Energy-Aware Self-Stabilizing Distributed Clustering Protocol for Ad Hoc Networks: the case of WSNs

Mandicou Ba[1*], Olivier Flauzac[1], Bachar Salim Haggar[1], Rafik Makhloufi[1], Florent Nolot[1] and Ibrahima Niang[2]
[1] CReSTIC - SysCom EA 3804, Université de Reims Champage-Ardenne
Reims, BP 1039-51687- France
[e-mail: {mandicou.ba, olivier.flauzac, bachar-salim.haggar, rafik.makhloufi, florent.nolot}@univ-reims.fr]
[2] Laboratoire d'Informatique de Dakar, Université Cheikh Anta Diop
Dakar, Fann BP 5005- Sénégal
[e-mail: iniang@ucad.sn]
*Corresponding author: Mandicou Ba

Received June 30, 2013; revised September 16, 2013; accepted October 24, 2013; published November 29, 2013

Abstract
In this paper, we present an Energy-Aware Self-Stabilizing Distributed Clustering protocol based on message-passing model for Ad Hoc networks. The latter does not require any initialization. Starting from an arbitrary configuration, the network converges to a stable state in a finite time. Our contribution is twofold. We firstly give the formal proof that the stabilization is reached after at most n+2 transitions and requires at most $n * \log(2n + k + 3)$ memory space, where n is the number of network nodes and k represents the maximum hops number in the clusters. Furthermore, using the *OMNeT*++ simulator, we perform an evaluation of our approach. Secondly, we propose an adaptation of our solution in the context of Wireless Sensor Networks (WSNs) with energy constraint. We notably show that our protocol can be easily used for constructing clusters according to multiple criteria in the election of cluster-heads, such as nodes' identity, residual energy or degree. We give a comparison under the different election metrics by evaluating their communication cost and energy consumption. Simulation results show that in terms of number of exchanged messages and energy consumption, it is better to use the Highest-ID metric for electing CHs.

Keywords: Ad Hoc and WSNs, Clustering, Energy-Aware, Self-stabilizing, OMNeT++.

A preliminary version of this paper appeared in 7th ACM ICUIMC (IMCOM) 2013, January 17-19, Kota Kinabalu, Malaysia. This version includes a study case in the context of Wireless Sensors Networks (WNSs) with energy constraint. This research was supported by Regional Council of Champagne-Ardenne and European Regional Development Fund through the CPER CapSec ROFICA project. Simulations are executed on Grid'5000 experimental testbed hosted at the ROMEO HPC Center. We express our thanks to Regional Council of Champagne-Ardenne, European Regional Development Fund, Crid'5000 and ROMEO HPC Center.

http://dx.doi.org/10.3837/tiis.2013.11.002

1. Introduction

Due to their properties and to their wide applications, Wireless Sensor Networks (WSNs) have been gaining growing interest in the last decades. These networks are used in various domains like: medical, scientific, environmental, military, security, smart homes etc. [1].
In a WSN, sensors have very limited energy resources due to their small size. This battery power is consumed by three operations: data sensing, communication and processing. Communication of messages is the activity that needs the most important quantity of energy while power required by CPU is minimal. For example, [2] shows that the energy cost of transmitting a *1KB* message over a distance of *100* meters is approximately equivalent to the execution of *3 million CPU* instructions by a *100 MIPS/W* processor. However, the most frequently used communication solution in these networks is diffusion, because it is simple and it requires few calculations. But, this method is expensive and may cause network saturation. Thus, saving communication power is more urgent in WSNs than optimizing processing. Consequently, to extend the sensor network lifetime, it is very important to carefully manage the very scarce battery power of sensors by limiting communications. This can be done through notably efficient routing protocols that optimize energy consumption.
To do this, one solution is to structure the network into trees [3,4,5] or into clusters [6,7,8]. Many previous studies (e.g. [9,10]) proved that clustering is an effective scheme in increasing the scalability and lifetime of wireless sensor networks. Clustering consists in partitioning the network into groups called clusters, thus giving a hierarchical structure [11]. A particular node called cluster-head manages each cluster. A node is elected as a cluster-head using a certain metric such as the mobility degree, node's identity, node's density, etc. or a combination of these parameters. Several self-stabilizing clustering solutions have been proposed. They are classified into *1-hop* and *k-hops* algorithms. In *1-hop* solutions [12,13,14,15], nodes are at a distance of *1* from the cluster-head and the maximum diameter of clusters is *2*. However, in k-hops solutions [16,17,18], nodes can be located at a distance of *k* from the cluster-head and the maximum diameter of clusters is *2k*. However, these approaches generate a lot of traffic and require considerable resources.
In this paper, we propose an Energy-Aware Self-Stabilizing Distributed Clustering protocol based on message-passing for heterogeneous wireless Ad Hoc networks. Our algorithm is completely distributed and self-stabilizing. *Dijkstra* defined a distributed system to be self-stabilizing if, regardless of the initial state, the system is guaranteed to reach a legitimate (correct) state in a finite time [19]. Our protocol builds non-overlapping clusters in *k-hops* and does not require initialization. It is based on the criterion of maximum identity attached to the nodes for cluster-head selection and relies only on the periodic exchange of messages with the *1-hop* neighbourhood. The choice of the identity metric provides more stability against dynamic criteria such as mobility degree and weight of nodes. We validate our approach through both formal poof and simulations.
On one hand, we prove that a legal configuration is reached after at most *n+2* transitions and it requires at most $n * \log(2n + k + 3)$ memory space, where n is the number of network nodes. Furthermore, using the *OMNeT++* simulator, we performed several simulations in order to evaluate the performance of our clustering approach.
On the other hand, we propose a study case in the context of Wireless Sensor Networks (WSNs) with energy constraint. We show that our solution optimizes energy consumption and thus prolongs the network lifetime by minimizing the number of exchanged messages

involved during the clusters formation. It also offers an optimized structure for routing. Our clustering protocol is generic and complete. It can be easily used for constructing clusters according to multiple criteria in the election of cluster-heads such as: nodes' identity, residual energy, and degree. We propose to validate our approach by evaluating its communication cost in terms of exchanged messages and energy consumption. Thus, we compare its performance in the case of using different cluster-heads election methods under the same clustering approach and the same testing framework.

The remainder of the paper is organized as follows. In Section 2 we describe some related works of self-stabilizing clustering solutions. Section 3 presents our contribution. In Section 4, we describe the computational model used in the paper and we give some additional definitions and concepts. In Section 5 we first present a broad and intuitive explanation of the proposed algorithm before defining it more formally. In Section 6 we establish the formal proof of our approach and give its memory space complexity. Section 7, illustrates the performance of our solution through simulations. A study case in the context of Wireless Sensor Networks (WSNs) with energy constraint is described in Section 8. Finally, we conclude and present some research perspectives in Section 9.

2. Related work

Several proposals of clustering have been done in the literature [12,13,14,15,16,17,18].
Self-stabilizing algorithms presented in [12,13,14,15] are 1-hop clusters solutions.
A metric called *density* is used by Mitton *et al.* in [12], in order to minimize the reconstruction of structures for low topology change. Each node calculates its density and broadcasts it to its neighbours located at *1-hop*. For the maintenance of clusters, each node periodically calculates its mobility and density.
Flauzac *et al.* [13] have proposed a self-stabilizing clustering algorithm, which is based on the identity of its neighbourhood to build clusters. This construction is done using the identities of each node that are assumed unique. The advantage of this algorithm is to combine in the same phase the neighbours discovering and the clusters establishing. Moreover, this deterministic algorithm constructs disjoint clusters, i.e., a node is always in only one cluster.
In [14], Johnen *et al.* have proposed a self-stabilizing protocol designed for the state model to build 1-hop clusters having a bounded size. This algorithm guarantees that the network nodes are partitioned into clusters where each one has at most *SizeBound* nodes. The cluster-heads are chosen according to their *weight* value. In this case, the node with the highest weight becomes cluster-head. In [15] Johnen *et al.* have extended the proposal from [14]. They have proposed a robust self-stabilizing weight-based clustering algorithm. The robustness property guarantees that, starting from an arbitrary configuration, after one asynchronous round, the network is partitioned into clusters. After that, the network stays partitioned during the convergence phase toward a legitimate configuration where clusters verify the ad hoc clustering properties. These approaches [14,15], based on state model, are not realistic in the context of wireless sensor networks.
Self-stabilizing algorithms proposed in [16,17,18] are k-hops clustering solutions.
In [16] Mitton *et al.* applied self-stabilization principles over a clustering protocol proposed in [12] and they present properties of robustness. Each node computes its *k-density* value based on its view {k+1}-neighbourhood and locally broadcasts it to all its neighbours at distance *k*. Thus, each node is able to decide by itself whether it wins in its 1-*neighbourhood* (as usual, the smallest *ID* will be used to decide between joint winners). Once a cluster-head is elected, the

