Memory efficient Self-Stabilizing distance-k Independent Dominating Set Construction*

Colette Johnen

Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France

Abstract. We propose a memory efficient self-stabilizing protocol building distance-k independent dominating sets[1]. A distance-k independent dominating set is a distance-k independent set and a distance-k dominating set.
Our algorithm, named $\mathcal{SID}$, is silent; it converges under the unfair distributed scheduler (the weakest scheduling assumption).
The protocol $\mathcal{SID}$ is memory efficient : it requires only $log(2((n+1)(k+1))^2)$ bits per node.
The correctness and the termination of the protocol $\mathcal{SID}$ is proven.
The computation of the convergence time of the protocol $\mathcal{SID}$ is an opened question.

keywords distributed algorithm, fault tolerance, self-stabilization, distance-k dominating set, distance-k independent set, distance-k independent dominating set, memory efficient

1 Introduction

The clustering of networks consists of partitioning network nodes into non-overlapping groups called clusters. Each cluster has a single head, called leader, that acts as local coordinator of the cluster, and eventually a set of standard nodes. leader. Clustering is found very attractive in infrastructure-less networks, like ad-hoc networks, since it limits the responsibility of network management only to leaders, and it allows the use of hierarchical routing.

Silent self-stabilizing protocols building k-hops clustering set are proposed [1,2,3,4]. In k-hop clusters, the distance between a standard node and its leader is at most k; the set of cluster heads can be not a distance-k independent set. The protocol of [1] is designed for $k = 2$. Routing tables are maintained by the cluster heads to store routing information to nodes

* Partially supported by the ANR project DISPLEXITY (ANR-11-BS02-014). This study has been carried out in the frame of "the Investments for the future" Programme IdEx Bordeaux CPU (ANR-10-IDEX-03-02)

[1] The protocol $\mathcal{SID}$ was presented in a brief announcement at SSS'13.

both within and outside the cluster. The goal of the protocol in [2] is to build bounded size clusters (each cluster has at most $Cluster\_Max$ nodes). The protocol of [3] is designed for weighted edges networks; it requires $O(log(k^4.\Delta^2.D^2.n^6))$ bits per node, where Δ is a bound on node degree and D is the network diameter. The protocol of [4] requires at least $log(2.k.n^2.n^{k+1}))$ bits per node.

In [5,6], Larsson and Tsigas propose self-stabilizing (l,k)-clustering protocols under various assumptions. These protocols ensure, if possible, that each node has l cluster-heads at distance at most k.

In [7], a silent self-stabilizing protocol extracting a minimal distance-k dominating set from any distance-k dominating set is proposed. A minimal distance-k dominating set has no proper subset which also a distance-k dominating set. The protocol requires at least $O(log(n^k))$ bits per node.

The paper [8] presents a silent self-stabilizing protocol building a small distance-k dominating set : the obtained dominating set contains at most $\lceil n/(k+1) \rceil$ nodes. The protocol of [8] requires $log(2.n^2.(n/k)^k)$ bits per node. The protocol of [9] builds competitive distance-k dominating sets : the obtained dominating set contains at most $1 + \lfloor (n-1)/(k+1) \rfloor$ nodes. The protocol of [9] requires $O(log(2.k.(\Delta+1)^3.n^3))$ bits per node.

Contribution. In this paper, we consider the problem of computing a distance-k independent dominating set in a self-stabilizing manner in case where $k > 1$. A nodes set is distance-k independent dominating set (also called maximal distance-k independent set) if and only if this set is a distance-k independent set and a distance-k dominating set. A set of nodes, I is distance-k independent if the distance between any pair of I's nodes is at least $k+1$. A set of nodes D is distance-k dominating if every node is within distance k of a node of D.

The protocol $\mathcal{SID}$ is simple : no use of the hierachical collateral composition, no need of leader election process, neither the building of spanning tree. It converges under the unfair distributed scheduler (the weakest scheduling assumption); and it is silent.

According to our knowledge, [10] is the only previous work proposing a silent self-stabilizing protocol building a maximal distance-k independent set assuming that $k > 1$. The protocol of [10] converges in $4n + k$ rounds; the computation of the convergence time of the protocol $\mathcal{SID}$ is an open question. The protocol in [10], requires $log((n+1)^{k+1})$ bits per node. The protocol $\mathcal{SID}$, requires less memory space - only $log(2.((n+1).(k+1))^2)$ bits per node. To achieve this result, the technique uses is quite different

and new; for instance two distincts total order relations on the same objects are used.

2 Specification of Problem and Computation Model

A distributed system S is an undirected graph $G = (V, E)$ where the vertex set, V, is the set of nodes and the edge set, E, is the set of communication links. A link $(u, v) \in E$ if and only if u and v can directly communicate (links are bidirectional); so, the node u and v are neighbors. N_v denotes the set of v's neighbors: $N_v = \{u \in V \mid (u, v) \in E\}$. The distance between the nodes u and v is denoted by $dist(u, v)$. The set of nodes at distance at most k to a node v is denoted by `k-neigborhood(v)` $= \{u \in V \mid dist(u, v) \in [1, k]\}$.

