Efficient Cluster-Based Self-Organization Scheme for Connectivity Control in Wireless Sensor Networks

Hervé Gokou Diédié[1], Michel Babri[2], and Boko Aka[1]
[1] Université Nangui Abrogoua, Abidjan 02 BP 801, Côte d'Ivoire
[2] Institut National Polytechnique Félix Houphouët-Boigny, Yamoussoukro BP 1093, Côte d'Ivoire
Email: {herve.diedie, michel.babri}@inphb.edu.ci; aka_b_m@yahoo.fr

Abstract —Despite progress made in recent years, cluster formation delay, load balancing, energy holes remain challenging for cluster-based topology control protocols. When, for energy efficiency purpose, one tries to address these problems simultaneously, one is confronted with latency, message overhead, and topological defects such as isolated Cluster Heads (CH), pairs of adjacent CHs etc. These unexpected outcomes are detrimental to both network capacity and lifetime. In this paper, we propose a fast cluster-based self-organization protocol that reduces time and energy wastes during cluster formation, minimizes the scope and frequency of cluster maintenance process, and mitigates energy holes in the sink's two-hop neighborhood. Simulation results show that our contribution is able to quickly construct a good quality communication topology while enhancing network lifetime.

Index Terms—Clustering, self-organization, leader election, topology control, connectivity, wireless sensor networks

I. Introduction

Wireless sensors are small size resources-constrained electronic machines that are capable of measuring physical values in their environment [1]. To collect information about a hostile region of interest, one has to randomly deploy (generally by air) an important number of such devices. The outcome is a dense and chaotic collection of nodes with redundant links [2]. In such a network, interference but also noise and obstacles have a negative impact on the quality of communications [3]. As a result, large amounts of packets are lost and retransmitted. Hence, energy wastes that shorten the lifetime of sensor nodes [4], [5]. It is therefore necessary to provide these nodes with some energy efficient self-organization abilities. Topology control is one of the techniques used for this purpose. It aims at optimizing the topological properties of the graph induced by the network such as nodes degree, area coverage degree etc. [6]. In this paper, we focus on nodes connectivity control whose goal is to construct and maintain an energy efficient communication infrastructure for network lifetime maximization [7], [8].Two types of connectivity control methods can be found in the literature, namely flat and hierarchical methods [7]. The latter are the most scalable, especially when based on distributed, localized and emergent strategies [9]. *Clustering* is one of these hierarchical techniques; it consists in grouping nodes in virtual sets (*clusters*), each led by a special node referred to as the Cluster Head (CH). This node is responsible for data aggregation and forwarding [7]. This helps reduce both the number and the size of packets sent to the sink.

In recent years, progress has been made to improve cluster formation metrics and strategies. Nevertheless, design issues such as cluster formation time, load balancing, and energy holes (especially around the sink) remain challenging. Yet, these problems need to be simultaneously addressed to significantly increase network lifetime. However, techniques that are commonly used to this end tend to enhance delay, message overhead and even introduce some topological defects (isolated CHs, pairs of adjacent CHs...). These unexpected outcomes contribute to sensor nodes energy quick depletion and network lifetime shortage.

In this paper, we propose a self-organization scheme to quickly construct and maintain good communication architecture while preserving network connectivity. These performances were achieved, by reducing the message overhead, waiting times required for CH election, and cluster maintenance frequency and scope. This helps avoid the common topological defects aforementioned. Furthermore, we mitigated energy holes by using a technique called *Sink-As-Cluster-Head* in the vicinity of the sink. Unlike Jain and Reddy [10], we implemented this strategy in its two-hop neighborhood, without any additional overhead. The resulting protocol outperforms some recent solutions in terms of speed, topological quality, and energy efficiency.

The remainder of this paper is organized as follows. Section II reviews relevant related work. Then after presenting motivation and the main objectives in section III, the proposed connectivity control protocol is detailed in section IV. Simulation results are shown and discussed in section V, and we finally conclude in section VI, with some perspectives.

II. Related Work

In this section, we discuss some relevant solutions for connectivity control proposed in the literature. But first it is necessary to specify the problem to solve.

A. Problem Statement

Connectivity control is a technique that belongs to topology control [8]. Its goal is to construct and maintain

Manuscript received February 16, 2016; revised July 19, 2016.
Corresponding author email: herve.diedie@inphb.edu.ci.
doi:10.12720/jcm.11.7.632-643

an efficient communication topology while preserving the connectivity degree required by the underlying application. Existing strategies are classified into flat and hierarchical methods. The former include essentially transmission power control and communication unit activity scheduling techniques; while the latter is aimed at the construction of connected dominating sets [7]. Usually considered more scalable, and easier to maintain, [8] hierarchical methods have attracted researchers' attention for several years. In this paper, we mainly focus on *clustering*, one of the techniques that provide ample opportunity for both energy losses reduction and scalability.

Clustering consists in grouping nodes into virtual subsets based on certain criteria [11]. More formally, let $G=(V, E)$ be the graph induced by the current physical communication topology of the network. V and E respectively denote the set of nodes and the set of links between nodes. Dividing G into k subsets ($k>1$) means transforming V into a family denoted by P_k : $P_k = \{V_i, \cdots, V_k\}$, such as $\forall V_i \in P_k$ we have $V_i \neq \varnothing$, $V_i \subset V$ and $V = \bigcup_i^k V_i$ and $\forall (V_i, V_j) \in P_k^2$, $i \neq j \Rightarrow V_i \cap V_j \neq \varnothing$.

These disjoint subsets are referred to as *Clusters*. Although non-disjoint subsets (overlapping clusters) are allowed, it is desirable in this type of network that each node is affiliated to only one cluster. This helps reduce contention for medium access [12]. The main objective is to minimize the number of clusters while maintaining nodes connectivity [12]. This problem is referred to as the graph partition problem, a well-known NP-hard problem in graph theory [13].

Furthermore, each cluster member must be under the responsibility of a master node, the Cluster Head (CH). The latter is chosen based on a combination of criteria that depends on the underlying application's needs. Hence, clustering raises another problem namely, the leader election problem; a well-known problem in distributed systems [14]. In addition to the above-mentioned objectives, it should also be ensured to have neither adjacent pair of CHs nor isolated ones while minimizing the number of clusters [15].

Since the purpose of many clustering protocols for WSNs is to construct a backbone to the sink [16], [17], clustering can also be formulated as a minimum connected dominating set problem. The latter is also a well-known NP-hard problem in graph theory.

B. Cluster-Based Connectictivity Control in WSNs

Cluster-based connectivity control protocols that exist in the literature are approximate solutions, since as mentioned above, clustering involves solving NP-hard sub-problems [7]. In this paper, we focus on solutions that exhibit features of self-organization; namely, the distributed or the localized ones. These approaches can be classified according to criteria such as objectives of the underlying application, CH election mode, intra or inter-cluster topology, CH service time duration etc. [7], [8], [17].

LEACH protocol by Heinzelman *et al.* [18] is certainly the most famous probabilistic cluster-based solution. Indeed, for CH election, authors use a method where each node calculates its probability of being a CH, knowing that it has not served as such during the last *1/p* rounds, where *p* is the desired percentage of CH. The goal is to balance the load among nodes that belong to the same cluster. A TDMA-based intra-cluster communication model is also used to prevent energy losses due to collisions and interferences. Although sensor nodes' energy consumption is well balanced by this random rotation, LEACH is not really scalable for WSNs since this protocol allows the creation of clusters only located near the sink. LEACH-like solutions that abound in the literature try to overcome these shortcomings [17].

