1 Introduction

Maximal independent sets have received a lot of attention in different areas. For instance, in wireless networks, the maximum independent sets can be used as a black box to perform communication (to collect or to broadcast information) (see [20, 9] for example). In self-stabilizing distributed algorithms, this problem is also a fundamental tool to transform an algorithm from one model to another [13, 29].

An independent set $I$ in a graph is a set of vertices such that no two of them form an edge in the graph. It is called maximal when it is maximal inclusion-wise (in which case it is also a minimal dominating set).

The maximal independent set (MIS) problem has been extensively studied in parallel and distributed settings, following the seminal works of [1, 19, 21]. Their idea is based on the fact that a node joins the “MIS under construction” $S$ according to the neighbors: node $v$ joins the set $S$ if it has no neighbor in $S$, and it leaves the set $S$ if at least one of its neighbors is in $S$. Most algorithms in the literature, including ours, are based on this approach.

The MIS problem has been extensively studied in the LOCAL model, [10, 23, 5] for instance (a synchronous, message-passing model of distributed computing in which messages can be arbitrarily large) and in the CONGEST model [22] (synchronous model where messages are $O(\log n)$ bits long). In the LOCAL model, Barenboim et al. [3] focus on an identified system and give a self-stabilizing algorithm producing a MIS within $O(\Delta + \log^* n)$ rounds. Balliu et al. [2] prove that the previous algorithm [3] is optimal for a wide range of parameters in the LOCAL model. In the CONGEST model, Ghaffari et al. [11] prove that there exists a randomized distributed algorithm that computes a maximal independent set in $O(\log \Delta \cdot \log \log n + \log \log n)$ rounds with high probability.

Self-stabilizing algorithms for maximal independent set have been designed in various models (anonymous network [25, 29, 28] or not [12, 16]). Up to our knowledge, Shukla et al. [25] present the first algorithm designed for finding a MIS in a graph using self-stabilization paradigm for anonymous networks. Some other self-stabilizing works deal with this problem assuming identifiers: with a synchronous daemon [12] or distributed one [16]. These two works require $O(n^2)$ moves to converge. Turau [27] improves these results to $O(n)$ moves under the distributed daemon. Recently, some works improved the results in the synchronous model. For non-anonymous networks, Hedetniemi [15] designed a self-stabilization algorithm for solving the problem related to dominating sets in graphs in particular for a maximal independent set which stabilizes in $O(n)$ synchronous rounds. Moreover, for anonymous networks, Turau [28] design some Randomized self-stabilizing algorithms for maximal independent set w.h.p. in $O(\log n)$ rounds. See the survey [14] for more details on MIS self-stabilizing algorithms.

Some variant of the maximal independent set problem have been investigated, as for example the 1-maximal independent set problem [26, 24] or Maximal Distance-$k$ Independent Set [4, 17]. Tanaka et al. [26] designed a silent
self-stabilizing 1-MIS algorithm under the weakly-fair distributed daemon for any identified network in \( O(nD) \) rounds (where \( D \) is a diameter of the graph).

In this paper, we focus on the construction of a MIS handling both transient and Byzantine faults. On one side, transient faults can appear in the whole system, possibly impacting all nodes. However, these faults are not permanent, thus they stop at some point of the execution. Self-stabilization \(^{[7]}\) is the classical paradigm to handle transient faults. Starting from any arbitrary configuration, a self-stabilizing algorithm eventually resumes a correct behavior without any external intervention. On the other side, (permanent) Byzantine faults \(^{[18]}\) are located on some faulty nodes and so the faults only occur from them. However, these faults can be permanent, i.e., they could never stop during the whole execution.

In a distributed system, multiple processes can be active at the same time, meaning they are in a state where they could make a computation. The definition of self-stabilizing algorithm is centered around the notion of daemon, which captures the ways the choice of which process to schedule for the next time step by the execution environment can be made. Two major types of daemon are the sequential and the distributed ones. A sequential daemon only allows one process to be scheduled for a given time step, while a distributed daemon allows the execution of multiple processes at the same time. Daemons can also be fair when they have to eventually schedule every process that is always activable, or adversarial when they are not fair. As being distributed instead of sequential (or adversarial instead of fair) only allows for more possibilities of execution, it is harder to make an algorithm with the assumption of a distributed (resp. adversarial) daemon than with the assumption of a sequential (resp. fair) daemon.

We introduce the possibility that some nodes, that we will call Byzantine nodes, are not following the rules of the algorithm, and may change the values of their attributes at any time step. Here, if we do not work under the assumption of a fair daemon, one can easily see that we cannot guarantee the convergence of any algorithm as the daemon could choose to always activate alone the same Byzantine node, again and again.

Under the assumption of a fair daemon, there is a natural way to express complexity, not in the number of moves performed by the processes, but in the number of rounds, where a round captures the idea that every process that wanted to be activated at the beginning of the round has either been activated or changed its mind. We give a self-stabilizing randomized algorithm that, in an anonymous network (which means that processes do not have unique identifiers to identify themselves) with Byzantine nodes under the assumption of a distributed fair daemon, finds a maximal independent set of a superset of the nodes at distance 3 or more from Byzantine nodes. We show that the algorithm stabilizes in \( O(\Delta n) \) rounds w.h.p., where \( n \) is the size and \( \Delta \) is the diameter of the underlying graph.

In this paper, we first present the model (Section \(^{[2]}\)). Then, we give a self-stabilizing randomized algorithm with Byzantine nodes under the fair daemon (Section \(^{[3]}\), which converges in \( O(\Delta n) \) rounds.

Then, in the last part, we give a self-stabilizing randomized algorithm that finds a maximal independent set in an anonymous network, under the assumption of a distributed adversarial daemon. We show that our algorithm converge in \( O(n^2) \) moves with high probability.

## 2 Model

A system consists of a set of processes where two adjacent processes can communicate with each other. The communication relation is represented by a graph \( G = (V, E) \) where \( V \) is the set of the processes (we will call node any element of \( V \) from now on) and \( E \) represents the neighbourhood relation between them, i.e., \( uv \in E \) when \( u \) and \( v \) are adjacent nodes. By convention we write \([V] = n\) and \([E] = m\). If \( u \) is a node, \( N(u) = \{v \in V | uv \in E\} \) denotes the open neighbourhood, and \( N[u] = N(u) \cup \{u\} \) denotes the closed neighbourhood. We note \( deg(u) = |N(u)| \) and \( \Delta = \max \{deg(u) | u \in V\} \).

We assume the system to be anonymous meaning that a node has no identifier. We use the state model, which means that each node has a set of local variables which make up the local state of the node. A node can read its local variables and all the local variables of its neighbours, but can only rewrite its own local variables. A configuration is the value of the local states of all nodes in the system. When \( u \) is a node and \( x \) a local variable, the \( x\)-value of \( u \) in configuration \( \gamma \) is the value \( x^\gamma_u \).

An algorithm is a set of rules, where each rule is of the form \( \langle \text{guard} \rangle \rightarrow \langle \text{command} \rangle \) and is parametrized by the node where it would be applied. The \( \text{guard} \) is a predicate over the variables of the said node and its neighbours. The \( \text{command} \) is a sequence of actions that may change the values of the node’s variables (but not those of its neighbours). A rule is enabled on a node \( u \) in a configuration \( \gamma \) if the guard of the rule holds on \( u \) in \( \gamma \). A node is activable on a
configuration $\gamma$ if at least one rule is enabled on $u$. We call move any ordered pair $(u, r)$ where $u$ is a node and $r$ is a rule. A move is said possible in a given configuration $\gamma$ if $r$ is enabled on $u$ in $\gamma$.

The activation of a rule on a node may only change the value of variables of that specific node, but multiple moves may be performed at the same time, as long as they act on different nodes. To capture this, we say that a set of moves $t$ is valid in a configuration $\gamma$ when it is non-empty, contains only possible moves of $\gamma$, and does not contain two moves concerning the same node. Then, a transition is a triplet $(\gamma, t, \gamma')$ such that: (i) $t$ is a valid set of moves of $\gamma$ and (ii) $\gamma'$ is a possible configuration after every node $u$ appearing in $t$ performed simultaneously the code of the associated rule, beginning in configuration $\gamma$. We will write such a triplet as $\gamma \xrightarrow{t} \gamma'$. We will also write $\gamma \rightarrow \gamma'$ when there exists a transition from $\gamma$ to $\gamma'$. $V(t)$ denotes the set of nodes that appear as first member of a couple in $t$.

