Robust Self-Stabilizing construction of bounded size weight-based clusters

Colette Johnen[1] and Fouzi Mekhaldi[2]

[1] Université Bordeaux, LaBRI UMR 5800, F-33405 Talence, France
[2] Université Paris-Sud, LRI UMR 8623, F-91405 Orsay, France

Abstract. We propose the first robust self-stabilizing protocol building 1-hop clusters whose size is bounded, moreover the clusterhead selection is weight-based. The protocol reaches quickly (in 4 rounds) a safe configuration, where the safety property is satistfied: network nodes are partitionned into bounded clusters (clusterheads are not the most suitable nodes). During the convergence to a legitimate configuration, where more desired properties are guaranteed, the safety property is preserved, ensuring then the continuity functioning of hierarchical protocols.

Keywords: Clustering, Self-stabilization, safety property, robustness.

1 Introduction

Clustering: An Ad Hoc or sensor network consists of wireless hosts that communicate without any pre-installed infrastructure. The clustering is introduced in such networks to facilitate the network management and increase the scalability. Clustering is a hierarchical organization which consists of partitioning network nodes into groups called clusters. Each cluster has a single clusterhead, and eventually a set of ordinary nodes. Clustering is attractive hence it allows the use of hierarchical routing which reduces the amount of stored routing informations; and decreases the transmission overhead. However, for more efficiency, the hierarchical structure should be established as soon as possible, and must be maintained over time. A well maintenance ensures the continuity functioning of protocols using the hierarchical organization like hierarchical routing protocols.

Bounded clusters: If a certain zone becomes densely populated with nodes, the clusterhead might not be able to handle all the traffic generated by nodes of its cluster. In addition, the power consumption of a clusterhead is proportional to the number of nodes in its cluster; thus the lifetime of a clusterhead is inversely proportional to its cluster's size. Therefore, controlling the number of nodes in a cluster will extend its clusterhead's lifetime, and will improve the stability of the cluster. Furthermore, keeping the number of nodes in a cluster smaller than a threshold facilitates the operation of the medium access control (MAC) protocol.

Self-Stabilization: A technique for designing solutions that tolerate transient faults is the self-stabilization. A self-stabilizing system guarantees that regardless of the current system's configuration, it will converge to a legitimate configuration in a finite number of steps. This convergence implies three important

properties of self-stabilization: embedded tolerance of arbitrary transient failures, unneeded of correct initialization (as the initial state does not have to be legitimate) and obvious adaptivity to dynamic changes of the network topology: ex. mobility of nodes, or node crashes (as any of them can be treated as a transient fault). This is why the self-stabilization property is attractive.

Robustness: In the other hand, there are some disadvantages of self-stabilizing protocols. They are in particular: tolerance of only non-permanent faults, and no explicit detection of accomplishing convergence. Moreover, self-stabilizing systems do not guarantee any property during the convergence period, so the system behaves arbitrarily along the convergence to a legitimate configuration. In addition, the convergence time may be proportional to the size of the network; particularly, in weight-based clustering protocols (see Sect. 5.2). Thus, self-stabilizing weight-based clustering protocols are not scalable. In order to overcome these drawbacks, we are interested to the robust self-stabilization. The robust self-stabilization guarantees that from an illegitimate configuration and without occurrence of faults, the system reaches quickly a safe configuration, in which the safety property is satisfied. During the convergence to a legitimate configuration, the safety property is preserved. The safety property has to be defined in such a way that the system still performs correctly its task once a safe configuration is reached. The safety property for our protocol is defined in Section 4.3.
The problem studied is the clustering of a network such that the number of nodes per cluster is bounded by a pre-defined threshold. This structure is also known as capacitated dominating set (dominating set with node capacities).

Related works: We are interested to protocols building 1-hop clusters (1-dominating sets) in which ordinary nodes are neighbor of their clusterhead. [1] presents a self-stabilizing protocol that constructs a minimal dominating set under synchronous scheduler. In [2,3], self-stabilizing protocols building a connected dominating set are provided. In [4], a self-stabilizing protocol to construct a maximal independent set (MIS) is presented. In [5], a probabilistic self-stabilizing MIS protocol is presented. A self-stabilizing protocol building a minimal k-dominating set under synchronous scheduler is presented in [6]. A self-stabilizing clustering protocol is presented in [7]; the density criteria (defined in [8]) is used to select the clusterheads. In [9], a robust self-stabilizing protocol building a minimum connected dominating set is proposed. In a safe configuration, the built set is just dominating set. In [10], a robust self-stabilizing version of DMAC [11] under synchronous scheduler is presented. A robust self-stabilizing weight-based clustering protocol is proposed in [12]. It is a robust self-stabilizing version of GDMAC [13] (an extended version of DMAC). In robust self-stabilizing protocols [10,12], a configuration is safe if each node belongs to a cluster.
To our knowledge, the only protocols building bounded size clusters are [14,15,16]. In [15], the obtained clusters have a size bounded by a lower and an upper bound. This solution cannot be applied to one-hop clusters, because the degree of nodes may be less than the lower bound. [14,15] are not self-stabilizing. Although [16] is self-stabilizing, it is not robust. During the convergence period, a node may not belong to a cluster even if it belongs initially to a well-formed cluster.