cluster-head *ID* and its density are locally broadcasted by all nodes that have joined this cluster. A cluster can then extend itself until it reaches a cluster frontier of another cluster-head. The approach proposed in [12,16] generates a lot of messages. The main reason is due to the fact that each node must know {k+1}-Neighbouring, computes its k-*density* value and locally broadcasts it to all its k-*neighbours*. This is very expensive in terms of messages and causes an important energy consumption.
Caron *et al.* [17], using as metric a unique ID for each process and weighted edges, have proposed a self-stabilizing *k-clustering* algorithm based on a state model. Note that *k-clustering* of a graph consists in partitioning network nodes into disjoints clusters, in which every node is at a distance of at most k from the cluster-head. This solution is partially inspired by Amis *et al.* [20] and finds a *k-dominating* set in a network of processes. It is a combination of several self-stabilizing algorithms and it uses an unfair daemon. Each process can read its own registers and those of its neighbours at distance $k + 1$, but can write only to its own registers. This algorithm executes in $O(n*k)$ rounds and requires $O(log(n) + log(k))$ memory space per process, where n is the network size.
In [18] using criterion of minimal identity, *Datta et al.* have proposed a self-stabilizing distributed algorithm called *MINIMAL*. This approach is designed for the state *model* (also called *shared memory model*) and uses an unfair daemon. Authors consider an arbitrary network *G* of processes with unique *IDs* and no designated leader. Each process can read its own registers and those of its neighbours at distance k, but can write only to its own registers. They compute a subset *D*, a minimal *k-dominating* set of graph *G*. *D* is defined as a k-dominating set if every process that is not in *D* is at distance at most k from a member of *D*. *MINIMAL* converges in $O(n)$ rounds. Using *D* as the set of cluster-heads, a partition of *G* into clusters, each of radius k follows. Authors show that $O(n^2)$ steps are sufficient for the phase clock to stabilize. And after stabilization, *MINIMAL* requires $O(n^2)$ steps to execute n actions. Thus, the system converges to a terminal configuration in $O(n^2)$ steps starting from any configuration and requires $O(log(n))$ memory space per process, with n the network size.

3. Contribution Description

We propose an efficient Energy-Aware Self-Stabilizing Distributed Clustering protocol for Ad Hoc networks that builds k-hops clusters. Our algorithm is based on message-passing model and does not require any initialization. It structures the network into non-overlapping clusters with a diameter at most equal to $2k$. This structuring is based only on information from neighbouring nodes at distance *1*. Contrary to other clustering algorithms, our solution uses a unique type of message to discover the neighbourhood of a node at distance 1 and to structure the network into *k-hops* clusters. Starting from an arbitrary configuration, the network converges to a legal configuration after a finite time. Furthermore, we propose a study case in the context of Wireless Sensor Networks (WSNs) with energy constraint. The proposed protocol is generic and complete. It can be easily used for constructing clusters according to multiple criteria in the election of cluster-heads such as: nodes' identity, residual energy, and degree. We show that our solution optimizes energy consumption and thus prolongs the network lifetime by minimizing the communication cost involved during clusters formation.

4. Problem Specification

The main concepts used along this paper are defined in this section according to [21, 22].

4.1 Computation Model

We consider our network as a distributed system that can be modelled by an undirected graph $G = (V, E)$. Where, V is the set of network nodes such that $|V| = n$ and $n \geq 2$. E represents all existing connections between nodes. An edge (u, v) exists if and only if u can communicate with v and vice-versa. This means that all links are bi-directional. In this case, the nodes u and v are neighbours. The set of neighbours' $v \in V$of node u is marked N_u. Each node u of the network has a unique identifier id_u such that $0 \leq id_u \leq n-1$ and can communicate with N_u. We define the distance $dist_{(u,v)}$ between nodes u and v in the graph G as the minimum number of edges along the path between u and v.

4.2 Message-passing Model

Our proposed algorithm is designed through an asynchronous message-passing model following the standard models given in [21, 22] for distributed systems. For this purpose, each pair of nodes is connected by a bi-directional link. Links are asynchronous and messages transit time is finite but not bounded. Moreover links are reliable. They do not create, corrupt or lose messages. Furthermore, each node u periodically sends to its neighbours a message that is received correctly within some finite but unpredictable time by all its 1-hop neighbours. Each node u maintains a table containing the current state of its direct neighbours. Upon receiving a message, a node u executes our clustering approach.

4.3 Self-Stabilizing System

4.3.1 Transition Concept

In distributed systems, *transitions* influence only a part of the *configuration*, (*i.e.*, the system's global state). Each configuration itself is a tuple and each transition refers to some components of this tuple. The component of the configuration includes the state of each node in the network. A *transition system* consists of a set of all possible states of the network.
Formally, a system transition is a triple $S = (C, \longmapsto, I)$, where C is the set of configurations, $\longmapsto$ is a binary transition relation on C, and I is a subset of C and represents all arbitrary initial configurations.
A *transition relation* is a subset of $C \times C$. Instead of $(\gamma, \delta) \in \longmapsto$, the more convenient notation $\gamma \longmapsto \delta$ is used. Let $S = (C, \longmapsto, I)$ be a transition system. An *execution* $\varepsilon = (\gamma_0, \gamma_1, \gamma_2, \cdots)$ of S is a maximal *sequence* where $\gamma_0 \in I$, and for all $i \geq 0, \gamma_i \longmapsto \gamma_{i+1}$. $\gamma_i \longmapsto \gamma_{i+1}$ consists in a transition (also called *step* or *round*) in which each node u executes all enabled rules of type: $< Rule > ::< Predicate > \longmapsto < Action >$. Where $< Predicate >$ is a boolean expression involving the local variables of nodes and their neighbours. $< Action >$ is an assignment that can modify local variables of nodes.
A *terminal configuration* is a configuration for which there is no δ such that $\gamma_i \longmapsto \gamma_{i+1}$.
A sequence $\varepsilon = (\gamma_0, \gamma_1, \gamma_2, \cdots)$ with $\gamma_i \longmapsto \gamma_{i+1}$ for all i, is *finite* if it ends in a terminal configuration γ. The terminal configuration is also called *legal configuration*. Let $\mathcal{L}_C$ be the set of all legal configurations of a transition system S.

4.3.2 Self-stabilizing Concept

Self-stabilization was introduced by Dijkstra [19] as a system, regardless of its starting configuration, is guaranteed to reach a legal configuration in a finite number of transitions.
Formally, we say that a transition system S is self-stabilizing if the following two conditions hold:

1. *Convergence*: a legal configuration δ is reachable from γ, notation $\gamma \rightsquigarrow \delta$, if exists a sequence $\varepsilon = (\gamma_0, \gamma_1, \gamma_2, \cdots, \gamma_k = \delta)$ with $\gamma_i \mapsto \gamma_{i+1}$ for all $0 \leq i < k$.
2. *Closure*: if an execution ε begins at γ member of $\mathcal{L}_C$, then all configuration γ_i of ε are members of $\mathcal{L}_C$.

Note that a legal configuration δ is reachable if it is reachable starting from an arbitrary configuration and without occurrence of faults.

Let S be a self-stabilizing transition system. The *stabilization time* in S is the cardinality of a sequence $\varepsilon \in S$ (*i.e.* number of transitions or steps in ε) such that ε is finite.

5. Self-stabilizing k-hops Clustering Algorithm

5.1 Preliminaries

In this section, we give definitions of main concepts and notations used in our algorithm.

Definition 1: (Cluster) we define a k-hops cluster as a connected sub-graph in the network, with a diameter less than or equal to $\boldsymbol{2k}$. The set of all the nodes of a cluster $\boldsymbol{i}$ is denoted $\boldsymbol{V_i}$.

Definition 2: (Cluster identifier) each cluster has a unique identifier corresponding to the highest node identity in its cluster. The identity of a cluster that owns a node $\boldsymbol{i}$ is denoted $\boldsymbol{cl_i}$.

In our clusters, each node u has a status noted $status_u$. Thus, a node can be Cluster-Head (*CH*), a *Simple Node* (*SN*) or a *Gateway Node* (*GN*). Moreover, each node selects a neighbour $v \in N_u$, noted gn_u, through which it can reach its *CH*.