HEED protocol by Younis and Fahmy [19] is another major probabilistic cluster-based solution. Unlike those of LEACH, the authors of HEED adopt a multi-hop approach to explicitly address the scalability problem. They also consider nodes residual energy for CH election to increase cluster stability. However, similar to LEACH, HEED message overhead remains high. HEED also depends on a predetermined number of CH to elect; which helps the presence of unaffiliated nodes, adjacent CHs or isolated CHs in the final topology. Many solutions have been proposed to cope with the limitations of HEED, among the most recent ones is BEE(M) protocol by Xu *et al.* [20]; these authors propose to add node density (node degree out of number of nodes) to the probabilistic metrics used for CH election. BEE (M) can reduce the number of iterations for CH election. However, BEE (M) does not scale and is difficult to implement in failure-prone networks.

Solutions that use deterministic criteria also abound in the literature [17]. Some of them consider only one metric such as nodes residual energy or CH service time estimation; while others consist in combining different criteria and weighting them according to the needs of the underlying application. WCA protocol by Chatterjee *et al.* [21] is one of the earliest weighted-based clustering methods. Its shortcomings have motivated several contributions in recent years [17].

Regardless of the approach used, and despite the significant progress made, some challenging issues remain; namely:

- CH election process message overhead (energy efficiency) and time complexity (self-organization latency);
- Cluster maintenance frequency(load balance) and scope (stability);
- Energy Holes, especially in the vicinity of the sink;

To speed up CH election process and overcome self-organization latency, some authors investigate to reduce the number of messages (declaration of candidacy, CH self-proclamation…). This strategy is utilized in both EAP protocol by Liu *et al.* [22] and LLEAP protocol by

El-Basioni *et al.* [23]. Indeed, these authors combine average residual energy and election duration to calculate the time to wait before any victory claim. Only nodes that have not yet received such a message will broadcast their victory claim at the end of their waiting time. However, error-prone or useless information collection process is still needed before election starts. For example, any neighbor's residual energy is inevitably overrated since the amount of energy lost when sending this information is not considered. Moreover, in these protocols, topological defects such as adjacent CHs are favored by the CH tree construction phase which follows the cluster formation process. Another approach consists in the reduction of the number of candidates by either defining a competition range below the communication range or using a random self-proclamation process. The former technique is implemented in several protocols like ACCA by Afsar *et al.* [24]. As for random self-proclamation scheme, it allows some nodes to act as temporary CHs which will be electing later on the permanent ones. SNDC protocol by Chiti *et al.* [25] is a shining example of that scheme. However, these number of candidates reduction techniques require strict nodes synchronization and long waiting times which increase self-organization latency. These techniques are also unable to eliminate some useless information gathering procedures at the beginning of the CH election. As mentioned above, these time-consuming procedures could lead to a large number of unaffiliated nodes or isolated CHs.

To reduce clusters maintenance frequency and scope, some authors propose the use of a vice-CH to avoid election process overhead. This method is applied in SNDC protocol by Chiti *et al.* [25]. Despite its simplicity and speed, the main concern with this solution is that vice-CH's *fitness value* may quickly become outdated especially in dynamic environments. Moreover, such a strategy requires each node to store additional information about the CH's deputy, which can be expensive. Finally, this approach cannot prevent either chain reaction, generally triggered during cluster maintenance or topological defects such as the pairs of adjacent CHs and the isolated CHs.

To cope with the sink's neighborhood energy holes problem, some authors propose the creation of non-uniform clusters by adapting the size of clusters to nodes distance from the sink. Among these authors, one can mention Liao *et al.* [26] and more recently, Hematkhah and Kavian [27] respectively with DBSCA and DCPVP protocols. Despite helping inter-cluster load balancing, the need of sensor-to-sink distance calculation is a detriment to speed and scalability. Another solution consists in implementing an approach called *Sink-As-Cluster-Head*. This strategy aims at helping the sink's neighbors save energy by preventing them from taking part to any CH election. As shown by Jain and Reddy [10], this solution can significantly increase network lifetime; provided it can be rapidly implemented with no additional overhead, to guarantee the connectivity of nodes located in this area.

III. Objectives

Speed, energy efficiency, and topology quality are still challenging for connectivity control protocols. Indeed, a good cluster-based connectivity control solution should lead to the rapid creation of a relatively small number of evenly distributed clusters; while ensuring good connectivity and load balancing for all nodes. Goals that many protocols identified in the literature, including those discussed above, are struggling to achieve effectively. The risk is the loss of all beneficial effects that clustering provides to the network (capacity increase, small number and size of packets...).

Hence, the specific objectives of this work are as follows:

- Self-organization latency reduction;
- Common topological defects (unaffiliated nodes, isolated CH, adjacent CHs…) elimination;
- CH election process overhead reduction;
- Sink's neighborhood energy holes risk mitigation;

IV. Contribution

In this section we detail our connectivity control protocol. But first, we make some assumptions before modeling the deployment area and the network.

A. Assumptions

We make the following assumptions:

- Each node is assumed to have a unique identifier (ID). This is a necessary condition that helps solving the leader election problem [28], [29].
- Nodes are also assumed to be equipped with omni-directional antenna and an efficient localization system.
- Network is homogeneous, i.e. sensor nodes are supposed to have the same characteristics (communication range, initial energy...).
- Nodes remain static after deployment.
- Finally, we assume that communication channel is not obstructed and messages are received in a bounded time with negligible propagation delays.

B. Deployment area Model

Nodes are randomly deployed in an area denoted by D; considered as a cube with dimension d and side denoted by l. More formally, $D = [0,l]^d$ while $l > 0$ and $d = 1,2,3$; let S denote the set of elements currently present in the deployment area (nodes and targets) and P a function that describes the position of any element in D according to duration T denoting the network lifetime. We have $P : S \times T \mapsto D$ where P *a*ssigns to each element in D at any time $t \in T$, a set of coordinates of dimension d. In the rest of this paper, we assume that $d = 2$.

C. Network Model

Let *V* denote the set of nodes in the deployment area *D*, at time $t \in T$. Let *E* be the set of relations (links) that exist between nodes. Each link $(u,v) \in V$ represents node *u* ability to reach node *v* with a communication range r_u. This ability depends on *d(u,v)* denoting the distance that separates these two nodes.

More formally, $E = \{(u,v) \in V \mid d(u,v) \le r_u\}$ while *d(u,v)* is calculated respectively from *u* and *v* coordinates at time *t* in *D*. Hence, we consider graph *G=(V,E)* induced by the network, as a spatiotemporal graph in *D*. In the rest of this paper we assume that *G* is undirected; i.e. links are symmetric.

D. Proposed Protocol Description

In this sub-section, we describe our connectivity control scheme. The resulting localized protocol is called SOCLES (Self-Organization scheme for Connectivity and Lifetime Extension in wireless Sensor networks).

1) Neighborhood analysis

We assume that an efficient neighbor discovery protocol is implemented on nodes. Each node must also keep in addition to its own identifier (ID), its Cluster Head's ID (CHID), a Boolean state variable (candidate or not candidate) and a variable for its current status in the network namely, Free Node (FN), Self-Proclaimed Cluster Head (SPCH), Cluster Head (CH), Ordinary Member (OM). Nodes are considered *free* or *unaffiliated* when they do not belong to any cluster. Only information about adjacent neighbors is recorded in a table.

