

A Self-Stabilizing Algorithm for Finding a Minimal Distance-2 Dominating Set in Distributed Systems

Ji-CHERNG LIN, TETZ C. HUANG, CHENG-PIN WANG
AND CHIH-YUAN CHEN[†]

*Department of Computer Science and Engineering
Yuan Ze University
Chungli, 320 Taiwan*

E-mail: csjclin@saturn.yzu.edu.tw

[†]*Department of Computer Science and Information Engineering
Nanya Institute of Technology
Chungli, 320 Taiwan*

The study of various dominating set problems is an important area within graph theory. In applications, a dominating set in a system can be considered as an ideal place for allocating resources. And, a minimal dominating set allows allocating a smaller number of resources. Distance-versions of the concept of minimal dominating sets are more applicable to modeling real-world problems, such as placing a smaller number of objects within acceptable distances of a given population. However, due to the main restriction that any processor in a distributed system can only access the data of its direct neighbors, a self-stabilizing algorithm for finding a minimal distance- k (with $k \geq 2$) dominating set is hard to get, and its correctness is hard to verify. In this paper, a self-stabilizing algorithm for finding a minimal distance-2 dominating set is proposed. The algorithm can be applied to any distributed system that operates under the central demon model. The correctness of the algorithm is verified.

Keywords: minimal distance-2 dominating set, self-stabilizing algorithm, Dijkstra's central demon model, distributed system, legitimate configuration

1. INTRODUCTION

A *distributed system* consists of a set of loosely connected processors that do not share a common or global memory. Each processor has one or more shared registers and possibly some non-shared local variables, used to specify the *local state* of the processor. Local states of all processors in the system at a certain time constitute the *global configuration* (or, simply, *configuration*) of the system at that time. The main restriction of a distributed system is that each processor in the system can only access the data (*i.e.*, read the shared data) of its neighbors. Since a distributed algorithm is an algorithm that works in a distributed system, it must abide by this main restriction. Depending on the purpose of a distributed system, a global criterion for the global configuration is defined. Those global configurations satisfying the criterion are called *legitimate configurations*, whereas other global configurations are called *illegitimate configurations*. When the system is in a legitimate configuration, the purpose of the system is fulfilled.

Received February 26, 2007; revised September 4, 2007; accepted October 18, 2007.
Communicated by Chin-Laung Lei.

1.1 Dijkstra's Central Demon Model

Dijkstra's *central demon model* of computations [1-3] of an algorithm in a distributed system has the following features:

- (a) The algorithm running on each processor consists of one or more rules. Each rule is of the form

condition part \rightarrow *action part*.

The *condition part* (or *guard*) is a Boolean function over the states of the processor and its neighbors; the *action part* is an assignment of values to some of the processor's shared registers. If the condition part of a rule in a processor is evaluated as true, we say that the processor is *privileged* to execute the action part (or to *make a move*).

- (b) At the initial configuration, if none of the processors is privileged, then the system is deadlocked. Otherwise, if a privileged processor exists, the *central demon* in the system will randomly select exactly one among all the privileged processors to make a move, in a single atomic step. The local state of the selected processor thus is changed and in the meantime results in the change of the global configuration of the system. The system will then repeat the above process to change global configurations as long as it does not encounter any deadlock situation. Thus, the behavior of the system under the action of the algorithm can be described by *executions*. An infinite or finite sequence of configurations $\Gamma = (\gamma_1, \gamma_2, \dots)$ of the system is called an *execution* (of the algorithm in the system) if for any $i \geq 1$, γ_{i+1} is obtained from γ_i after exactly one processor in the system makes the i^{th} move $\gamma_i \rightarrow \gamma_{i+1}$, and in the case that Γ is finite, it is further required that no node is privileged in the last configuration.

