belong to DS.
   - If $inMWDS_u(u) = true$ then u cannot execute any rule and will never can.
   - If $inMWDS_u(u) = false$ then u can execute *GC*7. Thus the guard of *GC*7 is true and will remain so until u executes this rule. Thus u will eventually join DS. According to the previous point, from here u will never can execute any rule.

   Thus, when u is in DS, we add u to X.

   Once u is in X, we consider the set V_u equals to all the neighbors of u that are not in $\{X \cup Y\}$. u belongs to DS and has the smallest $(weight, id)$ value among all nodes in V_u. So applying a similar justification than the one used with the neighbor of the leader (point 2), we can conclude that eventually we will add in Y all nodes in V_u.
4) We apply point 3 once again on the node with the new minimum $(weight, id)$ value in $V \setminus \{X, Y\}$.

Eventually, all node will belong to $X \cup Y$. Thus, the set of nodes in DS will eventually be a dominated set. Furthermore, eventually no node will be able to execute any rule. □

Lemma 1. *Let X be the set built by the algorithm in part 5. $\forall u \in X$, $u \neq leader$ we have $\exists v \in N_u : v \in X \wedge (weight(v), id(v)) < ((weight(u), id(u))$*

Proof. Let T be the tree built in the algorithm part 2 and let M be the MWDS build in the algorithm part 4. Let S be the set of the fathers in T of all nodes in M. Note that $X = M \cup S$. Indeed, the first line

of *GC*9 selects all nodes in *M* and the second line of *GC*9 selects all nodes in *S*. All nodes selected in *GC*9 guard are added to *X*. Finally, *GC*10 deletes from *X* all nodes that do not belong to $M \cup S$.

Let $u \neq leader$. Let us prove that $\exists v \in N_u : v \in X \wedge (weight(v), id(v)) < ((weight(u), id(u))$

- If $u \in M$: according to *GC*9, u' father in *T* is in *M*. The tree *T* is a minimal weighted tree, thus $(weight(v), id(v)) < ((weight(u), id(u))$.
- If $u \notin M$: The algorithm part 4 is stabilized, then according to Theorem 2 there is no true guard in *u* that belongs to this algorithm. In particular, *GC*7 guard is false. Then $\exists i \in N(j) : (weight(j), id(j)) < ((weight(u), id(u)) and in MWDS_u(j) = true$. Thus $j \in M$ and so $j \in X \wedge (weight(j), id(j)) < ((weight(u), id(u))$.

□

Theorem 3. *The algorithm in part 5 is self-stabilizing for the construction of a CDS.*

Proof. Let *X* be the set built by the algorithm in part 5. Let *M* be the MWDS built in the algorithm part 4. *X* contains all nodes in *M*, thus *X* is a dominated set. Furthermore according to Lemma 1, $\forall u \in X$: there exists a path from *u* to the leader such that the path only contains nodes in *X*. Thus the set *X* is connected. Thus *X* is a connected dominated set. □

Theorem 4. *Let M the MWCDS built in part 5.* $\forall i \notin M$, *the variable* $dom_i(i)$ *in part 6 eventually contains a node in M.*

Proof. Let *i* be a node not in *M*. If $dom_i(i) \notin M$, then GC11 is true, and *i* will select the nearest, in Euclidean distance, sensor in *M* to be its dominator. So, all sensors that are not in *M* have a dominator in *M*. □

Theorem 5. *The part 6 eventually builds a connected sub-graph of the network.*

Proof. Let *M* the MWCDS built in part 5. For the sub-graph to be connected, we need, $\forall (u, v) \in E$:

(1) if $u \in M \wedge v \in M$
(2) or if $u \notin M \wedge v \in M \wedge dom_u(u) = v$

then $range_u \geq dist(u, v) \wedge range_v \geq dist(u, v)$.

Case (1) if *u* and *v* belong to *M* and $range_u < dist(u, v)$, *u* will execute GC12 and the range of *u* will be $\geq dist(u, v)$.
Case (2) If $range_u < dist(u, v)$, then *u* will execute GC11 and $range_u$ will be $dist(u, v)$. If $range_v < dist(v, u)$, *v* will execute GC12 and $range_v$ will be $\geq dist(v, u)$.

