Memory Efficient Self-Stabilizing k-Independent Dominating Set Construction

Colette Johnen

Memory efficient Self-Stabilizing k-Independent Dominating Set Construction*
Labri Technical Report RR-1473-13

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

October 1, 2013

Abstract

We propose a memory efficient self-stabilizing protocol building k-independent dominating sets. A k-independent dominating set is a k-independent set and a k-dominating set. A set of nodes, I, is k-independent if the distance between any pair of nodes in I is at least $k+1$. A set of nodes, D, is a k-dominating if every node is within distance k of a node of D.
Our algorithm, named $\mathcal{SID}$, is silent; it converges under the unfair distributed scheduler (the weakest scheduling assumption).
We established that any k-independent sets contains at most $\lfloor (2n)/(k+2) \rfloor$ nodes, n being the network size.
The protocol $\mathcal{SID}$ is memory efficient : it requires only $2log((k+1)n+1)+1$ 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 opened question.

keywords distributed computing, fault tolerance, self-stabilization, k-dominating set, k-independent set, 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. In 1-hop clusters, the standard nodes are neighbor (at distance 1) of their 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. This is why numerous self-stabilizing 1-hop clustering protocols were proposed in the literature [9, 10, 8, 11, 13, 14, 15, 16, 19, 20].

*This work was partially supported by the ANR project `Displexity`.

Silent self-stabilizing protocols building k-hops clustering set are proposed for $k > 1$. In k-hop clusters, the distance between a standard node and its leader is at most k [1, 2, 3, 7]. The sets of cluster heads built by these protocols are not k-independent. The protocol of [1] is designed for $k = 2$. Routing tables are maintained by the cluster heads to store routing information to nodes 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). bits per node. The protocol of [3] is designed for weighted edges networks; it requires at least $O(k.log(n))$ bits per node. The protocol of [7] requires at least $2(k+1)log(n)$

In [17, 18], Larsson and Tsigas propose self-stabilizing (l,k)-clustering protocols under various assupmtions. These protocols ensure, if possible, that each node has l cluster-heads at distance at most k.

Related Works. In [4], a silent self-stabilizing protocol extracting a minimal k-dominating set from any k-dominating set is proposed. A minimal k-dominating set has no proper subset which also a k-dominating set. The protocol requires at least $O(k.log(n))$ bits per node.

The paper [6] presents a silent self-stabilizing protocol building a small k-dominating set : the obtained dominating set contains at most $\lceil n/(k+1) \rceil$ nodes. The protocol of [6] requires $O(log(n) + k.log(n/k))$ bits per node. The protocol of [5] builds competitive k-dominating sets : the obtained dominating set contains at most $1 + \lfloor (n-1)/(k+1) \rfloor$ nodes. The protocol of [5] requires $O(log(2k.2(\Delta+1).2n.D))$ bits per node, where D is the network diameter, and Δ is a bound on node degree. These both protocols use the hierachical collateral composition of several silent self-stabilizing protocols whose a leader election protocol and a spanning tree construction rooted to the elected leader. So they require more memory space than our protocol.

In [12], a fast silent self-stabilizing protocol building a k-independent dominating set is proposed the protocol converges in $4n + k$ rounds and it requires $(k+1)log(n+1)$ bits per node. The computation of the convergence time of the protocol $\mathcal{SID}$ is opened question.

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

The presented protocol, named $\mathcal{SID}$, is simple : no use of the the hierachical collateral composition, no need of leader election process, neither the building of spanning tree. The protocol $\mathcal{SID}$ converges under the unfair distributed scheduler (the weakest scheduling assumption). The algorithm $\mathcal{SID}$ is silent. The protocol $\mathcal{SID}$ is memory efficient : it requires only $2log((k+1)n+1)+1$ bits per node.

In section 2, we establish that any k-independent sets contain at most $\lfloor (2n)/(k+2) \rfloor$ nodes, n being the network size. So the protocol of [12] and the protocol $\mathcal{SID}$ have the same upper bound on the size of built k independent dominating sets : $\lfloor (2n)/(k+2) \rfloor$ nodes.

Paper outline. The rest of the paper is organised as follows. In section 2, communication and computation models are defined. The protocol $\mathcal{SID}$ is presented in section 3. In the section 4, the correctness of the $\mathcal{SID}$ protocol is proven. The termination of the $\mathcal{SID}$ protocol is established in the section 5.

