A Self-stabilizing Algorithm for the Center-finding Problem Under a More Realistic Demon

Submitted to

Workshop on Algorithms and Computational Molecular Biology

ABSTRACT

The problem of locating centers in distributed systems are especially important since they are ideal locations for placing resources that need to be shared among different processors in a network. In this paper, we design and prove the correctness of a self-stabilizing center-finding algorithm under a more realistic demon in a distributed system with a tree topology.

Correspondence to: Tetz C. Huang
Department of Computer Engineering and Science
Yuan-Ze University
135 Yuan-Tung Rd., Chung-Li Taoyuan 32026, Taiwan
E-mail: cstetz@saturn.yzu.edu.tw
TEL: 03-4638800 ext. 371 FAX: 03-4638850

Ji-Cherng Lin
Department of Computer Engineering and Science
Yuan-Ze University
135 Yuan-Tung Rd., Chung-Li Taoyuan 32026, Taiwan
E-mail: csjclin@saturn.yzu.edu.tw
TEL: 03-4638800 ext. 354 FAX: 03-4638850

Nathan Mou
Department of Computer Engineering and Science
Yuan-Ze University
135 Yuan-Tung Rd., Chung-Li Taoyuan 32026, Taiwan
E-mail: nathan24@ms57.hinet.net
TEL: 03-4935103 ext. 236 FAX: 03-4935203

Keywords: Self-stabilizing algorithm, model of computation, read/write separate atomicity, interleaving model, center.

A SELF-STABILIZING ALGORITHM FOR THE CENTER-FINDING PROBLEM UNDER A MORE REALISTIC DEMON

TETZ C. HUANG, JI-CHERNG LIN AND NATHAN MOU

Abstract. The problem of locating centers in distributed systems are especially important since they are ideal locations for placing resources that need to be shared among different processors in a network. In this paper, we design and prove the correctness of a self-stabilizing center-finding algorithm under a more realistic demon in a distributed system with a tree topology.

1. Introduction

E. W. Dijkstra first introduced the notion of self-stabilization in a distributed system in his pioneering paper [2] (cf. also [3][4]) in 1974, in which he coined the phrase and showed the feasibility of designing such algorithms in a distributed system. In addition to Dijkstra's classic papers [2][3][4], a good reference for the basics on the self-stabilizing system of Dijkstra type can be found in Bruell et al. [5]. Later in 1993, Dolev et al., introduced a new type of self-stabilizing system in their famous paper [1]. The computational model of the new type of system assumes the *read/write separate atomicity*. Under such an assumption, each atomic step in the system of Dolev type consists of internal computations

Date: July 2, 2002.
Key words and phrases. Self-stabilizing algorithm, center, rooted tree, model of computation, read/write separate atomicity, read/write composite atomicity.

and either a single read operation or a single write operation. In this setting, Dolev et al. presented two simple self-stabilizing algorithms in [1], one of which is for the mutual exclusion problem and the other is for the breadth-first search tree problem. As is proved in the paper, both algorithms are self-stabilizing under the computational model of Dolev type.

Self-stabilizing center-finding algorithms in a distributed system that uses the computational model of Dijkstra type have been investigated during the past [5][6]. In this paper, we design and prove the correctness of a self-stabilizing algorithm that finds the center(s) for any distributed system with a tree topology that uses the computational model of Dolev type. To the best of our knowledge, there has been no published paper so far that discusses the self-stabilizing center-finding algorithm in a distributed system whose computational model assumes the read/write separate atomicity.

The rest of this paper is arranged as follows. In Section 2, the algorithm is proposed and the meaning of the legitimate configuration is explained. The correctness proof of the algorithm is given in Section 3. Finally, in Section 4, some remarks conclude the whole discussion.