Definition 1 (distance-k independent dominating set).
*Let D be a subset of V; D is a **distance-k dominating set** if and only if $\forall v \in V/D$ we have* `k-neigborhood`$(v) \cap D \neq \emptyset$.
*Let I be a subset of V; I is a **distance-k independent set** if and only if $\forall u \in I$ we have* `k-neigborhood`$(u) \cap I = \emptyset$.
A subset of V is a distance-k independent dominating set if this subset is a distance-k dominating set and a distance-k independent set.

At every node v in the network is assigned an identifier, denoted by id_v. Two distinct nodes have different identifier. It is possible to order the identifier values.

Each node maintains a set of shared variables. A node can read its own variables and those of its neighbors, but it can modify only its variables. The *state* of a node is defined by the values of its local variables. The cartesian product of states of all nodes determines the *configuration* of the system. Let `var` be a shared variable, `var`$(v)_c$ denotes the value of `var` for the node v in the configuration c. The *program* of each node is a set of *rules*. Each rule has the form: $Rule_i :< Guard_i > \longrightarrow < Action_i >$. The *guard* of a v's rule is a boolean expression involving the state of the node v, and those of its neighbors. The *action* of a v's rule updates v's state. A rule can be executed by a node v only if it is *enabled*, i.e., its guard is satisfied by the node v. A node is said to be enabled if at least one of its rules is enabled. A configuration is *terminal* if and only if no node can execute a rule.

During a *computation step* from a configuration one or more enabled nodes simultaneously perform an action to reach another configuration.

A *computation* e is a sequence of configurations $e = c_0, c_1, ..., c_i, ...$, where c_{i+1} is reached from c_i by a single computation step, $\forall i \geqslant 0$. A computation e is *maximal* if it is infinite, or if it reaches a terminal configuration.

Definition 2 (Silent Self-Stabilization). *Let $\mathcal{L}$ be a predicate on the configuration. A distributed system S is a silent self-stabilizing system to $\mathcal{L}$ if and only if (1) all terminal configurations satisfy $\mathcal{L}$; (2) all computations reach a terminal configuration.*

3 The protocol $\mathcal{SID}$

In the following subsection, we gives the notation used by the protocol $\mathcal{SID}$.

3.1 k-augmentedID type

Definition 3. ***k-augmentedID type*** *An k-augmentedID value, a, is $\perp$ or an n-tuple (d, x) such that d is integer with $0 \leq d \leq k$, and x is a node identifier. Let $a = (d, x)$ be k-augmentedID value. We use the following notation $a.dist = d$ and $a.id = x$.*
Let v be a node of V, id_v^+ is the following k-augmentedID value: $(0, id_v)$.

Definition 4. ***The total order relation*** *dom* ***on k-augmentedID***

- *$dom(a, b) = a$ if $b = \perp$, $a.id < b.id$ or $a.id = b.id \ \wedge \ a.dist < b.dist$, otherwise $dom(a, b) = b$.*
- *The k-augmentedID value $a1$ dominates the k-augmentedID value $a2$ if and only if $dom(a1, a2) = a1$.*
- *Let X be a finite set of k-augmentedID values. $dom(X)$ is the k-augmentedID value, denoted dX, belonging to X such that any value of X is dominated by dX (i.e. $\forall y \in X$ we have $dom(dX, y) = dX$).*

Definition 5. ***The total order relation*** *min* ***on k-augmentedID***

- *$min(a, b) = a$ if $b = \perp$, $a.dist < b.dist$ or $a.dist = b.dist \ \wedge \ a.id < b.id$ otherwise $min(a, b) = b$.*
- *The k-augmentedID value $a1$ is larger than the k-augmentedID value $a2$ if and only if $min(a1, a2) = a2$.*
- *Let X be a finite set of k-augmentedID values. $min(X)$ is the k-augmentedID value, denoted mX, belonging to X such that any value of X is larger than mX (i.e. $\forall y \in X$ we have $min(mX, y) = mX$).*

The node $u1$ is *closer* to the node v than the node $u2$ iff $d1 = dist(u1, v)) < dist(u2, v) = d2$ or $id_{u1} < id_{u2}$. Notice that $(d2, id_{u2}))$ is larger than $min((d1, id_{u1})$.

Definition 6. ***The operation +1*** *on k-augmentedID is defined as follow :* $a+1 = a$ *if* $a = \perp$ *or if* $a.dist = k$ *otherwise* $a+1 = (a.dist+1, a.id)$

3.2 Code of the protocol $\mathcal{SID}$

The variables, the function and procedure specifications, the predicates and the rules of $\mathcal{SID}$ are presented in protocol 1. By lack of space, the code of the functions and the procedures are omitted in the paper, they can found in the technical report of LaBRI [11]).
The variable `firstH`(v) contains the identifier of the closest head to v (with its distance to v).
The variable `secondH`(v) contains the identifier of the second closest head to v (with its distance to v) inside its `k-neighborhood`. If a node v does not have two heads in its `k-neighborhood` then `secondH`(v) is set to $\perp$.