Definition 3: (Node status)

- $ClusterHead$ $(\boldsymbol{CH})$: a node u has CH status if it has the highest ID among all nodes of its cluster:
  - $status_u = CH \iff \forall\, v \in V_{cl_u}, (id_u > id_v) \wedge \left(dist_{(u,v)} \leq k\right).$
- $Simple\ Node$ $(\boldsymbol{SN})$: a node u has SN status if u and all its neighbours are in the same cluster:
  - $status_u = SN \iff (\forall\, v \in N_u, cl_u = cl_v) \wedge \{\exists\, w \in V_u\ ,\ (status_w = CH) \wedge \left(dist_{(u,w)} \leq k\right)\}.$
- $Gateway\ Node$ $(\boldsymbol{GN})$: a node u has *GN* status if is has at least one neighbouring node v in a different cluster:
  - $status_u = GW \iff \exists\, v \in N_u, (cl_u \neq cl_v).$

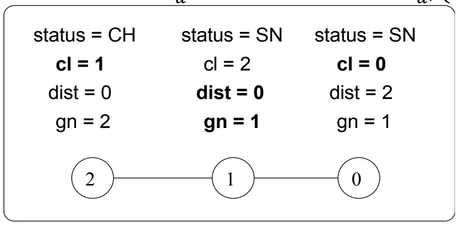


Fig. 1 Incoherent nodes

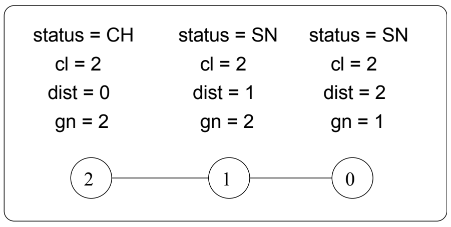


Fig. 2 Coherent nodes

Definition 4: (Node coherence)

A node u is a *coherent node if and only if* it is in one of the following states (**Fig. 1** and **Fig. 2**):

- $if\ status_u = CH, then\ (cl_u = id_u) \wedge \left(dist_{(u,CH_u)} = 0\right) \wedge (gn_u = id_u)$.
- $if\ status_u \in \{SN, GW\}\ then\ (cl_u \neq id_u) \wedge \left(dist_{(u,CH_u)} \neq 0\right) \wedge (gn_u \neq id_u)$

Definition 5: (Node stability)
A node *u* is *stable node* if and only if its variables no longer change, it is coherent and satisfies the following states:

- $if\ status_u = CH\ then\ \forall\ v \in N_u, (status_v \neq CH) \wedge \left\{\left((cl_v = cl_u) \wedge (id_u < id_u)\right) \vee \left((cl_u \neq cl_u) \wedge \left(dist_{(v,CH_v)} = k\right)\right)\right\}$. Example of node 9 in cluster V_9 as illustrated in **Fig. 3**.
- $if\ status_u = SN\ then\ \forall\ v \in N_u, (cl_u = cl_v) \wedge \left(dist_{(u,CH_u)} \leq k\right) \wedge \left(dist_{(v,CH_v)} \leq k\right)$. Example of node 10 in cluster V_{10} as illustrated in **Fig. 3**.
- $if\ status_u = GN\ then\ \exists\ v \in N_u, (cl_u \neq cl_u) \wedge \left\{\left(\left(dist_{(u,CH_u)} = k\right) \wedge \left(dist_{(v,CH_v)} \leq k\right)\right) \vee \left(\left(dist_{(u,CH_u)} \leq k\right) \wedge \left(dist_{(v,CH_v)} = k\right)\right)\right\}$. Example of node 2 in cluster V_9 or node 8 in cluster V_{10} as illustrated in **Fig. 3**.

Definition 6: (Network stability)

The network is *stable* if and only if all nodes are stable nodes. (See **Fig. 3**)

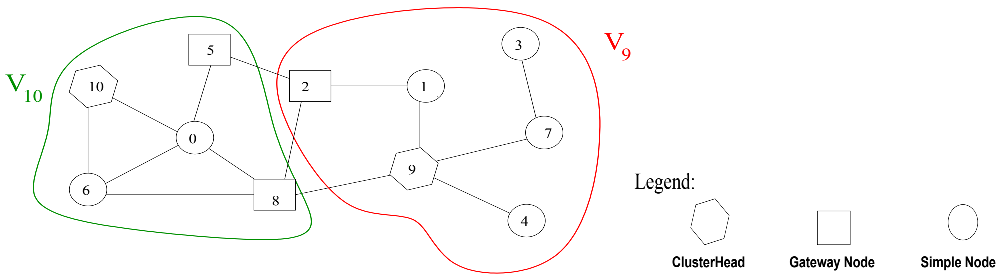
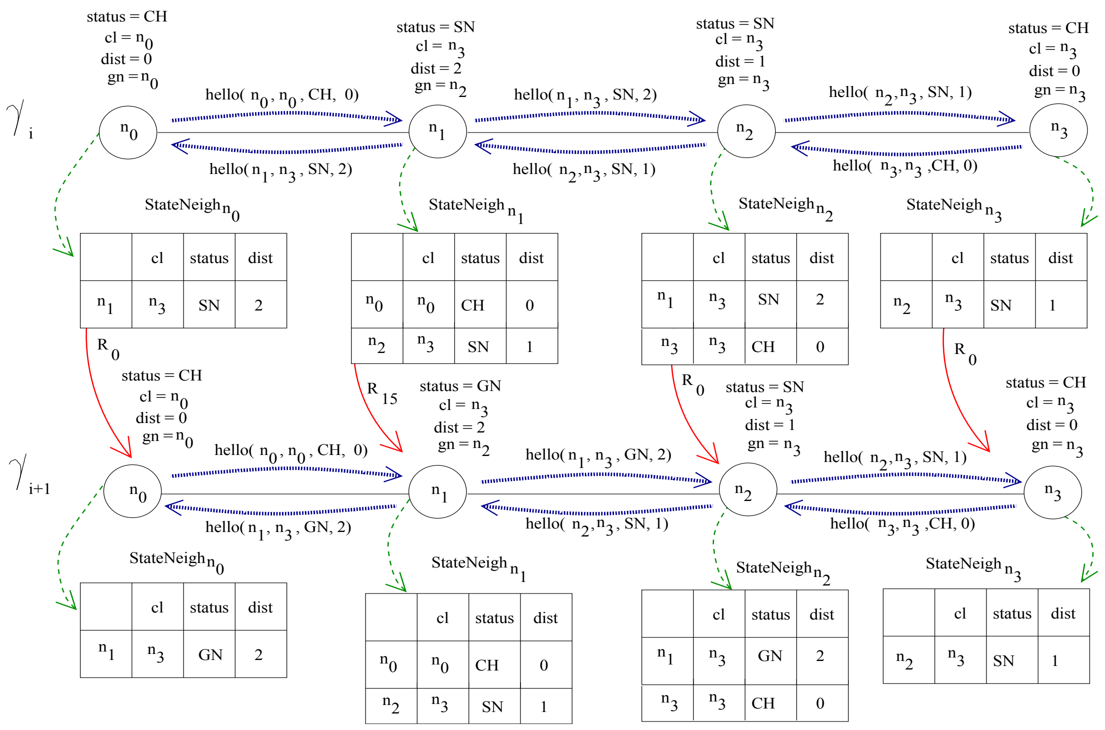