2. The Center-finding Algorithm

Let $T = (V, E)$ be an undirected tree that is used to model a distributed system with a tree topology. Each node $x \in V$ represents a processor in the system and each edge $\{x, y\} \in E$ represents the bidirectional link connecting processors x

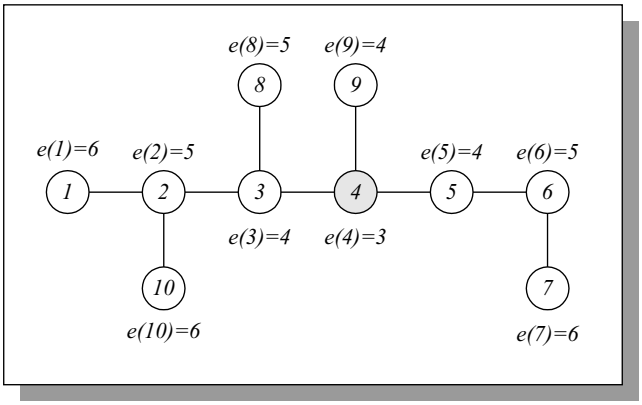

FIGURE 1. Eccentricities of nodes in a tree. The shaded node stands for the unique center of the tree.

and y. For any x, $y \in V$, let $d(x, y)$ denote the *distance* between x and y, that is, the length of the unique simple path in T that connects x and y. Let $e(x) = \max \{d(x, y) \mid y \in V\}$ denote the *eccentricity* of a node x, viz. the distance between x and a farthest vertex from x in T. Then a *center* of T is a node with the minimum eccentricity. The so-called *center-finding problem* for the system T is to identify the center(s) of the system. Proposition 1 states a well-known property regrading the center(s) of a tree. The proof of it can be found in Theorem 2.1 in [7].

Proposition 1. *A tree has a unique center or two adjacent centers (cf. Figure 1 and 2).*

For later use, for any $x \in V$, we define $N(x)$ to be the set of all x's neighbors. We also introduce some notations $p(x)$, $T(x)$ and $H(x)$ relating to T in the following definition. Then, we demonstrate some properties with regard to $H(x)$.

Definition 1. *Let $T = (V, E)$ be as above.*

Case 1. *T has a unique center c. In this case, we designate c as the root and T thus becomes a rooted tree at c. For any $x \in V - \{c\}$, the parent of x is denoted by $p(x)$. For any $x \in V$, let $T(x)$ represents the subtree of T rooted at x . Then we define $H(x) = \max\{d(x, y) \mid y$ is a leaf node in $T(x)\}$, i.e., the height of $T(x)$.*

Case 2. *T has two centers c_1, c_2. In this case, we first delete from T the edge connecting c_1 and c_2 and thus obtain two subtrees T_1 and T_2 of T, where $c_1 \in V(T_1)$ and $c_2 \in V(T_2)$. T_1 and T_2 can be considered as rooted trees at c_1 and c_2, respectively. For any $x \in V - \{c_1, c_2\}$, $p(x)$ denotes the parent of x in the rooted tree to which x belongs. For any $x \in V$, the meanings of $T(x)$ and $H(x)$ are also apparent.*

We now give two examples to assist readers in comprehending the notions given in above definition. The tree shown in Figure 3-(1) has a unique center c. It induces a rooted tree at c shown in Figure 3-(2). The shaded nodes constitutes

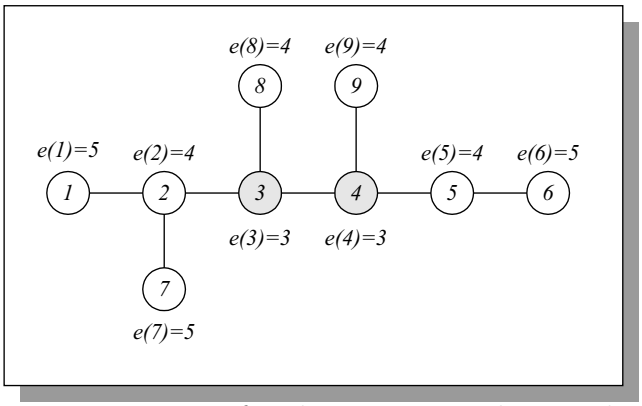

FIGURE 2. Eccentricities of nodes in a tree. The two shaded nodes stand for the two centers of the tree.