Under this central demon model, Dijkstra introduced the notion of self-stabilization of a distributed system in his classic paper [1] in 1974 [2, 3]. Following Dijkstra's idea, in this paper we define an algorithm to be *self-stabilizing* if every execution of the algorithm has a suffix in which all configurations are legitimate. Note that although this definition of self-stabilizing algorithms is not sufficient for some cases such as self-stabilization with respect to the mutual exclusion problem, it does suffice for most cases, including self-stabilization with respect to the minimal distance-2 dominating set problem to be discussed in this paper.

1.2 Main Result

Let $G = (V, E)$ be a simple connected undirected graph that models a distributed system, with each node $i \in V$ representing a processor in the system and each edge $\{i, j\}$ representing the bidirectional link connecting processors i and j . A subset D of V is called a *dominating set* in G if every node outside D has a neighbor in D . A dominating set D in G is *minimal* if no proper subset of D is a dominating set in G . Based on these basic definitions, variants of dominating sets can be defined. Self-stabilizing algorithms for various dominating set problems have been proposed and studied in the past [4-12]. The dominating set problems dealt with in all these works are of distance-1 type.

In this paper, we focus on finding a minimal distance-2 dominating set in G . A subset D of V is called a *distance-2 dominating set* in G if for any node x in V , there is a node y in D such that the distance between x and y is not greater than two. A distance-2 dominating set D in G is *minimal* if no proper subset of D is a distance-2 dominating set in G . Since, by definition, the requirement for a distance-2 dominating set is less strict than that for a distance-1 dominating set, a minimal distance-2 dominating set can be expected to be smaller and thus be able to further reduce the resources needed for allocation. However, due to the main restriction that any processor in a distributed system can only access the data of its direct (distance-1) neighbors, a self-stabilizing algorithm for finding a minimal distance-2 dominating set is harder to get, and its correctness is harder to verify. In this paper, a self-stabilizing algorithm for finding a minimal distance-2 dominating set is proposed. The algorithm can be applied to any distributed system that operates under the central demon model. A rigorous correctness proof for the algorithm is also provided.

1.3 Organization of the Paper

The rest of this paper is organized as follows: In section 2, we propose our algorithm and show that in any legitimate configuration a minimal distance-2 dominating set can be identified. An example is given to illustrate the execution of the algorithm in section 3. The correctness proof of the algorithm is provided in section 4, and finally in section 5 some remarks conclude the discussion in this paper.

2. THE ALGORITHM AND THE LEGITIMATE CONFIGURATIONS

We assume that the system in consideration is modeled by a simple connected undirected graph $G = (V, E)$. Each processor has a unique identifier, and we will use the unique identifier to represent the processor. Moreover, each processor x in the system maintains a shared register d_x whose value is 0, 1 or 2, and a shared register p_x whose value is the identifier of a neighbor of x . In all the following, $N(x) = \{y \in V \mid \{x, y\} \in E\}$, $N_0(x) = \{y \in N(x) \mid d_y = 0\}$, $N_1(x) = \{y \in N(x) \mid d_y = 1\}$, and $N_0^2(x) = \{y \in V \mid d(x, y) = 2 \text{ and } d_y = 0\}$, where $d(x, y)$ is the distance between nodes x and y .

Algorithm 1

{For any node x }

R1: $N_0(x) = \emptyset \wedge N_1(x) = \emptyset \wedge d_x \neq 0 \rightarrow d_x := 0$

R2: $N_0(x) \neq \emptyset \wedge d_x \neq 1 \rightarrow d_x := 1; p_x := \min N_0(x)$, where $\min N_0(x)$ is the smallest element in the set $N_0(x)$

R3: $N_0(x) = \emptyset \wedge N_1(x) \neq \emptyset \wedge [d_x = 1 \vee (d_x = 0 \wedge \exists y \in N_1(x) \text{ s.t. } p_y \neq x)] \rightarrow d_x := 2$

R4: $d_x = 1 \wedge N_0(x) \neq \emptyset \wedge p_x \notin N_0(x) \rightarrow p_x := \min N_0(x)$

The legitimate configurations are defined to be all those configurations in which no node in the system is privileged.