We say that a rule $r$ is executed on a node $u$ in a transition $\gamma \xrightarrow{t} \gamma'$ (or equivalently that the move $(u, r)$ is executed in $\gamma \xrightarrow{t} \gamma'$) when the node $u$ has performed the rule $r$ in this transition, that is when $(u, r) \in t$. In this case, we say that $u$ has been activated in that transition. Then, an execution is an alternate sequence of configurations and move sets $\gamma_0, \gamma_1, \gamma_2, \cdots$ where (i) the sequence either is infinite or finishes by a configuration and (ii) for all $i \in \mathbb{N}$ such that it is defined, $(\gamma_i, t_{i+1}, \gamma_{i+1})$ is a transition. We will write such an execution as $\gamma_0 \xrightarrow{t_1} \gamma_1 \cdots \xrightarrow{t_i} \gamma_i \cdots$. When the execution is finite, the last element of the sequence is the last configuration of the execution. An execution is maximal if it is infinite, or it is finite and no node is activable in the last configuration. It is called partial otherwise. We say that a configuration $\gamma'$ is reachable from a configuration $\gamma$ if there exists an execution starting in configuration $\gamma$ that leads to configuration $\gamma'$. We say that a configuration is stable if no node is activable in that configuration.

The daemon is the adversary that chooses, from a given configuration, which nodes to activate in the next transition. Two types are used: the adversarial distributed daemon that allows all possible executions and the fair distributed daemon that only allows executions where nodes cannot be continuously activable without being eventually activated.

Given a specification and $\mathcal{L}$ the associated set of legitimate configuration, i.e., the set of the configurations that verify the specification, a probabilistic algorithm is self-stabilizing when these properties are true: (correctness) every configuration of an execution starting by a configuration of $\mathcal{L}$ is in $\mathcal{L}$ and (convergence) from any configuration, whatever the strategy of the daemon, the resulting execution eventually reaches a configuration in $\mathcal{L}$ with probability 1.

The time complexity of an algorithm that assumes the fair distributed daemon is given as a number of rounds. The concept of round was introduced by Dolev et al. [8], and reworded by Cournier et al. [6] to take into account activable nodes. We quote the two following definitions from Cournier et al. [6]: “

**Definition 1** We consider that a node $u$ executes a disabling action in the transition $\gamma_1 \rightarrow \gamma_2$ if $u$ (i) is activable in $\gamma_1$, (ii) does not execute any rule in $\gamma_1 \rightarrow \gamma_2$ and (iii) is not activable in $\gamma_2$.

The disabling action represents the situation where at least one neighbour of $u$ changes its local state in $\gamma_1 \rightarrow \gamma_2$, and this change effectively made the guard of all rules of $u$ false in $\gamma_2$. The time complexity is then computed capturing the speed of the slowest node in any execution through the round definition [8].

Given an execution $\mathcal{E}$, the first round of $\mathcal{E}$ (let us call it $R_1$) is the minimal prefix of $\mathcal{E}$ containing the execution of one action (the execution of a rule or a disabling action) of every activable nodes from the initial configuration. Let $\mathcal{E}'$ be the suffix of $\mathcal{E}$ such that $\mathcal{E} = R_1 \mathcal{E}'$. The second round of $\mathcal{E}$ is the first round of $\mathcal{E}'$, and so on.”

Observe that Definition ?? is equivalent to Definition ??, which is simpler in the sense that it does not refer back to the set of activable nodes from the initial configuration of the round.

Let $\mathcal{E}$ be an execution. A round is a sequence of consecutive transitions in $\mathcal{E}$. The first round begins at the beginning of $\mathcal{E}$; successive rounds begin immediately after the previous round has ended. The current round ends once every node $u \in V$ satisfies at least one of the following two properties: (i) $u$ has been activated in at least one transition during the current round or (ii) $u$ has been non-activable in at least one configuration during the current round.

Our first algorithm is to be executed in the presence of Byzantine nodes; that is, there is a subset $B \subseteq V$ of adversarial nodes that are not bound by the algorithm. Byzantine nodes are always activable. An activated Byzantine node is free to update or not its local variables. Finally, observe that in the presence of Byzantine nodes all maximal executions are infinite. We denote by $d(u, B)$ the minimal (graph) distance between node $u$ and a Byzantine node, and we define for $i \in \mathbb{N}$: $V_i = \{u \in V|d(u, B) > i\}$. Note that $V_0$ is exactly the set of non-Byzantines nodes, and that $V_{i+1}$ is exactly the set of nodes of $V_i$ whose neighbours are all in $V_i$.

When $i, j$ are integers, we use the standard mathematical notation for the integer segments: $[i, j]$ is the set of integers that are greater than or equal to $i$ and smaller than or equal to $j$ (i.e. $[i, j] = [i, j] \cap \mathbb{Z} = \{i, \cdots, j\}$). $\operatorname{Rand}(x)$ with $x \in [0, 1]$ represents the random function that outputs 1 with probability $x$, and 0 otherwise.
3 With Byzantines Nodes under the Fair Daemon

3.1 The algorithm

The algorithm builds a maximal independent set represented by a local variable $s$. The approach of the state of the art is the following: when two nodes are candidates to be in the independent set, then a local election decides who will remain in the independent set. To perform a local election, the standard technique is to compare the identifiers of nodes. Unfortunately, this mechanism is not robust to the presence of Byzantine nodes.

Keeping with the approach outlined above, when a node $u$ observes that its neighbours are not in (or trying to be in) the independent set, the non-Byzantine node decides to join it with a certain probability. The randomization helps to reduce the impact of Byzantine nodes. The choice of probability should reduce the impact of Byzantine nodes while maintaining the efficiency of the algorithm.

Algorithm 1 Any node $u$ has two local variables $s_u \in \{\bot, \top\}$ and $x_u \in \mathbb{N}$ and may make a move according to one of the following rules:

(Refresh) $x_u \neq |N(u)| \rightarrow x_u := |N(u)|$ ( = $\text{deg}(u)$)

(Candidacy?) $(x_u = |N(u)|) \land (s_u = \bot) \land (\forall v \in N(u), s_v = \bot) \rightarrow$

if $\text{Rand}(\frac{1}{1+\max\{x_v,|N(u)|\}}) = 1$ then $s_u := \top$

(W Withrawal) $(x_u = |N(u)|) \land (s_u = \top) \land (\exists v \in N(u), s_v = \top) \rightarrow s_u := \bot$

Observe since we assume an anonymous setting, the only way to break symmetry is randomisation. The value of the probabilities for changing local variable must carefully be chosen in order to reduce the impact of the Byzantine node.

A node joins the MIS with a probability $\frac{1}{1+\max\{x_v,|N(u)|\}}$. The idea to ask the neighbours about their own number of neighbours (through the use of the $x$ variable) to choose the probability of a candidacy comes from the mathematical property $\forall k \in \mathbb{N}, (1 - \frac{1}{k+1})^k > e^{-1}$, which will allow to have a good lower bound for the probability of the event “some node made a successful candidacy, but none of its neighbours did”.

3.2 Specification

Since Byzantine nodes are not bound to follow the rules, we cannot hope for a correct solution in the entire graph. What we wish to do is to find a solution that works when we are far enough from the Byzantine nodes. One could think about a fixed containment radius around Byzantine nodes, but as we can see later this is not as simple, and it does not work with our approach.

Let us define on any configuration $\gamma$ the following set of nodes, that represents the already built independent set:

$I_\gamma = \{u \in V_1 | (s_u^\gamma = \top) \land \forall v \in N(u), s_v^\gamma = \bot\}$

We say that a node is locally alone if it is candidate to be in the independent set (i.e. its $s$-value is $\top$) while none of its neighbours are. In configuration $\gamma$, $I_\gamma$ is the set of all locally alone nodes of $V_1$.

Definition 2 A configuration is said legitimate when $I_\gamma$ is a maximal independent set of $V_2 \cup I_\gamma$.

3.2.1 An example:

Figure 1 gives an example of an execution of the algorithm. Figure 1 depicts a network in a given configuration. The symbol drawn above the node represents the local variable $s$. Each local variable $x$ contains the degree of its associated node. Byzantine node is shown with a square.

In the initial configuration, nodes $v_1$ and $v_2$ are in the independent set, and then are activable for Withdrawal. In the first step, the daemon activates $v_1$ (Withdrawal) and $v_2$ (Withdrawal) leading to configuration $\gamma_1$ (Fig. 1b). In the second step, the daemon activates $v_1$ (Candidacy?). Node $v_1$ randomly decides whether to set $s_{v_1} := \top$ leading to configuration $\gamma_2$ (Fig. 1b), or $s_{v_1} := \bot$ leading to configuration $\gamma_1$ (Fig. 1b). Assume that $v_1$ chooses $s_{v_1} := \top$. At this moment, node $v_1$ is "locally alone" in the independent set. In the third step, the daemon activates $b$ and $b$ makes a Byzantine move setting $s_b := \top$, leading to configuration $\gamma_3$ (Fig. 1b).