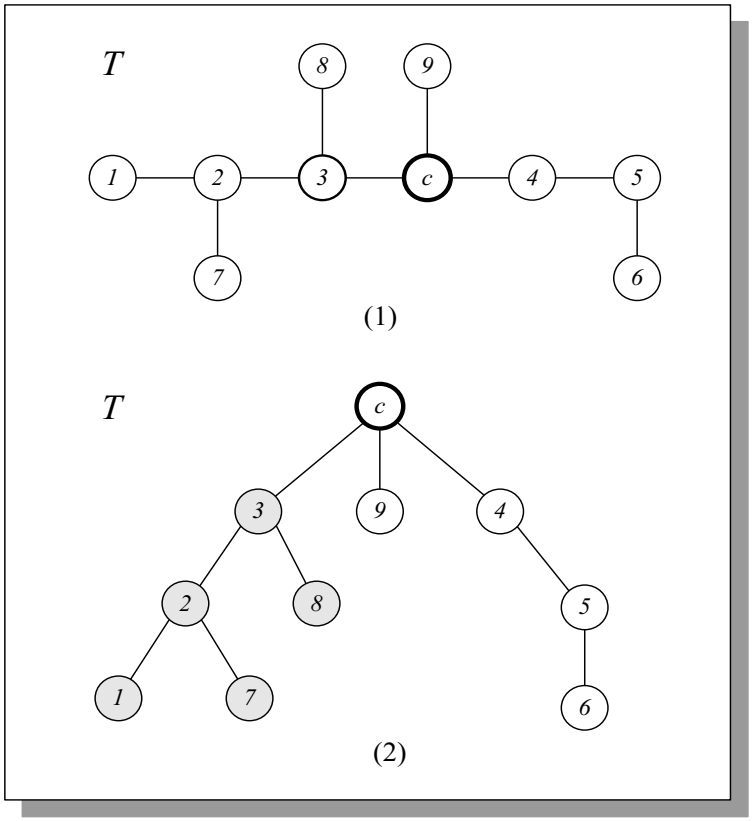

FIGURE 3. A tree with a unique center c and the induced rooted tree rooted at c.

the subtree $T(3)$ with $H(3) = 2$. Also note that $p(5) = 4$. Next, the tree shown in Figure 4-(1) has two adjacent centers c_1 and c_2. It induces two rooted trees T_1 and T_2 in Figure 4-(2). The shaded nodes represent the subtree $T(2)$ with $H(2) = 1$. Also note that $p(7) = 2$.

Lemma 1. *Suppose x is a node in T such that $\deg(x) > 1$ and x is not a center of T. Then $H(p(x)) \geq H(x) + 1$ and [$\forall y \in N(x) - \{p(x)\}$, $H(y) \leq H(x) - 1$] and [$\exists y_0 \in N(x) - \{p(x)\}$ such that $H(y_0) = H(x) - 1$].*

Lemma 2. *Suppose T has a unique center c. Then* [$\forall y \in N(c)$, $H(y) \leq H(c)-1$] *and* [$\exists y_1, y_2 \in N(c)$ *such that* $y_1 \neq y_2$ *and* $H(y_1) = H(y_2) = H(c) - 1$].

Lemma 3. *Suppose T has two centers c_1 and c_2. Then* $H(c_1) = H(c_2)$, [$\forall y \in N(c_1) - \{c_2\}$, $H(y) \leq H(c_1) - 1$] *and* [$\exists y_0 \in N(c_1) - \{c_2\}$ *such that* $H(y_0) = H(c_1) - 1$].

Later in this section, we will propose a self-stabilizing algorithm that finds the center(s) for the distributed system T with a tree topology. The underlying model of computation employed here in the system was introduced by Dolev et