2 Model and Concepts

A distributed system S is an undirected graph $G = (V, E)$ where vertex set V is the set of nodes and 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, u and v are neighbors. We denote by N_v 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 $dist(u, v)$. $\texttt{k-neigborhood}(v) = \{u \in V \mid dist(u, v) \in [1, k]\}$.

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

Lemma 1 *The size of a k-independent set is at most $max(\lfloor (2n)/(k+2) \rfloor,\ 1)$.*

Proof 1 *Let I be a k-independent set such that $|I| > 1$. Let v be a node of I. We denote by $\texttt{closer}(v)$ the set of nodes closer to v than any other node of I.*
Notice that $\bigcup_{w \in I} \texttt{closer}(w) \subset V$ and $\texttt{closer}(v) \cap \texttt{closer}(u) = \emptyset, \forall (u, v) \in I^2$. Let u be the closest node to v that belongs to I. Let x be node on the path from v to u such that $0 \leq dist(v, x) \leq \lfloor k/2 \rfloor$. Let w be a node of I other than v. We have $dist(w, x) > k - dist(v, x) \geq \lfloor k/2 \rfloor$ because $k < dist(w, v) \leq dist(v, x) + dist(x, w)$. So, $\texttt{closer}(v)$ contains the first $\lfloor k/2 \rfloor + 1$ nodes in the path from v to u. We conclude that $|I| \leq \lfloor (2n)/(k+2) \rfloor$. ■

The protocol $\mathcal{SID}$ builds k-independent sets. So, the obtained k-independent dominating set contains at most $\lfloor (2n)/(k+2) \rfloor$ nodes.
Furthermore, 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. The symbol $\perp$ denotes a value smaller than any identifier value in the network.

Each node v maintains a set of shared variables such that v 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. 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 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.

During a *computation step* $c_i \rightarrow c_{i+1}$, one or several enabled nodes perform an enabled action and the system reaches the configuration c_{i+1} from c_i. 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 one computation step: $\forall i \geqslant 0, c_i \rightarrow c_{i+1}$. We say that a computation e is *maximal* if it is infinite, or if it reaches a terminal configuration. We note by $\mathcal{C}$ the set of all possible configurations, and by $\mathcal{E}$ the set of all maximal computations. The set of maximal computations starting from a particular configuration $c \in \mathcal{C}$ is denoted $\mathcal{E}_c$. $\mathcal{E}_A$ denotes the set of all maximal computations where the initial configuration belongs to the set of configurations $A \subset \mathcal{C}$.

Definition 2 (Attractor) *Let B_1 and B_2 be subsets of $\mathcal{C}$. B_2 is an attractor from B_1, if and only if the following conditions hold:*

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

Definition 3 (Self-stabilization) *A distributed system S is self-stabilizing for the specification $\mathcal{SP}$ (a predicate on configurations) 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 $\mathcal{C}$;*
- *Configurations of $\mathcal{L}$ satisfied $\mathcal{SP}$.*

A self-stabilizing protocol is *silent* if all maximal computations are finite.

Stabilization time. 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 $e_1 = c_1, ..., c_j$, such that every enabled node in c_1 either executes a rule or it is neutralized during a computation step of e_1. A node v is *neutralized* during a computation step cs $c_i \rightarrow c_{i+1}$, if v is enabled in c_i and disabled in c_{i+1}, but it did not execute any action during cs.
Let e_2 be the suffix of e such that $e = e_1 e_2$. The second round of e is the first round of e_2, and so on.

The stabilization time is the number of rounds of a computation reaching a legitimate configuration from any initial one.

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 4 ***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 5 ***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-augmented value $a1$ dominates the k-augmented 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 belonging to X such that any value of X is dominated by $dom(X)$ (i.e. $\forall y \in X$ we have $dom(dom(X), y) = dom(X)$).*

Definition 6 ***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-augmented value $a1$ is larger than the k-augmented 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 belonging to X such that any value of X is larger than $min(X)$ (i.e. $\forall y \in X$ we have $min(min(X), y) = min(X)$).*

The node $u1$ is *closer* to the node v than the node $u2$ iff $dist(u1, v)) < dist(u2, v)$ or $id_{u1} < id_{u2}$. We have $min((dist(u1, v), id_{u1}), (dist(u2, v), id_u)) = (dist(u1, v); id_{u1})$.