Lemma 1 In any legitimate configuration, the set $D = \{x \in V \mid d_x = 0\}$ is a distance-2 dominating set in G .

Proof: Suppose that the system is in a legitimate configuration. Then no node in the system is privileged. Let u be any node in $V - D$. Then $d_u \neq 0$.

Case 1: $N_0(u) \neq \emptyset$. Then $N_0(u) \cup N_0^2(u) \neq \emptyset$.

Case 2: $N_0(u) = \emptyset$. Then since $d_u \neq 0$ and u is not privileged by $R1$, $N_1(u) \neq \emptyset$. Let v be any node in $N_1(u)$. Then $d_v = 1$. Since v is not privileged by $R1$, the condition $N_0(v) = \emptyset \wedge N_1(v) = \emptyset$ is false. Similarly, since v is not privileged by $R3$, the condition $N_0(v) = \emptyset \wedge N_1(v) \neq \emptyset$ is false. It follows that $N_0(v) \neq \emptyset$. Since $v \in N(u)$ and $d_u \neq 0$, $N_0(v) \subseteq N_0(u) \cup N_0^2(u)$. Hence $N_0(u) \cup N_0^2(u) \neq \emptyset$.

In both cases, we get that $N_0(u) \cup N_0^2(u) \neq \emptyset$. Thus there is a node y in D such that the distance between u and y is not greater than two. Therefore the set D is a distance-2 dominating set in G . \square

Theorem 1 In any legitimate configuration, the set $D = \{x \in V \mid d_x = 0\}$ is a minimal distance-2 dominating set in G .

Proof: Suppose that the system is in a legitimate configuration. Then no node in the system is privileged. Let x be any node in D . Then $d_x = 0$. Since x is not privileged by $R2$, $N_0(x) = \emptyset$. Let y be any node in $N(x)$. Then, since $d_x = 0$, $N_0(y) \neq \emptyset$. Since y is not privileged by $R2$, $d_y = 1$. Hence $N_1(x) \neq \emptyset$. This, together with the fact that $d_x = 0$, $N_0(x) = \emptyset$ and x is not privileged by $R3$, implies that for every $u \in N_1(x)$, $p_u = x$. In particular, $p_y = x$. Let z be any node in $N(y) - \{x\}$. Then since $d_y = 1$, $y \in N_1(z)$ and $N_1(z) \neq \emptyset$.

Case 1: $N_0(z) \neq \emptyset$. Then, since z is not privileged by $R2$, $d_z = 1$.

Case 2: $N_0(z) = \emptyset$. Then, since $N_1(z) \neq \emptyset$, $y \in N_1(z)$, $p_y = x \neq z$ and z is not privileged by $R3$, we have that $d_z = 2$.

From all the above, we see that for every $y \in N(x)$, $d_y = 1$ and for every $z \in N(y) - \{x\}$, $d_z \neq 0$. Thus $N_0(x) \cup N_0^2(x) = \emptyset$. It follows that the distance between x and any node in $D - \{x\}$ is greater than two. Therefore $D - \{x\}$ is not a distance-2 dominating set. This, together with Lemma 1, implies that D is a minimal distance-2 dominating set in G . \square

3. AN ILLUSTRATION

The example in Fig. 1 illustrates the execution of Algorithm 1. Note that in each configuration in Fig. 1, the shaded nodes represent the privileged nodes, whereas the shaded node with a bold circle represents the privileged node selected by the central demon to make a move.

4. CORRECTNESS PROOF

In this section we give a rigorous proof for the self-stabilization of Algorithm 1. For presentation's sake, for $i = 1, 2, 3, 4$, we say that a move $\gamma_m \rightarrow \gamma_{m+1}$ is an Ri -move made by a node x , if x executes rule Ri in the move $\gamma_m \rightarrow \gamma_{m+1}$.