Fig. 3 Stable nodes - Stable clusters

5.2 Basic Idea of Proposed Algorithm

Our algorithm is self-stabilizing and does not require any initialization. Thus, starting from any arbitrary configuration, with only one type of exchanged message, nodes are structured into non-overlapping clusters in a finite number of steps. This message is called *hello message* and it is periodically exchanged between each neighbour nodes. It contains the following four items information: node identity, cluster identity, node status and the distance to cluster-head. Thus, the *hello message* structure is $hello\ (id_u, cl_u, status_u, dist_{(u,CH_u)})$. Note that cluster identity is also the identity of the cluster-head. Furthermore, each node maintains a neighbourhood table $StateNeigh_u$ that contains the set of its neighbouring nodes states. Whence, $StateNeigh_u[u]$ contains the states of nodes *v* neighbour of *u*. Our solution proceeds as follows:
As soon as a node *u* receives a *hello message*, it executes Algorithm 1. During this algorithm, *u* executes three steps consecutively. The first step consists in updating neighbourhood. The next step is the coherence management and, finally, the last step is building the clusters. At the end of these three steps, *u* sends a *hello message* to its neighbours.
After updating the neighbourhood, all nodes check their coherence. For example, as a cluster-head has the highest identity, if a node *u* has *CH* status, its cluster identity must be equal to its identity. In **Fig. 1**, node 2 is cluster-head. Its identity is 2 and its cluster identity is 1. Thus, node 2 is not a coherent node. This is similar for nodes 1 and 0, because they do not

satisfy definition 4. Each node detects its incoherence and corrects it during the coherence management step. **Fig. 2** shows nodes that are coherent.
During the clustering step, each node compares its identity with those of its neighbours at distance 1. A node u elects itself as a cluster-head if it has the highest identity among all nodes of its cluster. If a node u discovers a neighbour v with a highest identity then it becomes a node of the same cluster as v with SN status. If u receives again a *hello message* from another neighbour, which is in another cluster than v, the node u becomes gateway node with *GN* status. As the *hello message* contains the distance between each node u and its cluster-head, u knows if the cluster diameter is reached. So it can choose another cluster.
Fig. 4 illustrates the transition from configuration $\gamma_i \longmapsto \gamma_{i+1}$. At γ_i, each node sends to its neighbours a *hello message*. γ_{i+1} is a legal configuration. Upon receiving messages from neighbours, each node updates its neighbour table and executes Algorithm 1. In this example, we have 2-hops clustering. Node n_1, is member (*SN*) of node $n_3$cluster. Furthermore, it is at distance 2 from n_3. It detects node n_0 as neighbour cluster. Thus, it becomes gateway node with *GN* status. After that, node n_1 sends a *hello message* to its neighbours for an update. Nodes n_3, n_2 and n_0 according to their current state and their neighbours, do not change their state. γ_{i+1} corresponds to a legal configuration.



γ_i: StateNeigh n_0

	cl	status	dist
n_1	n_3	SN	2

γ_i: StateNeigh n_1

	cl	status	dist
n_0	n_0	CH	0
n_2	n_3	SN	1

γ_i: StateNeigh n_2

	cl	status	dist
n_1	n_3	SN	2
n_3	n_3	CH	0

γ_i: StateNeigh n_3

	cl	status	dist
n_2	n_3	SN	1

γ_{i+1}: StateNeigh n_0

	cl	status	dist
n_1	n_3	GN	2

γ_{i+1}: StateNeigh n_1

	cl	status	dist
n_0	n_0	CH	0
n_2	n_3	SN	1

γ_{i+1}: StateNeigh n_2

	cl	status	dist
n_1	n_3	GN	2
n_3	n_3	CH	0

γ_{i+1}: StateNeigh n_3

	cl	status	dist
n_2	n_3	SN	1

Fig. 4 Transitions in 2-hops clusters

5.3 Formal Description of Proposed Algorithm

In our solution, each node u of the network has the following local variables defined in Table 1. These variables allow for each node to store information on its direct neighbours and on its *CH* located at most at k-hops. Upon receiving from a neighbour, each nodes u executes our clustering approach described in **Algorithm 1**.

Table 1. Constants and variables definition

Constants and variables definition
Parameters: $\|V\| = n$: *the total number of node in the netwrok* k: *maximun number of hop in clusters*
Local variables of node u: N_u: *neighbourhood at distance* 1 *of node u.* cl_u: *cluster identity of node u.* id_u: *identity of node u.* $status_u \in \{CH, SN, GN\}$: *status of node u.* $dist_{(u,CH_u)}$: *distance from node u to its CH.* gn_u : *identity of neighbor that allows u to reach its CH.* $StateNeigh_u$: *a neighbour table of node u containing all states of its neighbouring nodes at distance* 1. $StateNeigh_u[v]$: *contains the states of node v neighbour of node u.*
Macros: $NeighCH_u = \{id_v, v \in N_u \wedge status_v = CH \wedge cl_u = cl_v\}$ $NeighMax_u = (Max\{id_v, v \in N_u \wedge status_v \neq CH \wedge cl_u = cl_v\})$ $\wedge \left(dist_{(v,CH_v)} = Min\{dist_{(x,CH_x)}, x \in N_u \wedge cl_x = cl_v\}\right)$

6. Formal Proof of Self-stabilization and Memory Space Complexity

6.1 Proof of Convergence and Closure

In this section, we outline the main theorems and properties verified by our algorithm and its maximum memory space required. Due to space limitation, we give only sketches of proof. All details of proofs are described in [23].

To prove the convergence and the closure of our algorithm, we use the notion of *fixed* node. In our approach, nodes with the highest identity are *fixed* in the first place. Formally, A node u is said to be *fixed* at configuration γ_i if $\forall i \geq j$, cl_u is equal to cl_u at cl_j. Let $\mathcal{F}_i$ be the set of fixed nodes at γ_i. To make the proof of convergence, we study the value of $|\mathcal{F}_i|$ and prove that $|\mathcal{F}_i| = n$ if $\gamma_i = \delta$, where n is the number of nodes in the graph.

Lemma 1. **Let γ_0 *an arbitrary configuration* . At γ_1 $\forall u \in V, u$ *is coherent*.**

Sketch of Proof

Let γ_0 an arbitrary configuration. Each node, regardless its state, verifies its incoherence and corrects its during $\gamma_0 \longmapsto \gamma_1$ by applying rule R_{10} or R_{20}. At γ_1, all nodes become coherent.

Corollary 1. $|\mathcal{F}_1| \geq 1$ **and** $|\mathcal{F}_1| > |\mathcal{F}_0|$ **.**

Sketch of Proof

According to Lemma 1, all nodes are coherent at γ_1. We know that at γ_0, $\exists u$ *such that* $\forall v \in V, id_u > id_v$. At least u applies rule R_{11} at step *Cluster-2* during $\gamma_0 \longmapsto \gamma_1$ and is fixed at γ_1. Therefore $u \in \mathcal{F}_1$ and $|\mathcal{F}_1| \geq 1 \Rightarrow |\mathcal{F}_1| > |\mathcal{F}_0|$. This node is called CH_{Max}.

Theorem 1. $\forall\, i < k+1, |\mathcal{F}_{i+1}| > |\mathcal{F}_i|$.

Sketch of Proof This proof is made by induction on i.

The base case, $i = 0$: according to Corollary 1 we have $|\mathcal{F}_1| > |\mathcal{F}_0|$. Thus, the induction assumption is verified at the first range.

For the inductive step, assume the theorem holds $\forall\, i < k+1$ and prove that $|\mathcal{F}_{i+1}| > |\mathcal{F}_i|$.

At γ_2, all nodes at distance 1 of CH_{Max} such that defined in Corollary 1 are fixed by applying rule R_{13} or R_{16} according to their status. By induction, we prove that at γ_i, all nodes at distance $i-1$ of CH_{Max} are fixed by applying rule R_{14} or R_{17} according to their status. For $i = k$, by induction we have $|\mathcal{F}_{k+1}| > |\mathcal{F}_k|$. Therefore, $\forall\, i < k+1, |\mathcal{F}_{i+1}| > |\mathcal{F}_i|$.

Theorem 2. (***Convergence***) $From\ any\ configuration\ \gamma_0, we\ reach\ a\ legal\ configuration$ $\delta\ after\ at\ most\ n+2\ transitions.$

Sketch of Proof

$As\ |\mathcal{F}_1| \geq 1, we\ have\ |\mathcal{F}_{k+1}| \geq k$. By repeating the process form a new CH_{Max}, which is the maximum identity node $\notin \mathcal{F}_{k+1}$, we prove that $\forall\, i < n, |\mathcal{F}_{i+1}| > |\mathcal{F}_i|$ and $\mathcal{F}_i \subset \mathcal{F}_{i+1}$. For $i = n$, we have $|\mathcal{F}_{n+1}| > |\mathcal{F}_n|$. As $|V| = n$, thus $|\mathcal{F}_{n+1}| = n$. It is then necessary a supplementary transition to the status of nodes no longer change.