Definition 7 ***The operation +1*** *on k-augmentedID is defined as follow : $a + 1 = a$ if $a = \perp$ or if $a.dist = k$ otherwise $a + 1 = (a.dist + 1, a.id)$*

Protocol 1 : Variables, Predicates and Rules of the Protocol $\mathcal{SID}$ on the node v

Shared variables

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

Internal variable

- *beReal* a boolean variables used by some macros.

Notation

- $\texttt{firstAugmentedIdSet}(v) = \{a + 1 \in k\text{-augmentedID} \mid a = \texttt{firstHead}(u) \vee a = \texttt{secondHead}(u) \ with\ u \in N_v \wedge a.dist < k \wedge a.id \neq id_v\}$
- $\texttt{secondAugmentedIdSet}(v) = \{a \in \texttt{firstAugmentedIdSet} \mid a.id \neq \texttt{firstHead}(v).id\}$

Macros

- $\texttt{isDefended}(v)$ returns true iff $\texttt{firstAugmentedIdSet}(v) \neq \emptyset$.
- $\texttt{isDominated}(v)$ returns true iff $id_v^+ \neq dom(\texttt{firstAugmentedIdSet}(v) \cup id_v^+)$.
- $\texttt{correctFirstHead}(v)$ returns true iff $\texttt{firstHead}(v) == min(\texttt{firstAugmentedIdSet}(v))$.
- $\texttt{computingFirstHead}(v)$ returns the value of $min(\texttt{firstAugmentedIdSet}(v))$.
- $\texttt{correctSecondHead}(v)$ returns true iff $\texttt{secondHead}(v) == min(\texttt{secondAugmentedIdSet}(v) \cup \perp)$.
- $\texttt{computingsSecondHead}(v)$ returns the value of $min(\texttt{secondAugmentedIdSet}(v) \cup \perp)$.

Predicates

- $\texttt{Head}(v) \equiv \texttt{firstHead}(v) == (0, id_v)$
- $\texttt{toResign}(v) \equiv \texttt{isDominated}(v)$
- $\texttt{toElect}(v) \equiv \neg\texttt{isDefended}(v)$
- $\texttt{headToUpdate}(v) \equiv \texttt{firstHead}(v) \neq (0, id_v) \vee \texttt{secondHead}(v) \neq \perp$
- $\texttt{ordinaryToUpdate}(v) \equiv \neg\texttt{correctFirstHead}(v) \vee \neg\texttt{correctSecondHead}(v)$

Rules

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

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

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

$\mathbf{RC}(v) : \texttt{Head}(v) \wedge \neg\texttt{toResign}(v) \wedge \texttt{headToUpdate}(v) \longrightarrow \texttt{firstHead}(v) := (0, id_v);\ \texttt{secondHead}(v) := \perp;$

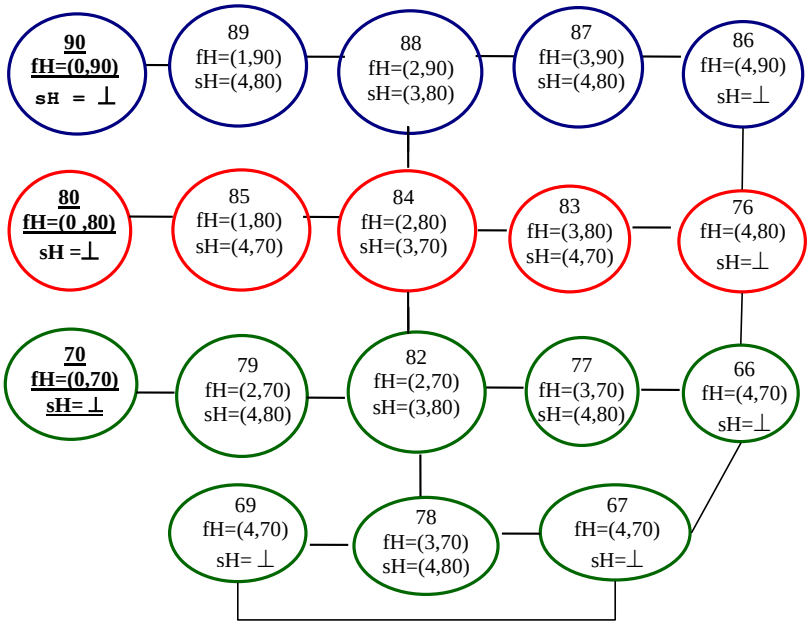
3.2 Code of the protocol $\mathcal{SID}$

A node v is said to be a head if $\texttt{firstHead}(v) = id_v^+$; otherwise it is an ordinary node. The heads set built by the protocol $\mathcal{SID}$ (defined in protocol 1) is a k-independent dominating set. The codes of the macros are defined in the protocols 2 to 7.