From (1) and (2), the network remains connected after reducing the transmission range. □

5. Performance evaluation

In this section, we present extensive simulations to evaluate our proposed distributed self-stabilizing algorithm for energy saving in WSN. This section is organized as follows. First, we describe the simulation environment and the parameters setting. Then, we investigate the average transmission range after applying our algorithm. Afterwards, we compare the average energy for end-to-end communication using our reduced backbone in the case of maximum transmission range and the case of optimal bounded energy. The following study focuses on the efficiency of our approximation algorithm of MWCDS. Finally, we analyze the optimal traffic overhead to maximize the network lifetime.

Table 1
Simulation parameters setting.

Simulation parameter	Value
Area	200 m × 200 m
Network size	200–5000
Maximum transmission range	25 m
Maximum battery energy	3 J
E_{amp}	100 pJ/(bit m^2)
E_{amp}	50 nJ/bit

5.1. Simulation configuration

To evaluate our proposal, we used Peersim simulation platform [23].

For each simulation configuration, 100 simulations are performed and confidence intervals are computed at 95%. Sensor nodes are randomly scattered in 200 m×200 m square. Each node has a maximum transmission range of 25 m and maximum battery energy of 3 J. The energy consumption model [24] is given by Eqs. (1) and (2).

$$E_t(k, d) = E_{elec} \cdot k + E_{amp} \cdot k \cdot d^2 \quad (1)$$

$$E_r(k) = E_{elec} \cdot k \quad (2)$$

E_t and E_r are respectively the energy used for transmission and receiving *k* bits with Euclidean distance *d*. In these formulas, the radio dissipates E_{elec} = 50 nJ/bit to run the transmitter or receiver circuitry, and the radio dissipates E_{amp} = 100 pJ/(bit m^2) to run transmit amplifier. The simulation parameters are summarized in Table 1.

5.2. The transmission range

Here, we intend to analyze the transmission range of the sensors when applying our algorithm for different sizes of the network varying between 200 and 5000 sensors. The average transmission ranges are catching after stabilization of the algorithm.

Fig. 2 shows that the average transmission ranges decreases with the increase of the network size, particularly between 200 and 500. When the network size is at 5000 sensors, the transmission range is 2.8 times lower than the maximum range of 25 m.

When the number of nodes increases, the network density increases too and then the average distance between nodes decreases. Thus, our algorithm takes care of the distance between nodes.

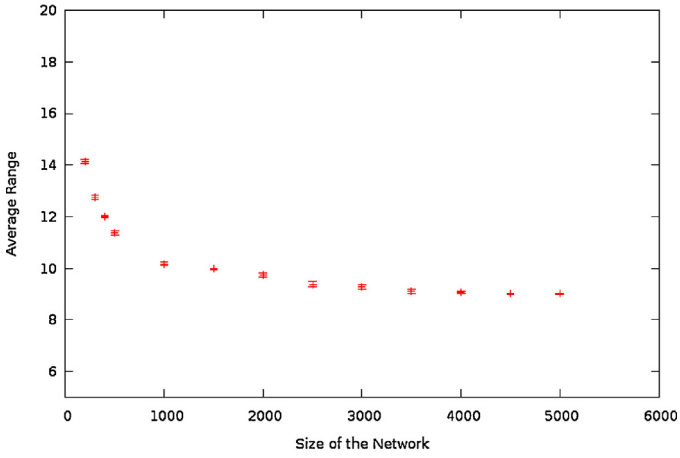


Fig. 2. The average range transmission.