Hence, each t_{discov} unit of time (given as a parameter) any node must update its table after a two-hop neighbor discovery. This process aims at helping sensor nodes check both the presence of their cluster mates and the existence of at least one CH in their two-hop neighborhood. It is noteworthy that the sink is regarded by sensor nodes as a full-fledged Cluster Head.

Therefore, on the basis of the information collected, a sensor node can make three kinds of decisions, namely, *Join the nearest CH, Trigger a new CH election process, or Keep its current status*. Respectively, when no cluster mate is present and at least one CH has been found, when no CH has been found and no neighbor is currently involved in any election process; and finally, when none of the two situations previously mentioned has been noticed.

2) Cluster head election and cluster formation

If none of its two-hop neighbors is neither a CH nor currently involved in any election process, a sensor node must trigger a new CH election process. It sends for that purpose an *ELECT* message containing an election ID. Both sender and recipients of such a message must be a candidate and calculate its fitness score using (1).

$$score = \alpha \times \left(\frac{n-1}{n}\right) + \beta \times \left(\frac{Er}{Ei}\right) + \delta \times (1 - CC) + \gamma \times (1 - \Omega) \quad (1)$$

where *Er, Ei and CC* denote respectively, the candidate's residual energy, initial energy, and clustering coefficient. As a reminder, on an undirected graph, the clustering coefficient CC of a vertex is the ratio of the number e of edges existing between its *n (n>0)* adjacent neighbors to the number of all possible edges between them.

$$CC = \frac{2 \times e}{n \times (n-1)} \quad (2)$$

$$\Omega = \frac{(\mu CH + \mu OPC)}{n} \quad (3)$$

It is worth noting that *score* $\in$ [0;1]. The goal of this metrics combination is to help the Cluster Head perform correctly its tasks. Indeed, we believe that compared with its neighbors, a CH must have lower energy consumption, greater communication abilities, and a smaller number of redundant links in its vicinity. The weight coefficients $\alpha, \beta, \delta, \gamma$ enable the underlying application to favor one specific criterion over others. However, to ensure the energy efficiency of this scheme, it is desirable to have:

$$\alpha + \beta + \delta + \gamma = 1 \ , \quad \beta > \max(\alpha, \delta, \gamma) \quad (4)$$

In most scenarios 0.2, 0.5, 0.1, 0.2 would be suitable values respectively for $\alpha, \beta, \delta,$ and γ. The parameter Ω is introduced to apply a penalty that greatly reduces a candidate's chances of becoming CH, if it is already adjacent to μCH Cluster Heads and μOPC neighbors involved in other election processes. This parameter contributes greatly to avoid having pairs of adjacent CHs in the final topology.

Hence, after computing its fitness score, a candidate sets a timer and waits until t_{win} whose value is calculated using (5); where t_{elect} (given as a parameter) denotes a CH election average duration.

$$t_{win} = t_{elect} - score \ , t_{elect} > 1 \quad (5)$$

Therefore, if after t_{win} a candidate has not received any WIN message, it changes its status to SPCH (Self-Proclaimed CH) and broadcasts to its one-hop neighbors a WIN message (containing its ID, the election ID, its score and degree) then waits until $t_{congrat}$ for CONGRAT messages; $t_{congrat}$ is calculated using (6). Where t_{reply} and Λ respectively denote the average waiting time for a reply and the maximum number of hierarchical levels in a cluster (here, Λ=3).

$$t_{congrat} = t_{reply} \times \Lambda \quad (6)$$

It is worth mentioning that election ID parameter helps a candidate recognize and accept only *valid messages*, i.e. only those sent by its opponents. Thus, if a candidate, whose timeout t_{win} has not yet expired, receives a WIN message from any opponent, it has to give up immediately the competition by changing its status to OM (Ordinary Member). It also has to record the ID,

score and degree of the sender. However, it must keep on receiving any other WIN message until t_{win}.

Therefore, a defeated candidate (OM), whose t_{win} has expired, will choose as its CH the best among all the self-proclaimed winners. Degrees and IDs are used to break ties. The defeated node then sends to the CH it has chosen a CONGRAT message, and finally broadcasts a RECRUIT message in its one-hop vicinity. This message helps recruit its unaffiliated one-hop neighbors i.e. nodes with a FN (Free Node) status. Several sensor nodes have this status in a newly deployed network.

Hence, when a Free Node receives a RECRUIT message, it has to join the sender's cluster and change its status to OM (Ordinary Member).

A Self-Proclaimed CH (SPCH) must consider any valid WIN message as a CONGRAT message, if the sender's ID is lower than its own ID; otherwise, it must change its status to OM and sets its new t_{win} timer to t_{elect}.

Similarly, a SPCH must leave the election process at the end of its $t_{congrat}$. Then it either changes its status to CH, if it has received a least one CONGRAT message, or regains its status as a Free Node (FN). The CH must ensure the effective functioning of its cluster i.e. all the members-to-sink, members-to-members and sink-to-members communication.

To prevent two nodes from being adjacent CHs, especially when they have been a candidate in two different electoral processes, the one with the lowest degree or ID must resign. Therefore, it has to modify its status to OM, choose the runner-up as the new CH and notify other members of this change. To avoid any overhead, the runner-up's ID may be piggybacked in the schedule message sent to members for the data gathering process. Although this process is frequent in many cluster-based connectivity control protocols, it is made very rare by SOCLES for two reasons. Firstly, because of the parameter Ω we introduced into the fitness score calculation (1). Secondly, because any sensor node is allowed to postpone its decision to trigger a new CH election if it realizes that at least another one is taking place ($\mu OPC \geq 1$) in its two-hop neighborhood.

At the end of its service time, a CH regains its Free Node (FN) status and starts a new affiliation process. Similarly, its old cluster mates when noticing its absence may join neighboring clusters without necessarily triggering a new election process. CH service time limitation is introduced for load balancing purpose. This duration of time is defined as a parameter.

The affiliation process is repeated until the sensor node's residual energy reaches a threshold *Ethr*. The latter is also defined as a parameter.

The state transition diagram of a sensor node during CH election and cluster formation processes is described in Fig. 1.

The rationale behind SOCLES protocol is described in Fig. 2.