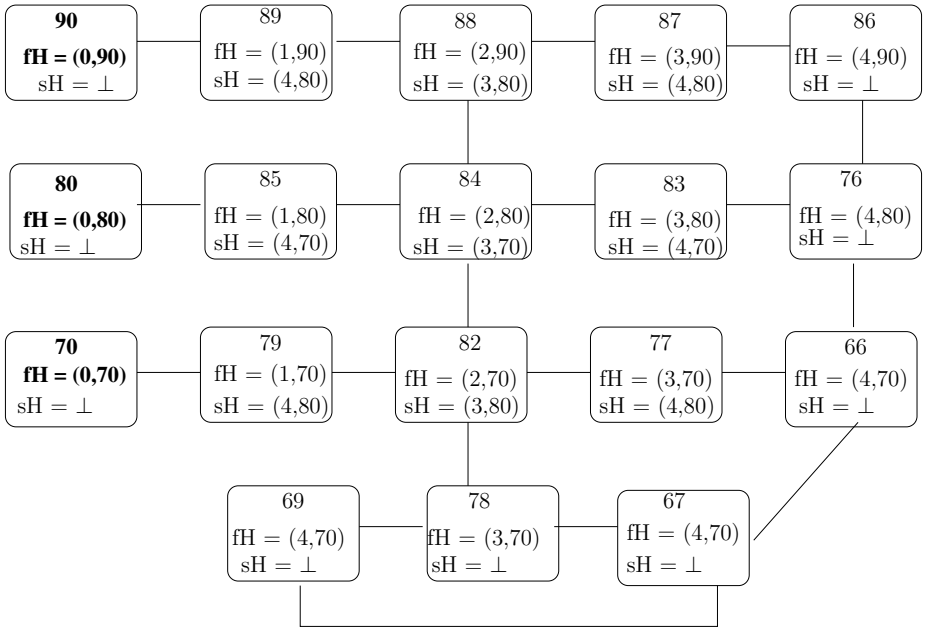

Fig. 1. A legitimate configuration of $\mathcal{SID}$ with $k = 4$

A node v is said to be a head if $\texttt{firstH}(v) = id_v^+ = (0, id_v)$; otherwise v is an ordinary node. We will prove that in any terminal configuration the Head set built by the protocol $\mathcal{SID}$ is a distance-k independent dominating set. We will also establish that all computations are finite.

In the figure 1 is presented a terminal configuration of $\mathcal{SID}$ in the case where $k = 4$. In each node, it is indicated the value of `firstH`, denoted by fH, and the value of `secondH` denoted by sH. The legitimate configuration has three heads. On the same network with the same value for k, is presented another terminal configuration having a single head in the figure 2.

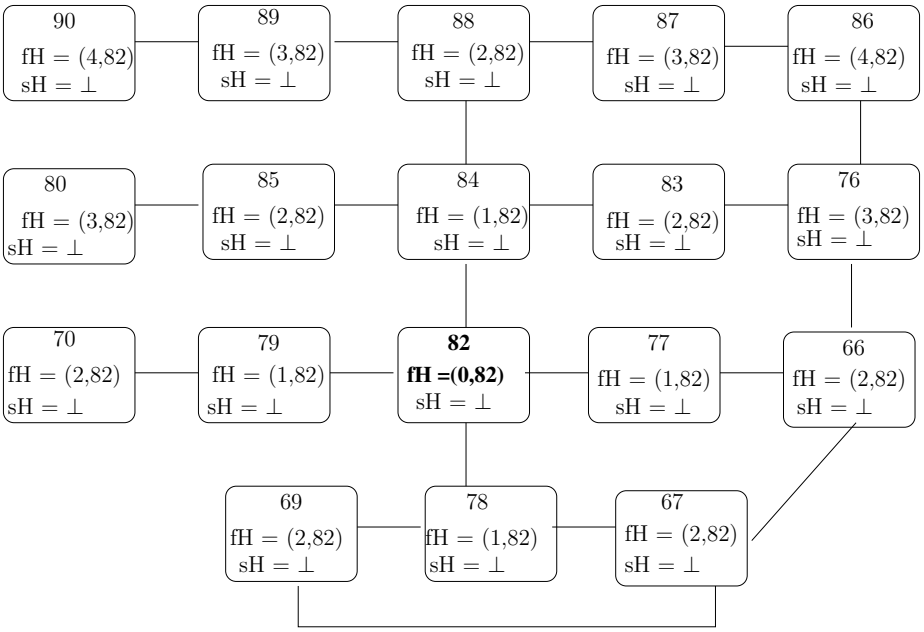

Fig. 2. A terminal configuration of $\mathcal{SID}$ having a single head

The function $\texttt{isDefended}(v)$ returns true if the set $\texttt{FirstS}(v)$ is not empty otherwise the function returns false.

The function $\texttt{isDominated}(v)$ returns true if a value x of $\texttt{FirstS}(v)$ dominates the value $id_v^+ = (0, id_v)$; otherwise the function returns false.