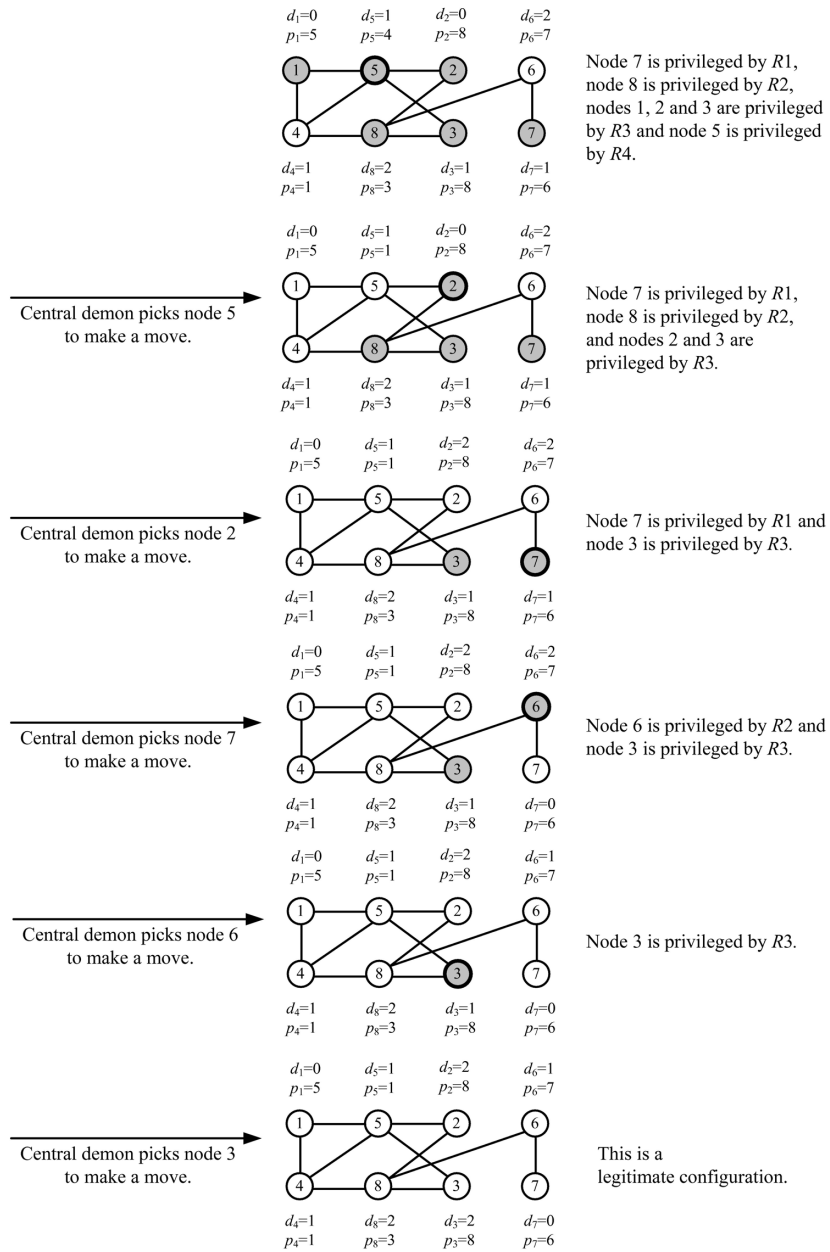


Fig. 1. An example to illustrate an execution of Algorithm 1. (In the last configuration, which is a legitimate configuration, a minimal distance 2-dominating set $D = \{1, 7\}$ can be identified.)

Lemma 2 Suppose x is a node in the system and $\Gamma = (\gamma_1, \gamma_2, \dots)$ is an execution of Algorithm 1. If $\gamma_p \rightarrow \gamma_{p+1}$ and $\gamma_q \rightarrow \gamma_{q+1}$ are two consecutive R_1 -moves in Γ made by x , then the first move made by x after γ_{p+1} must be an R_3 -move.