Algorithm 1: *Energy-Aware Self-Stabilizing Distributed Clustering algorithm*

```
/* Upon receiving message from a neighbour*/
Predicates
$P_1(u) \equiv (status_u = CH)$
$P_2(u) \equiv (status_u = SN)$
$P_3(u) \equiv (status_u = GN)$
$P_{10}(u) \equiv (cl_u \neq id_u) \vee (dist_{(u,CH_u)} \neq 0) \vee (gn_u \neq id_u)$
$P_{20}(u) \equiv (cl_u = id_u) \vee (dist_{(u,CH_u)} = 0) \vee (gn_u = id_u)$
$P_{40}(u) \equiv \forall v \in N_u, (id_u > id_v) \wedge (id_u \geq cl_v) \wedge (dist_{(u,v)} \leq k)$
$P_{41} \equiv \exists v \in N_u, (status_v = CH) \wedge (cl_v > cl_u)$
$P_{42} \equiv \exists v \in N_u, (cl_v > cl_u) \wedge (dist_{(v,CH_v)} < k)$
$P_{43} \equiv \forall v \in N_u, (cl_v > cl_u) \wedge (dist_{v,CH_v} = k)$
$P_{44} \equiv \exists v \in N_u, (cl_v \neq cl_u) \wedge \{(dist_{(v,CH_v)} = k) \vee (dist_{(u,CH_u)} = k)\}$

Rules
/*Update neighbourhood*/
$StateNeigh_u[v] := (id_v, cl_v, status_v, dist_{(v,CH_v)});$

/*Cluster-1: Coherent management*/
$R_{10}(u) :: P_1(u) \wedge P_{10}(u) \rightarrow cl_u := id_u; gn_u := id_u; dist_{(u,CH_u)} := 0;$
$R_{20}(u) :: \{P_2(u) \vee P_3(u)\} \wedge P_{20} \rightarrow status_u := CH; cl_u := id_u; gn_u := id_u; dist_{(u,CH_u)} := 0;$

/*Cluster-2: Clustering*/
$R_{11}(u) :: \neg P_1(u) \wedge P_{40}(u) \rightarrow status_u := CH; cl_u := id_u; gn_u := id_u; dist_{(u,CH_u)} := 0;$
$R_{12}(u) :: \neg P_1(u) \wedge P_{41}(u) \rightarrow status_u := SN; cl_u := id_v; dist_{(u,v)} := 1; gn_u := NeighCH_u;$
$R_{13}(u) :: \neg P_1(u) \wedge P_{42}(u) \rightarrow status_u := SN; cl_u := cl_v; dist_{(u,CH_u)} := dist_{(v,CH_v)} + 1; gnu := NeighMax_u;$
$R_{14}(u) :: \neg P_1(u) \wedge P_{43}(u) \rightarrow status_u := CH; cl_u := id_u; gn_u := id_u; dist_{(u,CH_u)} := 0;$
$R_{15}(u) :: P_2(u) \wedge P_{44}(u) \rightarrow status_u := SN;$
$R_{16}(u) :: P_1(u) \wedge P_{41}(u) \rightarrow status_u := SN; cl_u := id_v; dist_{(u,v)} := 1; gn_u := NeighCH_u;$
$R_{17}(u) :: P_1(u) \wedge P_{42}(u) \rightarrow status_u := SN; cl_u := cl_v; dist_{(u,CH_u)} := dist_{(v,CH_v)} + 1; gnu := NeighMax_u;$

/*Sending hello message*/
$R_0\,(u) :: hello(id_u, cl_u, status_u, dist_{(u,CH_u)})$
```

Theorem 3. (***Closure***) $From\ a\ legal\ configuration\ \gamma_i, without\ occurence\ of\ faults,$ $each\ node\ remains\ in\ \ a\ legal\ configuration\ at\ \gamma_{i+1}$.

Sketch of Proof

Let γ_i a legal configuration. $\forall u \in V$, u is fixed γ_i and only rule R_0 will be executed. Thus, $\forall\, j < i$, at γ_i, we have a legal configuration.

6.2 Memory Space Complexity

Lemma 2. *The memory space occupation required for each node u is at most* $n * \log(2n + k + 3)$.

Sketch of Proof

Each node u stores for each neighbour v the identity (id_v), the cluster identity (cl_v), the status $(status_v)$ and the distance $(dist_{(v,CH_v)})$. Thus, we have n possible values for identity variable; n possible values for cluster identity variable; k possible values for the distance and 3 kinds for the status. Therefore, the memory space required for each neighbour v *is* $\log(2n + k + 3)$. Furthermore, each node u stores the same information for itself. Finally, our algorithm requires at most $n * \log(2n + k + 3)$ memory space.

6.3 Analytical Comparison

Table 2 illustrates a comparison of stabilization time and memory space between our approach and others approaches designed for the state model. We note that the stabilization time of our solution does not depend on the k parameter contrary to approach proposed in [17]. Proposed algorithm has a unique phase to discover the neighbourhood and to build *k-hops* clusters. It also has a unique stabilization time contrary to approach described in [18]. Furthermore, our solution considers a 1-hop neighbourhood at opposed to [17, 18]. In fact, our approach is only based on locally neighbourhood information.

Table 2. Theoretical comparison of stabilization time and memory space

	Stabilization time	Memory space per neighbour	Neighbourhood
Our approach	$n + 2$	$\log(2n + k + 3)$	*1-hop*
Datta *et al.* [18]	$O(n), O(n^2)$	$O(\log(n))$	*k-hops*
Caron et al. [17]	$O(n * k)$	$O(\log(n) + \log(k))$	*{k+1}-hops*

Furthermore in Ba et *.al* [24], we have compared our proposed algorithm with one of the most referenced self-stabilizing solutions based on message-passing model [16]. This shows that we reduce communication cost and energy consumption by a factor of at least 2.

7. Performance Analysis of Our Approach

As we have proved in Section 6, with our approach the network is stable after at most *n*+2 transitions. This reflects the worst case of a topology where nodes form an ordered chain. However, Ad Hoc networks are often characterized by random topologies. In order to evaluate the performance of our solution in a random topology, we have implemented proposed algorithm using the *OMNeT++* [25] simulation environment. For generating random graphs, we have used SNAP [26] library. Simulations were carried out using *Grid'5000* [27] platform.

7.1 Impact of Network Size and Nodes Degree on Stabilization Time

First, we study the impact of network size and degree on stabilization time. In **Fig. 5**, we have fixed *k* = 2. For each node degree fixed at 3, 5 and 7 we consider a network size from 100 to 1000 nodes. Note that, we generate *d-regular graphs models* using *SNAP* library, where *d* represents nodes degree (number of neighbours for each node). For each given network size, we compute several series of simulations. We give the average stabilization time as the

average of all values corresponding to simulations results. We note that the stabilization time increases as the number of nodes in the network increases. Furthermore, we note that for arbitrary topologies, the average stabilization time is below $n + 2$, formal value proved in the worst case. Moreover, the number of transitions needed to reach a legal configuration appears stable when the network size increases (500 to 1000 nodes).
To better observe the impact of nodes degree on the stabilization time as illustrated in **Fig. 6**, we consider a network size of 100, 200 and 400 nodes and we vary the nodes degree. We observe that the stabilization time decreases as the nodes degree increases. The main reason is due to the fact that each node has more neighbours. Thus during each transition, we have more nodes that fixed at the same time. With our approach, we have a better stabilization time under networks with high degree. However, the Ad Hoc networks are often characterized as dense networks.

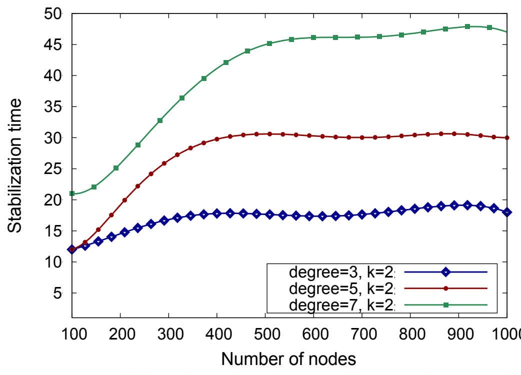


Fig. 5. Stabilization time according to the number of nodes

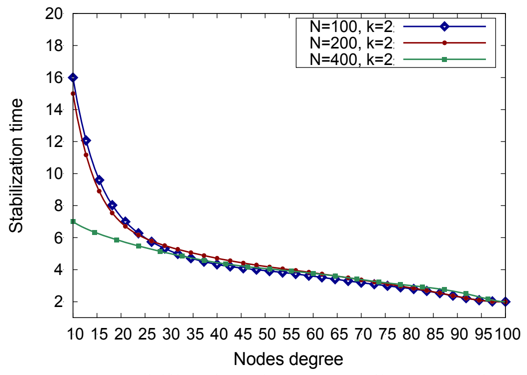


Fig. 6. Stabilization time according to nodes degree

7.2 Scalability

To examine the scalability of our approach, we vary the number of nodes in the network at the same time as the density of connectivity. For $k = 2$, we consider a network size of 100 to 1000 nodes. For each network size, we vary the density from 10% to 100%. Note that we generate *Erdos-Renyi random graphs models* using *SNAP* library. Thus, we obtain the *3D* curve illustrated in **Fig. 7**.
We note that except for low densities (10% and 20%), the stabilization time varies slightly with the increasing number of nodes. In case of a low density, we observe a peak due to longer chains in the network topology. With these series of simulation, we can make two main remarks. (*i*) The only determining factor for our approach is the density of connectivity and our solution is scalable. (*ii*) On average, for networks with an arbitrary topology, the stabilization time is far below the one of the worst-case scenario ($n + 2$ transitions).