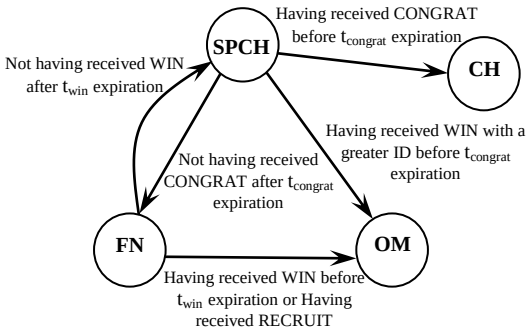


Fig. 1. The state diagram of a sensor node in the CH election phase of SOCLES

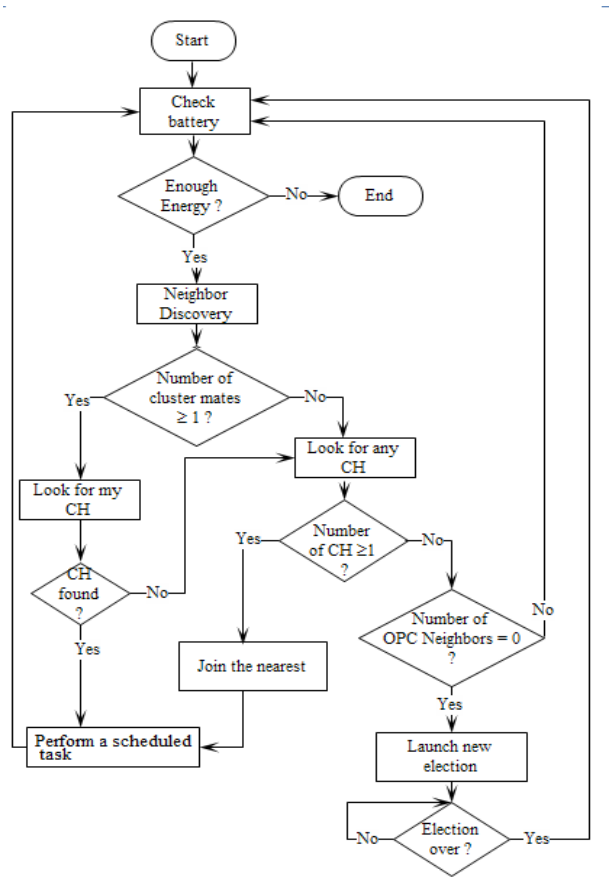


Fig. 2. Flowchart of SOCLES'node affiliation process.

Processes namely, CH election, cluster formation, neighbor discovery information analysis, message analysis, affiliation decision making are detailed respectively in Algorithm 1, 2 and 3.

Algorithm 1 Cluster_Formation ( )

Inputs: $Ei, Ethr, t_{discov}, t_{elect}, t_{reply}, \alpha, \beta, \delta, \gamma$

1: $Er \leftarrow$ Calculate residual energy
2: **while** ($Er > Ethr$) **do** ▷ residual energy enough
3:
4: **if** ($\neg candidate \wedge current\_time() = DISCOV\_timeout$) **then**
5: $N \leftarrow$ Neighborhood Discovery ▷ 2-hop discovery
6: **if** ($N \neq \varnothing$) **then**
7: Neighborhood_Analysis ( ) ▷ (see Algorithm 2)
8: **end if**

```
9:      DISCOV_timeout ← current_time() + t_discov
10:   end if
11:
12:   Messages_Analysis ( )      ▷ (see Algorithm 3)
13:
14:   ▷ timers handling
15:
16:   if ( candidate ∧ (current_time() = WIN_timeout) ) then
17:     if ( W = ∅ ) then
18:       status ← SPCH
19:       broadcast WIN ( election_id)
20:       CONGRAT_timeout ← current_time() + t_congrat
21:     else
22:       Choose in W nearest neighbor v as CH
23:       broadcast RECRUIT (election_id)
24:       send CONGRAT (election_id) to chosen CH
25:       candidate ← False
26:     end if
27:     WIN_timeout ← 0
28:   end if
29:   if ( candidate ∧ (current_time() = CONGRAT_timeout) ) then
30:     candidate ← False
31:     CONGRAT_timeout ← 0
32:     if ( D = ∅ ) then
33:       status ← FN
34:     else
35:       Organize both inter and intra-cluster communication
36:       SERVICE_timeout ← current_time() + t_service
37:     end if
38:   end if
39:   if ( current_time() = SERVICE_timeout ) then
40:     status ← FN
41:     SERVICE_timeout ← 0
42:   end if
43:
44:   Er ← Calculate residual energy
45:  end while
```

Algorithm 2 Neighborhood_Analysis ( )

```
1:    C ← {v ∈ N : CHID = CHID(v)}   ▷ List of my cluster mates
2:    L ← ∅      ▷ List of CH
3:    CH_search ← True      ▷ CH search necessary
4:    if ( status ≠ FN ∧ C ≠ ∅ ) then
5:      if ( status = CH ) then
6:        CH_search ← False  ▷ CH search no more necessary
7:      else
8:        L ← Search N for my CH
9:      end if
10:   end if
11:
12:   if ( CH_search ∧ L = ∅ )
13:     L ← Search N for any CH
14:   end if
15:
16:   OPC ← Search N for other processes candidates
17:   μCH ← |L|
18:   n ← |N|
19:   μOPC ← |OPC|
20:
21:   if ( CH_search ∧ μCH = 0 ∧ μOPC = 0 ) then
22:       status ← FN
23:       broadcast ELECT (election_id)
24:       candidate ← True
25:       W ← ∅      ▷ List of self-proclaimed winners
26:       D ← ∅      ▷ List of defeated candidates
27:       score ← Calculate my score   ▷ (see Equation 1)
28:       WIN_timeout ← current_time() + t_win  ▷ (see Equation 5)
29:   else
30:       if ( μCH ≠ 0 ) then
31:         Choose the best node v of L as my CH
32:         status ← OM
33:       end if
34:   end if
```

Algorithm 3 Messages_Analysis ( )

```
1:    receive (message) from neighbor v
2:    switch (message.type)
3:      case ELECT :
4:        if ( ¬candidate ) then
5:          status ← FN
6:          candidate ← True
7:          W ← ∅      ▷ List of self-proclaimed winners
8:          D ← ∅      ▷ List of defeated candidates
9:          score ← Calculate score      ▷ (see Equation 1)
10:         WIN_timeout ← current_time() + t_win  ▷ (see Equation 5)
11:       endif
12:     case WIN :
13:       if ( candidate ) then
14:         status ← OM
15:         W ← W ∪ {v}   ▷ Add up the sender of the message
16:       end if
17:     case RECRUIT :
18:       if ( candidate ) then
19:         Choose the CH of v
20:         status ← OM
21:       end if
22:     case CONGRAT :
23:       if ( status = SPCH ) then
24:         status ← CH      ▷ I become the new CH
25:         D ← D ∪ {v}   ▷ Add up the sender of the message
26:       end if
27:     end switch
```

To fully understand these algorithms we give some important definitions.

Definition 1 (Affiliable node). Let *N(u)* denote the set of a node *u*'s neighbors; *u* is affiliable, if $|N(u)| > 0$.

Definition 2 (Affiliated node). Let *CHID(u)* denote the ID of node *u*'s Cluster Head; *u* is affiliated, if $CHID(u) > 0$.

Definition 3 (Unaffiliated node or Free node). Let *CHID(u)* denote the ID of node *u*'s Cluster Head; *u* is unaffiliated or free, if $CHID(u) = 0$.

Definition 4 (Cluster mates set). A cluster mate of a node *u*, is a neighbor *v* that is also affiliated to *u*'s Cluster Head. Formally, let *N(u)* denote the set of node *u*'s neighbors and *C(u)* denote the set of its cluster mates. Hence, we have

$$C(u) = \{v \in N(u) \mid CHID(u) = CHID(v)\}$$

where *CHID(u)* is the ID of the node *u* 's Cluster Head.

Definition 5 (Other Process Candidate set). For a node *u*, an Other Process Candidate (OPC) is a neighbor *v* that is currently involved in a different CH election process. Formally, let *N(u)* denote the set of node *u*'s neighbors and *OPC(u)* denote the set of OPC neighbors. Hence, we have

$$OPC(u) = \{v \in N(u) \mid \text{state}(v) = \text{candidate} \wedge EID(u) \neq EID(v)\}$$

where *EID(u)* is the ID of the election process in which node *u* is currently involved.