In the fourth step, the daemon activates $v_1$ (Withdrawal) and $b$ that sets $s_b := \bot$. The configuration is now the same as the first configuration (Fig. 1b). The daemon is assumed to be fair, so nodes $v_2$ and $v_3$ need to be activated before
the execution can be called an infinite loop. These activations will prevent the node \( v_1 \) to alternate forever between in and out of the independent set, while the rest of the system remains out of it. In the fifth step, the daemon activates \( v_1 \) (\textit{Candidacy}?), \( v_2 \) (\textit{Candidacy}?) and \( v_3 \) (\textit{Candidacy}?). They randomly decide to change their local variable \( s \). Assume that \( v_1 \), \( v_2 \) and \( v_3 \) choose \( s_{v_1} := \top \), \( s_{v_2} := \bot \), and \( s_{v_3} := \top \), leading to configuration \( \gamma_5 \) (Fig. 1b). At this moment, node \( v_3 \) is “locally alone” in the independent set. \( v_3 \) is far enough from the Byzantine node then it will remain in the independent set whatever \( b \) does.

\[
\begin{array}{ccc}
\bot & \top & \top \\
\top & \bot & \top \\
\bot & \bot & \bot \\
\end{array}
\]

(a) Initial configuration \( \gamma_0 \). \( v_1 \) and \( v_2 \) execute rule Withdrawal.

\[
\begin{array}{ccc}
\top & \top & \bot \\
\bot & \bot & \bot \\
\bot & \bot & \bot \\
\end{array}
\]

(b) Configuration \( \gamma_1 \): \( v_1 \) executes \textit{Candidacy}?.

\[
\begin{array}{ccc}
\bot & \top & \top \\
\top & \bot & \bot \\
\bot & \bot & \bot \\
\end{array}
\]

(c) Configuration \( \gamma_2 \): \( b \) sets \( s_b := \top \).

\[
\begin{array}{ccc}
\bot & \bot & \bot \\
\bot & \bot & \bot \\
\bot & \bot & \bot \\
\end{array}
\]

(d) Configuration \( \gamma_3 \): \( v_1 \) and \( b \) change their \( s \)-value.

\[
\begin{array}{ccc}
\bot & \top & \bot \\
\top & \bot & \bot \\
\bot & \bot & \bot \\
\end{array}
\]

(e) Configuration \( \gamma_4 \): Nodes \( v_2 \) and \( v_3 \) execute rule \textit{Candidacy}.

\[
\begin{array}{ccc}
\bot & \top & \top \\
\bottom & \bot & \bot \\
\top & \bot & \bot \\
\end{array}
\]

(f) Configuration \( \gamma_5 \): A \( V_1 \)-stable configuration.

Figure 1: Execution

3.2.2 About the specification:

Our goal is to design an algorithm that builds a maximal independent set of the subgraph induced by a set of nodes where node “too close” to Byzantine nodes have been removed. The question now is to define what does “too close” mean. One could think about a fixed containment radius only excluding nodes at distance at most 1 from Byzantine nodes. This set of nodes has been previously defined as \( V_1 \). Indeed, in Figure 3.2.3(a), \( v_1 \) and \( v_2 \) belong to \( V_1 \) and their local view of the system is correct, then they have no reason to change their states. Moreover, Byzantine nodes are too far away to change that: whatever the value of the state of \( v_0 \), the view of \( v_1 \) remains correct. Thus a containment radius of 1 could seem correct. However, in Figure 3.2.3(b), if the Byzantine node does not make any move, then \( v_0 \) remains in the MIS while \( v_1 \) remains out of it. Thus, in this example, if we only consider nodes in \( V_1 \), the \( \top \)-valued nodes of \( V_1 \) are not a MIS of \( V_1 \). If \( V_1 \) is not always a good choice, neither is \( V_2 \). See Figure 3.2.3(c), as one can see that all nodes in \( V_1 \) will never change their local state. The same can be said for \( V_k \) for any \( k \), see example Figure 3.2.3(d) for \( V_3 \). The solution is then to consider a set of nodes defined from a fixed containment radius to which we add locally alone neighboring nodes. The smallest containment radius that works with this approach is 3 (which corresponds to set \( V_2 \)). Note that it depends on the current configuration and not only on the underlying graph.

3.2.3 About the choice of probability to join the MIS:

We could have gone with the same probability for every node, but that comes with the cost of making the algorithm very sensitive to the connectivity of the underlying graph. As we rely for convergence on the event where a node is candidate alone (i.e. switch to \( \top \) without other node doing the same in its neighborhood), the probability of progress in a given number of rounds would then be exponentially decreasing with the degrees of the graph.

We could have gone with something depending only on the degree of the node where the rule is applied. While it could have been an overall improvement over the uniform version above, the minoration of the probability of progress that can be made with a local scope is no better. We cannot exclude that a finer analysis would lead to a better overall improvement, but it would require to deal with far more complex math. On a smaller scale, we can also note that this choice would introduce a bias toward small degree nodes, while we might not want that (depending on the application).

Then, we have chosen the version where nodes takes into account their degree and what they know of the degree of their neighbors. On one hand, the first concern you could rise here would be the potential sabotage by Byzantine nodes. Here is the intuition of why this cannot be a problem here. If a node \( u \) is at distance at least 2 from any Byzantine node and if \( u \) is “locally alone” in the independent set, then whatever the Byzantine nodes do, \( u \) will forever remain in the independent set. To maximize the harm done, the Byzantine nodes have to prevent indirectly such a node to join the
independent set. To do so it has to maximize the probability of its neighbours to be candidate to the independent set. But Byzantine nodes cannot lie efficiently in that direction, as the probability is upper-bounded by the degree values of both the node and its non-Byzantine neighbours. On the other hand, this choice allows us to address the problem that we had with the previous solution. Here, we can indeed frame the probability to be candidate alone between two constant bounds with a simple local analysis. Thus, we can ensure that the convergence speed does not depend on the connectivity of the underlying graph. Again, on a smaller scale, we also greatly reduce the bias toward small degree nodes compared to the previous option.

![Diagram](image)

Figure 2: What is the good containment radius?

### 3.3 The proof

Every omitted proof can be seen in the appendix.

We say that in a configuration $\gamma$, a node $u$ is degree-stabilized if rule Refresh is not enabled on it. The configuration $\gamma$ is then said to be degree-stabilized if every non-Byzantine node is degree-stabilized. Observe the two following facts about degree-stabilization (proofs in appendix):

**Lemma 3** Any reachable configuration from a degree-stabilized configuration is degree-stabilized.

**Proof:** No rule can change the $x$ value of a degree-stabilized non-Byzantine node. ∎

**Lemma 4** From any configuration $\gamma$, the configuration $\gamma'$ after one round is degree-stabilized.

**Proof:** Let $u$ be a non-byzantine node.

If $x_u^\gamma = \deg(u)$, no activation of rule can change that thus $x_u^{\gamma'} = \deg(u)$.

If $x_u^\gamma \neq \deg(u)$, then $u$ is activable in $\gamma$ and remains so until it is activated. Rule Refresh is then executed on $u$ in the first round, and since no rule can change the value of $x_u$ afterward we have $x_u^{\gamma'} = \deg(u)$. ∎

All locally alone nodes in $V_1$ (i.e. nodes of $I_\gamma$) remain locally alone during the whole execution.

**Lemma 5** If $\gamma \rightarrow \gamma'$, $I_\gamma \subseteq I_{\gamma'}$.

**Proof:** Let’s consider $u \in I_\gamma$. The only rules that may be enabled on $u$ in $\gamma$ is Refresh since Candidacy? can’t be executed on $u$ because $s_u^\gamma = \top$, and Withdrawal can’t be executed on $u$ because $\forall v \in N(u), s_v^\gamma = \bot$. We have then $s_u^{\gamma'} = \top$.

Now let’s consider $v \in N(u)$. By definition of $I_\gamma$, $s_v^\gamma = \bot$, and $v$ has $u$ as a neighbour that has value $\top$ in $\gamma$. Then the only rule that may be enabled on $v$ in $\gamma$ is Refresh, and we have $s_v^{\gamma'} = \bot$.

Thus, $u \in I_{\gamma'}$. ∎

We now focus on the progression properties of enabled rules after one round. We start with the Withdrawal rule. When the Withdrawal rule is enabled on a node $u \in V_1$, the conflict is solved by either making $u$ locally alone or setting $s_u = \bot$.

**Lemma 6** If $\gamma$ is a degree-stabilized configuration and if Withdrawal is enabled on $u \in V_1$ then after one round either Withdrawal has been executed on $u$ or in the resulting configuration $\gamma'$ we have $u \in I_{\gamma'}$. 

6
Proof: Since $\gamma$ is degree-stabilized, no Refresh move can be executed in any future transition. Then, since $s^*_u = \top$, only Withdrawal can be executed on $u$ or any of its non-Byzantine neighbours until $u$ has been activated. Since $u \in V_1$, it is in fact true for every neighbour of $u$.