Contributions: We propose the first robust self-stabilizing protocol building 1-hop clusters having bounded size, where the clusterhead selection is weight-based. Our protocol ensures the fault-tolerance, load-balancing, best choice of clusterheads, and reliability. The load balancing is achieved by building bounded size clusters. The number of nodes that a clusterhead handles is bounded by a threshold ($SizeBound$). Thus, none of clusterheads are overloaded. The fault-tolerance and the reliability are achieved by the robust self-stabilization property.

Our protocol reaches a safe configuration, in at most 4 rounds, where a minimum service is guaranteed: the network is partitioned into bounded size clusters having an effectual leader (it is not the most suitable node within the cluster). From that, the protocol converges to a legitimate configuration in at most $\frac{7*|V|}{2} + 5$ rounds ($|V|$ is the size of network). Any self-stabilizing protocol building weight-based clusters needs $O(V)$ rounds (see subsection 5.2) from some configurations and some networks. During the convergence to a legitimate configuration, the safety property stays verified. Thus, the minimal service is continuously provided. In a legitimate configuration, a best quality of service is achieved: clusterheads are the most suitable nodes in their cluster, and their number is locally minimized.

2 Model and Concepts

A distributed system S is modelled by an undirected graph $G = (V, E)$ in which, V is the set of (mobile) nodes and E is the set of edges. There is an edge $\{u, v\} \in E$, if and only if u and v can communicate between them (links are bidirectional). In this case, we say that u and v are neighbors, and we note by N_v the set of neighbors of the node v. Every node v in the network is assigned a unique identifier id.

The *state* of a node is defined by the value of its local variables. A *configuration* of the system S is an instance of the node states. The *program* of each node is a set of *rules*. Each rule has the following form: $Rule_i : Guard_i \longrightarrow Action_i$. The *guard* of a rule of a node v is a Boolean expression involving the local variables of v, and those of its neighbors. The *action* of a rule of v updates one or more variables of v. A rule can be executed only if it is *enabled*, i.e., its guard evaluates to true. A node is said to be enabled if at least one of its rules is enabled. In a *terminal configuration*, no node is enabled.

A *computation step* $c_i \rightarrow c_{i+1}$ consists of one or more enabled nodes executing a rule. A *computation* is a sequence of consecutive computation steps. We say that a computation e is maximal if it is infinite, or if it reaches a terminal configuration. A computation is *fair*, if for any node v which is continuously enabled along this computation, eventually performs an action. In this paper, we study only fair computations. We note by $\mathcal{C}$ the set of all configurations, and by $\mathcal{E}$ the set of all (fair) computations.The set of (fair) computations starting from a particular configuration $c \in \mathcal{C}$ is denoted $\mathcal{E}_c$. $\mathcal{E}_A$ is the set of computations where the initial configuration belongs to $A \subset \mathcal{C}$.

A node v is neutralized in the computation step e, $c_i \rightarrow c_{i+1}$, if v is enabled in c_i and not enabled in c_{i+1}, but v did not execute any action during e.

We use the round notion to measure the time complexity. The first round of a computation $e = c_1, ..., c_j, ...$ is the minimal prefix $e' = c_1, ..., c_j$ such that every node v enabled in c_1, either executes a rule or becomes neutralized in c_j. Let e'' be a suffix of e such that $e = e'e''$. The second round of e is the first round of e'', and so on. We use the *attractor* notion to define the self-stabilization.

Definition 1. ***(Attractor)****. Let B_1 and B_2 be subsets of configurations of $\mathcal{C}$. B_2 is an attractor from B_1, if and only if the following conditions hold:*

- ***Convergence:***
  $\forall e \in \mathcal{E}_{B_1}(e = c_1, c_2, ...), \exists i \geqslant 1 : c_i \in B_2$.
  $\forall c \in B_1$, *If* $(\mathcal{E}_c = \emptyset)$ *then* $c \in B_2$.
- ***Closure:*** $\forall e \in \mathcal{E}_{B_2}(e = c_1, ...), \forall i \geqslant 1 : c_i \in B_2$.