7.3 Impact of k Parameter

In order to observe the impact of the k parameter, we set the node degree at 5 and we consider a network size of 100, 200, 400, 500 and 1000 nodes. For each network size, we vary the k parameter from 2 to 10. **Fig. 8** shows the stabilization time according to the k parameter variation. We observe that, the stabilization time decreases as the k parameter increases. In fact, if k parameter increases and because the hello message contains the distance between each node u and its cluster-head, the sphere of influence of the largest nodes increases. Thus, nodes carry fewer transitions to be fixed at a *CH*. Finally, we have fewer clusters. Nevertheless, in the case of a small value of the *k* parameter, we have more clusters with small diameters. Therefore, it requires more transitions to reach a stable state in all clusters. Note that, regardless the value of the *k* parameter, the stabilization time is far below the worst-case scenario ($n + 2$ transitions).

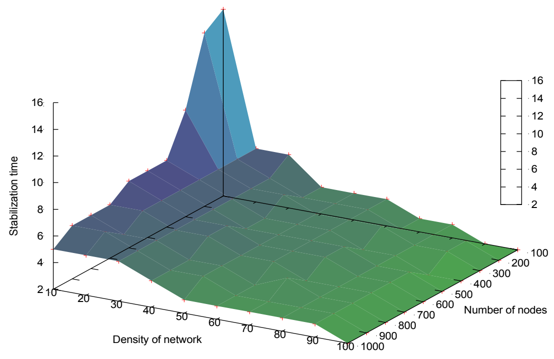


Fig. 7. Scalability

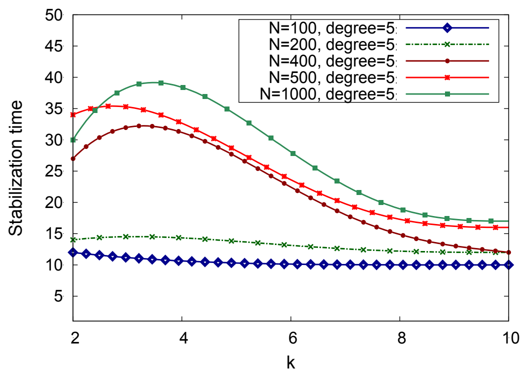


Fig. 8. Impact of k parameter

7.4 Size and Number of Clusters

As the density of connectivity is the determining factor in our algorithm, we evaluate the number of clusters obtained according to the network density. For $k = 2$, we consider a network size of 100, 500 and 1000 nodes. We vary the node degree from 5 to 100 neighbours. **Fig. 9** shows that regardless the number of nodes in network, we get less clusters when the number of neighbours increases. In fact, in denser networks, nodes with the largest identity absorb more nodes into their clusters.

As we have more clusters with low density, we consider a network size of 1000 nodes with 5 neighbours for each node. We evaluate nodes distribution between clusters. We note that, as illustrated in **Fig. 10**, we have clusters of variable size. We obtain 39 singleton clusters representing around 4% of the total number of nodes. We also note that the highest identity clusters include more nodes. The main reason is due to the fact that nodes choose as *CHs* those with the highest identity.

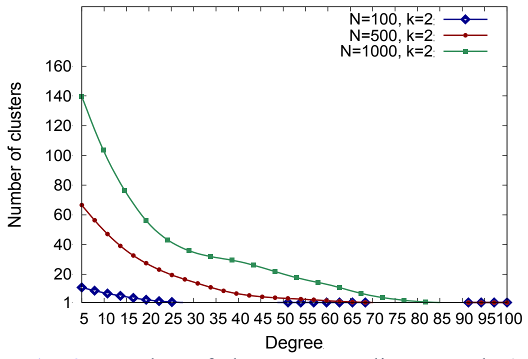


Fig. 9. Number of clusters according to nodes' degree

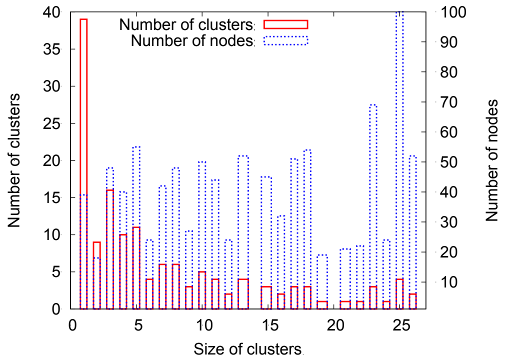


Fig. 10. Size and number of clusters

8. Study Case in Context of WSNs: Evaluation Study of Cluster-head Election Criteria

We have proposed a generic clustering approach for wireless Ad Hoc networks. The latter can be easily applied for Wireless Sensor Networks (WSNs). Our clustering algorithm is an effective scheme in increasing the scalability and lifetime of wireless sensor networks. This protocol optimizes energy consumption and prolongs the network lifetime by minimizing the number of messages involved in the construction of clusters and by minimizing stabilization time. Our clustering protocol is generic and can be easily used for constructing clusters according to multiple criteria in the election of cluster-heads, such as: nodes' identity, residual energy or degree. We propose to validate our approach in context of WSNs under the different election metrics by evaluating its communication cost in terms of exchanged messages and energy consumption. In the following, we assume that, the ID criterion is supposed unique and is integer type as defined in Section **4.1**. The degree criterion is an integer type (≥ 1) and represents le total number of neighbours as defined in following Section **8.2**. Energy criterion (noted $E_r, Risdual\ Energy$) is a double type such that $E_i \geq E_r > 0.0$ and represents the remaining battery power level. Note that E_i represents the initial energy of sensor nodes as defined in **Table 4**.
In order to implement our clustering approach in a realistic way, we use standard models for representing both the energy consumption and the network structure.

8.1 Energetic Model

To model the energy consumption for a node when it sends/receives a message, we use the first order radio model proposed in [28] and used in many other studies like [9, 29]. A sensor node consumes E_{Tx} amount of energy to transmit one l-bits message over a distance d (in meters). As shown in equation (1), when the distance is higher than a certain threshold d_0, a node consumes more energy according to a different energetic consumption model.

$$E_{Tx}(l,d) = \begin{cases} l * E_{elec} + l * \varepsilon_{fs} * d^2, & if\ d < d_0 \\ l * E_{elec} + l * \varepsilon_{mp} * d^4, & if\ d \geq d_0 \end{cases} \qquad (1)$$

Each sensor node will consume E_{Rx} amount energy when receiving a message, as shown in equation (2).

$$E_{Rx}(l) = l * E_{elec} \qquad (2)$$

The values of the parameters used in equations (1) and (2) to model energy are summarized in **Table 3**:

Table 3. Radio modelling parameters

Parameters	Definition	Value
E_{elec}	Energy dissipation rate to run radio	50 nJ/bit
ε_{fs}	Free space model of transmitter amplifier	$10 pj/bit/m^2$
ε_{mp}	Multi-path model of transmitter amplifier	$0{,}0013\ pj/bit/m^4$
d_0	Distance threshold	$\sqrt{\varepsilon_{fs} \div \varepsilon_{mp}}\ m$

8.2 Network Model

We consider WSNs as networks represented by an arbitrary random graph based on Erdos-Renyi model [30] with probability p = 0,1 for all network sizes. Our system can be modelled by an undirected graph $G = (V, E)$. V *is* the set of network nodes and E represents all existing connections between nodes. An edge exists if and only if the distance between two nodes is less or equal than a fixed radius $r \leq d_0$. This r represents the radio transmission range, which depends on wireless channel characteristics including transmission power. Accordingly, the neighbourhood of a node u is defined by the set of nodes that are inside a circle with centred at u and radius r and it is denoted by $N_r(u) = N_u = \{\forall v \in V \backslash \{u\}, dist_{(u,v)} \leq r\}$. The degree of a node u in G is the number of edges which are connected to u, and it is equal $\deg_u = |N_r(u)|$.

8.3 Framework Validation

We have also used *ONMeT++* [23] simulator to compare the performance in context of WSNs under the different election metrics by evaluating its communication cost in terms of messages, stabilization time, energy consumption and number of clusters. For generating random graphs, we have also used the SNAP [24] library. All simulations were carried out using *Grid'5000* [25] platform. Simulations parameters are summarized in **Table 4**.