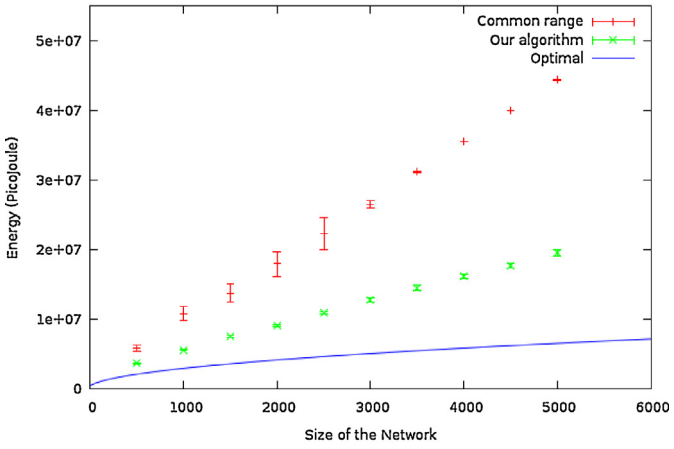


Fig. 3. The average energy consumed in the network.

5.3. The energy consumption

Reducing energy consumption is our main goal in this work. To show the efficiency of our algorithm, we compute the average energy consumed in the network to route end-to-end messages. More precisely, for any couple (u, v) of nodes, we compute the energy consumed for the following end-to-end communication: u sends a 1-bit message to v. The final result is the average energy consumed of all these communications. We use this scenario in the 3 following cases:

(case 1 – our algorithm) In the simulation, the communication from the scenario are routed using the backbone built by our algorithm, and nodes use the transmission range computed by our algorithm.

(case 2 – common range) In the simulation, the communication from the scenario are routed using the shortest path in hop-distance between u and v, and each node uses its initial transmission range (25 m).

(case 3 – optimal) We compute here the scenario in the case of optimal transmission range. We use the optimal path between u and v to route the message, i.e. we consider an optimal placement of sensors between u and v. Each routing node uses the minimal range in order to send the message to the next node in the path.

Fig. 3 shows the energy consumed by the network to send one bit depending on the number of sensors in the network for the three scenarios.

The energy consumption for end-to-end messages is based on energy consumption model presented in Section 5.1 and data collected from the simulations.

In the following, we give the computation of the average energy consumed in the optimal case:

The average energy consumed in the network for end-to-end communication using the shortest path ($E_{end\text{-}to\text{-}end}$) is the sum of the energies consumed for the successive transmissions(E_{ST}) and the energies consumed for successive receptions(E_{SR}) as shown by Eq. (3). Eqs. (4) and (5) represent respectively the calculation formulas of (E_{ST}) and (E_{SR}).

$$E_{end\text{-}to\text{-}end} = E_{ST} + E_{SR} \tag{3}$$

$$E_{ST} = HD * E_t \tag{4}$$

$$E_{SR} = HD * D * E_r \tag{5}$$

where HD is the average hop-distance between the sender and the receiver and D is the average degree. It is worth noticing that in Eq. (4), the Euclidean distance used in E_t is the transmission range squared average.

To compute a lower bound of the energy consumption using the optimal transmission range, we consider that the sensors are placed in the optimal placement between the sender and the receiver.

The surface covered by a sender (scs) is given by Eq. (6) and the number of sensors per square meter (nss) by Eq. (7).

$$scs = \pi * \left(\frac{d'}{m}\right)^2 \tag{6}$$

$$nss = \frac{n}{(200)^2} \tag{7}$$

where m is the number of sensors placed at regular Euclidean distance between the sender and the receiver, including the sender and d' is the Euclidean distance between the sender and the receiver.

The energy consumed by the network for a communication between two sensors at Euclidean distance d' is defined by the function f as represented by Eq. (8).

$$f(m) = m\left(E_{elec} + E_{amp} * \left(\frac{d'}{m}\right)^2 + E_{elec} * nss * scs\right) \tag{8}$$

In order to find a lower bound of f, we have to find $m1$ such as $f'(m1) = 0$. $f'(m)$ is given by Eq. (9) and $m1$ by Eq. (10).