The function $\texttt{correctFirstH}(v)$ returns true if the value of $\texttt{firstH}(v)$ is $min(\texttt{FirstS}(v))$; otherwise or if the set $\texttt{FirstS}(v)$ is empty then the function returns false.

Algorithm 1: : code of $\mathcal{SID}$ on the node v

Shared variables

- $\texttt{firstH}(v)$ and $\texttt{secondH}(v)$. They take value in k-augmentedID.

Internal variable

- *beReal* is a boolean variable used by some funtions.

Notation

- $\texttt{FirstS}(v) = \{a+1 \in k\text{-augmentedID} \mid a = \texttt{firstH}(u) \ \vee \ a = \texttt{secondH}(u)$ $with\ u \in N_v \wedge a.dist < k \wedge a.id \neq id_v\}$
- $\texttt{secondS}(v) = \{a \in \texttt{FirstS} \mid a.id \neq \texttt{firstH}(v).id\}$

Boolean function specifications

- $\texttt{isDefended}(v)$ returns true iff $\texttt{FirstS}(v) \neq \emptyset$.
- $\texttt{isDominated}(v)$ returns true iff $id_v^+ \neq dom(\texttt{FirstS}(v) \cup id_v^+)$.
- $\texttt{correctFirstH}(v)$ returns true iff $\texttt{firstH}(v) = min(\texttt{FirstS}(v))$.
- $\texttt{correctSecondH}(v)$ returns true iff $\texttt{secondH}(v) = min(\texttt{secondS}(v) \cup \perp)$.

Procedure specifications

- $\texttt{computingFirstH}(v)$ sets $\texttt{firstH}(v)$ to $min(\texttt{FirstS}(v))$.
- $\texttt{computingsSecondH}(v)$ sets $\texttt{secondH}(v)$ to $min(\texttt{secondS}(v) \cup \ \perp)$.

Predicates

- $\texttt{Head}(v) \equiv \texttt{firstH}(v) = (0, id_v)$
- $\texttt{toResign}(v) \equiv \texttt{isDominated}(v)$
- $\texttt{toElect}(v) \equiv \neg\texttt{isDefended}(v)$
- $\texttt{headToUpdate}(v) \equiv \texttt{secondH}(v) \neq \perp$
- $\texttt{ordinaryToUpdate}(v) \equiv \neg\texttt{correctFirstH}(v) \vee \neg\texttt{correctSecondH}(v)$

Rules

$\mathbf{RE}(v) : \neg\texttt{Head}(v) \wedge \texttt{toElect}(v) \longrightarrow \texttt{firstH}(v) := (0, id_v);\ \texttt{secondH}(v) := \perp$

$\mathbf{RU}(v) : \neg\texttt{Head}(v) \wedge \neg\texttt{toElect}(v) \wedge \texttt{ordinaryToUpdate}(v) \longrightarrow$
$\texttt{computingFirstH}(v);\ \texttt{computingSecondH}(v)$

$\mathbf{RR}(v) : \texttt{Head}(v) \wedge \texttt{toResign}(v) \longrightarrow$
$\texttt{computingFirstH}(v);\ \texttt{computingSecondH}(v)$

$\mathbf{RC}(v) : \texttt{Head}(v) \wedge \neg\texttt{toResign}(v) \wedge \texttt{headToUpdate}(v) \longrightarrow \texttt{secondH}(v) := \perp$

The procedure computingFirstH(v) sets firstH(v) to min(FirstS(v)) if the set FirstS(v) is not empty; otherwise the value of firstH(v) is $\perp$. In the latter case, v verifies the predicate toElect(v) and it does not verify the predicate toResign(v). So the procedure computingFirstH(v) is never preformed when set FirstS(v) is empty.

The function correctSecondH(v) returns true if the value of secondH(v) is min(secondS(v) $\cup \perp$); otherwise the function returns false. The procedure computingSecondH(v) sets secondH(v) to min(secondS(v)$\cup \perp$).

Once the system is stabilized, the set FirstS(v) contains some heads in k-neighborhood of v. More precisely, this set contains the closest and second closest head to v if there are at least one Head in the k-neighborhood of v.

If the k's neighborhood of a node v does not contain any head then the set FirstS(v)) is empty. So the predicate toElect(v) is verified. If v is an ordinary node then v is enabled (the rule **RE** is enabled). Therefore, the heads set is a distance-k dominating set, in a terminal configuration.

If one or several Heads have in their k-neighborhood another Head then at least one of these Heads is enabled. Let us name, v, the Head having the largest identifier among the Heads that have Heads in their k-neighborhood. Once the system is stabilized, the FirstS(v) contains a value (d, id_u) such that $id_v > id_u$ and $d < k$. The node v is enabled : it verifies the predicate toResign. So, the set of heads is a distance-k independent set, in any terminal configuration.