In the figure 1 is presented a terminal configuration having three heads. On the same network is presented another terminal configuration having a single head, in the figure 2.

The variable $\texttt{firstHead}(v)$ contains the identifier of the closest head to v (with its distance to v).

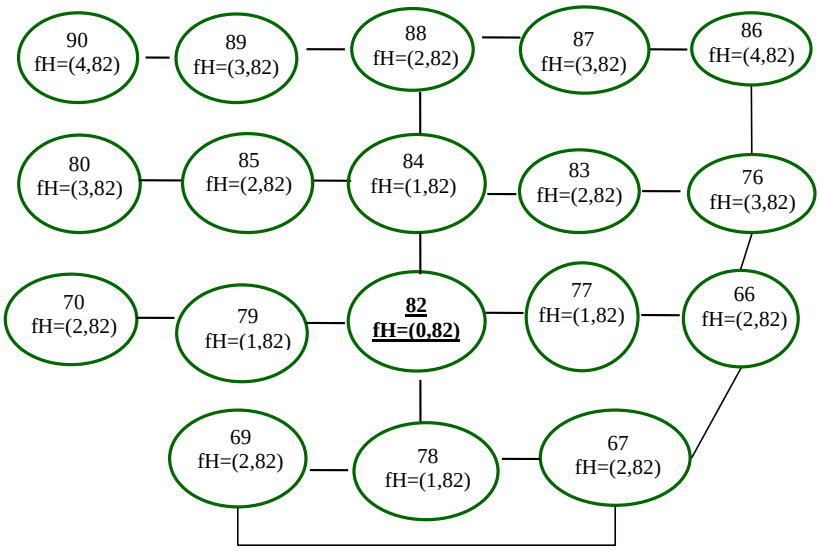
The variable $\texttt{secondHead}(v)$ contains the identifier of the second closest head to v (with its distance to v) inside its `k-neighborhood`. If a node v does not have two heads in its `k-neighborhood` then $\texttt{secondHead}(v)$ is set to $\perp$. The execution of the rules **RU** or the rules **RC** updates the two variables $\texttt{firstHead}(v)$, and $\texttt{secondHead}(v)$ without changing the status of v (i.e. v stays ordinary or head).



k=4. The head identifiers are underlined. In each node, it is indicated the value of **`firstHead`** and the value of **`secondHead`**. The color of a node is the color of its closest head.

Figure 1: A terminal configuration of $\mathcal{SID}$

The macro $\texttt{isDefended}(v)$ (defined in protocol 2) returns true if



k=4. The head identifier is underlined. In each node, it is indicated the value of **`firstHead`**

Figure 2: Another terminal configuration of $\mathcal{SID}$

the set `firstAugmentedIdSet`(v) is not empty otherwise the macro returns false.

Protocol 2 : macro `isDefended`(v)

---

for $u \in N_v$ do
  if `firstHead`$(u).dist < k \wedge$ `firstHead`$(u).id \neq id_v$ then return true;
  if `secondHead`$(u).dist < k \wedge$ `secondHead`$(u).id \neq id_v$ then return true;
done; return false;

The macro `isDominated`(v) (defined in protocol 3) returns true if a value x of `firstAugmentedIdSet`(v) dominates the value id_v^+; otherwise the macro returns false.

The macro `correctFirstHead`(v) (defined in protocol 4) returns true if the value of `firstHead`(v) is min(`firstAugmentedIdSet`(v)); otherwise the macro returns false. Notice that if the set `firstAugmentedIdSet`(v)) is empty the macro returns false;

The macro `computingFirstHead`(v) (defined in protocol 5) returns min(`firstAugmentedIdSet`(v)) if the set `firstAugmentedIdSet`(v) is not empty; otherwise the macro returns $\perp$.