Definition 6 (Cluster Head). Let *CHID(u) and ID(u)* respectively denote the ID of node *u*'s Cluster Head and its own ID*; u* is a Cluster Head, if *CHID(u) =ID(u)* .

Definition 7 (Isolated Cluster Head). Let *CHID(u) ID(u) and C(u)* respectively denote the ID of node *u*'s Cluster Head, its own ID and its cluster mates set *; u* is an isolated Cluster Head , if *CHID(u) =ID(u)* and $|C(u)|=0$.

V. Performance Evaluation

To validate our contribution we conducted the simulation campaign with the OMNeT++ 4.6 simulator [30]. We describe in the following sub-sections the experiments we carried out, before discussing their results. Parameters we used are presented in Table I. We also used the energy consumption model by Heinzelman *et al.* [18] described in (7)-(9). Where E_{Tx} denotes the amount of energy expended after transmitting *k* bits over a distance *d*. While E_{Rx} denotes the amount of energy expended receiving these *k* bits. Whereas E_{elec}, e_{fs}, e_{amp} denote respectively the amount of energy required to process and to transmit 1 bit over a distance superior or inferior to a threshold d_0. Results are compared with those we obtained with four related protocols recently proposed in the literature, namely:

- ACCA by Afsar *et al.* [24];
- DCPVP by Hematkhah and Kavian [27];
- EEC-SCH by Jain & Reddy [10];
- SNDC by Chiti *et al.* [25].

But we start our performance evaluation with an analysis of message and time complexity.

TABLE I: SIMULATION PARAMETERS

Parameter	Value
Deployment area	1000 m X 1000 m
Number of sensors	100 ~ 1000
Sink position	(450,200)
Sink/Sensor radio range	250 m /109 m
Initial Energy	0.2 J
E_{elec}	50 nJ/bit
e_{fs}	10 nJ/bit/m ²
e_{amp}	0,0013 pJ/bit/m^4
d_0	87 m
CH service time	0.5 s

$$E_{Tx}(k,d) = \begin{cases} k \times (E_{elec} + e_{fs} \times d^2), d < d_0 \\ k \times (E_{elec} + e_{amp} \times d^4), d \geq d_0 \end{cases} \quad (7)$$

with $$d_0 = \sqrt{\frac{e_{fs}}{e_{amp}}} \quad (8)$$

$$E_{Rx}(k) = k \times E_{elec} \quad (9)$$

A. Analytic Performance Evaluation

In this sub-section, we analyze the time and message complexity of SOCLES.

1) Analysis of time complexity

Theorem 1. In the worst case, time complexity of SOCLES is $O(1)$.

Proof. During CH election, candidates have to wait at most until t_{win} to know the results. As shown by (5), t_{win} is constant and upper bounded by t_{elect}. Then, after sending a WIN message, a self-proclaimed CH has to wait until $t_{congrat}$. As shown by (6) this waiting time depends on both the hierarchical level of the CH in the cluster and the reply average time. Since all these waiting times are constant in the worst case, the time complexity is $O(1)$.

2) Analysis of message complexity

Theorem 2. In the worst case, message complexity of SOCLES is $O(n)$ where *n* denotes the number of sensor nodes.

Proof. During the election process only one WIN message is sent by the winner, then each defeated node *v* sends a CONGRAT message to the self-proclaimed CH and finally a RECRUIT message is broadcasted to the neighboring free nodes. Therefore, in the worst case, 2 messages are sent by each candidate during the election process. Hence, $O(n)$ messages are sent by the *n* sensor nodes.

In Table II we compare the time and message complexities of the analyzed protocols.

TABLE II: ANALYTIC PERFORMANCE COMPARISON

Protocol	Time complexity	Message complexity
ACCA	*O (n)*	*O (n)*
DCPVP	*O (n)*	*O* (n^2)
EEC-SCH	*O (n)*	*O* (n^2)
SNDC	*O (n)*	*O* (n^2)
SOCLES	*O (1)*	*O (n)*

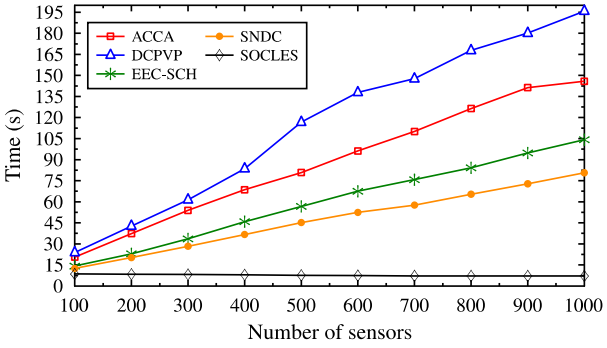


Fig. 3. Self-organization latency vs. Number of nodes

B. Simulation Setup

1) Self-organization latency

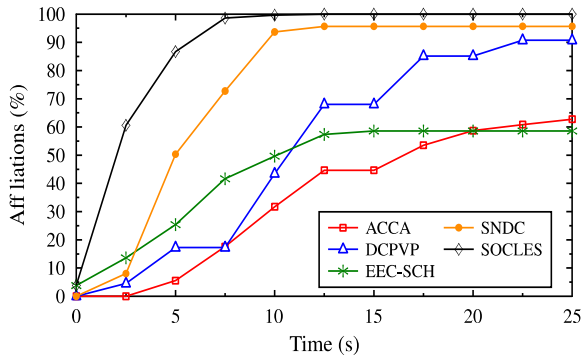
This experiment aims at evaluating the speed of SOCLES' first topology construction process i.e. nodes connectivity construction during the period of time between the end of deployment and the stabilization of nodes affiliation rate). We deploy, for that purpose, a series of networks consisting of randomly and uniformly distributed sensors while the sink's position is fixed.

To measure the impact of the number of sensors on this latency we vary with a 100 units step, the population of sensor nodes from 100 to 1000. All other parameters are presented in Table I. Experiment is repeated 100 times as the population is varied. The average times we obtained with each protocol are shown in Fig. 3 with a 95% confidence interval.

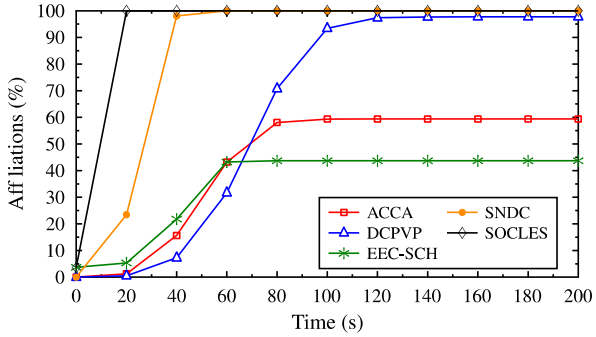
Three categories of networks can be noticed:

- The first one includes networks whose population *N* is between 100 and 400. These networks are characterized by a large number of isolated nodes (*unaffiliable* nodes) i.e. the ones whose degree equals to zero.
- The second category consists of networks whose population *N* varies from 500 to 700. These networks are characterized by a giant connected component. Here, all nodes are affiliable (i.e. their degree≥ 1).
- The third one consists of networks whose population *N* is between 800 and 1000. These networks exhibit a connected set or sometimes a strongly connected one.