Since $u$ is activable in $\gamma$, we have two cases: (i) If $u$ has performed a Withdrawal move in the next round there is nothing left to prove. (ii) If it is not the case, $u$ must have been unactivated by fairness hypothesis, that means that each neighbour $v \in N(u)$ that had $s$-value $\top$ in $\gamma$ have been activated. By the above, they must have performed a Withdrawal move that changed their $s$-value to $\bot$. Also $u$ is supposed not to have performed any rule, so it keeps $s$-value $\bot$ in the whole round: its neighbours -that are non-Byzantine since $u \in V_1$- cannot perform any Candidacy move. As such, in the configuration $\gamma'$ at the end of the round, every neighbour of $u$ has $s$-value $\bot$, and $u$ has $s$-value $\top$. Since $u \in V_1$ that means that $u \in I_{\gamma'}$. □

When Candidacy? rule is enabled on a node $u \in V_1$, then Candidacy? is executed on $v \in N[u]$ within one round.

Lemma 7 If in $\gamma$ degree-stabilized we have the Candidacy? rule enabled on $u \in V_1$ (i.e., $s^*_u = \bot$ and $\forall v \in N(u), s^*_v = \bot$) then after one round Candidacy? have either been executed on $u$, or on at least one neighbour of $u$.

Proof: Since $\gamma$ is degree-stabilized, no Refresh move can be executed in any future transition. Then, until $u$ or one of its neighbours have been activated, only Candidacy? can be executed on them since it’s the only rule that can be activated on a node with $s$-value $\bot$.

Since $u$ is activable in $\gamma$, by fairness, we have two cases:

(i) If $u$ is activated before the end of the round, the only rule that it could have performed for its first activation is the Candidacy? rule since its $s$-value in $\gamma$ is $\bot$ and the configuration is supposed degree-stabilized.

(ii) If not, it has been unactivated, which means that at least one neighbour $v \in N(u)$ has been activated. As $u \in V_1$, $v$ cannot be Byzantine. The only rule that it could have performed for its first activation is the Candidacy? rule since its $s$-value in $\gamma$ is $\bot$ and the configuration is supposed degree-stabilized. □

Fact 8 $\forall k \in \mathbb{N}, \left(1 - \frac{1}{k+1}\right)^k > e^{-1}$

Proof: For $k = 0$ it is true ($(1 - \frac{1}{0+1})^0 = 1 > \frac{1}{e}$).

Suppose now that $k \geq 1$. A basic inequality about $\ln$ is that $\forall x \in \mathbb{R}^+, \ln(x) \geq 1 - x$, with equality only when $x = 1$.

For $x = \frac{k}{k+1}$, it gives us $\ln\left(\frac{k}{k+1}\right) \geq 1 - \frac{k+1}{k} = -\frac{1}{k}$, with equality only when $k = 1$. But since $\frac{k}{k+1}$ cannot have value 1 for any value of $k$, we have then $\ln\left(\frac{k}{k+1}\right) > -\frac{1}{k}$.

Then, by multiplying by $k$ on each side, we have $k \ln\left(\frac{k}{k+1}\right) > -1$, and thus, taking the exponential:

$$\left(1 - \frac{1}{k+1}\right)^k = e^{k \ln\left(\frac{k}{k+1}\right)} > e^{-1}$$

If node $u \in V_1$ executes Candidacy? rule, then $u$ becomes a locally alone node with a certain probability in the next configuration. So it implies that set $I$ grows.

Lemma 9 If $\gamma$ is a degree-stabilized configuration, and in the next transition rule Candidacy? is executed on a node $u \in V_1$, there is probability at least $\frac{1}{\gamma_{\bot}(\deg(u))}$ that in the next configuration $\gamma'$, $s^*_u = \top$ and $\forall v \in N(u), s^*_v = \bot$.

Proof: For any node $y$, we write $\varphi(y) = \frac{1}{1 + \max(\{\deg(v)|v \in N[y]\})}$.

Since Candidacy? can be executed on $u$ in $\gamma$, we know that $\forall v \in N[u], s^*_v = \bot$. Since $\gamma$ is degree-stabilized and no neighbour of $u$ can be Byzantine by definition of $V_1$, we have $\forall v \in N[u], x^*_v = \deg(v)$. The probability of $s^*_u = \top$ knowing the rule has been executed is then $\varphi(u)$.

Then, for a given $v \in N(u)$, node $v$ is not Byzantine since $u \in V_1$, and the probability that $s^*_v = \top$ is either 0 (if Candidacy? has not been executed on $v$ in the transition) or $\frac{1}{1 + \max(\{x^*_w|w \in N[v]\})}$. Since $\deg(u) = x_u \leq \max(\{x^*_w|w \in N[v]\})$, that probability is then at most $\frac{1}{1 + \deg(u)}$. $\square$
From Lemma 10, we have three possibilities after one round:

Proof: 

Let us denote by \( I_\gamma \) a configuration such that \( I_\gamma \) is not a maximal independent set of \( V_2 \cup I_\gamma \), then after at most one round one of the following events happens:

1. Rule Candidacy? is executed on a node of \( V_1 \)
2. A configuration \( \gamma' \) such that \( I_\gamma \subseteq I_\gamma' \) is reached.
3. A configuration \( \gamma' \) such that rule Candidacy? is enabled on a node of \( V_2 \) in \( \gamma' \) is reached.

Proof: Suppose \( \gamma' \) is a degree-stabilized configuration such that \( I_\gamma \cup V_2 \) is not a maximal independent set of \( V_2 \). As \( \gamma' \) is supposed degree-stabilized, we will only consider the possibility of moves that are not Refresh moves. (a) If Candidacy? is enabled on a node of \( V_2 \) in \( \gamma' \), Condition 3 holds. (b) If it is not the case, then there exists \( u \in V_2 \) such that at least one neighbour \( v \in V_1 \) such that \( s'_u = s'_v = \top \) (otherwise \( I_\gamma \) would be a maximal independent set of \( V_2 \cup I_\gamma \)).

In the first case there is nothing left to prove. In the second case, Withdrawal is enabled on \( u \in V_2 \) in \( \gamma' \) and from Lemma 6 we have two possible cases: (i) if \( \gamma' \) is the configuration after one round, \( u \in I_\gamma \) and Condition 2 holds; (ii) \( u \) perform a Withdrawal move before the end of the round.

In the first case there is nothing left to prove. Suppose now that we are in the second case and that \( u \) is activated only once before the end of the round, without loss of generality since Condition 1 would hold otherwise as it second activation would be a Candidacy? move and \( u \in V_1 \). Then:

- If within a round a configuration \( \gamma' \) is reached where a node \( w \in N(u) \) is such that \( s'_w = \top \) and \( w \) is not activable we have: \( w \notin I_\gamma \) (as \( u \) is \( \top \)-valued in \( \gamma' \) which gives \( I_\gamma \subseteq I_\gamma \cup \{w\} \subseteq I_\gamma' \) by Lemma 5 thus Condition 2 holds.
- If we suppose then that no such event happens until the end of the round in configuration \( \gamma' \), we are in either of those cases: (i) Every neighbour of \( u \) have value \( \bot \) in \( \gamma' \) and and \( s'_u = \bot \) (as we would be in the previous case if it was \( \top \)), thus Candidacy? is enabled on \( u \) in \( \gamma' \) and Condition 3 holds. (ii) A Candidacy? has been performed within the round on a neighbour of \( u \) and Condition 1 holds.

Thus, in every possible case, one of the three conditions holds.

Every 2 rounds, the set of locally alone nodes strictly grows with some probability.

Lemma 11 If \( \gamma \) is degree-stabilized, with \( I_\gamma \) not being a maximal independent set of \( V_2 \cup I_\gamma \), then after two rounds the probability for the new configuration \( \gamma' \) to be such that \( I_\gamma \subseteq I_\gamma' \) is at least \( \frac{1}{\Delta + 1} \).

Proof:

From Lemma 10, we have three possibilities after one rounds:

- If we are in Case 1, let us denote by \( \gamma'' \) the resulting configuration after the transition where the said Candidacy? move have been executed on \( u \in V_1 \). Then using Lemma 8 we have \( u \in I_\gamma \) with probability at least \( \frac{1}{\Delta + 1} \). Since we have \( u \notin I_\gamma \) (if \( u \) was in \( I_\gamma \), Candidacy? could not have been executed on \( u \) after configuration \( \gamma \)) and by Lemma 5 we have then \( I_\gamma \subseteq I_\gamma' \) with probability at least \( \frac{1}{\Delta + 1} \).
• If we are in Case 2, there is nothing left to prove.