3.3 Illustration of $\mathcal{SID}$ behavior

In the figure 3, an execution with $k = 2$ under the synchronous schedule is presented. During the first computation step, the node having the identifier 8 detectes that its neighbor having the identifier 7 is a Head, so it becomes ordinary by executing the rule **RE** (it sets its firstH variable to $(1, 7)$). Also during the first step, the node at distance 1 of the Head 4 updates its shared variables (i.e. it executes the rule **RU**). During the 2th step (starting at the configuration b), two Heads detect that there are at distance 2 of the Head 4, as their identifier are larger than 4, they execute the rule **RR** (i.e. they become ordinary). During the 3rd step (starting at the configuration c), two ordinary nodes (the node of identifier 8 and the node of identifier 9) detect that they have no Head in their 2-neighborhood so they become Head (i.e. they execute the rule

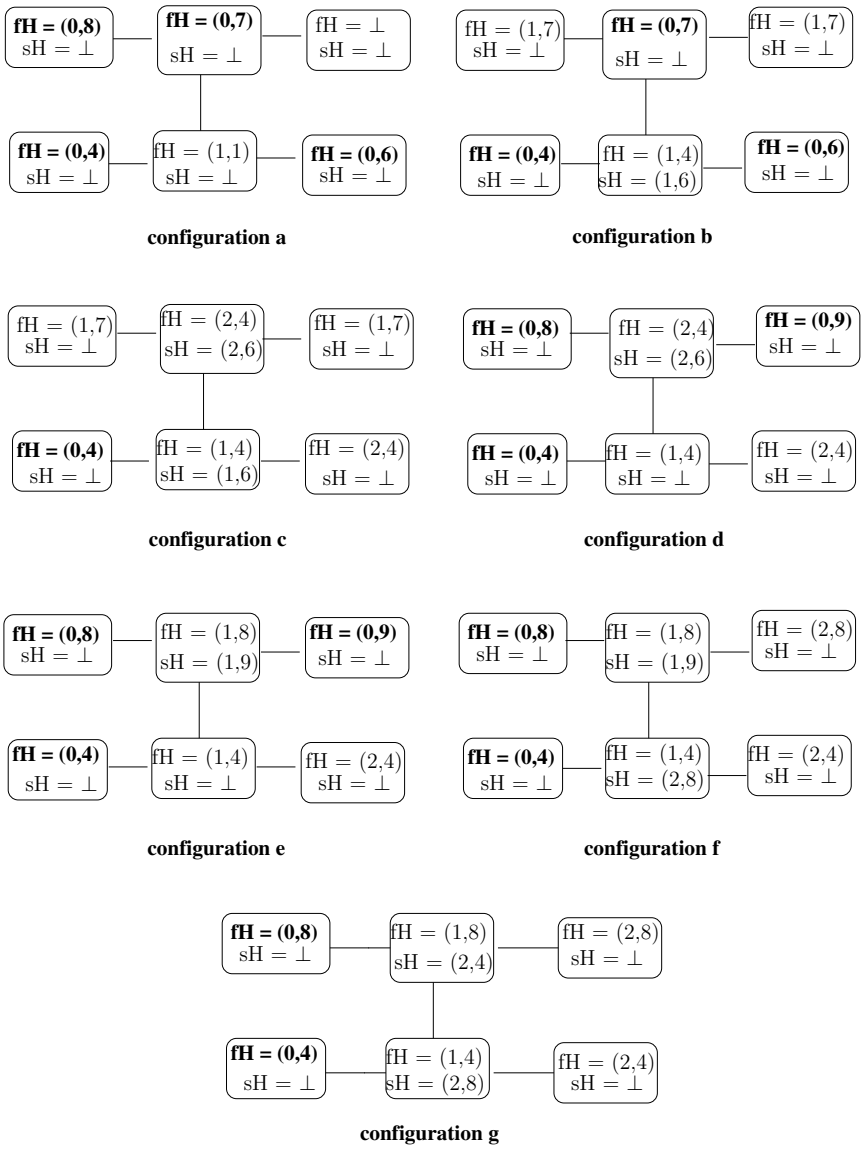

Fig. 3. An execution of $\mathcal{SID}$ with $k = 2$

RE). During the 5th step, the node 9 detects that it is at distance 2 of the Head 8; so it resigns. during the last computations step, the only rule executed is **RU** to update the variable `secondH`. So, no node will change its status (i.e. to become a Head or Ordinary). The configuration g is terminal and also legitimate.

4 Correctness of the protocol $\mathcal{SID}$

In this section, we prove that all terminal configurations of $\mathcal{SID}$ protocol are legitimate: the set of heads is a distance-k independent dominating set.

Definition 7. *The property* `OrdinaryPr`*(i) defined for all* $i \in [1, k]$ *is verified if the two following statements are satisfied:*

- `OrdinaryPrFirst`*(i): for all ordinary node* v, $\texttt{firstH}(v) = (i, id_u)$ *if and only if* u *is the closest head to* v *and* i *is the distance between* u *and* v.
- `OrdinaryPrSecond`*(i): for all node* v, $\texttt{secondH}(v) = (i, id_w)$ *if and only if* w *is the second closest head to* v *and* i *is the distance between* w *and* v.