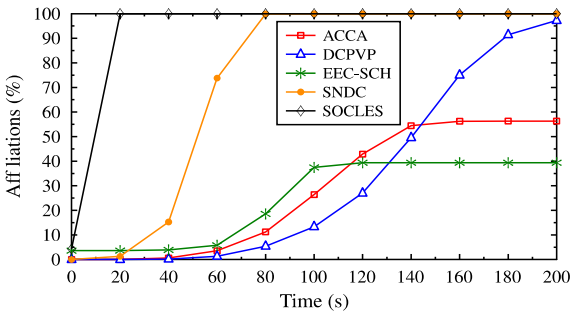
Fig. 4 (a)-(c) show the evolution of affiliation rate over time for three networks that represent each of these categories.



(a)



(b)



(c)

Fig. 4. Evolution of affiliation rate over time: (a) for networks with N=100, (b) for networks with N=500, (c) for networks with N=1000, where N denotes the number of deployed sensor nodes.

TABLE III: QUALITY OF THE FIRST TOPOLOGY (N=100).

Metrics	Protocols				
	ACCA	DCPVP	EEC-SCH	SNDC	SOCLES
Affiliable and Affiliated	59,4	86	55,6	90,6	94,8
Affiliable but Unaffiliated	36,4	8,8	37,1	0	0
Unaffiliable but Affiliated	0	0	2,1	1,2	0
Unaffiliable and Unaffiliated	5,2	5,2	5,2	4	5, 2
Number of Clusters	29,71	22 ,41	16,76	47 ,13	22,62
Cluster size	2,41	3 ,87	4,08	2,66	3,76
Isolated CH	8,2	0	15,78	16,43	0
Pairs of adjacent CH	0,98	0	0	12,37	0

TABLE IV: QUALITY OF THE FIRST TOPOLOGY (N=500).

Metrics	Protocols				
	ACCA	DCPVP	EEC-SCH	SNDC	SOCLES
Affiliable and Affiliated	305,2	486,5	218,4	499,9	500
Affiliable but Unaffiliated	194,8	13,5	281,6	0	0
Unaffiliable but Affiliated	0	0	0	0,1	0
Unaffiliable and Unaffiliated	0	0	0	0	0
Number of Clusters	54,84	27 ,57	10,73	321 ,52	39,23
Cluster size	5,54	17 ,65	20,49	2,92	19,43
Isolated CH	1,72	0	9,73	228,63	0
Pairs of adjacent CH	16,05	0,57	0	160,62	0

2) Quality of the first topology

The goal of this experiment is to evaluate the quality of the first logical organization built by SOCLES during the period of time between the end of the deployment and the stabilization of the affiliation rate. Networks are created and deployed in the same conditions described in the previous experiment. The metrics we chose are related on the one hand to the nature of nodes affiliations and on the other to the characteristics of the clusters created (number, size etc.). The experiment is repeated 100 times. The average values obtained with a 95% confidence interval are presented in Table III-Table V. These results are

those of networks that represent of each of the three categories listed in above.

TABLE V: QUALITY OF THE FIRST TOPOLOGY (N=1000).

Metrics	Protocols				
	ACCA	DCPVP	EEC-SCH	SNDC	SOCLES
Affiliable and Affiliated	563,1	972,1	409,4	1000	1000
Affiliable but Unaffiliated	436,9	27,9	590,6	0	0
Unaffiliable but Affiliated	0	0	0	0	0
Unaffiliable and Unaffiliated	0	0	0	0	0
Number of Clusters	55,01	32,37	9,7	903 ,54	43,46
Cluster size	10,26	30 ,04	36	2,36	35,09
Isolated CH	0	0	8,7	832,69	0
Pairs of adjacent CH	0	0,79	0	225,8	0

3) Energy efficiency

This experiment aims at the evaluation of the amount of energy lost mainly during construction and maintenance processes i.e. the impact of SOCLES protocol on network lifetime. Networks are created and deployed in the same conditions as described in the previous experiments.

We consider the following two types of duration as network lifetime:

- **FND (First Node Dead)**: The period of time between the end of the deployment and the death of a node.
- **ASND (All Sink's Neighbors Dead)**: The period of time between the end of the deployment and the death of the sink's last neighbor.

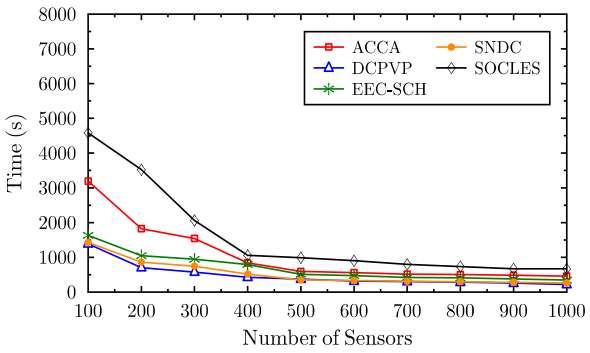


Fig. 5. Network lifetime (First Node Dead) vs. Number of nodes

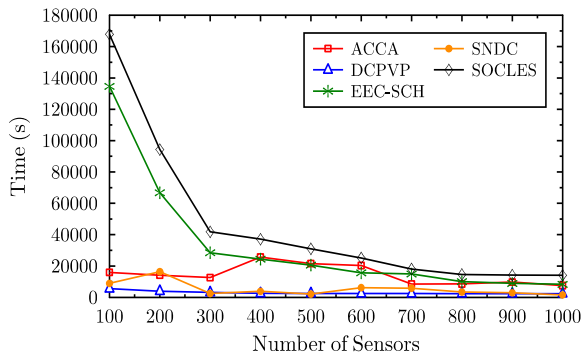


Fig. 6. Network lifetime (All Sink's Neighbors Dead) vs. Number of nodes.

These durations are considered because they can help appreciate how well both loads and sink neighborhood energy holes are respectively balanced and prevented. An energy penalty is regularly imposed to all the *affiliable-but-unaffiliated* nodes by setting their remaining energy to a value equals to the average residual energy of all *affiliable-affiliated* ones. Experiments are repeated 100 times. The results we obtained with of all the evaluated protocols are shown in Fig. 5 and Fig. 6. All these results are averaged with a 95% confidence interval.

We also considered the evolution of sensor nodes population over the ASND network lifetime. These results are shown in Fig. 7 (a)-(c) with a 95% confidence interval. Networks that are considered represent each of the three categories we identified at the beginning of the simulation campaign (see the self-organization latency sub-section above).