Definition 2. ***(Self-stabilization)****. A distributed system S is self-stabilizing if and only if there exists a non-empty set $\mathcal{L} \subseteq \mathcal{C}$, called set of legitimate configurations, such that the following conditions hold:*

- *$\mathcal{L}$ is an attractor from $\mathcal{C}$.*
- *All configurations of $\mathcal{L}$ satisfy the specification problem.*

Definition 3 (Robustness under Input Change [12]). *Let $\mathcal{SP}$ be the safety predicate, that stipulates safe configurations. Let $\mathcal{IC}$ be a set of input changes that can occur in the system. A self-stabilizing system is robust under any input changes of $\mathcal{IC}$ if and only if the set of configurations satisfying $\mathcal{SP}$ is:*

- *closed under any computation step.*
- *closed under any input changes of $\mathcal{IC}$.*

3 Weight-based bounded size clustering

The problem studied consists to build 1-hop clusters having bounded size. Both problems of finding the minimum number of 1-hop clusters (i.e., a minimal dominating set), and the minimum number of bounded clusters (i.e., a minimal capacitated dominating set) are NP-hard [17,18]. The goal of our protocol is not to give a distributed solution to these problems, but to propose a robust self-stabilizing protocol building an useful clustering. Our solution satisfies some desired properties like: a best choice of clusterheads, number of nodes per cluster is bounded, and the number of clusterheads neighbor is minimal.

Specifications: The final clusters provided by our protocol satisfy the **well-balanced clustering properties**, informally defined as follows:

- ***Affiliation condition***: each ordinary node affiliates with a neighbor clusterhead, such that the weight of the clusterhead is greater than its weight.
- ***Size condition***: each cluster contains at most $SizeBound$ ordinary nodes.
- ***Clusterheads neighbor condition***: if a clusterhead v has a neighbor clusterhead u such that $w_u > w_v$, then the size of u's cluster is $SizeBound$ (v cannot join u's cluster without violating the size condition).

As clusterheads have more tasks to ensure than ordinary nodes, then each clusterhead must be more suitable than ordinary nodes inside its cluster. This is the goal of the *Affiliation* condition. Our protocol selects clusterheads according to their weight value. Each node has an input variable, its weight, named w, representing its suitability to be clusterhead. The higher the weight of a node, the more suitable this node is for the role of clusterhead.
A significant node's weight can be obtained by a sum of different normalised parameters like: node mobility, memory and processing capacity, bandwidth, battery power, and so on. The computation of the weight value is out the scope of this paper. Nevertheless, we consider that the weight value of a node can increase or decrease, reflecting changes in the node's status. We assume that nodes weight are different (the *id* of nodes breaks the tie).

The proposed protocol provides bounded size clusters; at most $SizeBound$ ordinary nodes are in a cluster. This limitation on the number of nodes that a clusterhead handles, ensures the load balancing: no clusterhead is overloaded.

As clusters have bounded size, several clusterheads may be neighbors. To limit locally the number of clusterheads, the *clusterheads neighbor* condition is used. A node v stays clusterhead only if it cannot join any neighbor cluster: all neighbor clusters are full (it contains $SizeBound$ members), or v's weight is bigger than all neighbor clusterhead's weight. Notice that *clusterhead neighbor* condition ensures that the clusterhead set S, is minimal : there is not clustering structure satisfying the *affiliation* and *size* conditions where the clusterheads set, S' is a proper subset of S.

Let we note that, a trivial configuration in which each node of the network is a clusterhead does not satisfy the specification of the problem: it does not satisfy the *clusterheads neighbor* condition. Because, some clusterheads can become ordinary without violating the *affiliation* and *size* conditions.

4 Robust Self-Stabilizing protocol for Bounded Size Weight-based Clusters

4.1 Variables and Macros

The variables, and macros are presented in Protocol 1. Each node v has three possible status. It can be a clusterhead ($HS_v = CH$), a nearly ordinary node ($HS_v = NO$), or an ordinary node ($HS_v = O$). A node v which is clusterhead or nearly ordinary, is the leader of its cluster, and it is responsible to manage it.
To prevent the violation of the *size condition*, a node u cannot freely join a cluster: u needs the permission of its potential new clusterhead. More precisely, only nodes belonging to the set CD_v may join v's cluster.
The set N_v^+ indicates the clusterhead neighbors of v that are more suitable than v's current clusterhead. If the set N_v^+ is not empty, then v must change its clusterhead in order to choose a more suitable one: in v's neighborhood, there is a clusterhead u having a bigger weight than v and v's clusterhead, and u accepts the v's affiliation (i.e., $v \in CD_u$).

Protocol 1 : Variables and macros on node v.