Table 4. Simulations parameters

Parameters	Values
Messages size	2000 bits
Distance between two nodes	100 meters
Initial energy E_i	{1,2,3} Joules
Ideal degree	{5,20,40}
Energy threshold	{0.1, 0.01} %
Number of nodes	[100,1000]
Random graph model	Erdos-Renyi
Network density	0.1
Number of runs for each simulation result	100

8.4 Simulations Results

8.4.1 Communication Cost (Exchanged Messages)

In order to evaluate the validity of our clustering approach in the context of WSNs with energy constraint, we first measure the necessary cost in terms of exchanged messages to achieve the clustering procedure.

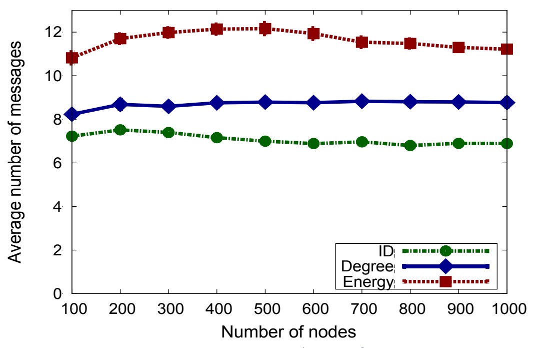


Fig. 11. Average number of messages

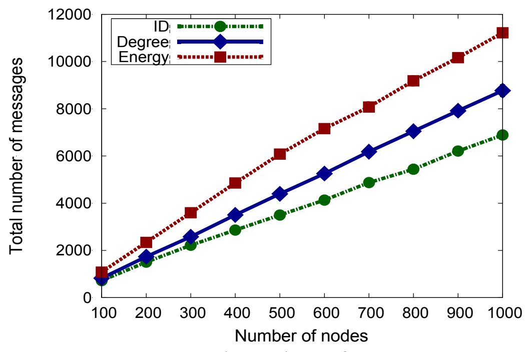


Fig. 12. Total number of messages

Based on the same network topology, the clustering based on the criterion of ID generates fewer messages as shown in **Fig. 11** and **Fig. 12**. The main reason is that the ID criterion brings greater stability during the clustering phase. In addition, the ID criterion is simpler and deterministic compared to degree or energy criteria. Indeed, for the degree criterion, it is necessary for nodes to receive a message from their neighbours in order to calculate their degree. Then, the degree is broadcasted and the clustering phase begins. This is expensive in terms of messages. Also, the energy criterion ration generates more messages compared to ID and degree criteria. As energy is a parameter that decreases during the clustering phase, it provides less stability and requires more messages to reach a stable state in the entire network.

8.4.1 Energy Consumption

We have also measured the energy consumption required for building clusters in the entire network. As illustrated in **Fig. 13** and **Fig. 14**, the ID criterion consumes less energy during the clustering phase. Indeed, as illustrated in **Fig. 11** and **Fig. 12**, both degree and energy criteria generate more messages than ID criterion during the clusters formation. However, in sensor networks communications are the major source of energy consumption.

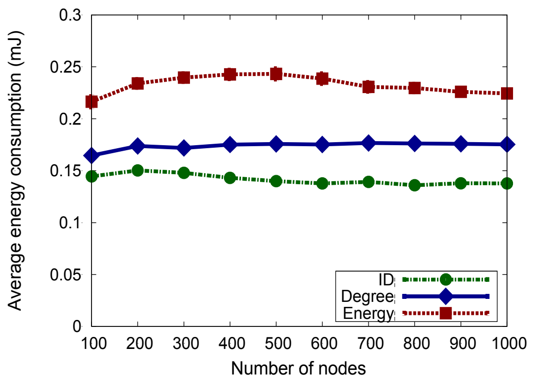


Fig. 13. Average energy consumption

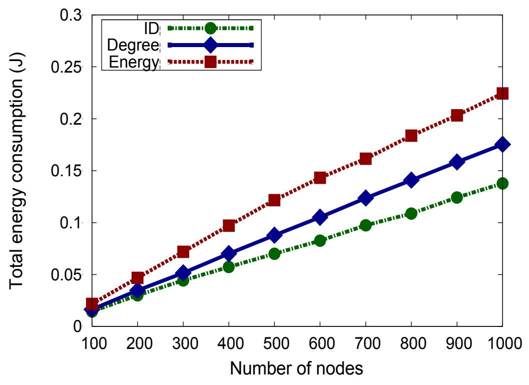


Fig. 14. Total energy consumption

8.4.2 Impact of Highest and Ideal Degree

The highest degree limits delay and communications. Nevertheless a CH handles up to a certain number of nodes in its cluster. To master the number of nodes managed by cluster-head, one solution is to setup an Ideal degree ρ. Thus, a cluster-head is the node minimizing its distance to this ideal degree ρ ($\Delta_d = |\deg_u - \rho|$). To evaluate the impact of highest and ideal degree, we fix Δ_d to 5, 20 and 40 and then we evaluate energy consumption required for building clusters in the entire network. We observe a slight increase in the energy consumption for ideal degree 5, 20 and 40 as illustrated in **Fig. 15**. However, the ideal degree has the advantage to allow parameterizing the number of nodes managed by a cluster-head.

8.4.3 Impact of Residual Energy or Energy Threshold

High level of residual energy on CHs prolongs network lifetime because CHs need more important battery than other nodes. However, residual energy level evolves through time and can lead to frequently changes of CH candidates for a negligible energy difference. To avoid this, we use an energy threshold to limit abrupt changes of nodes when the energy of CHs decreases substantially. To evaluate the impact of residual energy or energy threshold metrics, we set the energy threshold to 0.1% and 0.01% and we measure energy consumption. **Fig. 16** shows that the energy threshold criterion reduces energy consumption during the clustering phase. Indeed, nodes no longer change after slight decrease of their CHs energy. This entails fewer messages exchanges and less energy consumption.

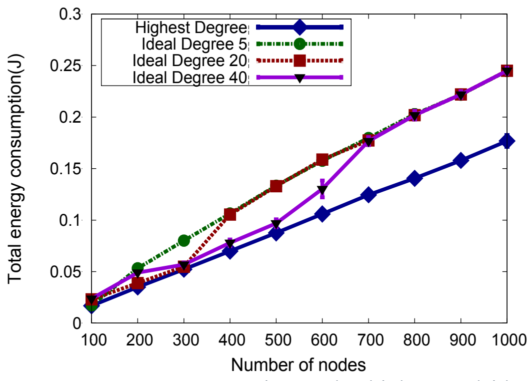


Fig. 15. Energy consumption under highest and ideal, degree metrics

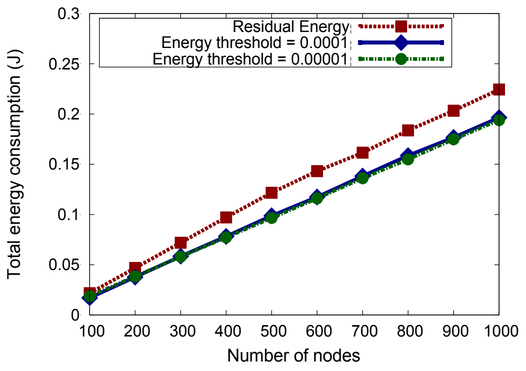


Fig. 16. Energy consumption under residual and energy threshold metrics

9. Conclusion

In this paper, we have proposed a generic Energy-Aware Efficient Self-Stabilizing Distributed Clustering protocol for Ad Hoc networks. Our algorithm structures the network into non-overlapping clusters with a diameter at most equal to $2k$. This does not require any initialization. Furthermore, it is based only on information from neighbouring nodes at distance 1. Contrary to the other clustering algorithms, we have used a unique type of messages to discover the neighbourhood of a node at distance 1 and to structure the network into non-overlapping *k-hops* clusters.
On one hand, we proved that, starting from an arbitrary configuration, the network converges to a legal configuration after at most in $n + 2$ transitions and requires at most $n * \log(2n + k + 3)$ memory space. Experimental results show that for arbitrary topology networks, the stabilization time is far below the worst-case scenario.
On the other hand, we proposed a study case in the context of Wireless Sensor Networks (WSNs) with energy constraint. Our solution can be easily used for constructing clusters according to multiple criteria in the election of cluster-heads such as: nodes' identity, residual energy, and degree. We have proposed to validate our approach by evaluating its communication cost and energy consumption. We show that our solution optimizes energy