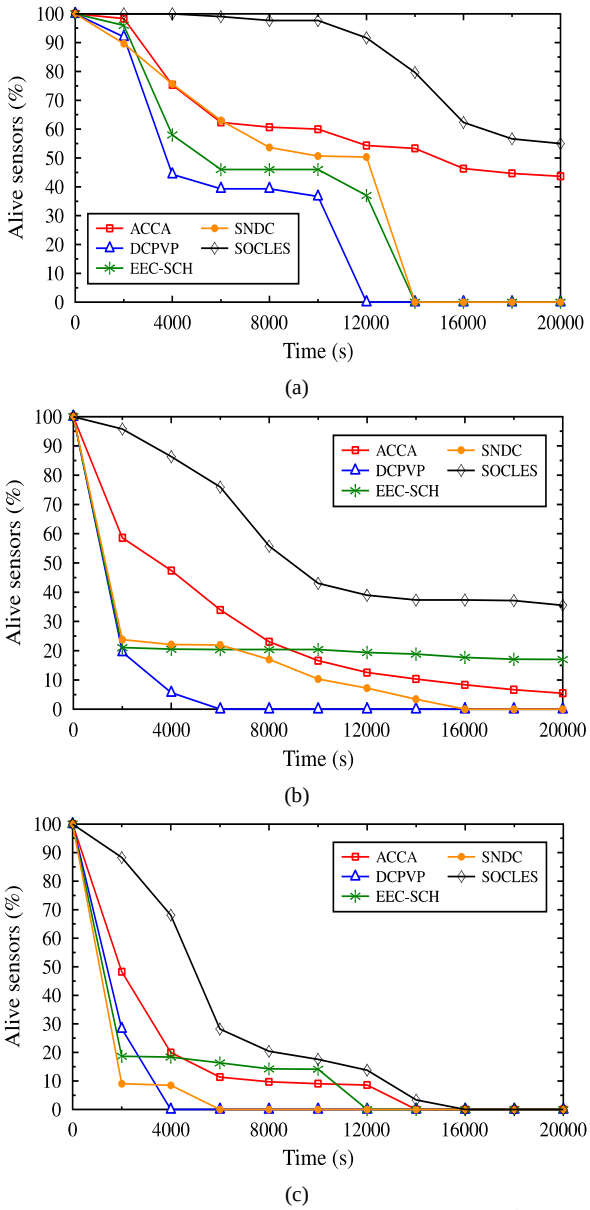


Fig. 7. Sensor nodes population evolution over network lifetime, i.e. until All Sink's Neighbors are Dead (ASND): (a) For networks with N=100, (b) For networks with N=500, (c) For networks with N=1000; where N denotes the number of deployed sensor nodes.

C. Results and Discussion

In this sub-section, we analyze and interpret the results of experiments we described above.

1) Self-organization latency

The results shown in Fig. 3 suggest that the self-organization latency of the SOCLES protocol is the lowest (7.5s on average) and remains constant. For the other evaluated protocols, this latency increases as the number of sensors grows. In other words, SOCLES scales better than these protocols. It is worth noting that this result corroborates the worst case time complexity analysis we carried out. The reason is that, unlike SOCLES' nodes waiting times (t_{win} and $t_{congrat}$); those used by the other evaluated protocols depend either on the degree of a node or on the number of competitors (ACCA). In SOCLES, the waiting time t_{win} depends on candidates' fitness score while $t_{congrat}$ is proportional to nodes' hierarchical level, which is constant by definition.

Moreover, in our SOCLES protocol, CH election process is triggered by only one message. We also avoided the time-consuming pre-election messages used by other protocols to average some metrics.

2) Quality of the first topology

This experiment aimed at the analysis of nodes' affiliations properties and rate. We also evaluated the characteristics of clusters that were created. These studies concern only the topology constructed from the end of the deployment phase until the affiliation rate stabilizes.

Fig. 4(a)-(c) show that like the other protocols we evaluated, the node affiliation rate of SOCLES increases gradually as the population of nodes grows but stabilizes faster. Table III-Table V help a fine-grained analysis of the nature of these affiliations. For instance, in Table II (100 sensors deployed, with only 94.8 affiliable nodes on average), when topology stabilizes, 100% of affiliable nodes get affiliated with SOCLES, only 95.6% (i.e. 90.6 out of 94.8) with SNDC, 90.7% with DCPVP, 62.7% with ACCA, 58.6% with EEC-SCH. Likewise, several affiliable nodes are not affiliated, especially with ACCA and EEC-SCH; while some unaffiliable nodes get affiliated with SNDC and EEC-SCH. Tables IV and V show that with most protocols we evaluated, these topological defects get worse as the number of nodes grows (N=500 and N=1000). In other words, unlike SOCLES and SNDC the 3 other evaluated protocols struggle at times to properly ensure the connectivity of all deployed nodes. This situation is due to the shortcomings of the strategies used by ACCA and DCPVP that respectively consist in limiting the number of candidates and creating non-uniform clusters. These topological defects are also due to the use of shorts waiting times like those noticed with EEC-SCH when nodes density were relatively low.

Analysis of the characteristics of clusters as illustrated in Table III-Table V, suggest that a particularly high number of clusters are created when using the SNDC protocol. This number is medium with SOCLES or ACCA and low with EEC-SCH and DCPVP. This trend continues as the size of the network grows. However, we must analyze the low values observed with EEC-SCH and DCPVP in the light of the large amounts of *affiliable-but-unaffiliated* nodes we obtained when using these protocols. Indeed, with EEC-SCH, on average, 40% of active sensors are unaffiliated. Hence the artificially small number of clusters. By contrast, the large number of clusters we obtained with SNDC is due to both its inability to create clusters with more than 3 members (on average) and its tendency to elect isolated Cluster Heads. These are the consequences of the random self-proclamation strategy used by this protocol during CH election.

Table III-Table IV also show a large number of adjacent pairs of Cluster Heads obtained with the ACCA, DCPVP and SNDC protocols. This is due to the fact that, unlike SOCLES, these protocols do not explicitly use any strategy to prevent this kind of topological defect. Indeed, in our contribution, we introduced the parameter Ω into the candidate's fitness score calculation (1) mainly for that purpose.

3) Energy efficiency

Messages exchanged during CH election and cluster maintenance processes are the main reasons for the energy losses. Therefore, in our contribution, we made an effort to reduce the number, frequency and scope of CH elections. Indeed, after regaining its Free Node status, any former member of a cluster can join if possible, a new one in its vicinity. It may also postpone its decision to trigger a new election if at least one neighbor is already involved in such a process. This strategy helps prevent CH election domino effect in the entire network. Hence, the FND-type network lifetime values shown in Figure 5. For all protocols we evaluated, network lifetime decreases as network size grows. This is due to the fact that message overhead essentially depends on node degree. However, in SOCLES network lifetime decreases slower, since as mentioned above, all the useless energy consuming messages were eliminated during both cluster formation and maintenance processes.

Furthermore, the *Sink-as-Cluster-Head* strategy helped both SOCLES and EEC-SCH have the best ASND-type network lifetimes as illustrated in Figure 6. However, we implemented this strategy in the sink's two-hop neighborhood without any additional overhead. Indeed, the free nodes recruit process at the end of the CH election does not need any acknowledgment message. Therefore, loads are more balanced in this region, hence the percentage of alive nodes that decreases more slowly with SOCLES as shown in Fig. 7 (a) -(c).

VI. Conclusion

In this paper, we have tackled the clustering-based connectivity control problem. We aimed to mitigate energy holes, topology construction delays, and defects, as well as maintenance frequency and scope. For that

purpose, we proposed a localized protocol called SOCLES (Self-Organization Scheme for Connectivity and Lifetime Extension in wireless Sensor networks). We used a combination of metrics that eliminate the time and energy consuming information collection process at the beginning of the Cluster Head election. To avoid sink's neighbors' overload, our scheme selects the sink as a cluster-head and build a cluster in its 2-hop vicinity. We showed that our contribution is able to prevent common topological defects, improve self-organization latency, balance loads and increase network lifetime.