Constants

$w_v \in \mathbb{R}$; Weight of node v.
$SizeBound \in \mathbb{N}$; Maximum number of ordinary nodes that may belong to a cluster.

Local variables

$HS_v \in \{CH, O, NO\}$; Hierarchical status of node v.
$Head_v \in \{IDs\}$; Identity of v's clusterhead.
$wh_v \in \mathbb{R}$; Weight of v's clusterhead.
$CD_v \subseteq \{IDs\}$; List of nodes that can join v's cluster.
$S_v \in \mathbb{N}$; Locale value about the size of v's cluster.

Macros

Size of v's cluster: $Size_v := |\{z \in N_v : Head_z = v\}|$;
v's neighbors could be clusterheads of v:
$N_v^+ := \{z \in N_v : (v \in CD_z) \wedge (HS_z = CH) \ \wedge (w_z > w_{Head_v}) \wedge (w_z > w_v)\}$;
Computation of $CD2_v$:

Begin
$CD0_v := \{z \in N_v : wh_z < w_v \wedge w_z < w_v\}$;
If $|CD0_v| \leqslant SizeBound - Size_v$ **then** $CD1_v := CD0_v$;
Else $CD1_v$ contains the $SizeBound - Size_v$ smallest members of $CD0_v$;
If $CD_v \subseteq CD1_v \cup \{z \in N_v : Head_z = v\}$ **then** $CD2_v := CD1_v$;
Else $CD2_v := \emptyset$;
End

4.2 Predicates and rules

The predicates and rules are illustrated in Protocol 2. The election rule allows a node to become clusterhead. The affiliation rule allows an ordinary or a nearly ordinary node to affiliate with an existing clusterhead. The resignation rule allows a clusterhead to become nearly ordinary node. The correction rules update if necessary the value of v's local variables: $Head_v, CD_v, S_v,$ and wh_v without changing its hierarchical status.

Election and Affiliation processes: When an ordinary node v does not satisfy the *affiliation* or *size* conditions, the predicate `Change` is satisfied. In this case, v has to change its cluster (it will affiliate with another clusterhead or it will become clusterhead). The rule executed by v depends on N_v^+ value. If $N_v^+ = \emptyset$, then no node can be clusterhead of v. So, v must become clusterhead (*Election* rule). Otherwise ($N_v^+ \neq \emptyset$), v has a neighbor that could be its new clusterhead. So, v affiliates with the best clusterhead of N_v^+ ($Affiliation$ rule).

Resignation process: A clusterhead v has to resign its status when it does not satisfy the *clusterheads neighbor* condition, i.e. $N_v^+ \neq \emptyset$. In this case, v executes the *Resignation* rule. In order to maintain the hierarchical structure over the network, the clusterhead v does not take directly the ordinary status: v takes the nearly ordinary status, and still performs correctly its task of clusterhead. Nevertheless, $HS_v = NO$ and $CD_v = \emptyset$, i.e., no node can join v's cluster. All members of v's cluster verify the predicate `Change`. So, they will quit the v's cluster. Thus, the v's cluster will eventually be empty ($Size_v = 0$). Then,

either v affiliates with an existing clusterhead ($Affiliation$ rule) if $N_v^+ \neq \emptyset$, or it becomes again a clusterhead ($Election$ rule). This mechanism guarantees that during the construction/maintenance of clusters, no clusterhead abandons its leadership. Thus, the hierarchical structure of the network is continuously available even during its reorganization.

Protocol 2 : Robust Self-Stabilizing Clustering Protocol on node v.

Predicates

/* `true` if a node has to change its clusterhead */
`Change(v)` $\equiv (Head_v \notin N_v \cup \{v\}) \vee$
$(w_v > w_{Head_v}) \vee (HS_{Head_v} \neq CH) \ \vee (S_{Head_v} > SizeBound)$

/* The guard of Election rule */
`Election-g(v)` $\equiv [(HS_v = O) \wedge (N_v^+ = \emptyset) \wedge$ `Change(v)`$] \vee [(HS_v = NO) \wedge (N_v^+ = \emptyset)]$

/* The guard of Affiliation rule */
`Affiliation-g(v)` $\equiv [(HS_v = O) \wedge (N_v^+ \neq \emptyset)] \vee [(HS_v = NO) \wedge (Size_v = 0) \wedge (N_v^+ \neq \emptyset)]$

/* The guard of resignation rule */
`Resignation-g(v)` $\equiv (HS_v = CH) \wedge (N_v^+ \neq \emptyset)$