$$f'(m) = E_{elec}\left(1 - \pi \cdot \left(\frac{d'}{(200 \cdot m)}\right)^2 \cdot n\right) - E_{amp} \cdot \left(\frac{d'}{m}\right)^2 \tag{9}$$

$$m1 = d' \cdot \sqrt{\left(\frac{E_{amp}}{E_{elec}}\right) + \pi \cdot \frac{n}{(200)^2}} \tag{10}$$

To compute the average energy consumed by the network between two sensors, we need the average Euclidean distance between two sensors in the network. The sensors are uniformly distributed in a square of 200 m×200 m. Hence, we define d' by Eq. (11) that represents the average distance between two points in this square [25].

$$d' = 200 \cdot \left((2 + \sqrt{2})15 + \ln\frac{(1 + \sqrt{2})}{3}\right) \tag{11}$$

To plot the optimal energy represented by Eq. (8), we replace m by the value of $m1$ (see Eq. (10)) and d' by Eq. (11).

We can observe, from Fig. 3, that our solution is better than routing with common ranges (maximum ranges). In addition, our solution is not too far from the optimal case. The ratio between our solution and the optimal solution varies between 1.71 and 2.98, while the ratio between the optimal solution and routing with common ranges varies between 2.79 and 6.76 where sensors vary between 500 and 5000.

5.4. The battery level of the dominators

In our solution, sensors of the backbone consume more energy than the others because of their routing role. In addition, backbone sensors have higher transmission range than the others because of the specific algorithm for reducing transmission range.

To study the energy of this sensors, we randomly affect a battery level between 1% and 100% to all sensors in the network. After the stabilization of our algorithm, we caught the energy of the sensors in the backbone. This study allows to evaluate the efficiency of our MWCDS algorithm.

It is worth noticing that the average battery level of the sensors in the network is 50.5%.

Fig. 4 shows the percentage of energy in the backbone sensors according to the number of sensors in the network varying between 500 and 5000.

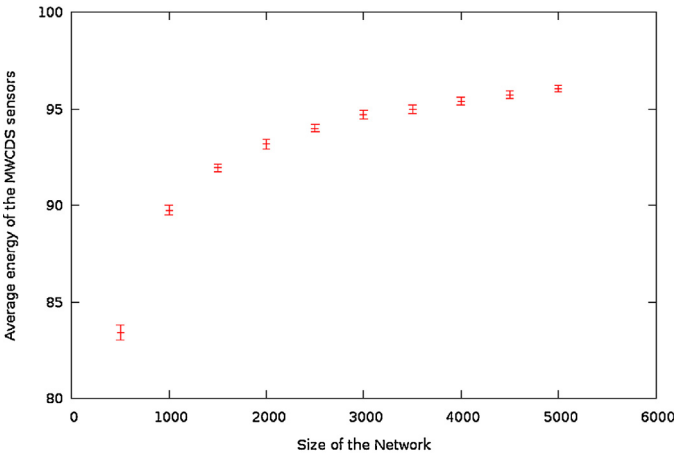


Fig. 4. The average energy of the dominators.

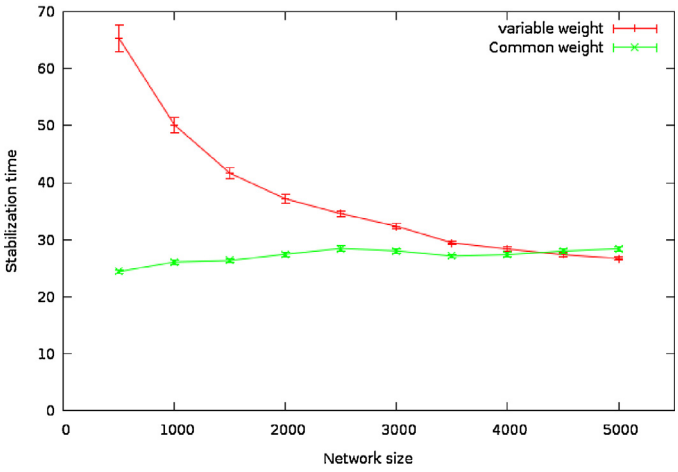


Fig. 5. The stabilization time variation.