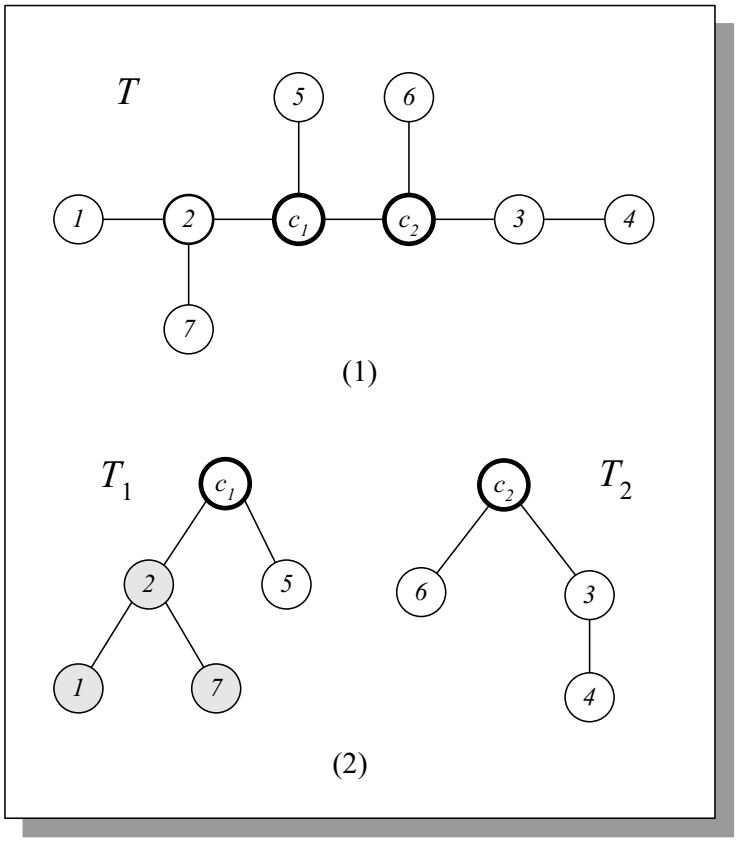

FIGURE 4. A tree with two centers c_1 and c_2 and the induced rooted trees rooted at c_1 and c_2.

al. in [1] (cf. also [8]) that assumes the read/write separate atomicity instead of the commonly used read/write composite atomicity. Thus, for each $x \in V$ and for each $y \in N(x)$, let x maintain a register h_{xy}, in which x writes and from which y reads. The register is serializable with respect to read and write operations. For each processor x with $\deg(x) > 1$ and for each $y \in N(x)$, let x also maintain a local variable r_{yx}, in which x stores the value that it reads from the shared register h_{yx} of the neighbor y. The values of each register h_{xy} and each local variable r_{yx} are in the range $N = \{0, 1, 2, \ldots\}$. Figure 5 illustrates the distributed system that assumes the read/write separate atomicity and is equipped with the proposed algorithm. $N_{x,r} = \{r_{yx} \mid y \in N(x)\}$ denotes the multi-set of values of all x's local variables whereas $N^-_{x,r} = N_{x,r} - \{\max N_{x,r}\}$ denotes the set $N_{x,r}$ with one maximum value in it removed. For example, if $N_{x,r} = \{3, 4, 4\}$, then $N^-_{x,r} = \{3, 4\}$. The legitimate configurations for the system are defined to be those configurations in which $\forall x \in V$, [$\deg(x) = 1 \wedge h_{xy} = 0$ for the unique $y \in N(x)$] or [$\deg(x) > 1 \wedge (\forall y \in N(x),\ r_{yx} = h_{yx} \wedge\ h_{xy} = 1 + \max N^-_{x,r})$].

Theorem 1 (Uniqueness)**.** *If the system $T = (V, E)$ is in any legitimate configuration, then $\forall x \in V$ and $\forall y \in N(x)$, $h_{xy} = H(x)$, the height of $T(x)$.*

The above theorem shows the meaning and the uniqueness of the legitimate configuration. The converse is also true, which shows the existence of the legitimate configuration.

Theorem 2 (Existence)**.** *The configuration in which* [$\forall x \in V$ *and* $\forall y \in N(x)$, $h_{xy} = H(x)$] *and* [$\forall x \in V$ *with* $\deg(x) > 1$ *and* $\forall y \in N(x)$, $r_{yx} = h_{yx}$] *is a legitimate configuration.*

The above two theorems reveal that there is actually a unique legitimate configuration, that is, the configuration in the statement of Theorem 2, and when the system is in the legitimate configuration, for any $x \in V$ and for any $y \in N(x)$, the register h_{xy} records the height $H(x)$ of $T(x)$.

Now we equip the system with the algorithm.

Self-stabilizing center-finding algorithm

{*For every leaf node* x *in the system*}

1. repeat forever

2. *if* $h_{xy} \neq 0$ *then write*$(h_{xy} := 0)$ *endif*

(where y is the unique neighbor of x.)