/* The guards of Correction rules */
`Cor-guardCH(v)` $\equiv (HS_v = CH) \wedge$
$[(Head_v \neq v) \ \vee (CD_v \neq CD2_v) \ \vee (S_v \neq Size_v) \vee (wh_v \neq w_v)]$
`Cor-guardNO(v)` $\equiv (HS_v = NO) \wedge [(Head_v \neq v) \vee (CD_v \neq \emptyset) \vee (S_v \neq 0) \vee (wh_v \neq w_v)]$
`Cor-guardO(v)` $\equiv (HS_v = O) \wedge [(CD_v \neq \emptyset) \vee (S_v \neq 0) \vee (wh_v \neq w_{Head_v})]$

Rules

/* Clustering Construction rules */
Election : `Election-g(v)` $\longrightarrow$
$HS_v := CH; Head_v := v; CD_v := CD2_v; S_v := Size_v; wh_v := w_v;$

Affiliation : `Affiliation-g(v)` $\longrightarrow$
$HS_v := O; \ Head_v := max_{w_z}\{z \in N_v^+\}; \ CD_v := \emptyset; S_v := 0; \ wh_v := w_{Head_v};$

Resignation : `Resignation-g(v)` $\longrightarrow$
$HS_v := NO; \ Head_v := v; \ CD_v := \emptyset; \ S_v := 0; \ wh_v := w_v;$

/* Correction rules */
Correction-CH : $\neg$`Resignation-g(v)` $\wedge$ `Cor-guardCH(v)` $\longrightarrow$
$Head_v := v; \ CD_v := CD2_v; \ S_v := Size_v; \ wh_v := w_v;$

Correction-NO : $\neg$`Election-g(v)` $\wedge \neg$`Affiliation-g(v)` $\wedge$ `Cor-guardNO(v)` $\longrightarrow$
$Head_v := v; \ CD_v := \emptyset; S_v := 0; \ wh_v := w_v;$

Correction-O : $\neg$`Election-g(v)` $\wedge \neg$`Affiliation-g(v)` $\wedge$ `Cor-guardO(v)` $\longrightarrow$
$CD_v := \emptyset; \ S_v := 0; \ wh_v := w_{Head_v};$

4.3 Safety predicate

A safe configuration is a configuration satisfying the safety predicate $\mathcal{SP}$. The safety predicate ensures that the following properties are satisfied:
• each node belongs to one cluster having an effectual leader (no condition on leader's weight, but its status is not ordinary node);

• each cluster has less than $SizeBound$ ordinary members.

Definition 4. *The safety predicate $\mathcal{SP}$ is defined as follow:*
• $\mathcal{SP} \equiv \forall v,\ \mathcal{SP}_v = True$
• $\mathcal{SP}_v \equiv (Head_v \in N_v \cup \{v\})\ \wedge (HS_{Head_v} \neq O)\ \wedge P_s(v)$
• $P_s(v) \equiv |Cluster_v \cup CD_v| \leqslant SizeBound$
• $Cluster_v = \{z \in N_v : Head_z = v\}$. *$Cluster_v$ is the v's cluster (i.e., the set of nodes having chosen v as their clusterhead).*

Our protocol is robust under the following input changes: (i.e., the safety predicate $\mathcal{SP}$ is preserved) (i) the change of node's weight, (ii) the crash of ordinary nodes, (iii) the failure of a link between (1) a clusterhead and a nearly ordinary node, (2) two clusterheads, (3) two nearly ordinary nodes, or (4) two ordinary nodes, (iv) the joining of a sub-network that verifies the predicate $\mathcal{SP}$.

The difficulty is to preserve the size condition after any computation step. A cluster whose clusterhead v satisfies the predicate $P_s(v)$, verifies the size condition in the current configuration and after any computation step. On the contrary, a cluster whose clusterhead v does not satisfy the predicate $P_s(v)$, may not verify the size condition after the specific computation step in which all nodes of CD_v join v's cluster (this feature is illustrated in Figure 1). In the initial configuration, $Cluster_6 = \{1\}$, and $CD_6 = \{2, 3, 4\}$. Thus, the size condition is satisfied, but the $P_s(6)$ predicate is not satisfied: $|CD_6 \cup Cluster_6| = |\{1, 2, 3, 4\}| > 3$. After the computation step where all nodes of CD_6 join 6's cluster, the size condition is no more satisfied.