The macro `correctsecondHead`(v) (defined in protocol 6) returns true if the value of

Protocol 3 : macro `isDominated`(v)

for $u \in N_v$ do
  if `firstHead`(u).$dist < k \wedge$ `firstHead`(u).$id < id_v$ then return true;
  if `secondHead`(u).$dist < k \wedge$ `secondHead`(u).$id < id_v$ then return true;
done; return false;

Protocol 4 : Macro `correctFirstHead`(v)

$beReal :=$ false;
for $u \in N_v$ do
  if `firstHead`(u).$id \neq id_v \wedge$ `firstHead`(u).$dist < k$ then
    if `firstHead`(v) $\neq min($`firstHead`(v), `firstHead`(u) $+ 1)$ then return false;
    if `firstHead`(v) $=$ (`firstHead`(u) $+ 1)$ then $beReal :=$ true;
  if `secondHead`(u) $\neq \perp \wedge$ `secondHead`(u).$id \neq id.v \wedge$ `secondHead`(u).$dist < k$ then
    if `firstHead`(v) $\neq min($`firstHead`(v), `secondHead`(u) $+ 1)$ then return false;
    if `firstHead`(v) $=$ (`secondHead`(u) $+ 1)$ then $beReal :=$ true;
done; return $beReal$;

`secondHead`(v) is $min($`firstAugmentedIdSet`(v)$\cup\perp)$; otherwise the macro returns false.

The macro `computingSecondHead`(v) (defined in protocol 7) returns $min($`secondAugmentedIdSet`(v)) if the set `secondAugmentedIdSet`(v) is not empty; otherwise the macro returns $\perp$.

Once the system is stabilized, the set `firstAugmentedIdSet`(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 in the `k-neighborhood` of v. If the k's neighborhood of a node v does not contain any head then the set `firstAugmentedIdSet`(v)) is empty. So the predicate `isDefended`(v) is not verified. If v is an ordinary node then v is enabled (the rule **RE** or the rule **RU** is enabled). Therefore, the heads set is a k-dominating set, in a terminal configuration.

If `toResign`(v) is verified then v has in its `k-neighborhood` a head u having a smaller identifier than v's identifier (i.e. $id_v > id_u$). In this case, if v is a head, it is enabled. So, the set of heads is a k-independent set, in any terminal configuration.

The proof of the protocol $\mathcal{SID}$ has two parts.

- In the section 4, we prove that a terminal configuration of $\mathcal{SID}$ protocol is legitimate : the set of heads is a k-independent dominating set.
- In the section 5, we prove that all maximal computations under any unfair dis-

Protocol 5 : Macro `computingFirstHead`(v)

`firstHead`(v) :=$\perp$;
for $u \in N_v$ do
  if `firstHead`$(u).id \neq id_v \wedge$ `firstHead`$(u).dist < k$ then
    `firstHead`$(v) := min($`firstHead`$(v),$ `firstHead`$(u) + 1)$;
  if `secondHead`$(u) \neq \perp \wedge$ `secondHead`$(u).id \neq id.v \wedge$ `secondHead`$(u).dist < k$ then
    `firstHead`$(v) := min($`firstHead`$(v),$ `secondHead`$(u) + 1)$;
done;

Protocol 6 : Macro `correctSecondHead`(v)

$beReal$:= false;
for $u \in N_v$ do
  if `firstHead`$(u).id \neq id_v \wedge$ `firstHead`$(u).id \neq$ `firstHead`$(v).id \wedge$
    `firstHead`$(u).dist < k$ then
    if `secondHead`$(v) \neq (min($`secondHead`$(v),$ `firstHead`$(u) + 1)$ then return false;
    if `secondHead`$(v) =$ `firstHead`$(u) + 1$ then $beReal$ = true;
  if `secondHead`$(u) \neq \perp \wedge$ `secondHead`$(u).id \neq id_v \wedge$
    `secondHead`$(u).id \neq$ `firstHead`$(v).id \wedge$ `secondHead`$(u).dist < k$ then
    if `secondHead`$(v) \neq min($`secondHead`$(v),$ `secondHead`$(u) + 1)$ then return false;
    if `secondHead`$(v) =$ `secondHead`$(u) + 1$ then $beReal$ = true;
done;
if $\neg beReal$ and (`secondHead`$(v) \neq \perp$) then return false;
return true;

Protocol 7 : Macro `computingSecondHead`(v)