• If we are in Case 3, let us denote by \( \gamma'' \) the first configuration where Candidacy? is enabled on some \( u \in V_2 \). Then using Lemma 7, after at most one more round Candidacy? will be executed on \( v \in N[u] \). Let us denote by \( \gamma'' \) the resulting configuration after the transition where the said Candidacy? move have been executed on \( v \). Since \( u \in V_2 \) we have \( v \in V_1 \) and using Lemma 9 \( v \in I_{\gamma''} \) with probability at least \( \frac{1}{(\Delta+1)e} \) and \( v \notin I_{\gamma} \) by the same argument as above. Thus, since \( I_{\gamma''} \subseteq I_{\gamma'} \), the probability that \( I_{\gamma} \subseteq I_{\gamma'} \) is at least \( \frac{1}{(\Delta+1)e} \).

In every case, the property is true, thus the lemma holds. \( \square \)

Since the number of locally alone nodes cannot be greater than \( n \), the expected number of rounds before stabilization can be computed. We will use the notation \( \alpha = \frac{1}{(\Delta+1)e} \) to simplify the formulas in the remaining of the paper.

Lemma 12: For any \( p \in [0, 1] \). From any degree-stabilized configuration \( \gamma \), the algorithm is self-stabilizing for a configuration \( \gamma' \) where \( I_{\gamma'} \) is a maximal independent set of \( V_2 \cup I_{\gamma'} \). The time for this to happen is less than \( \max \left(-\alpha^{-2} \ln p, \frac{\sqrt{2} \cdot n}{\sqrt{\alpha - 1}} \right) \) rounds with probability at least \( 1 - p \).

Proof: Consider a degree-stabilized configuration \( \gamma_0 \), and \( \Omega \) the set of all complete executions of the algorithm starting in configuration \( \gamma_0 \). The probability measure \( \mathbb{P} \) is the one induced by the daemon choosing the worst possibility for us at every step.

Consider for \( i \in \mathbb{N} \) the random variables \( X_i \) that denotes the configuration after the \( i \)-th round has ended (\( X_0 \) is the constant random variable of value \( \gamma_0 \)).

Consider \((\mathcal{F}_i)_{i \in \mathbb{N}}\) the natural filtration associated with \( X_i \).

Consider the function \( f : \gamma \mapsto |I_\gamma| \).

\( Y_i = \mathbb{I}_{f(X_i) - f(X_{i-1}) > 0} \) is the random variable with value 1 if the size of \( I \) increased in the \( i \)-th round, else 0.

Consider the stopping time \( \tau \) (random variable describing the number of rounds the algorithm take to stabilize on \( V_1 \)) defined by:

\[ \tau(\omega) = \inf \{ n \in \mathbb{N} | \text{I does not change after round } n \text{ in execution } \omega \} \]

As \( Y_i \) has values in \{0; 1\}, we have \( \mathbb{P}(Y_i = 1|\mathcal{F}_{i-1}) = \mathbb{E}[Y_i|\mathcal{F}_{i-1}] \). Also, from Lemma 10 we get \( \mathbb{P}(Y_i = 1|\mathcal{F}_{i-1}) \geq \alpha \cdot \mathbb{I}_{\tau \geq i-1} \). Thus combining the two relations we get:

\[ \mathbb{E}[Y_i|\mathcal{F}_{i-1}] \geq \alpha \cdot \mathbb{I}_{\tau \geq i-1} \tag{1} \]

Consider \( S_i = \sum_{k=1}^{i} Y_k \) the random variable representing the number of rounds where there have been an increment. Since this cannot happen more time than there are nodes, we get:

\[ S_i \leq n \tag{2} \]

Consider \( A_i = \sum_{k=1}^{i} \mathbb{E}[Y_k|\mathcal{F}_{k-1}] \) the random variable representing the sum of the expected values of the increments at each step.

When \( \tau > i \), every for very value of \( k \in [1, i] \) we have \( \mathbb{I}_{\tau \geq k-1} = 1 \). Then using (1) we get:

\[ \tau > i \Rightarrow A_i \geq i\alpha \tag{3} \]
Consider then the random variable $M_i = \sum_{k=1}^i Y_k - \mathbb{E}[Y_k|\mathcal{F}_{i-1}]$ (do note that it is the same as the difference $S_i - A_i$).

$$\mathbb{E}[M_{i+1}|\mathcal{F}_i] = \mathbb{E} \left[ \sum_{k=1}^{i+1} Y_k - \mathbb{E}[Y_k|\mathcal{F}_{k-1}] \right]$$

$$= \mathbb{E} [M_i + Y_{i+1} - \mathbb{E}[Y_{i+1}|\mathcal{F}_i]]$$

$$= \mathbb{E} [M_i|\mathcal{F}_i] + \mathbb{E} [Y_{i+1}|\mathcal{F}_i] - \mathbb{E} [\mathbb{E}[Y_{i+1}|\mathcal{F}_i]|\mathcal{F}_i]$$

$$= M_i + \mathbb{E} [Y_{i+1}|\mathcal{F}_i] - \mathbb{E} [Y_{i+1}|\mathcal{F}_i]$$

$$= M_i$$

Thus $(M_i)_{i\in\mathbb{N}}$ is a martingale with respect to the filtration $(\mathcal{F}_i)_{i\in\mathbb{N}}$.

We also have $|M_{i+1} - M_i| = |Y_{i+1} - \mathbb{E}[Y_{i+1}|\mathcal{F}_i]| \leq \max(Y_{i+1}, \mathbb{E}[Y_{i+1}|\mathcal{F}_i]) \leq 1$. 


Thus by Azuma inequality:
\[
\forall \beta \leq 0, \mathbb{P}(M_i \leq \beta) \leq e^{-\frac{2\beta^2}{i}}
\] (4)

Then using (2) and (3) we get \( \tau > i \Rightarrow S_i - A_i \leq n - i\alpha, \) i.e. \( \tau > i \Rightarrow M_i \leq n - i\alpha \)

Thus we have \( \mathbb{P}(\tau > i) \leq \mathbb{P}(M_i \leq n - i\alpha) \) and for \( i \geq \frac{n}{\alpha} \) we can apply (4) to get:
\[
\mathbb{P}(\tau > i) \leq e^{-\frac{2(n-i\alpha)^2}{i}}
\]

For \( i \geq \frac{\sqrt{2}}{\sqrt{2}-1} \alpha \) (it implies that \( i \geq \frac{n}{\alpha} \)) we have \( \frac{1}{2}(i\alpha)^2 \leq (n - i\alpha)^2 \), which gives for such \( i \):
\[
\mathbb{P}(\tau > i) \leq e^{-i\alpha^2}
\]

For \( i \geq -\alpha^{-2} \ln p \), we have \( e^{-i\alpha^2} \leq p \)

Mixing the two above inequalities, when \( i \geq \max \left(-\alpha^{-2} \ln p, \frac{\sqrt{2}}{\sqrt{2}-1} \frac{n}{\alpha}\right) \), we get:
\[
\mathbb{P}(\tau > i) \leq p
\]

Which concludes the proof. \(\square\)

**Theorem 13** For any \( p \in [0,1] \). From any configuration \( \gamma \), the algorithm is self-stabilizing for a configuration \( \gamma' \) where \( I_{\gamma'} \) is a maximal independent set of \( V_2 \cup I'_{\gamma}, \) and reach such a configuration in \( 1 + \max \left(-\alpha^{-2} \ln p, \frac{\sqrt{2}}{\sqrt{2}-1} \frac{n}{\alpha}\right) \) rounds or less with probability \( 1 - p \).

**Proof:** From Lemma \( 12 \) we reach a degree-stabilized configuration after at most one round. Then from that configuration, we apply Lemma \( 12 \). \(\square\)

## 4 In an Anonymous System under the Adversary Daemon

### 4.1 The algorithm

The algorithm builds a maximal independent set represented by a local variable \( s \).

We could have used the previous algorithm as fairness was only needed to contain Byzantine influence. But the complexity would have been something proportional to \( \Delta n^2 \), and as we will prove, we can do better than that.

We keep the idea of having nodes making candidacy and then withdrawing if the candidate’s situation is not correct. We still do not have identifiers, and so we do need a probabilistic tie-break. But contrary to the byzantine case, we move the probabilities to the Withdrawal rule: a non-candidate node with no candidate neighbour will always become a candidate when activated, but a candidate node with a candidate neighbour will only withdraw with probability \( \frac{1}{2} \) when activated.

**Algorithm 2** Any node \( u \) has a single local variable \( s_u \in \{\bot, \top\} \) and may make a move according to one of the following rules:

(Candidacy) \( (s_u = \bot) \land (\forall v \in N(u), s_v = \bot) \Rightarrow s_u := \top \)

(Waiting?) \( (s_u = \top) \land (\exists v \in N(u), s_v = \top) \Rightarrow \text{if Random}(\frac{1}{2}) = 1 \text{ then } s_u := \bot \)

The following lemma guarantees the correctness of the algorithm.