Proof: Since $\gamma_p \rightarrow \gamma_{p+1}$ and $\gamma_q \rightarrow \gamma_{q+1}$ are R1-moves made by x , $d_x = 0$ in γ_{p+1} and $d_x \neq 0$ in γ_q . Thus there must be a move made by x in the segment $(\gamma_{p+1}, \dots, \gamma_q)$ and let $\gamma_s \rightarrow \gamma_{s+1}$ be the first such move. Then $d_x = 0$ in $(\gamma_{p+1}, \dots, \gamma_s)$. Since $d_x \neq 1$ in γ_s , $\gamma_s \rightarrow \gamma_{s+1}$ cannot be an R4-move.

Now, we prove that $\gamma_s \rightarrow \gamma_{s+1}$ cannot be an R2-move. Since $\gamma_p \rightarrow \gamma_{p+1}$ is an R1-move made by x , $N_0(x) = N_1(x) = \emptyset$ in γ_{p+1} . Hence for any $y \in N(x)$, we have $d_y = 2$ in γ_{p+1} . Since $x \in N(y)$ and $d_x = 0$ in $(\gamma_{p+1}, \dots, \gamma_s)$, $N_0(y) \neq \emptyset$ in $(\gamma_{p+1}, \dots, \gamma_s)$. Thus y cannot execute R1 in $(\gamma_{p+1}, \dots, \gamma_s)$. This, together with the fact that $d_y = 2$ in γ_{p+1} , implies that $d_y \neq 0$ in γ_s . Hence $N_0(x) = \emptyset$ in γ_s . It follows that the move $\gamma_s \rightarrow \gamma_{s+1}$ cannot be an R2-move.

Since $\gamma_p \rightarrow \gamma_{p+1}$ and $\gamma_q \rightarrow \gamma_{q+1}$ are two consecutive R1-moves made by x , the move $\gamma_s \rightarrow \gamma_{s+1}$ cannot be an R1-move. Therefore $\gamma_s \rightarrow \gamma_{s+1}$ is an R3-move. \square

For any node $x \in V$ and for any configuration γ , we define $M_\gamma(x) = \{y \in V \mid d(x, y) = 2, y < x \text{ and } d_y = 0 \text{ in } \gamma\}$.

Lemma 3 Suppose x is a node in the system and $\Gamma = (\gamma_1, \gamma_2, \dots)$ is an execution of Algorithm 1 in which there is no R1-move made by any node $y < x$. If $\gamma_p \rightarrow \gamma_{p+1}$ and $\gamma_q \rightarrow \gamma_{q+1}$ are two consecutive R1-moves in Γ made by x , then $|M_{\gamma_p}(x)| > |M_{\gamma_q}(x)|$.

Proof: Let $\gamma_s \rightarrow \gamma_{s+1}$ be the first move made by x after γ_{p+1} . Then $d_x = 0$ in $(\gamma_{p+1}, \dots, \gamma_s)$. By Lemma 2, $\gamma_s \rightarrow \gamma_{s+1}$ is an R3-move. This, together with the fact that $d_x = 0$ in γ_s , implies that there exists a $u \in N(x)$ such that $d_u = 1$ and $p_u \neq x$ in γ_s . Since x executes R1 in the move $\gamma_p \rightarrow \gamma_{p+1}$, $N_0(x) = N_1(x) = \emptyset$ in γ_{p+1} . Thus $d_u = 2$ in γ_{p+1} . Since $d_u = 1$ in γ_s , u must make an R2-move in $(\gamma_{p+1}, \dots, \gamma_s)$. Let $\gamma_t \rightarrow \gamma_{t+1}$ be the last such move. Then $d_u = 1$, $p_u = \min N_0(u)$ in γ_{t+1} and u cannot make any R2-move in $(\gamma_{t+1}, \dots, \gamma_s)$.

Case 1: u never makes any R4-move in $(\gamma_{t+1}, \dots, \gamma_s)$. Then p_u can never change in $(\gamma_{t+1}, \dots, \gamma_s)$. Since $p_u \neq x$ in γ_s , $p_u \neq x$ in γ_{t+1} .