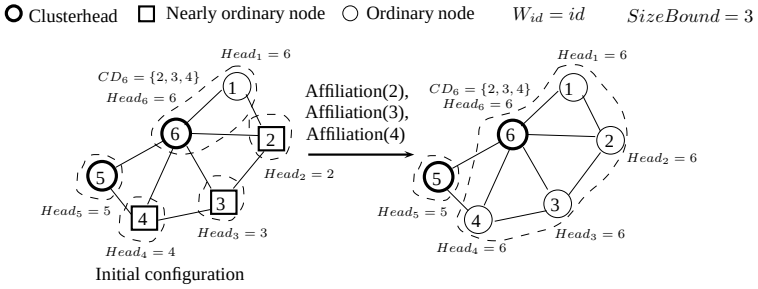

Fig. 1. Violation of the size condition from a configuration not satisfying $P_s(v)$

The variable CD_v is computed in such a way that the predicate $P_s(v)$ stays verified after any computation step. For each clusterhead v, the macro $CD2_v$ is used to compute CD_v value. $CD2_v$ is computed in 3 steps. $CD0_v$ is the set of v's neighbors that want to enter into v's cluster, i.e., their weight and their clusterhead's weight are smaller than v's weight. The size of $CD0_v$ can be greater than $SizeBound - Size_v$: $CD1_v$ is a subset of $CD0_v$, containing at most $SizeBound - Size_v$ elements. The set $CD2_v$ is a subset of $CD1_v$ ensuring that the predicate $P_s(v)$ stays verified by v after any computation step from the current configuration (assuming that $P_s(v)$ is verified in the current configuration).

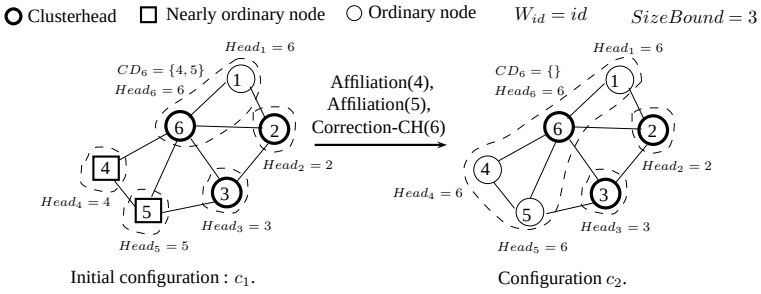

Fig. 2. Illustration of CD value computation

The Figure 2 illustrates the computation of CD_v value. In the initial configuration, there are 5 clusters satisfying the size condition, and $CD_6 = \{4, 5\}$. For simplicity, the weight of a node is its identity. Thus, the clusterhead 6 has the highest weight in its neighborhood. Nodes $2, 3, 4$ and 5 want to belong to $Cluster_6$ (the node 1 is already in $Cluster_6$); so, $CD0_6 = N_6 - Cluster_6 = \{2, 3, 4, 5\}$.
$CD1_6$ contains only two nodes, because $SizeBound = 3$ and $|Cluster_6| = 1$; so, $CD1_6 = \{2, 3\}$. In the reached configuration c_2, $CD_6 = \emptyset$ because
$CD_6(c_1) \nsubseteq \{CD1_6(c_1) \cup Cluster_6(c_1)\}$. Notice that in c_2, $P_s(6)$ is still verified: $|CD_6(c_2) \cup Cluster_6(c_2)| \leqslant SizeBound$.

5 Convergence

Proofs of convergence to a safe and legitimate configurations, and robustness are omitted due to lack of space, they can be found in [19]. A legitimate configuration is a terminal configuration in which *the well balanced clustering properties* are verified. The convergence process from an arbitrary configuration to a legitimate configuration is done in two steps. First, the system converges quickly to a safe configuration; and after that, it progresses to reach a legitimate configuration.

5.1 Illustration of the convergence process

The convergence process from an unsafe to a safe configuration, is straightforward: once an ordinary node locally detects an unsafe situation (ex: its cluster has more than $SizeBound$ members), it becomes clusterhead if it cannot join a neighbor cluster without violating the *affiliation* and *size* conditions. Along any computation, the time to reach a safe configuration is at most 4 rounds.
The convergence to a legitimate configuration is explained informally in follows (technical proofs are in [19]). Figure 3 illustrates the convergence process from an arbitrary configuration to a legitimate configuration. For simplicity in this example, the weight of a node is its identifier.
The initial configuration 3.a is not safe, because the cluster of 5 does not have a leader. Node 5 eventually performs the *Affiliation* rule to affiliate with the clusterhead 7. The reached configuration, 3.b, is safe.

From a safe configuration, the stabilization is done in phases. At the end of i^{th} phase, nodes of Set_i have their final state (have stabilized).