3. endrepeat

{*For every non-leaf node* x *in the system*}

01. repeat forever

02. for each $y \in N(x)$ *do*

03. *read* $(r_{yx} := h_{yx})$

04. endfor

05. for each $y \in N(x)$ *do*

06. *if* $h_{xy} \neq 1 + \max N^-_{x,r}$ *then write* $(h_{xy} = 1 + \max N^-_{x,r})$ *endif*

07. endfor

08. endrepeat

It should be understood that each processor in the system runs its own program indefinitely and at its own pace, and the running of the program has to follow the order of the statements in the program.

3. Correctness Proof

We now give the correctness proof in the following theorem. To facilitate the presentation in the following proof, we define some terminologies. We say that a node x with $\deg(x) > 1$ *just completes a full round of reading all its neighbors* whenever x just completes a full execution of the loop from statement 02 to statement 04 in the algorithm. Likewise, we say that a node x with $\deg(x) > 1$ *just completes a full round of writing all its registers* whenever x just completes a full execution of the loop from statement 05 to statement 07. For any time instant t, we use $h_{xy}(t^+)$ to denote the value of h_{xy} right after t and $h_{xy}(t^-)$ to denote the value of h_{xy} right before t. If $h_{xy}(t^+) = h_{xy}(t^-)$, the value of h_{xy} at t is well-defined and is denoted as $h_{xy}(t)$; otherwise, $h_{xy}(t)$ is undefined. Likewise, $r_{yx}(t^+)$ and $r_{yx}(t^-)$ stand for the value of r_{yx} right after t and the value of r_{yx} right before t, respectively. If $r_{yx}(t^+) = r_{yx}(t^-)$, the value of r_{yx} at t is well-defined and is denoted as $r_{yx}(t)$; otherwise, $r_{yx}(t)$ is undefined.

Theorem 3 (Self-stabilization)**.** *Regardless of any initial state, the system will converge to the legitimate state and then stay in the legitimate state thereafter.*

Proof. Let $t = 0$ be the initial time instant and L be the diameter of T. Let $m = \lfloor \frac{L}{2} \rfloor$.

Claim. $\forall j \in \{0, 1, \ldots, m\}$, there exists an instant $t_j > 0$ such that $\forall t > t_j$, [$\forall x \in V$ with $0 \leq H(x) \leq j$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$] and [$\forall x \in V$ with $H(x) > j$ and $\forall y \in N(x)$, $h_{xy}(t^+) \geq j$].

Proof of the Claim. We prove the claim by induction on j. For $j = 0$, in view of statement 2 in the algorithm, it is obvious that for each $x \in V$ with $H(x) = 0$, there exists a $t_0(x) > 0$ such that $\forall t > t_0(x)$, $h_{xy}(t) = 0$ for $y \in N(x)$. Let $t_0 = \max_{x \in M_0} t_0(x)$. Then, $\forall t > t_0$, [$\forall x \in V$ with $H(x) = 0$, $h_{xy}(t) = 0$ for $y \in N(x)$] and [$\forall x \in V$ with $H(x) > 0$ and $\forall y \in N(x)$, $h_{xy}(t^+) \geq 0$ (since, as mentioned before, the register h_{xy} always attains a value in $N = \{0, 1, 2, \ldots\}$ for any $x \in V$ and for any $y \in N(x)$)]. Hence the claim is true for $j = 0$. Let $0 \leq k < m$. Assume that for $j = k$, the claim is true, that is, there exists a $t_k > 0$ such that $\forall t > t_k$, [$\forall x \in V$ with $0 \leq H(x) \leq k$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$] and [$\forall x \in V$ with $H(x) > k$ and $\forall y \in N(x)$, $h_{xy}(t^+) \geq k$].

Subclaim 1. If $x \in V$ with $H(x) = k + 1$, then there exists a $t_1(x) > t_k$ such that $\forall t > t_1(x)$, $\max N^-_{x,r}(t) = k$.

Proof of Subclaim 1.