Case 2: u makes an R4-move in $(\gamma_{t+1}, \dots, \gamma_s)$. Let $\gamma_r \rightarrow \gamma_{r+1}$ be the first such move. Then $p_u \notin N_0(u)$ in γ_r and u never makes any R4-move in $(\gamma_{t+1}, \dots, \gamma_r)$. Since $d_x = 0$ in $(\gamma_{p+1}, \dots, \gamma_s)$, $x \in N_0(u)$ in γ_r . Hence $p_u \neq x$ in γ_r . Since u never makes any R2- or R4-move in $(\gamma_{t+1}, \dots, \gamma_r)$, p_u never changes in $(\gamma_{t+1}, \dots, \gamma_r)$. Thus $p_u \neq x$ in γ_{t+1} .

In both cases, we get that $p_u \neq x$ in γ_{t+1} . Since $p_u = \min N_0(u)$ in γ_{t+1} , $\min N_0(u) \neq x$ in γ_{t+1} . Since $d_x = 0$ in γ_{t+1} , $x \in N_0(u)$ in γ_{t+1} . Hence $\min N_0(u) < x$ in γ_{t+1} .

Let $w = \min N_0(u)$ in γ_{t+1} . Then $p_u = w < x$ and $d_w = 0$ in γ_{t+1} . Since $d_x = 0$ in $(\gamma_{p+1}, \dots, \gamma_s)$, for any $v \in N(x)$, we have that $N_0(v) \neq \emptyset$ in $(\gamma_{p+1}, \dots, \gamma_s)$. Hence v cannot execute R1 in $(\gamma_{p+1}, \dots, \gamma_s)$. Since $N_0(x) = N_1(x) = \emptyset$ in γ_{p+1} , $d_v = 2$ in γ_{p+1} . Thus $d_v \neq 0$ in γ_{t+1} . However, $d_w = 0$ in γ_{t+1} . So $w \notin N(x)$. This, together with the fact that $w \in N(u)$ and $u \in N(x)$, implies that $d(x, w) = 2$. Thus $w \in M_{\gamma_{t+1}}(x)$.

Since x executes R1 in the move $\gamma_q \rightarrow \gamma_{q+1}$, $N_0(x) = N_1(x) = \emptyset$ in γ_q . Hence $d_u = 2$ in γ_q . Since $d_u = 1$ in γ_s , u must make an R3-move $\gamma_l \rightarrow \gamma_{l+1}$ in $(\gamma_s, \dots, \gamma_q)$. Thus $N_0(u) = \emptyset$ in γ_{l+1} . It follows that $d_w \neq 0$ in γ_{l+1} . Since $w < x$, w never makes an R1-move in Γ . In particular, w never makes an R1-move in $(\gamma_{t+1}, \dots, \gamma_q)$. Hence $d_w \neq 0$ in γ_q . This implies that $w \notin M_{\gamma_q}(x)$.

Consequently, $w \in M_{\gamma_{t+1}}(x)$ and $w \notin M_{\gamma_q}(x)$. On the other hand, since for any node y

$< x, y$ cannot make any $R1$ -move in Γ , and any move in Γ cannot produce any new 0 -node (*i.e.*, node with 0 d -value) of which the identifier is smaller than x . It follows that $M_{\gamma_p}(x) \supseteq M_{\gamma_{p+1}}(x) \supseteq \dots \supseteq M_{\gamma_l}(x) \supseteq M_{\gamma_{l+1}}(x) \supseteq \dots \supseteq M_{\gamma_q}(x)$. Thus $M_{\gamma_{l+1}}(x) \supsetneq M_{\gamma_q}(x)$ and thus $M_{\gamma_p}(x) \supsetneq M_{\gamma_q}(x)$. Therefore $|M_{\gamma_p}(x)| > |M_{\gamma_q}(x)|$ and the lemma is proved. \square

Lemma 4 Suppose x is a node in the system and Γ is an execution of Algorithm 1. If for every node $y < x$, Γ contains a finite number of $R1$ -moves made by y , then Γ contains a finite number of $R1$ -moves made by x .