**Lemma 14** A configuration \( \gamma \) is stable if and only if \( \beta(\gamma) \) is a maximal independent set of \( G \).

**Proof:** Suppose \( \gamma \) is not stable.

- If Waiting? is activable on a node \( u \), it means that \( u \notin \beta(\gamma) \) as \( u \) has a neighbor with \( s \)-value \( \top \). But as \( u \) has \( s \)-value \( \top \) it also means that none of its neighbors is in \( \beta(\gamma) \), thus \( \beta(\gamma) \cup \{u\} \) is an independent set greater than \( \beta(\gamma) \).
- Else, Candidacy is activable on a node \( u \). It means that \( u \notin \beta(\gamma) \) as \( s_u = \bot \). It also means that none of its neighbors is in \( \beta(\gamma) \) since they have all \( s \)-value \( \bot \), thus \( \beta(\gamma) \cup \{u\} \) is an independent set greater than \( \beta(\gamma) \).
Thus, by contraposition, if $\beta(\gamma)$ is a maximal independent set of $G$, then $\gamma$ is stable.

Suppose now that $\gamma$ is stable. As Withdrawal? cannot be enabled on any node, any node with $s$-value $\top$ is in $\beta(\gamma)$. But then as Withdrawal? cannot be enabled on any node, every node has a node with $s$-value $\top$, and thus a node of $\beta(\gamma)$ in its closed neighborhood. $\beta(\gamma)$ is then a maximal independent set of $G$. □

Now that we have correction, we want to prove the convergence of our algorithm. The idea that we will develop is that we can give a non-zero lower bound on the probability that a connected component of candidate nodes eventually collapses into at least one definitive member of the independent set. As every transition with Candidacy makes such scenes appear, and Withdrawal? moves make those collapse into at least one member of the independent set with non-zero probability, this should converge towards a maximal independent set.

Looking at the guards of the rules of the algorithm, we observe the following fact:

**Fact 15** In a configuration $\gamma$, when Withdrawal? is enabled on a node, Candidacy is not enabled on any of its neighbours.

Given a configuration $\gamma$, we define $\beta(\gamma) = \{u \in V \mid s_u = \top \land \forall v \in N(u), s_v = \bot\}$. Note that $\beta(\gamma)$ is always an independent set since two distinct members cannot be neighbors (as they have both $s$-value $\top$).

**Lemma 16** If $\gamma \rightarrow \gamma'$, then $\beta(\gamma) \subseteq \beta(\gamma')$.

**Proof:** For every $u \in \beta(\gamma)$, we have by definition:

1. $s_u^{\gamma} = \top$;
2. for every neighbour $v$ of $u$, $s_v^{\gamma} = \bot$.

Point 1 implies that Candidacy is not enabled on any vertex of $N[u]$, and Point 2 implies that Withdrawal? is not enabled on any vertex of $N[u]$.

For every vertex $v \in N[u]$, since no rule is enabled on $v$, we have $s_v^{\gamma} = s_v^{\gamma'}$. Thus, $u \in \beta(\gamma')$. □

Now that we know that $\beta$ cannot shrink, it remains to prove that it does grow, and within a reasonable amount of time. We are introducing the following concepts to this end:

**Definition 17** A set of node $X \subseteq V$ is said to be a candidate set of a configuration $\gamma$ if $\forall u \in X, (s_u^{\gamma} = \top) \land \forall v \in N(u), s_v^{\gamma} = \bot \Rightarrow v \in X$. We will say that it is a connected candidate set when $X$ is a connected component of $G$.

### 4.2 The proof

First, we transform all executions into executions having suitable properties, using the two following lemmas for the simplicity of the proof.

**Lemma 18** Let $\gamma_1$ a configuration and $t$ a valid set of moves in $\gamma_1$. Let also $t_w \subseteq t$ (resp. $t_c$) the set of Withdrawal? moves (resp Candidacy moves) in $t$.

Then making two successive transition using the set of moves $t_w$ and then $t_c$ is equivalent to make a transition with the set of move $t$ (in the sense that the distribution of probability on the values of nodes variables is the same).

i.e. we may replace $\gamma_1 \xrightarrow{t} \gamma_2$ with $\gamma_1 \xrightarrow{t_w} \gamma' \xrightarrow{t_c} \gamma_2$.

**Proof:** Consider $\gamma_2$ the random variable describing the state of the configuration attained from $\gamma_1$ by executing the set of move $t$.

Let $W$ (resp. $C$) be the set of the nodes that appear in a Withdrawal? (resp. Candidacy) move of $t$.

There is only one possible $\gamma'$ such that $\gamma_1 \xrightarrow{t_w} \gamma'$ since the rule Candidacy is deterministic, consider this configuration $\gamma'$.

Due to Fact 15 no node in $W$ has a node of $C$ as neighbor. Thus, nodes in $W$ and their neighbors did not change their local variable in the transition $\gamma_1 \xrightarrow{t_w} \gamma'$. Then, Withdrawal? is enabled on every node of $W$ in $\gamma'$ and $t_w$ is a valid set of move in $\gamma'$.

Consider then the random variable $\gamma_2'$ such that $\gamma' \xrightarrow{t_w} \gamma_2'$. 

12
• For \( u \in C \), the transitions are deterministic and we have \( s_u^{\gamma_2} = \top = s_u^{\gamma_2} \).
• For \( u \in W \), note that as long as Withdrawal? is executed, its results always follow the same probability law.

Thus, as nodes that executed Withdrawal are the same in \( \gamma' \xrightarrow{t_u} \gamma_2 \) and \( \gamma_1 \xrightarrow{t} \gamma_2 \), the probability distribution of \( s_u \) is the same in \( \gamma_2 \) and \( \gamma_2' \).
• For nodes that are in neither \( C \), or \( W \), the didn’t execute any rules in the considered transitions and thus they have the same \( s \)-value in every configuration we considered.

Thus, \( \gamma_2 \) has the same probability distribution as \( \gamma_2' \), and hence the result. \( \square \)

**Lemma 19** Let \( \gamma_1 \) be a configuration, \( t \) a valid set of moves in \( \gamma_1 \) containing only Withdrawal? moves, and \( A \) a candidate set of \( \gamma_1 \). Let also \( t_A \subset t \) the set of moves on node of \( A \) in \( t \).

Then making two successive transition using the set of moves \( t_A \) and then \( t \setminus t_A \) is equivalent to make a transition with the set of move \( t \) (in the sense that the distribution of probability on the values of nodes variables is the same).

i.e. we may replace \( \gamma_1 \xrightarrow{t} \gamma_2 \) with \( \gamma_1 \xrightarrow{t_A} \gamma' \xrightarrow{\Delta^t_A} \gamma_2 \).

**Proof:** The proof follow the same pattern as the one of Lemma 19 and rely in the same way on the fact that a rule -here Withdrawal- applied on a node \( u \) have the same distribution of probability of \( s \)-value after the transition whatever the starting configuration of the transition.

The only thing left to prove is that \( t \setminus t_A \) is a valid set of move in any \( \gamma' \) such that \( \gamma_1 \xrightarrow{t_A} \gamma' \).

Let’s denote by \( B \) the set of the nodes that appear in moves of \( t \setminus t_A \).

Consider \( u \in B \). As Withdrawal? is enabled on \( u \) in \( \gamma_1 \) we know that \( s_u^{\top} = \top \). Then, by definition of a candidate set, \( u \) cannot be a neighbor of a node in \( A \), and as a result no node of \( N[u] \) is activated in the transition \( \gamma_1 \xrightarrow{t_A} \gamma' \). The closed neighborhood of \( u \) having the same \( s \)-values in \( \gamma_1 \) and \( \gamma' \), Withdrawal? must be enabled on \( u \) in \( \gamma' \).

It is then the case for every node of \( B \), and thus \( t \setminus t_A \) is a valid set of move in \( \gamma' \). \( \square \)

Using those lemmas, we will now only consider executions satisfying the three following properties:

1. Each transition is composed of only one move type.
2. For each transition \( \gamma_{i-1} \xrightarrow{t} \gamma_i \) containing Withdrawal? moves, only nodes of the same connected candidate set in \( \gamma_{i-1} \) execute a move.

Using Lemmas 18 and 19, all executions can be transformed into execution satisfying the two last property without increasing the number of moves.

From now, all the executions we consider satisfy the two previous properties.

Let \( E \) be an execution \( \gamma_0 \xrightarrow{t_1} \gamma_1 \cdots \gamma_{i-1} \xrightarrow{t_i} \gamma_i \cdots \).