Observation 1 *In a terminal configuration,*

1. *An ordinary node* v *does not verify* $\texttt{OrdinaryToUpdate}(v)$*;*
   so $\texttt{firstH}(v) = min(\texttt{firstS}(v))$ *and*
   $\texttt{secondH}(v) = min(\texttt{secondtSet}(v) \cup \perp)$.
2. *A head* u *does not verify* $\texttt{HeadToUpdate}(u)$*;*
3. *Let* w *be a node (head or ordinary),* $\texttt{firstH}(w) \neq \perp$*;*
4. *if* v *is an ordinary node then* $\texttt{firstH}(v).dist > 0$ *;*
5. *if* $\texttt{secondH}(v) \neq \perp$ *then* $\texttt{secondH}(v).dist > 0$ *;*
6. *if* $\texttt{secondH}(v) \neq \perp$ *then* $\texttt{secondH}(v).dist \geq \texttt{firstH}(v).dist$ *because* $\texttt{secondS}(v) \subset \texttt{firstS}(v)$, $\texttt{firsthead}(v) = min(\texttt{firstS}(v))$ *and* $\texttt{secondhead}(v) = min(\texttt{secondS}(v))$

Lemma 1. *In a terminal configuration of protocol* $\mathcal{SID}$*, the property* $\texttt{OrdinaryPr}(1)$ *is verified.*

Proof.
Let v be an ordinary node, in a terminal configuration of protocol $\mathcal{SID}$, named c. Assume that $(1, x) \in \texttt{firstS}(v)$. So v has a neighbor u such that $\texttt{firstH}(u) = (0, x)$ or $\texttt{secondH}(u) = (0, x)$.

According to Observation 1.5 $\texttt{secondH}(u).dist > 0$ or $\texttt{secondH}(u) = \perp$. So v has a neighbor u such that $\texttt{firstH}(u) = (0, x)$. According to Observation 1.4 u is a head; so $x = id_u$. Notice that $\forall a \in \texttt{firstS}(v)$, we have $a.dist > 0$, in c.

Proof of $\texttt{OrdinaryPrFirst}(1)$. If v has a head at distance 1 then v has a neighbor u such that $\texttt{firstH}(u) = (0, id_u)$. So, we have $\texttt{firstH}(v) = (1, id_u)$ with u being the head in v's neighborhood having the smallest identifier.
If v has not a head at distance 1 then for any u neighbor, we have $\texttt{firstH}(u).dist > 0$. and $\texttt{secondH}(u).dist > 0$ or $\texttt{secondH}(u) = \perp$ (according to Observation 1.5). In this case, $\texttt{firstH}(v).dist > 1$.

Proof of $\texttt{OrdinaryPrSecond}(1)$. If v has several heads at distance 1 then v has a neighbor w such that $\texttt{firstH}(w) = (0, id_w)$ with $id_w \neq \texttt{firstH}(v).id$. So, $\texttt{secondH}(v) = (1, id_w)$ with w being the head in v's neighborhood having the second smallest identifier.
If v has at most one head at distance 1 then v has not a neighbor w such that $\texttt{firstH}(w) = (0, id_w)$ with $id_w \neq \texttt{firstH}(v).id$. In this case, $\texttt{secondH}(v).dist$ is larger than 1 or $\texttt{secondH}(v) = \perp$. □

Lemma 2. *Let i be a positive integer smaller than k. In a terminal configuration of protocol $\mathcal{SID}$, if the properties* $\texttt{OrdinaryPr(j)}$ *are verified for all $j \in [1, i]$ then the property* $\texttt{OrdinaryPr}(i+1)$ *is verified.*

Proof. Let us assume that the properties $\texttt{OrdinaryPr(j)}$ are verified for all $j \in [1, i]$ in any terminal configuration of protocol $\mathcal{SID}$.
In a terminal configuration c, $(j, x) \in \texttt{firstS}(v)$ iff v has a neighbor u such that $\texttt{firstH}(u) = (j-1, x)$, or $\texttt{secondH}(u) = (j-1, x)$. If $j = 1$ then u is a head in c, according to Observation 1. If $1 < j \leq i+1$ then x is the identifier of a head in c at distance $j-1$ of u, according to the property $\texttt{OrdinaryPr}(j-1)$. So x is the identifier of a head at distance at most j of v, in c.

Proof of $\texttt{OrdinaryPrFirst}$(i+1). Let v' be the closest head to v and d' the distance from v' to v in the terminal configuration c. Assume that $0 < d' \leq i+1$. v has a neighbor u at distance $d'-1$ to v'. In c, the node v' is the closest head of u; so $\texttt{firstH}(u) = (d'-1, id_{v'})$, according to the properties $\texttt{OrdinaryPr}(d'-1)$. According to the properties $\texttt{OrdinaryPr}(j)$ $\forall\ j \in [1, i]$, in c, we have the following properties,

- if $(l, id) \in \texttt{firstS}(v)$ then $l \geq d'$; and
- if $(d', id) \in \texttt{firstS}(v)$ then $id \geq id_{v'}$. In c,

We conclude that $\texttt{firstH}(v) = (d', id_{v'})$, in c.