Case 1. x is not a center. Then, by Lemma 1, [$H(p(x)) \geq k + 2$], [$\forall y \in$

$N(x)-\{p(x)\}$, $H(y) \leq k$] and [$\exists y_0 \in N(x)-\{p(x)\}$ such that $H(y_0) = k$]. Thus, by the induction hypothesis, $\forall t > t_k$, [$h_{p(x)x}(t^+) \geq k$], [$\forall y \in N(x) - \{p(x)\}$, $h_{yx}(t) = H(y) \leq k$] and [$h_{y_0x}(t) = H(y_0) = k$]. Let $t_{p(x)} > t_k$ be an instant at which x reads $r_{p(x)x} = h_{p(x)x}$. Then $\forall t \geq t_{p(x)}$, if we let $t'_{p(x)}$ be the last instant in the time interval $(t_k, t]$ at which x reads $r_{p(x)x} = h_{p(x)x}$, then we can see that $r_{p(x)x}(t^+) = h_{p(x)x}(t'_{p(x)})$ and thus $r_{p(x)x}(t^+) \geq k$. Similarly, $\forall y \in N(x) - \{p(x)\}$, $\exists t_y > t_k$ such that [$\forall t \geq t_y$, $r_{yx}(t^+) \leq k$] and [$\forall t \geq t_{y_0}$, $r_{y_0x}(t^+) = k$]. Let $t_1(x) = \max\limits_{y \in N(x)} t_y$. Then $t_1(x) > t_k$ and $\forall t \geq t_1(x)$, max $N^-_{x,r}(t^+) = k$. Consequently, $\forall t > t_1(x)$, max $N^-_{x,r}(t^+) = k$.

Case 2. x is the unique center of T. By employing Lemma 2 and arguing analogously as in Case 1, we will get a $t_1(x) > t_k$ such that $\forall t > t_1(x)$, max $N^-_{x,r}(t^+) = k$.

Case 3. x is one of the two centers of T. By employing Lemma 3 and arguing analogously as in Case 1, we will obtain a $t_1(x) > t_k$ such that $\forall t > t_1(x)$, max $N^-_{x,r}(t^+) = k$. Therefore, Subclaim 1 is proved.

Let x and $t_1(x)$ be as in Subclaim 1. Let $t_2(x) > t_1(x)$ be the first instant after $t_1(x)$ at which x just completes a full round of reading all its neighbors. Let $y \in N(x)$ be arbitrary. (a) If after $t_1(x)$, the value of h_{xy} is never changed, then $h_{xy}(t_2(x)) = 1 + \max N^-_{x,r}(t_2(x))$. (Otherwise, x will execute statement 06 in the algorithm to change the value of h_{xy} after $t_2(x)$). Hence, $h_{xy}(t_2(x)) = 1 + k$, by Subclaim 1. Therefore, we have ($\sharp$): $\forall t \geq t_1(x)$, $h_{xy}(t) = h_{xy}(t_2(x)) = 1+k$. (b) If after $t_1(x)$, the value of h_{xy} is ever changed, then let $\bar{t}_y > t_1(x)$ be the first instant

after $t_1(x)$ at which the value of h_{xy} is changed. Then $h_{xy}(\bar{t}_y^+) = 1 + \max N_{x,r}^-(\bar{t}_y^+)$. Hence, $h_{xy}(\bar{t}_y^+) = 1 + k$, again by Subclaim 1. Since after $\bar{t}_y$, the guard condition in statement 06 in the algorithm never evaluates to true, node x will never execute the write action. Therefore, we have (♯♯): $\forall t > \bar{t}_y$, $h_{xy}(t) = 1 + k$. From (♯) and (♯♯) above, it follows obviously that there exists a $t^* > t_k$ such that $\forall t > t^*$ and $\forall y \in N(x)$, $h_{xy}(t) = k+1$. Thus, we have shown (■): $\forall x \in V$ with $H(x) = k+1$, $\exists t^* > t_k$ such that $\forall t > t^*$ and $\forall y \in N(x)$, $h_{xy}(t) = k + 1$. By arguing in the same manner as in the above proof of Subclaim 1 and by employing Lemmas 1, 2 and 3 again, we will obtain

Subclaim 2. If $x \in V$ with $H(x) > k + 1$, then there exists a $t_1^*(x) > t_k$ such that $\forall t \geq t_1^*(x)$, $\max N_{x,r}^-(t^+) \geq k$.