Proof: Suppose that for any node $y < x$, Γ contains only a finite number of $R1$ -moves made by y . Then there exists a suffix Γ' of Γ such that for any $y < x$, y does not make any $R1$ -move in Γ' . Let $\gamma_{p_1} \rightarrow \gamma_{p_1+1}, \gamma_{p_2} \rightarrow \gamma_{p_2+1}, \dots$, where $p_1 < p_2 < \dots$, be all the $R1$ -moves made by x in Γ' . Then, by Lemma 3, we have $|M_{\gamma_{p_1}}(x)| > |M_{\gamma_{p_2}}(x)| > \dots$, a decreasing sequence of nonnegative integers. Therefore Γ' contains a finite number of $R1$ -moves made by x , and thus Γ contains a finite number of $R1$ -moves made by x . \square

Lemma 5 Every execution Γ of Algorithm 1 is finite.

Proof: Let $V = \{x_1, x_2, \dots, x_n\}$ such that $x_1 < x_2 < \dots < x_n$. By applying the above lemma n times to x_1, x_2, \dots, x_n in the order of (x_1, x_2, \dots, x_n) , we get that Γ contains a finite number of $R1$ -moves. Hence there exists a suffix Γ' of Γ such that no node in the system makes any $R1$ -move in Γ' . If a node x ever executes $R3$ in the move $\gamma_l \rightarrow \gamma_{l+1}$ in Γ' , then $d_x = 2$ and $N_0(x) = \emptyset$ in γ_{l+1} . Since no node makes any $R1$ -move in Γ' , $N_0(x) = \emptyset$ from γ_{l+1} on. Hence x never executes $R2$ (or $R1$ of course) after the move $\gamma_l \rightarrow \gamma_{l+1}$. Thus d_x remains to be 2 , and hence can not execute $R3$ after the move $\gamma_l \rightarrow \gamma_{l+1}$. This shows that each node in the system can execute $R3$ at most once in Γ' . Therefore, there exists a suffix Γ'' of Γ' such that in Γ'' , no node executes $R1$ or $R3$. Hence, by the similar argument to that for $R3$ above, in Γ'' each node in the system can execute $R2$ at most once. Therefore, there exists a suffix Γ''' of Γ'' such that in Γ''' , no node executes $R1, R2$ or $R3$. Thus, no node changes its d -value in Γ''' and hence in Γ''' , each node in the system can execute $R4$ at most once. It follows from all the above that Γ is a finite execution. \square

Theorem 2 Algorithm 1 is self-stabilizing under the central demon model and solves the minimal distance-2 dominating set problem.

Proof: By Lemma 5, every execution of Algorithm 1 is finite. By the definition of a finite execution, no node is privileged in the last configuration of every finite execution. Hence the last configuration of every execution of Algorithm 1 is a legitimate configuration. It follows that Algorithm 1 is self-stabilizing. From this and Theorem 1, it follows that the minimal distance-2 dominating set problem is solved. \square

5. CONCLUDING REMARKS

We have proposed a self-stabilizing algorithm for finding a minimal distance-2 dominating set in a general network in which the central demon model is assumed. We have also provided the correctness proof. Due to the main restriction that any processor

in a distributed system can only access the data of its direct (distance-1) neighbors, and due to that the distance-2 dominating set problem requires a processor to know the information from beyond its direct neighbors, it is not easy to design and verify the correctness of a self-stabilizing algorithm for the minimal distance-2 dominating set problem. Moreover, the worst-case stabilization time of such an algorithm is even harder to compute. Although we think that the worst-case stabilization time of such an algorithm is $O(n^i)$ for some constant i , we cannot prove it.

There are several directions for further research:

1. To generalize the results in this paper from the minimal distance-2 dominating set problem to the minimal distance- k dominating set problem (k is an arbitrary positive integer).
2. To generalize the results in this paper from the central demon model to the distributed demon model.
3. To prove that the worst-case stabilization time of the proposed algorithm in this paper is $O(n^i)$ for some constant i , where n is the number of processors in the system.