Proof of $\texttt{OrdinaryPrSecond}$(i+1). Assume that the network has several heads. Let v" be the second closest head to v and d" the distance from v" to v, in a terminal configuration c. v has a neighbor u at distance $d" - 1$ to v" in c. (we have $d" > 0$). v" is the first or second closest head to u, in c. Assume that $d" \leq i+1$. According to the property $\texttt{OrdinaryPr}(d" - 1)$, $\texttt{firstH}(u) = (d" - 1, id_{v"}) \vee \texttt{secondH}(u) = (d" - 1, id_{v"})$, in c. According to the properties $\texttt{OrdinaryPr}(j)\ \forall\ j \in [1, i)$, in c, we have the following properties,

- if $(l, id) \in \texttt{secondS}(v)$ then $l \geq d"$;
- if $(d", id) \in \texttt{secondS}(v)$ then $id \geq id_{v"}$.

We conclude that $\texttt{secondH}(v) = (d", id_{v"})$. □

The following corollary is a direct result of lemmas 1 and 2. It establishes that the set of heads is a distance-k dominating set.

Corollary 1. *Let v be a ordinary node, in a terminal configuration of protocol $\mathcal{SID}$.* $\texttt{firstH}(v).id$ *is the closest head to v; their distance is* $\texttt{firstH}(v).dist \leq k$. *If* $\texttt{secondH}(v) = \perp$ *then v has a single head in its k-neigborhood; otherwise* $\texttt{secondH}(v).id$ *is the second closest head to v; their distance is* $\texttt{secondH}(v).dist$.

The following theorem establishes that the set of heads is a distance-k independent set in any terminal configuration.

Theorem 1. *Let v be a head, in a terminal configuration of protocol $\mathcal{SID}$. v has not head in its k-neigborhood.*

Proof. We will prove that if a head has another head in its k-neigborhood then the configuration c is not terminal.

Let $wrongHeadSet$ the set of heads having one or several heads in their k-neigborhood. Assume that $wrongHeadSet$ is not empty. We denote by $v1$ the node of $wrongHeadSet$ having the largest identifier. We denote by $v2$, the closest head to $v1$ and by d the distance between $v1$ and $v2$. We have $0 < d \leq k$ and $id_{v2} < id_{v1}$.

The node $v1$ has a neighbor u at distance $d - 1$ of $v2$. The node $v2$ is the first or the second closest head to u. According to corollary 1, $(d - 1, id_{v2}) = \texttt{firstH}(u)$ or $(d - 1, id_{v2}) = \texttt{secondH}(u)$. $v1$ is enabled because $v1$ satisfied the predicate $\texttt{toResign}(v1)$. □

5 Termination of the protocol $\mathcal{SID}$

In this section, we prove that all maximal computations of protocol $\mathcal{SID}$ under any unfair distributed scheduler are finite by *reductio ad absurdam* arguments.

Lemma 3. *Let e be a maximal computation.*
The values taken by `firstH` *and* `seconHead` *along e by any node belong to the same set containing $3nk$ k-augmentedID values.*

Proof. Let e be a maximal computation starting from a configuration, named $c0$. In a configuration c reached by e, for any node v, $\texttt{firstH}(v)_c.id$ is either the identifier of an node or this value appears in the initial configuration (i.e. there is a node u, such that $\texttt{firstH}(v)_c.id = \texttt{firstH}(u)_{c0}.id \vee \texttt{firstH}(v)_c.id = \texttt{secondH}(u)_{c0}.id$). So, the value taken by a variable `firstH` in e belongs to a set having $3nk$ values. Similary we prove that the value taken by a variable `secondH` along e belongs to the same bounded set. □

Observation 2 *Along any computation, a node performs at most one time the rule* **RC**.

Lemma 4. *Let e be a maximal computation. e has a suffix in which the only rule performed is* **RU**.

Proof. Assume that a or several nodes perform infinitely often the action **RE** or the action **RR** along e. Between two consecutive actions **RE** by a node u, this node has performed on time the action **RR**. So a node u that infinitely often performs the action **RE** or the action **RR** changes its status infinitely often. We name u^+ the node having the smallest identifier among the nodes that change their status infinitely often. e has a suffix $e1$ where only nodes having a identifier larger than id_{u^+} changes their status (i.e. they perform the action **RE** or the action **RR**).

As the set of value taken by $\texttt{firstH}(u^+)$ is bounded (lemma 3) along $e1$, infinitely often after the action $\mathbf{RR}(u^+)$, $\texttt{firstH}(u^+)$ has the same value, denoted by $(l+1, id)$. Notice that $id < id_{u^+}$ and $0 < l < k$. So u^+ has a neighbor u_l such that, infinitely often before the action $\mathbf{RR}(u^+)$, u_l verifies $\texttt{firstH}(u_l) = (l, id)$ or $\texttt{secondH}(u_l) = (l, id)$.
At time, where u^+ becomes head, we have $\texttt{firstS}(u^+) = \emptyset$. So, the values of u_l variables are infinitely often larger than (l, id). Thus, u_l gives infinitely often to one of its variables the value (l, id), but also gives a

larger value to the same variable.