We see that the average energy is much more than 50.5%, that is the average level in the whole network. Thus our algorithm selects nodes with a lot of energy to belong to the backbone. In addition, we can observe that the more we increase the size of the network, the larger the energy is.

We can conclude that with our solution, we reduce the energy consumption induced by communications. Moreover, only sensors with the highest battery level are in charge of the messages routing.

5.5. Network lifetime

In this simulation, we aim to study the network lifetime taking into account the regular reconstruction of the backbone. Indeed, in our algorithm, we periodically update the power leading to a rebuilding of the backbone and a readjustment of the transmission ranges. Note that nodes in the backbone are chosen according to their energy. Since these nodes will use their energy faster than the other nodes, we have to regularly build a new backbone. To do that, first, the energy variable is up-to-date, second, the algorithm stabilizes and finally the system works properly. It is then crucial to find a good value for the time period between two reconstructions. If it is too short, the system will never works properly, if it is too big, the backbone will never be up-to-date with the new energy value in the network.

This is the scenario of our simulation: there are 500 nodes in the network, we choose randomly a sender and a receiver and we perform an end-to-end communication between these two nodes. Then we do it again until the network dies. We call these messages the "application messages". In parallel of the application messages, our algorithm in running and regularly rebuilds the backbone. We compute the network lifetime for different duration between two backbone reconstructions. To study the lifetime of the network, we have to give a definition of what we intend by the network lifetime. Several definitions of WSN lifetime exist in the literature; network lifetime is the time from the deployment to the instant when the network is considered non functional. When a network should be considered as non-functional is, however, application-specific. It can be, for example, the instant when the first sensor dies [26], a percentage of sensors die [27], the network partitions [28], or the loss of coverage occurs [29]. In this simulation, we consider the network lifetime as the number of end-to-end communication of the application messages achieved before one sensor runs out of its battery energy. This is the strongest definition among the 4 previous one.

We have in this simulation two kind of messages: the application messages and the control messages, *i.e.* messages that are sent by the algorithm presented in this paper.

The scenario of the simulation is to send end-to-end application messages in the network by varying the duration between two backbone reconstructions. We assume here that the frequency of the application messages is uniformly distributed over the time. Thus, we consider that the duration between two backbone reconstruction is a number of application messages. For lifetime computation, we need to know the frequency of control messages. Indeed, the higher this frequency is, the smaller the convergence time of our algorithm is. We fix the percentage of time for a good behavior of our algorithm to 90%, if there is no error. Thus, the convergence time has to be 10% of the duration between two backbone reconstructions. We can then deduce the control messages frequency.

Fig. 5 shows the stabilization time (reset time) according to the network size in case of common and variable consumed energy. The stabilization time is represented by the number of control messages sent by each node in the network.

As we can see in Fig. 5, in case of common consumed energy, stabilization time is fairly steady with the increase of the network size. While, in case of variable consumed energy, the stabilization time decreases with the increase of the network size. We also observe that the range of confidence interval decreases with the increase of network size in case of variable consumed energy.

From these two observations, we can deduce that the impact of the consumed energy on the backbone reconstruction time decreases with the increase of the network size (network density).

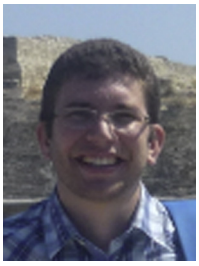
This result can be explained by the tree construction algorithm. In common consumed energy, the behavior of this algorithm is the same as the algorithm without weight. But in variable consumed energy, we can have two neighbors' nodes with a small energy consumed, and each one selects the other one as father, so the weight of each node will increase slowly. This phenomenon delays the selection of a good father in the tree. We also note that the more the graph is dense, the more the energy consumed by the nodes in the tree is small and hence less we meet this problem.

Fig. 6 shows the network lifetime according to the time (the number of end-to-end messages) between two backbone reconstructions.