`secondHead`(v) :=$\perp$;
for $u \in N_v$ do
  if `firstHead`$(u).id \neq id_v \wedge$ `firstHead`$(u).id \neq$ `firstHead`$(v).id \wedge$
    `firstHead`$(u).dist < k$ then
      `secondHead`$(v) := (min($`secondHead`$(v),$ `firstHead`$(u) + 1)$;
  if `secondHead`$(u) \neq \perp \wedge$ `secondHead`$(u).id \neq id_v \wedge$
    `secondHead`$(u).id \neq$ `firstHead`$(v).id \wedge$ `secondHead`$(u).dist < k$ then
      `secondHead`$(v) := min($`secondHead`$(v),$ `secondHead`$(u) + 1)$;
done;

---

tributed scheduler are finite by *reductio ad absurdam* arguments.

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

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

Definition 8 *The property* `OrdinaryPr`*(i) defined for all* $i \in [1, k]$ *is verifierd if the two following statements are satisfied:*

- `OrdinaryPrFirst`*(i): for all ordinary node* v, `firstHead`$(v) = (i, id_u)$ *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, `secondHead`$(v) = (i, id_w)$ *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* `OrdinaryToUpdate`(v)*;*
   so `firstHead`$(v) = min($`firstAugmentedIdSet`$(v))$ *and*
   `secondHead`$(v) = min($`secondtAugmentedIdSet`$(v) \cup \perp)$.
2. *A head* u *does not verify* `HeadToUpdate`(u)*;*
3. *Let* w *be a node (head or ordinary),* `firstHead`$(w) \neq \perp$*;*
4. *if* v *is an ordinary node then* `firstHead`$(v).dist > 0$ *;*
5. *if* `secondHead`$(v) \neq \perp$ *then* `secondHead`$(v).dist > 0$ *;*
6. *if* `secondHead`$(v) \neq \perp$ *then* `secondHead`$(v).dist \geq$ `firstHead`$(v).dist$ *because* `secondAugmentedIdSet`$(v) \subset$ `firstAugmentedIdSet`(v).

Lemma 2 *In a terminal configuration of protocol* $\mathcal{SID}$*, the property* `OrdinaryPr`(1) *is verified.*

Proof 2
Let v *be an ordinary node, in a terminal configuration of protocol* $\mathcal{SID}$*, named* c.
Assume that $(1, x) \in$ `firstAugmentedIdSet`(v). *So* v *has a neighbor* u *such that* `firstHead`$(u) = (0, x)$ *or* `secondHead`$(u) = (0, x)$.
According to observation 1.4 `secondHead`$(u).dist > 0$ *or* `secondHead`$(u) = \perp$. *So* v *has a neighbor* u *such that* `firstHead`$(u) = (0, x)$. *According to observation 1.3* u *is a head; so* $x = id_u$.
Notice that $\forall a \in$ `firstAugmentedIdSet`(v)*, we have* $a.dist > 0$*, in* c.

Proof of `OrdinaryPrFirst`*(1). If* v *has a head at distance* 1 *then* v *has a neighbor* u *such that* `firstHead`$(u) = (0, id_u)$. *So, We have* `firstHead`$(v) = (1, id_u)$ *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* $\texttt{firstHead}(u).dist > 0$. *and* $\texttt{secondHead}(u).dist > 0$ *or* $\texttt{secondHead}(u) = \perp$ *(according to observation 1.4). In this case,* $\texttt{firstHead}(v).dist > 1$.

Proof of `OrdinaryPrSecond`*(1). If v has several heads at distance* 1 *then v has a neighbor w such that* $\texttt{firstHead}(w) = (0, id_w)$ *with* $id_w \neq \texttt{firstHead}(v).id$. *So,* $\texttt{secondHead}(v) = (1, id_w)$ *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* $\texttt{firstHead}(w) = (0, id_w)$ *with* $id_w \neq \texttt{firstHead}(v).id$. *In this case,* $\texttt{secondHead}(v).dist$ *is larger than* 1 *or* $\texttt{secondHead}(v) = \perp$. ■

Lemma 3 *Let i be a positive integer smaller than k. In a terminal configuration of protocol* $\mathcal{SID}$, *if the properties* `OrdinaryPr(j)` *are verified for all* $j \in [1, i]$ *then the property* $\texttt{OrdinaryPr}(i+1)$ *is verified.*