We say that a candidate set \( X \) of \( \gamma \) is alive in \( \gamma \) if Withdrawal is enabled on at least a node of \( A \). Note that for connected candidate set it is equivalent to having cardinality at least 2.

In a transition \( \gamma \xrightarrow{t} \gamma' \), we say that an alive candidate set \( X \) of \( \gamma \) vanishes if \( \forall u \in X, s_u^{\gamma'} = \bot \) or \( u \in \beta_{\gamma'} \).

**Lemma 20** Suppose that \( \gamma \) is an alive connected candidate set of \( \gamma \), and that \( t \) is a valid set of move in \( \gamma \) that involve nodes of \( X \) (recall that we transformed our execution to make withdrawal transitions only act on a given connected candidate set).

Then, if \( \gamma' \) is the random variable describing the state of the system after performing the set of move \( t \) from \( \gamma \), the probability that \( \beta(\gamma') \setminus \beta(\gamma) \neq \emptyset \) knowing that \( X \) vanishes is at least \( \frac{2}{3} \).

**Proof:** If there is one node \( u \) in \( X \) that is not activated in the transition, we know that \( s_u^{\top} = \top \). As \( X \) is supposed to vanish in the transition, it implies that \( u \in \beta(\gamma') \).

Else, every node of \( X \) have performed Withdrawal in the transition.

Consider the distribution of the \( s \)-values after performing the set of move \( t \) in configuration \( \gamma \) (without the condition on the candidate set vanishing). Let’s write \( k = |X| \) As every node of \( X \) flip a coin independently, every outcome has the same probability \( \left( \frac{1}{2} \right)^k \). In particular,
• The event where every node of \(X\) change their \(s\)-value to \(\bot\) in the transition.
• Each of the \(k\) events where every node of \(X\) change their \(s\)-value to \(\bot\) in the transition except exactly one.

Those events are compatible with the constraint of the candidate set vanishing. Every other event is either incompatible with the constraint, or a situation where at least two nodes of \(X\) is in \(\beta(\gamma')\), as at least two nodes have \(s\)-value \(\top\) in \(\gamma'\) and \(X\) is supposed to vanish in the transition.

Thus, if denote by \(\lambda\) the sum of the probability of the events that are compatible with the constraint of the candidate set vanishing, we can say that \(\lambda \geq (k + 1) \left(\frac{1}{2}\right)^k\), and thus the probability to have \(\beta(\gamma_{j'}) \setminus \beta(\gamma_{j' - 1}) \neq \emptyset\) with the condition is \(1 - \left(\frac{1}{\lambda}\right)^k \geq 1 - \left(\frac{1}{(k + 1)\frac{1}{2}}\right)^k = 1 - \frac{1}{k + 1} = \frac{k}{k + 1}\).

Moreover, \(X\) is not empty by hypothesis. Consider \(u \in X\), it has at least a neighbor \(v\) such that \(s_u = \top\) as otherwise we would have \(u \in \beta(\gamma)\). But then we have \(v \in X\), as \(X\) is a candidate set. Thus \(k \geq 2\).

Thus the wanted probability is at least \(\frac{2}{3}\). \(\square\)

Now that we know that every time that an alive connected candidate set vanishes \(\beta\) progresses with good probability, it reamins to guarantee that it ever happens. To do so, we will need the following lemmas about life and death of candidate set.

**Lemma 21** If \(\gamma \xrightarrow{t} \gamma'\) is a transition with \(t\) containing only Candidacy moves, either \(\beta(\gamma) \subsetneq \beta(\gamma')\) or \(\exists X\) an alive connected candidate set of \(\gamma'\) such that \(\forall u \in X\), \(s_u = \bot\).

**Proof:** As a transition must contains at least one move, there is at least one node \(u\) appearing in \(t\).

If no neighbor of \(u\) appear in \(t\), we know that no neighbor of \(u\) change its \(s\)-value in the transition. But since Candidacy is enabled on \(u\) in \(\gamma\) we know that their \(s\)-value in \(\gamma\) is \(\bot\). Thus, \(s'_u = \top\) (since \(u\) executed Candidacy in the transition), and every of its neighbor has \(s\)-value \(\bot\) in \(\gamma'\), which is the definition for \(u \in \beta(\gamma')\). And since Candidacy is enabled on \(u\) in \(\gamma\), \(s_u = \bot\), thus \(u \notin \beta(\gamma')\). Thus \(\beta(\gamma) \subsetneq \beta(\gamma')\) (recall that from Lemma 16 \(\beta\) cannot lose nodes in a transition).

Suppose now that at least a neighbor of \(u\) appear in \(t\). Then consider the set \(X\) of the nodes that are accessible from \(u\) in the subgraph induced by the nodes appearing in \(t\). Every node \(v \in X\) executes Candidacy in the transition, thus \(s'_u = \bot\) and \(s'_v = \top\). Moreover, for every neighbor of a node of \(X\) not in \(X\), the condition of Candidacy imply that they have \(s\)-value \(\bot\) in \(\gamma\), and thus in \(\gamma'\) too since they did not perform a rule in the transition. Thus \(X\) is a candidate set of \(\gamma'\), connected by construction, and of cardinality at least 2 by hypothesis thus alive. \(\square\)

**Lemma 22** If \(\gamma \xrightarrow{t} \gamma'\) is a transition and \(X\) is an alive connected candidate set of \(\gamma\), one of those is true:

• \(X\) vanishes in the transition.
• \(\exists X'\) an alive connected candidate set of \(\gamma'\) such that \(X' \subseteq X\)

**Proof:** If \(t\) is a set of Candidacy moves, or a set of Withdrawal? moves on a connected candidate set different from \(X\), then the transition does not affect neither nodes of \(X\) or their neighbor thus \(X \subseteq X'\) is still an alive connected candidate set of \(\gamma'\).

Else, \(t\) is a set of Withdrawal? moves on nodes of \(X\). Suppose \(X\) does not vanish in the transition. By definition of vanishing, it means that it exists a node \(u \in X\) such that \(s_u = \top\) and \(u \notin \beta(\gamma')\). It implies that there exists \(v \in N(u)\) such that \(s_v = \bot\). But then, as \(t\) contains only Withdrawal? moves, \(s_v = \top\). By definition of a candidate set, we must have \(v \in X\). Now consider \(X'\) the set of nodes in the connected component of \(u\) in the subgraph of \(G\) induced by the nodes with \(s\)-value \(\top\) in \(\gamma'\). As only Withdrawal? moves have been executed in the transition, this imply that \(X'\) is a subset of the set of nodes in the connected of the connected component of \(u\) in the subgraph of \(G\) induced by the nodes with \(s\)-value \(\top\) in \(\gamma\), which is exactly \(X\). Moreover, if \(w \notin X'\) is a neighbor of a node in \(X'\), by definition of \(X'\) it is such that \(s_w = \bot\). Hence, \(cX'\) is a connected candidate set of \(\gamma'\), alive since of cardinality at least 2. \(\square\)

**Lemma 23** There cannot be more than \(\frac{n}{2} - |\beta(\gamma)|\) different connected candidate set alive in a configuration \(\gamma\).

**Proof:** An alive candidate set must have at least two nodes. Two candidate set that share a node must be the same candidate set as the definition of a candidate set prevents to have \(s\)-value \(\top\) in the neighborhood of a candidate set. \(\square\)

**Lemma 24** The execution ends with probability 1.
Proof: Every transition is either a transition with only Withdrawal moves, or only Candidacy moves.

Note that from Lemma 24, the number of alive connected component cannot shrink without one vanishing, and shrink by exactly one when it does happen.

Every transition with Withdrawal moves that do not make an alive connected candidate set vanish makes an alive connected candidate set shrink with probability at least $\frac{3}{2}$. Moreover when it makes one of those disappear is increase the size of $\beta$ with probability at least $\frac{2}{3}$.

Every transition with Candidacy moves either make a new alive candidate set appear, or increase the size of $\beta$. But as there can not be more than $\frac{2}{3} - |\beta|$ candidate set alive, there cannot be a transition that make a new alive candidate set appear while the number of alive candidate is maximal, and thus one must vanish before it becomes possible again (which will increase the size of $\beta$ with probability at least $\frac{2}{3}$).

By induction on the number of alive connected candidate set and their size, as long as Candidacy or Withdrawal are enabled, $\beta$ will grow within a finite time. As its size cannot be greater than the number of nodes, the execution must end.

Now, we may suppose that every alive candidate set vanishes in finite time, and use our Lemma 20.

Lemma 25 Let $1 > p \geq 0$. Any execution has at most $\max\left(-\frac{9}{4} \ln p, \frac{\sqrt{2} \lambda n}{\sqrt{2} - 1}, \frac{3\lambda n}{2}\right)$ transitions in which a connected candidate set vanishes with probability at least $1 - p$.