We can observe that for a reconstruction period between 0 and 100,000 application messages, the network lifetime increases fastly. The period here is too short, then the overhead of the control messages is high. Furthermore, between two backbone reconstructions, there is a small number of application messages, so the energy

[16] P.-J. Wan, K.M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, Mobile Netw. Appl. 9 (2) (2004) 141–149.
[17] S. Kamei, H. Kakugawa, A self-stabilizing approximation for the minimum connected dominating set with safe convergence, in: T.P. Baker, A. Bui, S. Tixeuil (Eds.), Proceedings of the 12th International Conference on Principles of Distributed Systems, (OPODIS'08), LNCS, Springer-Verlag, 2008, pp. 496–511.
[18] H. Raei, M. Tabibzadeh, B. Ahmadipoor, S. Saei, A self-stabilizing distributed algorithm for minimum connected dominating sets in wireless sensor networks with different transmission ranges, in: Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT'09), IEEE CS Press, 2009, pp. 526–530.
[19] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (2) (1982) 329–343.
[20] Y. Wang, W. Wang, X.-Y. Li, Efficient distributed low-cost backbone formation for wireless networks, IEEE Trans. Parallel Distrib. Syst. 17 (2006) 681–693.
[21] C. Ambühl, T. Erlebach, M. Matus, M. Nunkesser, Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs, in: Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Springer, 2006, pp. 3–14.
[22] S. Dolev, Self-stabilization, MIT Press, 2000.
[23] M. Jelasity, A. Montresor, G.P. Jesi, S. Voulgaris, The Peersim simulator, http://peersim.sf.net.
[24] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 8 (HICSS'00), IEEE CS, 2000.
[25] E.W. Weisstein, "Square line picking." from mathworld—a wolfram web resource, http://mathworld.wolfram.com/SquareLinePicking.html.
[26] I. Kang, R. Poovendran, Maximizing static network lifetime of wireless broadcast adhoc networks, in: IEEE International Conference on Communications (ICC'03), IEEE CS Press, 2003.
[27] E.J. Duarte-Melo, M. Liu, Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks, in: Global Telecommunications Conference (GLOBECOM'02), IEEE CS Press, 2002.
[28] J.-H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, IEEE/ACM Trans. Netw. 12 (2004) 609–619.
[29] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, C. Gill, Integrated coverage and connectivity configuration for energy conservation in sensor networks, ACM Trans. Sens. Netw. 1 (2005) 36–72.

Dr. Ben-Othman received his BSc and MSc degrees both in Computer Science from the University of Pierre et Marie Curie (Paris 6) France in 1992 and 1994, respectively. He received his PhD degree from the University of Versailles, France, in 1998. He was an Assistant Professor at the University of Orsay (Paris 11) and University of Pierre et Marie Curie (Paris 6), in 1998 and 1999 respectively and Associate Professor at the University of Versailles from 2000 to 2011. He is now full Professor at University Paris 13. His research interests are in the area of wireless ad hoc and sensor networks, Broadband Wireless Networks, multiservices bandwidth management in WLAN (IEEE 802.11), WMAN (IEEE 802.16), WWAN (LTE), security in wireless networks in general and wireless sensor and ad hoc networks in particular.



Karim Bessaoud received his Engineer Diploma from University of Oran in Algeria in 2008 and his MS in Computer Science from Pierre et Marie Curie University (Paris 6) in 2009. He is currently a PhD student at Prism Laboratory in UVSQ, France. His research interests include Wireless Sensor Networks, self-stabilizing and distributed algorithms.

Alain Bui is full professor of computer science PRiSM-CNRS laboratory at the University of Versailles Saint-Quentin, France. He received his PhD in Computer Science in 1994 from University Paris 7 and INRIA. His interests include distributed algorithms, random walks and self-stabilization.

Laurence Pilard is associate professor at University of Versailles-St-Quentin-en-Yvelines at PRiSM CNRS laboratory. Currently her main topics are Self-Stabilizing Systems and Large Scale Sensor Networks. Her main collaborators are Ted Herman, Joffroy Beauquier, Fredrik Manne and Alain Bui. She is currently advising a PhD student on sensor networks structuration.