Then, arguing analogously as right after Subclaim 1, we will eventually get (★): For any $x \in V$ with $H(x) > k + 1$, there exists a $\tilde{t} > t_k$ such that $\forall t > \tilde{t}$ and $\forall y \in N(x)$, $h_{xy}(t^+) \geq k+1$. Consequently, by (■) and (★) above, there exists an instant $t_{k+1} > 0$ (e.g. t_{k+1} can be chosen to be $\max\{t^*, \tilde{t}\}$) such that $\forall t > t_{k+1}$, [$\forall x \in V$ with $0 \leq H(x) \leq k + 1$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$] and [$\forall x \in V$ with $H(x) > k + 1$ and $\forall y \in N(x)$, $h_{xy}(t) \geq k + 1$], that is, the claim is true for $j = k + 1$. Therefore, by the postulate of mathematical induction, the claim at the beginning is proved.

According to above claim, there exists a $t_m > 0$ such that $\forall t > t_m$, $\forall x \in V$ with $0 \leq H(x) \leq m$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$. This obviously implies that $\forall x \in V$ and $\forall y \in N(x)$, $h_{xy} = H(x)$ never changes after t_m. Consequently,

in view of the algorithm, there exists a $t_m^* > t_m$ such that after t_m^*, $\forall x \in V$ with $H(x) > 0$ and $\forall y \in N(x)$, $r_{yx} = h_{yx}$. Therefore, after t_m^*, [$\forall x \in V$ with $H(x) = 0$, $h_{xy} = 0$ for $y \in N(x)$] and [$\forall x \in V$ with $H(x) > 0$ and $\forall y \in N(x)$, $h_{xy} = H(x) \wedge r_{yx} = h_{yx}$], that is, the system is in the legitimate configuration. Hence, the proof is completed. □

4. Concluding Remarks

In the above, we have shown that the proposed algorithm is indeed self-stabilizing in a distributed system whose underlying computational model assumes the read/write separate atomicity and in the legitimate configuration, all h_{xy}'s records the height $H(x)$ of $T(x)$. Arguing analogously as in the proof of Theorem 1, we can get that the h-value $h(x)$ defined in [5] and $H(x)$ defined in this paper are actually the same. Thus, by Theorem 4.4 in [5], as soon as the system reaches the legitimate configuration, identifying a center of T is to select a node x that satisfies $h_{xy} \geq h_{yx}$ for any $y \in N(x)$. Hence the center-finding problem is solved.

References

[1] S. Dolev, A. Israeli and S. Moran, Self-stabilization of dynamic systems assuming only read/write atomicity, *Distributed Computing* 7, 3-16, (1993).

[2] E. W. Dijkstra, Self-stabilizing systems in spite of distributed control, *Communications of the Association of the Computing Machinery* 17, 643-644, (1974).

[3] E. W. Dijkstra, Self-stabilization in spite of distributed control. In *Selected writings on computing: a personal perspective*, 41-46, Berlin-Heidelberg-New York: Springer-Verlag, (1982).

[4] E. W. Dijkstra, A belated proof of self-stabilization, *Distributed Computing 1*, 5-6, (1986).

[5] S. C. Bruell, S. Ghosh, M. H. Karaata and S. V. Pemmaraju, Self-stabilizing algorithms for finding centers and medians of trees, *SIAM Journal on Computing* 29 (2), 600-614, (1999).

[6] Tetz C. Huang ,Ji-Cherng Lin and H. J. Chen, A self-stabilizing algorithm which finds a 2-center of a tree, *Computers and Mathematics with Applications* 40, 607-624, (2000).

[7] F. Buckley and F. Harary, *Distance in Graphs*, Addison-Wesley, Redwood City, CA, (1990).

[8] S. Dolev, Self-stabilization, *MIT Press,* (2000).

Department of Computer Engineering and Science, Yuan-Ze University, 135 Yuan-Tung Rd., Chung-Li Taoyuan 32026, Taiwan

E-mail address: cstetz@saturn.yzu.edu.tw, csjclin@saturn.yzu.edu.tw, nathan24@ms57.hinet.net