consumption and thus prolongs the network lifetime by minimizing the number of messages involved during the clustering phase. It also offers an optimized structure for routing. Experimental results show that in terms of number of messages and energy consumption, it is better to use the Highest-ID metric for electing CHs.
As future work, we plan to implement mechanisms to balance clusters, maintaining the formed ones in case of topology changes. We are also working on the proposal of a routing process based on our clustering approach.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, 38 (4), pp. 393 – 422, 2002, Article (CrossRef Link)
[2] G. J. Pottie, W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, 43 (5), pp. 51–58, 2000. Article (CrossRef Link)
[3] I. Lavallée, "Self-stabilizing distributed spanning tree and leader election algorithm," *International Journal of Computer and Information Science*, pp.119–125, 2000.
[4] H. Baala, O. Flauzac, J. Gaber, M. Bui, T. El-Ghazawi, "A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks," *Journal of Parallel and Distributed Computing*, pp. 97 – 104, 2003. Article (CrossRef Link)
[5] L. Blin, M. G. Potop-Butucaru, S. Rovedakis, "Self-stabilizing minimum degree spanning tree within one from the optimal degree," *Journal of Parallel and Distributed Computing*, pp. 438 – 449, 2011. Article (CrossRef Link)
[6] C. Johnen, L. Nguyen, "Self-stabilizing weight-based clustering algorithm for ad hoc sensor networks," *Algorithmic Aspects of Wireless Sensor Networks*, pp. 83–94, 2006. Article (CrossRef Link)
[7] A. Datta, L. Larmore, P. Vemula, "Self-stabilizing leader election in optimal space," *Stabilization, Safety, and Security of Distributed Systems*, pp. 109–123, 2008. Article (CrossRef Link)
[8] O. Dagdeviren, K. Erciyes, "A hierarchical leader election protocol for mobile ad hoc networks," in *Proc. of the 8th international conference on Computational Science*, pp. 509–518, 2008. Article (CrossRef Link)
[9] J. Yu, Y. Qi, G. Wang, X. Gu, "A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution," *AEU - International Journal of Electronics and Communications*, pp. 54 – 61, 2012. Article (CrossRef Link)
[10] O. Younis, S. Fahmy, "Heed: a Hybrid, Energy-Efficient, Distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, pp. 366 – 379, 2004. Article (CrossRef Link)
[11] C. Johnen, L. Nguyen, Self-stabilizing weight-based clustering algorithm for ad hoc sensor networks, in: Algorithmic Aspects of Wireless Sensor Networks, 2006, pp. 83–94. Article (CrossRef Link)
[12] N. Mitton, A. Busson, E. Fleury, "Self-organization in large scale ad hoc networks," in *Proc. of the Mediterranean ad hoc Networking Workshop*, 2004.
[13] O. Flauzac, B. S. Haggar, F. Nolot, "Self-stabilizing clustering algorithm for ad hoc networks," in *Proc. of the International Conference on Wireless and Mobile Communications*, pp. 24–29, 2009. Article (CrossRef Link)
[14] C. Johnen, L. Nguyen, "Self-stabilizing construction of bounded size clusters," in *Proc. of the International Symposium on Parallel and Distributed Processing with Applications*, pp. 43–50, 2008. Article (CrossRef Link)
[15] C. Johnen, L. H. Nguyen, "Robust self-stabilizing weight-based clustering algorithm," *Theoretical Computer Science*, pp. 581 – 594, 2009. Article (CrossRef Link)
[16] N. Mitton, E. Fleury, I. Guerin Lassous, S. Tixeuil, "Self-stabilization in self-organized multihop wireless networks," in *Proc. of the 2nd International Workshop on Wireless Ad Hoc Networking*, pp. 909–915, 2005. Article (CrossRef Link)
[17] E. Caron, A. K. Datta, B. Depardon, L. L. Larmore, "A self-stabilizing k-clustering algorithm for weighted graphs," *Journal of Parallel and Distributed Computing*, 1159–1173, 2010. Article (CrossRef Link)
[18] A. K. Datta, S. Devismes, L. L. Larmore, "A self-stabilizing O(n)-round k-clustering algorithm," in *Proc. of the 28th IEEE International Symposium on Reliable Distributed Systems*, pp. 147–155, 2009. Article (CrossRef Link)
[19] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Commun. ACM*, pp. 643–644, 1974. Article (CrossRef Link)
[20] A. Amis, R. Prakash, T. Vuong, D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *Proc. of the INFOCOM*, pp. 32–41, 2000. Article (CrossRef Link)
[21] H. Attiya, J. Welch, "Distributed computing: fundamentals, simulations, and advanced topics," *Wiley series on Parallel and Distributed Computing*, 2004.
[22] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, 2000.

[23] M. Ba, O. Flauzac, B. S. Haggar, F. Nolot, I. Niang, "Self-stabilizing k-hops clustering algorithm for wireless ad hoc networks, " in *Proc. of 7th ACM ICUIMC (IMCOM)*, pp. 38:1–38:10 2013. Article (CrossRef Link)
[24] M. Ba, O. Flauzac, R. Makhloufi, F. Nolot, and I. Niang, "Comparison between self-stabilizing clustering algorithms in message-passing model, " in *Proc. of 9th International Conference on Autonomic and Autonomous Systems*, pp. 27–32, 2013. Article (CrossRef Link)
[25] A. Varga, R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshop*, pp. 60:1–60:10, 2008. Article (CrossRef Link)
[26] SNAP: Stanford Network Analysis Platform, http://snap.stanford.edu, 2013. Article (CrossRef Link)
[27] F. Cappello *et al.*, "Grid'5000: A large scale and highly reconfigurable experimental grid testbed," *International Journal of High Performance Computing Applications*, pp. 481–494, 2006. Article (CrossRef Link)
[28] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. of the 33rd Annual Hawaii International Conference on System Sciences*, pp. 8020 – 8030, 2000. Article (CrossRef Link)
[29] J. Wang, J.-U. Kim, L. Shu, Y. Niu, S. Lee, "A distance-based energy aware routing algorithm for wireless sensor networks," *Sensors*, pp. 9493–9511, 2010. Article (CrossRef Link)
[30] P. Erdos, A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, pp. 17–61, 1960.

Mandicou Ba is currently a Ph.D. student at CReSTIC laboratory of the University of Reims Champagne-Ardenne, France. He received his Master degree on Networking and Telecommunication from the University Cheikh Anta Diop (UCAD), Dakar, Senegal, in October 2010. His research interests include self-stabilization, clustering, energy optimization, routing protocols, aggregation, security, performance evaluation and simulation in ad hoc and wireless sensor networks. He is also ACM Student Member since December 2011.

Olivier Flauzac is actually Professor in the University of Reims hampagne-Ardenne. He obtained his PhD degree in Computer Sciences in 2000 at the University Of Technology of Compiègne. The same year, he became associate professor in the University of Reims Champagne-Ardenne. In 2005 he obtained a "Habilitation à Diriger des Recherches" then became Professor. From 2007 to 2011 he was responsible of the Master of Computer Science of the University of Reims Champagne-Ardenne. His research activities deal with (i) distributed self-adaptive structures construction and management; (ii) distributed autonomous security management and grid of security design; (iii) distributed data and services management in p2p grids environments.

Bachar Salim Haggar received his Ph.D. in computer science in 2011 from University of Reims Champagne-Ardenne. He obtained his Master and engineer degrees in computer science respectively on 2007 and 2006 from University of Reims Champagne-Ardenne and University of Tiaret (Algeria). His research interests include self-stabilization, clustering, spanning tree and routing protocol in ad hoc networks.

Florent Nolot obtained his PhD degree in Computer Sciences in 2002. He is an associate professor at the University of Reims Champagne-Ardenne, France, where he is in charge of a Master in Systems Administration and Network Security. Florent NOLOT has been involved in undergraduate and graduate courses in Computer Sciences for 10 years. He created a Cisco Networking Academy in June 2007 in his university and he is certified Cisco CCDP and CCNP. He is doing his research on ad-hoc networks and working on clustering problem. He is also developing a new security solution for the network, called grid of security. The goal of this new approach is to deploy a distributed security solution where communication between each device can be controlled in a collaborative manner without central decision.

Ibrahima Niang is currently an Associate Professor in Computer Science at the Faculty of Sciences and Technology of the Université Cheikh Anta Diop de Dakar (UCAD), Senegal. He received the Ph.D. degree in computer science in 2002 at Paris V University. His research activities concern QoS, mobility and security in wireless networks and distributed systems (P2PSIP). In recent years, he worked on research and development related to water management using biosensor networks.