However, like any other pure cluster-based connectivity control protocol, energy savings are still relatively low. Hence, the future work we envision is adding link quality estimation and power control techniques to enhance energy efficiency. We also plan to transform the proposed scheme into a formal fault and mobility tolerant k-hop clustering protocol.

REFERENCES

[1] S. S. Iyengar, K. G. Boroojeni, and N. Balakrishnan, "Introduction to distributed sensor networks," in *Mathematical Theories of Distributed Sensor Networks*, Berlin, Springer-Verlag, 2014, ch. 1, pp. 1-12.

[2] M. Younis and K. Akkaya, "Strategies and techniques for nodes placement in wirelss sensor networks: A survey," *Adhoc Networks*, vol. 6, no. 4, pp. 621-655, June 2008.

[3] L. H. A. Correia, D. F. Macedo, A. L. Dos Santos, and J. M. S. Nogueira, "Transmission power control techniques in ad hoc networks," in *Guide to Wireless Sensor Networks*, S. Misra, I. Woungang, S. C. Misra, Berlin, Springer-Verlag, 2009, ch. 18, pp. 469-491.

[4] S. M A. Oteafy and H. S. Hassanein, "Resilience and post-deployment maintenance," in *Dynamic Wireless Sensor Networks*, Berlin, Wiley-ISTE, 2014, ch. 3, pp. 19-31.

[5] Y. Wang, M. C. Vuran, and S. Goddard "Stochastic modeling of delay, energy consumption, and lifetime," in *Art of Wireless Sensor Network*, vol. 2, H. M. Ammari, Berlin Springer-Verlag, 2014, ch. 2, pp. 324-370.

[6] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, *Wireless Sensor and Actuator Networks Technologies, Analysis and Design*, Elsevier, 2008, ch. 4, pp. 65-113.

[7] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks" in *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 1, pp. 121-144, Feb. 2013.

[8] M. Li, Z. Li, and A. V. Vasilakos "A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2538-2557, Dec. 2013.

[9] C. Prehofer and C. Bettstetter, "Self-organization in communication networks: Principles and design paradigms," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 78-85, July 2005.

[10] A. Jain and B. V. R. Reddy, "Sink as cluster head: An energy efficient clustering method for wireless sensor networks," in *Proc. International Conference on Data Mining and Intelligent Computing*, New Delhi, Sept. 2014, pp. 1-6.

[11] S. E. Schaeffer, "Survey: Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27-64, August 2007.

[12] N. Nasser and L. M. Arboleda "Clustering in wireless sensor networks: A graph theory perspective," in *Algorithms and Protocols for Wireless Sensor Networks*, NJ, Wiley, 2009, ch. 7, pp. 161-195.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, CA, USA, 1979.

[14] R. Wattenhoffer and T. Moscriba "How to structure chaos: Initializing a WSN," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, J. Wu, New York, CRC Press Auerbach Publications, 2005, ch. 21, pp. 347-368.

[15] H. Karl and A. Willig, *Network Architecture Protocols and Architectures for Wireless Sensor Networks*, Berlin, Wiley, 2005, ch. 10, pp. 275-276.

[16] H. Liu, A. Nayak, and I. Stojmenovíc, "Energy-Efficient backbones and broadcasting in sensor and actuator networks," in *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, Berlin, Wiley, 2010, ch. 2, pp. 33-74.

[17] M. M. Afsar and M. H. Tayarani-N, "Clustering in sensor networks: A literature survey," *Journal of Network and Computer Applications*, vol. 46, pp. 198-226, Nov. 2014.

[18] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660-670, Dec. 2002.

[19] O. Younis and S. Fahmy "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366-379, Oct.-Dec. 2004.

[20] L. Xu; G. M. P O'Hare, and R. Collier "A balanced energy-efficient multihop clustering scheme for wireless sensor networks," in *Proc. 7th IFIP Wireless and Mobile Networking Conference*, Vilamoura, May 2014, pp. 1-8.

[21] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: A weighted clustering algorithm for mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 193-204, April 2002.

[22] M. Liu, J. Cao, G. Chen, and X. Wang, "An energy-aware routing protocol in wireless sensor networks," *Sensors*, vol. 9, no. 1, pp. 445-462, Jan. 2009.

[23] B. M. M. El-Basioni, S. M. Abd El-kaderb, H. S. Eissab, and M. M. Zahra, "An optimized energy-aware routing protocol for wireless sensor network," *Egyptian Informatics Journal*, vol. 12, no. 2, pp. 61-72, July 2011.

[24] M. Afsar, M. H Tayarani-N, and M. Aziz, "An adaptive competition-based clustering approach for wireless sensor networks," *Telecommunication Systems*, vol. 61, no. 1, pp. 181-204, Jan. 2016.

[25] F. Chiti, R. Fantacci, R. Mastandrea, G. Rigazzi, Á. S. Sarmiento, and E. M. M. López, "A distributed clustering scheme with self-nomination: proposal and application to critical monitoring," *Wireless Networks*, vol. 21, no. 1, pp. 329-345, Jan. 2015.
[26] Y. Liao, H. Qi, and W. Li, "Load-Balanced clustering algorithm with distributed self-organization for wireless sensor networks," *Sensors Journal, IEEE*, vol. 13, no. 5, pp. 1498-1506, May 2013.
[27] H. Hematkhah and Y. S. Kavian, "DCPVP: Distributed clustering protocol using voting and priority for wireless sensor networks," *Sensors (Basel, Switzerland)*, vol. 15 no. 3, pp. 5763–5782, Dec. 2015.
[28] N. Santoro, *Design and Analysis of Distributed Algorithms*. New Jersey, Wiley, 2007, pp. 99-101.
[29] M. Raynal, *Distributed Algorithms for Message-Passing Systems*, Berlin, Springer-Verilag, 2013, pp. 77-79.
[30] OMNeT++. 4.6 Network Simulation Framework. [Online]. Available: http://www.omnetpp.org.

Hervé Gokou Diédié is currently a Ph.D. student in Computer Science at Université Niangui Abrogoua (UNA) in Abidjan, Côte d'Ivoire. He received his MS in Computer Science from the same university in 2012. His research interests include distributed algorithms, topology control and self-organization in wireless sensor networks. He is a member of both Mathematics and Computer Science Research Laboratory at Université Nangui Abrogoua and Laboratoire Abidjanais de Recherche en Informatique et Télécoms (LARIT) at Institut National Polytechnique Félix Houphouët-Boigny (INP-HB) in Côte d'Ivoire since 2010.

Michel Babri received his Ph.D. in Computer Science from Université de Clermont-Ferrand, France in 1995. Since 1987, he is a Professor and Researcher in Computer Science at Institut National Polytechnique Félix Houphouët-Boigny (INP-HB) of Yamoussoukro in Côte d'Ivoire. His research interests include operating systems and distributed systems. He is a member of Laboratoire Abidjanais de Recherche en Informatique et Télécoms (LARIT) since 2006, and currently manages this laboratory.

Boko Aka received his Ph.D. in Physics from Université de Strasbourg 1, France in 1989. Since 1990, he is a Professor and Researcher in Physics and Computer Science in the Fundamental and Applied Sciences Departement at Université Niangui Abrogoua in Abidjan, Côte d'Ivoire. His research interests include physics, electronics and computer networks. He is the deputy head of the Mathematics and Computer Science Research Laboratory at Université Nangui.Abrogoua. He is also a member of Institut de Recherche en Energie (IREN) in Abidjan.