Notation 1 *We denote the set of safe configurations by A_3, and we denote by $V_i = V - Set_i$ the set of nodes not having their final state after the i^{th} phase.*

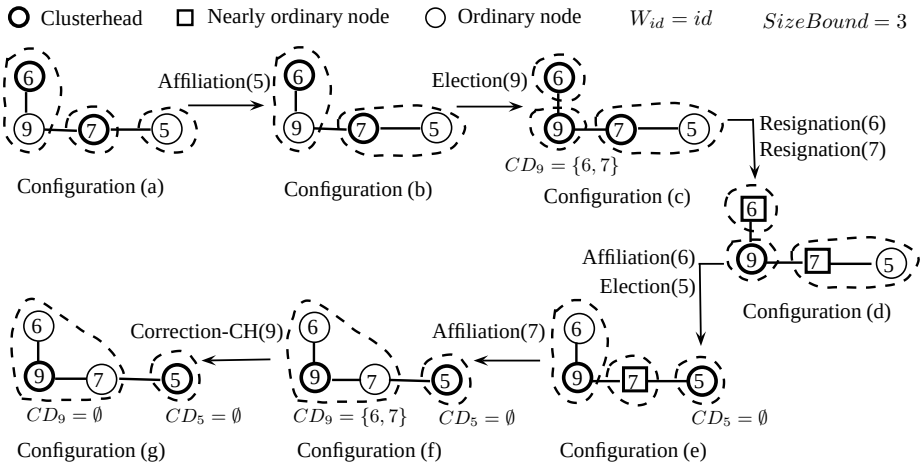

Fig. 3. Illustration of convergence to a legitimate configuration.

• Set_0 is the set of nodes having initially their terminal state (they will not do any action). Usually, $Set_0 = \emptyset$.

• We name vh_0, the node having the highest weight in $V_0 = V - Set_0$. In Figure 3, $vh_0 = 9$. The node 9 has to be clusterhead (it verifies `Election-g`). Once node 9 has performed the *Election* rule, the system reaches a configuration of $L_1 = A_3 \cap \{c \in C \mid HS_{vh_0} = CH\}$ where vh_0 will never change its status. In Figure 3, the configuration 3.c belongs to L_1. We prove, in [19], that L_1 is an attractor, and it is reached from A_3 in at most one round along any computation.

• Let $L_1^{'} = L_1 \cap \{c \in C| \ |Cluster_{vh_0}| = Min(SizeBound, |N_{vh_0} \cap V_0|)\}$ be the set of configurations where the vh_0's cluster is stable (no node will quit or join this cluster). In Figure 3, the configuration 3.f belongs to $L_1^{'}$. We prove that $L_1^{'}$ is an attractor, and along any computation, it is reached from L_1 in at most five rounds.

• In the last step of phase 1, all nodes of vh_0's cluster reach their final state (a configuration of $L_1^{''}$). In Figure 3, the configuration 3.g belongs to $L_1^{''}$, where $L_1^{''} = L_1^{'} \cap \{c \in \mathcal{C}| \ \forall \ v \in Set_0 \cup \{vh_0\} \cup \{Cluster_{vh_0}\}, CD_v = \emptyset \ \}$; We prove that $L_1^{''}$ is an attractor, and it is reached from $L_1^{'}$ in at most one round.

At the end of first phase, the set of nodes having their final state is $Set_1 = Set_0 \cup \{vh_0\} \cup \{Cluster_{vh_0}\}$. Each phase i is similar to the first one: the node of $V_i = V - Set_i$ having the highest weight, named vh_i, becomes clusterhead. vh_i's cluster is filled out, and the members of vh_i's cluster get their final state. At the end of i^{th} phase, the set of nodes having their final state is $Set_i =$

$Set_{i-1} \cup \{vh_i\} \cup \{Cluster_{vh_i}\}$. Each phase building a cluster with at least one ordinary node requires at most 7 rounds, whereas the construction of clusters having just the clusterhead requires 1 round. The number of phases is equal or less than number of clusters. Thus, the convergence time to a legitimate configuration is at most $\frac{7*|V|}{2}$ rounds from a safe configuration.

5.2 Upper Bound of stabilization time

Theorem 1. *The convergence time of a self-stabilizing weight-based clustering protocol is intrinsically proportional to the network size.*

Proof. Let us study the example network presented in Figure 4.
In the initial configuration 4.a, there are $(|V| - 1)/2$ clusters where $|V|$ is the network size. For any value of i that is old, X_i is a clusterhead, X_{i+1} affiliates with X_i, and X_0 is member of X_1's cluster. The node weights are ordered as follows: $X_i > X_{i+1}$.
The legitimate configuration (assuming that $SizeBound \geqslant 1$) is defined as: for any value of i that is even, X_i is a clusterhead, and X_{i+1} affiliates with X_i.
To reach a legitimate configuration, each node has to change its role. X_{i+1} detects that it has to change its role only after a change on X_i's role. Clearly, X_i can change its role only after i rounds. Therefore, the convergence time is $O(V)$ rounds. □