Proof 3 *Let us assume that the properties* `OrdinaryPr(j)` *are verified for all* $j \in [1, i]$ *in any terminal configuration of protocol* $\mathcal{SID}$.
In a terminal configuration c, $(j, x) \in \texttt{augmentedIdSet}(v)$ *iff v has a neighbor u such that* $\texttt{firstHead}(u) = (j-1, x)$, *or* $\texttt{secondHead}(u) = (j-1, x)$. *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 `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* $\texttt{firstHead}(u) = (d'-1, id_{v'})$, *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* $(l, id) \in \texttt{firstAugmentedIdSet}(v)$ *then* $l \geq d'$; *and*
- *if* $(d', id) \in \texttt{firstAugmentedIdSet}(v)$ *then* $id \geq id_{v'}$. *In c,*

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

Proof of `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" ≤ i + 1. According to the property* $\texttt{OrdinaryPr}(d"-1)$, $\texttt{firstHead}(u) = (d"-1, id_{v"}) \vee \texttt{secondHead}(u) = (d"-1, id_{v"})$, *in c. According to the properties* $\texttt{OrdinaryPr}(j)$ *forall* $j \in [1, i)$, *in c, we have the following properties,*

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

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

The following corollary is a direct result of lemmas 2 and 3. It establishes that the set of heads is a k-dominating set.

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

The following theorem establishes that the set of heads is a k-independent set.

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

Proof 4 *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 denoted 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, $(d-1, id_{v2}) =$ $\texttt{firstHead}(u)$ *or $(d-1, id_{v2}) =$* $\texttt{secondHead}(u)$. *$v1$ 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.

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

5.1 RR and RE rules

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

Assume that a or several nodes perform infinitely often the action **RE** or the action **RR**. 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 perform the action **RE** or the action **RR** changes its status infinitely often. We name u^+ the node the smallest identifier among nodes that changes their status infinitely often. e has a suffix $e1$ where only nodes having a identifier larger than id_{u^+} changes their status (i.e. to perform the action **RE** or the action **RR**).

As the set of value taken by `firstHead`(u^+) is bounded; along $e1$, infinitely often after the action **RR**(u^+), `firstHead`(u^+) has the same value, denoted $(l+1, id)$. Notice that $id < id_{u^+}$ and $0 < l < k$. So u^+ has a neighbor u_l such that, infinitely often before the action **RR**(u^+), u_l verfies `firstHead`$(u_l) = (l, id)$ or `secondHead`$(u_l) = (l, id)$.
At time, where u^+ becomes head, we have `firstAugmentedIdSet`$(u^+) = \emptyset$. So, the values of u_l variables are infinitely often larger than (l, id) : So 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.
So e has a suffix, named $e2$, in which the only rule performed is **RU**.

5.2 RU rule

Assume that a node or several nodes changing infinitely often their value `firstHead` or their value `secondHead` along $e2$. We named min^+ the smallest value infinitely often allocates to the variable `firstHead` or to the variable `secondHead` of one of these nodes. Let $e3$ be the suffix of $e2$ in which no variable `firstHead` and no variable `secondHead` 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 `firstHead` or its variable `secondHead`; and infinitely often, u^+ performs **RU** action to set to the same variable a value larger than min^+.

Let $c \rightarrow c'$ be a computation step of $e3$ where u^+ performs **RU** action to set a value larger than min^+ to its variable `firstHead` or to its variable `secondHead`. In c, min^+ is smaller than min(`firstAugmentedIdSet`(u^+)) or min^+ is smaller than
min(`secondAugmentedIdSet`(u^+)) . This property stays verified along $e3$: u^+ never sets the value min^+ to its variable `firstHead` (resp. to its variable `secondHead`). There is a contradiction.

We have established that $e2$ has a suffix $e4$ where no rule is executed. A terminal configuration is reached.

6 Conclusion

A simple and silent self-stabilizing protocols building k-independent dominating sets is presented. The obtained k-independent dominating set contains at most $\lfloor (2n)/(k+2) \rfloor$ nodes. The protocol converges under the unfair distributed scheduler (the weakest scheduling assumption). The protocol is memory efficient : it requires only $2log((k+1)n+1)+1$ bits per node.

The computation of the convergence time of the protocol $\mathcal{SID}$ is opened question.

References

[1] D. Bein, A. K. Datta, C. R. Jagganagari, and V. Villain. A self-stabilizing link-cluster algorithm in mobile ad hoc networks. In *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05)*, pages 436–441, 2005.