Proof: It is the same as the proof of Lemma 12 but the event we track is the vanishing of connected candidate set (which happens in finite time with probability $1$ from Lemma 24), the lower bound on the probability of the interesting event to happen when such a vanishing happens is $\frac{2}{3}$ from Lemma 20.

Theorem 26 Consider $p \in [0, 1]$. From any configuration, the algorithm is self-stabilizing for a configuration $\gamma$ in which $\beta(\gamma)$ is a maximal independent set, with at most $O(n^2)$ moves with probability at least $1 - p$.

Proof: Consider $p' \in [0, 1]$, $p' = \frac{p}{2}$, and $\lambda = \max\left(-\frac{9}{4} \ln p', \frac{\sqrt{2} \lambda n}{\sqrt{2} - 1}, \frac{3\lambda n}{2}\right)$. We denote by $x$ the number of vanishing of connected candidate sets in the execution. Using Lemma 25, we know that $x > \lambda$ with probability at most $p'$.

First, we count on the number of moves of type (u,Candidacy). For each transition in which a connected candidate set vanish, those nodes have had their $s$-value set to $\top$ by the last transition containing Candidacy moves on the said candidate set (or it does not exists they had $s$-value $\top$ in the initial state). This makes for at most $n$ Candidacy move for every connected candidate set that vanish, thus $xn$ in total.

To count for Withdrawal? moves, we will first count the number of successful (meaning the $s$-value of the node it is applied to change in the transition). For every connected candidate set that vanish, as they may only shrink from the time they appear due to Lemma 22, it takes at most $n$ successful Withdrawal? moves to make it disappear. Note that any Withdrawal? move acts on an alive connected candidate set, thus every Withdrawal? participates in the shrinking of an alive connected candidate set. Then, in total, we need at most $xn$ successful Withdrawal? moves to have $x$ vanishings of connected candidate sets.

Each Withdrawal? move has probability $\frac{1}{2}$ to be successful, a probability which does not depend on anything but the fact that rule Withdrawal? is applied, and are thus the outcome of different Withdrawal? moves are independant. The number of successful Withdrawal? is then connected to the total number of Withdrawal? by a binomial law of parameter $\lambda$ and $\frac{1}{2}$, where $\lambda$ is the number of Withdrawal? moves. Using the standard bound obtained by Hoeffding inequality for the binomial law, the probability to have $\lambda n$ or less successful Withdrawal? after $2(\lambda n + \sqrt{\lambda n \log p' - 1})$ Withdrawal? moves is at most $p$. This means that the probability to not have finished the execution after $2(\sqrt{\lambda n \log p' - 1} - 1)$ Withdrawal? moves is at most $2p' = p$.

Then, using the union bound, the probability for $x$ to be such that $x > \lambda$ or the number of Withdrawal? moves in the execution to be greater than $2(\lambda n + \sqrt{-\lambda n \log p' - 1}) + \lambda n = O(n^2)$ move in total is at least $1 - p$, which concludes the proof.
References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. URL: https://www.sciencedirect.com/science/article/pii/0196677486900192 doi:https://doi.org/10.1016/0196-6774(86)90019-2.

[2] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikael Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 481–497, 2019. doi:10.1109/FOCS.2019.00037.

[3] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-iterative distributed $(\delta + 1)$: -coloring below szegedy-vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. pages 437–446, 07 2018. doi:10.1145/3212734.3212769.

[4] Badreddine Benreguia, Hamouma Moumen, Soheila Bouam, and Chafik Arar. Self-stabilizing algorithm for maximal distance-2 independent set. *CoRR*, abs/2101.11126, 2021. URL: https://arxiv.org/abs/2101.11126, arXiv:2101.11126.

[5] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *Distrib. Comput.*, 33(3–4):349–366, jun 2020. doi:10.1007/s00446-020-00376-1.

[6] Alain Cournier, Stephane Devismes, and Vincent Villain. Snap-stabilizing pif and useless computations. In *Proceedings of the 12th International Conference on Parallel and Distributed Systems - Volume 1*, ICPADS ’06, page 39–48, USA, 2006. IEEE Computer Society. doi:10.1109/ICPADS.2006.100.

[7] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, nov 1974. doi:10.1145/361179.361202.

[8] Shlomi Dolev, Amos Israeli, and Shlomo Moran. Uniform dynamic self-stabilizing leader election. In Sam Toueg, Paul G. Spirakis, and Lefteris Kirousis, editors, *Distributed Algorithms*, pages 167–180, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[9] Xiaofeng Gao, Xudong Zhu, Jun Li, Fan Wu, Guihai Chen, Ding-Zhu Du, and Shaojie Tang. A novel approximation for multi-hop connected clustering problem in wireless networks. *IEEE/ACM Trans. Netw.*, 25(4):2223–2234, aug 2017. doi:10.1109/TNET.2017.2690359.

[10] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’16, page 270–277, USA, 2016. Society for Industrial and Applied Mathematics.

[11] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’21, page 2904–2923, USA, 2021. Society for Industrial and Applied Mathematics.

[12] 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 *Proceedings International Parallel and Distributed Processing Symposium*, pages 14 pp.–, 2003. doi:10.1109/IPDPS.2003.1213302.

[13] Maria Gradinariu and Sebastien Tixeuil. Conflict managers for self-stabilization without fairness assumption. In *27th International Conference on Distributed Computing Systems (ICDCS ’07)*, pages 46–46, 2007. doi:10.1109/ICDCS.2007.95.

[14] Nabil Guellati and Hamamache Khedouci. A survey on self-stabilizing algorithms for independence, domination, coloring, and matching in graphs. *J. Parallel Distrib. Comput.*, 70(4):406–415, apr 2010. doi:10.1016/j.jpdc.2009.11.006.

[15] Stephen Hedetniemi. *Self-Stabilizing Domination Algorithms*, pages 485–520. 01 2021. doi:10.1007/978-3-030-58892-2_16.

[16] Michiyio Ikeda, Sayaka Kamei, and Hirotugu Kakugawa. A space-optimal self-stabilizing algorithm for the maximal independent set problem. 01 2002.

[17] Colette Johnen and Mohammed Haddad. Efficient self-stabilizing construction of disjoint MDSs in distance-2 model. Research report, Inria Paris, Sorbonne Université ; LaBRI, CNRS UMR 5800 ; LIRIS UMR CNRS 5205, February 2021. URL: https://hal.archives-ouvertes.fr/hal-03138979.

[18] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982. doi:10.1145/357172.357176.
[19] Nathan Linial. Distributive graph algorithms global solutions from local data. In 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), pages 331–335, 1987. doi:10.1109/SFCS.1987.20

[20] Tao Liu, Xiaodong Wang, and Le Zheng. A cooperative swift scheme for wirelessly powered sensor networks. IEEE Transactions on Communications, PP:1–1, 03 2017. doi:10.1109/TCOMM.2017.2685580

[21] M Luby. A simple parallel algorithm for the maximal independent set problem. In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC ’85, page 1–10, New York, NY, USA, 1985. Association for Computing Machinery. doi:10.1145/22145.22146

[22] David Peleg. Distributed Computing: A Locality-Sensitive Approach. Society for Industrial and Applied Mathematics, USA, 2000.

[23] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization, page 350–363. Association for Computing Machinery, New York, NY, USA, 2020. URL: https://doi.org/10.1145/3357713.3384298

[24] Zhengnan Shi, Wayne Goddard, and Stephen T. Hedetniemi. An anonymous self-stabilizing algorithm for 1-maximal independent set in trees. Information Processing Letters, 91(2):77–83, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0020019004000985, doi:https://doi.org/10.1016/j.ipl.2004.03.010

[25] Sandeep Shukla, Daniel Rosenkrantz, and S. Ravi. Observations on self-stabilizing graph algorithms for anonymous networks. Proceedings of the Second Workshop on Self-Stabilizing Systems, 01 1995.

[26] Hideyuki Tanaka, Yuichi Sudo, Hirotsugu Kakugawa, Toshimitsu Masuzawa, and Ajoy K. Datta. A self-stabilizing 1-maximal independent set algorithm. Journal of Information Processing, 29:247–255, 2021. doi:10.2197/ipsjjip.29.247

[27] Volker Turau. Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler. Inf. Process. Lett., 103(3):88–93, jul 2007. doi:10.1016/j.ipl.2007.02.013

[28] Volker Turau. Making Randomized Algorithms Self-stabilizing, pages 309–324. 07 2019. doi:10.1007/978-3-030-24922-9_21

[29] Volker Turau and Christoph Weyer. Randomized self-stabilizing algorithms for wireless sensor networks. In Proceedings of the First International Conference, and Proceedings of the Third International Conference on New Trends in Network Architectures and Services Conference on Self-Organising Systems, IWSS’06/EuroNGI’06, page 74–89, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11822035_8