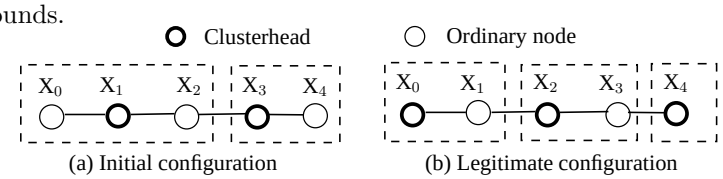

Fig. 4. Convergence time

6 Conclusion

Compared to the self-stabilizing protocol presented in [16], our solution is well-suited for large-scale modern distributed systems such as mobile ad hoc and sensor networks. Our protocol unlike [16], is scalable and during the reorganization of clusters, it ensures the maintain of hierarchical organization. The first benefit is due to the constant time (4 rounds at most) required to reach a configuration ensuring a minimum useful service. The second one is due to the robustness property which ensures that this minimum useful service still provided during the convergence to a legitimate configuration.

The cost of robustness property is the require of more time to ensure the stabilization (i.e., the convergence to a legitimate configuration). In fact, the upper bound on stabilization time for [16] is $|V|$ rounds, whereas for our protocol is $\frac{7*|V|}{2} + 5$ rounds.

Our protocol is designed for the state model. Nevertheless, it can be easily transformed into a protocol for the message-passing model. Each node v broadcasts periodically a message containing its state. Based on this message, v's neighbors decide whether to update their states or not.

References

1. Xu, Z., Hedetniemi, S.T., Goddard, W., Srimani, P.K.: A synchronous self-stabilizing minimal domination protocol in an arbitrary network graph. In: IWDC'03, Springer LNCS 2918. (2003) 26–32
2. Drabkin, V., Friedman, R., Gradinariu, M.: Self-stabilizing wireless connected overlays. In: OPODIS'06, Springer LNCS 4305. (2006) 425–439
3. Jain, A., Gupta, A.: A distributed self-stabilizing algorithm for finding a connected dominating set in a graph. In: PDCAT'05. (2005) 615–619
4. Goddard, W., Hedetniemi, S.T., Jacobs, D.P., Srimani, P.: Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In: IPDPS'03. (2003) 162.2
5. Dolev, S., Tzachar, N.: Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. Theoretical Computer Science **410** (2009) 514–532
6. Datta, A., Devismes, S., Larmore, L.: A self-stabilizing $o(n)$-round k-clustering algorithm. In: SRDS'09. (2009)
7. Mitton, N., Fleury, E., Guérin-Lassous, I., Tixeuil, S.: Self-stabilization in self-organized multihop wireless networks. In: WWAN'05. (2005) 909–915
8. Mitton, N., Busson, A., Fleury, E.: Self-organization in large scale ad hoc networks. In: MED-HOC-NET'04. (2004)
9. Kamei, S., Kakugawa, H.: A self-stabilizing approximation for the minimum connected dominating set with safe convergence. In: OPODIS'08, Springer LNCS 5401. (2008) 496–511
10. Kakugawa, H., Masuzawa, T.: A self-stabilizing minimal dominating set algorithm with safe convergence. In: APDCM'06. (2006)
11. Basagni, S.: Distributed clustering for ad hoc networks. In: ISPAN'99. (1999) 310–315
12. Johnen, C., Nguyen, L.H.: Robust self-stabilizing weight-based clustering algorithm. Theoretical Computer Science **410**(6-7) (2009) 581–594
13. Basagni, S.: Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In: VTC'99. (1999) 889–893
14. Chatterjee, M., Das, S.K., Turgut, D.: WCA: A weighted clustering algorithm for mobile ad hoc networks. Journal of Cluster Computing **5**(2) (2002) 193–204
15. Tomoyuki, O., Shinji, I., Yoshiaki, K., Kenji, I., Kaori, M.: An adaptive maintenance of hierarchical structure in ad hoc networks and its evaluation. In: ICDCS'02. (2002) 7–13
16. Johnen, C., Nguyen, L.H.: Self-stabilizing construction of bounded size clusters. In: ISPA'08. (2008) 43–50
17. M. Dom, D. Lokshtanov, S.S., Villanger, Y.: Capacitated domination and covering: A parameterized perspective. In: Proceedings of Third International Workshop IWPEC 2008, Springer LNCS 5018. (2008) 78–90
18. Kuhn, F., T, M.: Distributed approximation of capacitated dominating sets. In: SPAA '07. (2007) 161–170
19. Johnen, C., Mekhaldi, F.: Robust self-stabilizing construction of bounded size weight-based clusters. Technical Report N^o 1518, LRI, http://www.lri.fr/∼bibli/Rapports-internes/2009/RR1518.pdf (2009)