Assume that $l > 0$. At time where u_l gives the value (l, id) to one of its variable : u_l has a neigbor u_{l-1}, having the value $(l-1, id)$. At time where u_l gives a larger value than (l, id) to the same variable : u_{l-1} has a larger value than $(l-1, id)$. We conclude that there is a series of $l+1$ nodes : $u_l, u_{l-1}, ..u_0$ such that u_i has infinitely often has the value (i, id) and infinitely often does not have this value along $e1$.

Along $e1$, u_0 performs infinitely often the action **RR** and the action **RE**. We have $id = id_{u_0} < id_{u^+}$: there is a contradiction. □

Lemma 5. *Let e be a maximal computation. e has a suffix in which no rule is performed.*

Proof. According to lemma 4, e has a suffix, named $e2$, in which the only rule performed is **RU**. Assume that a node or several nodes changing infinitely often their value `firstH` or their value `secondH` along $e2$. We named min^+ the smallest value infinitely often allocated to the variable `firstH` or to the variable `secondH` of one of these nodes. Let $e3$ be the suffix of $e2$ in which no variable `firstH` and no variable `secondH` gets a value smaller than min^+. Along $e3$, infinitely often, a node, named u^+, performs **RU** action to set the value min^+ to its variable `firstH` or its variable `secondH`; and infinitely often, u^+ performs **RU** action to set to the same variable a value larger than min^+.

Let $c \to c'$ be a computation step of $e3$ where u^+ performs **RU** action to set a value larger than min^+ to its variable `firstH` or to its variable `secondH`. In c, min^+ is smaller than $min(\texttt{firstS}(u^+))$ or min^+ is smaller than $min(\texttt{secondS}(u^+))$. This property stays verified after this computation step along $e3$. So u^+ never sets the value min^+ to its variable `firstH` (resp. to its variable `secondH`). There is a contradiction. □

As no computation can be infinite, any maximal computation reaches a terminal configuration.

Corollary 2. *under the unfair distribued scheduler, Any maximal computation reaches a terminal configuration.*

6 Conclusion

A simple and silent self-stabilizing protocol building distance-k independent dominating sets is presented. The protocol converges under the unfair distributed scheduler (the weakest scheduling assumption). The computation of the convergence time of the protocol is an open question. In

[10], we establish that any distance-k independent sets contain at most $\lfloor(2n)/(k+2)\rfloor$ nodes, n being the network size. So the protocol of [10] and the presented protocol have the same upper bound on the size of built k independent dominating sets : $\lfloor(2n)/(k+2)\rfloor$ nodes.
The protocol $\mathcal{SID}$ is memory efficient : it requires only $log(2.((n+1).(k+1))^2)$ bits per node.

References

1. Bein, D., Datta, A.K., Jagganagari, C.R., Villain, V.: A self-stabilizing link-cluster algorithm in mobile ad hoc networks. In: International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05). (2005) 436–441
2. Bui, A., Clavière, S., Datta, A.K., Larmore, L.L., Sohier, D.: Self-stabilizing hierarchical construction of bounded size clusters. In: 18th International Colloquium Structural Information and Communication Complexity (SIROCCO'11), Springer LNCS 6796. (2011) 54–65
3. Caron, E., Datta, A.K., Depardon, B., Larmore, L.L.: self-stabilizing k-clustering algorithm for weighted graphs. Journal of Parallel and Distributed Computing **70** (2010) 1159–1173
4. Datta, A.K., Larmore, L.L., Vemula, P.: A self-stabilizing o(k)-time k-clustering algorithm. The Computer Journal **53**(3) (2010) 342–350
5. Larsson, A., Tsigas, P.: A self-stabilizing (k,r)-clustering algorithm with multiple paths for wireless ad-hoc networks. In: IEEE 31th International Conference on Distributed Computing Systems, (ICDCS'11), IEEE Computer Society (2011) 353–362
6. Larsson, A., Tsigas, P.: Self-stabilizing (k,r)-clustering in clock rate-limited systems. In: 19th International Colloquium Structural Information and Communication Complexity, (SIROCCO'12), Springer, LNCS 7355. (2012) 219–230
7. Datta, A., Devismes, S., Larmore, L.: A self-stabilizing $o(n)$-round k-clustering algorithm. In: 28th IEEE Symposium on Reliable Distributed Systems (SRDS'09). (2009) 147–155
8. Datta, A.K., Larmore, L.L., Devismes, S., Heurtefeux, K., Rivierre, Y.: Self-stabilizing small k-dominating sets. International Journal of Networking and Computing **3**(1) (2013) 116–136
9. Datta, A.K., Larmore, L.L., Devismes, S., Heurtefeux, K., Rivierre, Y.: Competitive self-stabilizing k-clustering. In: IEEE 32th International Conference on Distributed Computing (ICDCS'12). (2012) 476–485
10. Johnen, C.: Fast, silent self-stabilizing distance-k independent dominating set construction. Information Processing Letters **114**(10) (2014) 551–555
11. Johnen, C.: Memory efficient self-stabilizing k-independent dominating set construction. Technical Report RR-1473-13, Univ. Bordeaux, LaBRI, UMR 3800, F-33400 Talence, France (June 2013)