REFERENCES

1. E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, Vol. 17, 1974, pp. 643-644.
2. E. W. Dijkstra, "Self-stabilization in spite of distributed control," *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, Berlin-Heidelberg, New York, 1982, pp. 41-46.
3. E. W. Dijkstra, "A belated proof of self-stabilization," *Distributed Computing*, Vol. 1, 1986, pp. 5-6.
4. S. Shukla, D. J. Rosenkrantz, and S. S. Ravi, "Observations on self-stabilizing graph algorithms on anonymous networks," in *Proceedings of the Workshop on Self-Stabilizing Systems*, 1995, pp. 7.1-7.15.
5. M. Ikeda, S. Kamei, and H. Kakugawa, "A space-optimal self-stabilizing algorithm for the maximal independent set problem," in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2002, pp. 70-74.
6. 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 of the International Parallel and Distributed Processing Symposium, Workshop on Advances in Parallel and Distributed Computational Models*, 2003, pp. 22-26.
7. W. Goddard, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, "A self-stabilizing distributed algorithm for minimal total domination in an arbitrary system graph," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium, Workshop on Formal Methods for Parallel Programming*, 2003, pp. 240-244.
8. S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, "Self-stabilizing algorithms for minimal dominating sets and maximal independent sets," *Computers and Mathematics with Applications*, Vol. 46, 2003, pp. 805-811.
9. T. C. Huang, C. Y. Chen, and C. P. Wang, "A linear-time self-stabilizing algorithm

for the minimal 2-dominating set in general networks,” *Journal of Information Science and Engineering*, Vol. 24, 2008, pp. 175-187.

10. T. C. Huang, J. C. Lin, C. Y. Chen, and C. P. Wang, “A self-stabilizing algorithm for finding a minimal 2-dominating set assuming the distributed demon model,” *Computers and Mathematics with Applications*, Vol. 54, 2007, pp. 350-356.
11. S. Kamei and H. Kakugawa, “A self-stabilizing algorithm for the distributed minimal k -redundant dominating set problem in tree networks,” in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 720-724.
12. 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 International Workshop on Distributed Computing*, 2003, pp. 26-32.



Ji-Cherng Lin (林基成) received his B.S. degree in Mathematics from National Tsing Hua University, Taiwan in 1983, and M.S. and Ph.D. degrees in Computer Science from National Tsing Hua University, Taiwan in 1985 and 1991, respectively. He is now an Associate Professor of the Department of Computer Science and Engineering of Yuan Ze University, Taiwan. His current research interest is focused mainly on self-stabilizing algorithms.



Tetz C. Huang (黃哲志) received the B.S. degree in Mathematics from Fu Jen Catholic University, Taiwan, in 1975. From 1979 to 1986, he studied at the University of Chicago and received the M.S. degree in Mathematics. He was a lecturer in mathematics at the University of Chicago from 1981 to 1984. Since 1989, he has joined in Yuan Ze University, Taiwan. He is currently a professor in Yuan Ze University. He has publications on variational inequalities and self-stabilizing systems. His current research interest is focused mainly on self-stabilizing systems.



Cheng-Pin Wang (王承斌) received the B.S. and M.S. degrees in Mathematics from Chung Yuan Christian University, Taiwan, in 1996 and 1998, respectively. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering, Yuan Ze University and a part-time lecturer at Department of Applied Mathematics, Chung Yuan Christian University. His current research interest is self-stabilizing systems.



Chih-Yuan Chen (陳志遠) received the B.S. and M.S. degrees in Mathematics from Chung Yuan Christian University, Taiwan, in 1996 and 1998, respectively. In 2007, he received the Ph.D. degrees in Computer Science and Engineering from Yuan Ze University, Taiwan. He is currently an assistant professor in the Department of Computer Science and Information Engineering, Nanya Institute of Technology, Taiwan. His current research interest is self-stabilizing systems.