[2] A. Bui, S. Clavière, A. K. Datta, L. L. Larmore, and D. Sohier. Self-stabilizing hierarchical construction of bounded size clusters. In *18th International Colloquium Structural Information and Communication Complexity (SIROCCO'11), Springer LNCS 6796*, pages 54–65, 2011.

[3] E. Caron, A. K. Datta, B. Depardon, and L. L. Larmore. self-stabilizing k-clustering algorithm for weighted graphs. *Journal of Parallel and Distributed Computing*, 70:1159–1173, 2010.

[4] A. Datta, S. Devismes, and L. Larmore. A self-stabilizing $o(n)$-round k-clustering algorithm. In *28th IEEE Symposium on Reliable Distributed Systems (SRDS'09)*, pages 147–155, 2009.

[5] A. K. Datta, L. L. Larmore, S. Devismes, K. Heurtefeux, and Y. Rivierre. Competitive self-stabilizing k-clustering. In *IEEE 32th International Conference on Distributed Computing (ICDCS'12)*, pages 476–485, 2012.

[6] A. K. Datta, L. L. Larmore, S. Devismes, K. Heurtefeux, and Y. Rivierre. Self-stabilizing small k-dominating sets. *International Journal of Networking and Computing*, 3(1):116–136, 2013.

[7] A. K. Datta, L. L. Larmore, and P. Vemula. A self-stabilizing o(k)-time k-clustering algorithm. *The Computer Journal*, 53(3):342–350, 2010.

[8] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani. A fault-local self-stabilizing clustering service for wireless ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(9):912–922, 2006.

[9] S. Dolev and N. Tzachar. Empire of colonies self-stabilizing and self-organizing distributed algorithms. In *the 10th International Conference On Principles Of Distributed Systems (OPODIS'06), Springer LNCS 4305*, pages 230–243, 2006.

[10] V. Drabkin, R. Friedman, and M. Gradinariu. Self-stabilizing wireless connected overlays. In *Conference On Principles Of Distributed Systems (OPODIS'06), Springer LNCS 4305*, pages 425–439, 2006.

[11] W. Goddard, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In *the 5th IPDPS Workshop on Advances in Parallel and Distributed Computational Models (WAPDCM'03)*, 2003.

[12] C. Johnen. Fast self-stabilizing k-independant dominating set construction. Technical Report RR-1472-13, Univ. Bordeaux, LaBRI, UMR 3800, F-33400 Talence, France, June 2013.

[13] C. Johnen and F. Mekhaldi. Robust self-stabilizing construction of bounded size weight-based clusters. In *the 16th International Euro-Par Conference (Euro-Par'10), Springer LNCS 6271*, pages 535–546, 2010.

[14] C. Johnen and L. H. Nguyen. Self-stabilizing construction of bounded size clusters. In *the IEEE 8th International Symposium on Parallel and Distributed Processing and Applications (ISPA'08)*, pages 43–50, 2008.

[15] C. Johnen and L. H. Nguyen. Robust self-stabilizing weight-based clustering algorithm. *Theoretical Computer Science*, 410(6-7):581–594, 2009.

[16] S. Kamei and H. Kakugawa. A self-stabilizing approximation for the minimum connected dominating set with safe convergence. In *the 12th Conference On Principles Of Distributed Systems (OPODIS'08), Springer, LNCS 5401*, pages 496–511, 2008.

[17] A. Larsson and P. Tsigas. 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)*, pages 353–362. IEEE Computer Society, 2011.

[18] A. Larsson and P. Tsigas. Self-stabilizing (k,r)-clustering in clock rate-limited systems. In *19th International Colloquium Structural Information and Communication Complexity, (SIROCCO'12), Springer, LNCS 7355*, pages 219–230, 2012.

[19] N. Mitton, E. Fleury, I. Guérin-Lassous, and S. Tixeuil. Self-stabilization in self-organized multihop wireless networks. In *International Conference on Distributed Computing Systems Workshops (WWAN'05)*, pages 909–915, 2005.

[20] Z. Xu, S. T. Hedetniemi, W. Goddard, and P. K. Srimani. A synchronous self-stabilizing minimal domination protocol in an arbitrary network graph. In *Proceedings of the 5th International Workshop on Distributed Computing (IWDC'03), Springer LNCS 2918*, pages 26–32, 2003.