A Self-Stabilizing Algorithm for the Center-finding Problem Assuming Read/Write Atomicity

Ji-Cherng Lin Tetz C. Huang Nathan Mou

Department of Computer Science and Engineering, Yuan Ze Uinversity

No. 135 Yuan-Tung Road, Chungli, Taoyuan, Taiwan, R.O.C.

csjclin@saturn.yzu.edu.tw cstetz@saturn.yzu.edu.tw nathan24@ms57.hinet.net

Abstract— The problem of locating centers of graphs has a variety of applications in the areas of transportation and communication in distributed systems. In this paper, we design and prove the correctness of a self-stabilizing algorithm which finds the center(s) for a distributed system with a tree topology. The computational model employed in this paper was introduced by Dolev et al. in [6] that assumes the read/write separate atomicity.

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

I. Introduction

E. W. Dijkstra first introduced the notion of self-stabilization in a distributed system in his pioneering paper [3] (cf. also [4][5]) in 1974, in which he coined the phrase and showed the feasibility of designing such algorithms in a distributed system. According to him, a distributed system is self-stabilizing if regardless of any initial system configuration, the system can automatically adjust itself to eventually converge to a legitimate configuration (or configurations) and then stay in legitimate configuration thereafter unless it incurs a subsequent transient fault. In the self-stabilizing system of Dijkstra type, communications among neighboring processors are carried out by use of *shared registers* (hereafter *registers*), namely, each processor in the system is allowed to write values into its own registers and read those values stored in the registers owned by its neighbors. The *interleaving model* is used to reason about the behavior of the system. In this model, it is assumed that, at each given time, only a single processor is activated by a scheduler, the so-called *central daemon*, to make a *move*. In other words, the behavior of the system can be described by an *execution sequence* $E = (C_0, m_1, C_1, m_2, C_2, \ldots)$ in which $\forall i \geq 0$, C_i represents a *system configuration* (or, simply, *configuration*) and m_i stands for a move and C_{i+1} is obtained from C_i after a unique processor in the system makes the move m_{i+1}. In the self-stabilizing system of Dijkstra type, since the computational model assumes the *read/write composite atomicity*, a single move (or *atomic*

step) by a processor consists of reading registers of all its neighbors, making internal computations and then rewriting its own register (or registers). In addition to Dijkstra's classic papers [3][4][5], a good reference for the basics on this type of self-stabilizing system can be found in Bruell et al. [1]. Later in 1993, Dolev et al., introduced a new type of self-stabilizing system in their famous paper [6]. 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 and either a single read operation or a single write operation. In this setting, Dolev et al. presented two simple self-stabilizing algorithms in [6], 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 [1][8]. 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 in which the computational model is 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. In Section 3, an example illustrates the execution of the algorithm. The correctness proof of the algorithm is given in Section 4. Finally, in Section 5, some remarks conclude the whole discussion.

II. 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 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 below states a well-known property regrading the center(s) of a tree. The proof of it can be found in Theorem 2.1 in [2].

Proposition 1: A tree has a unique center or two adjacent centers (cf. Figures 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 $H(x)$.

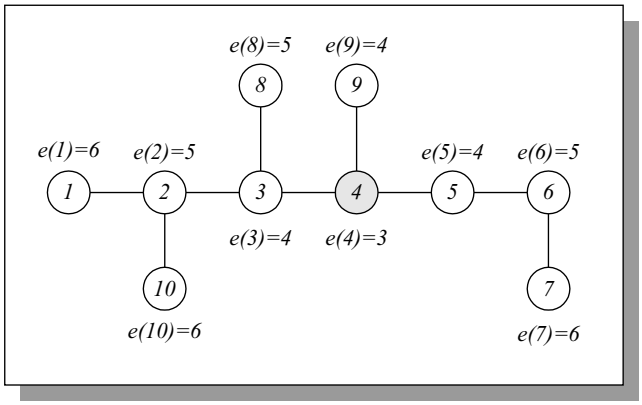

Fig. 1. Eccentricities of nodes in a tree. The shaded node stands for the unique center of the tree.

Definition 2: 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.

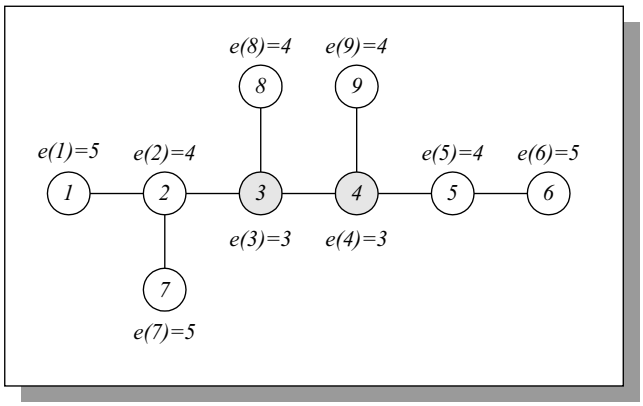

Fig. 2. Eccentricities of nodes in a tree. The two shaded nodes stand for the two centers of the tree.

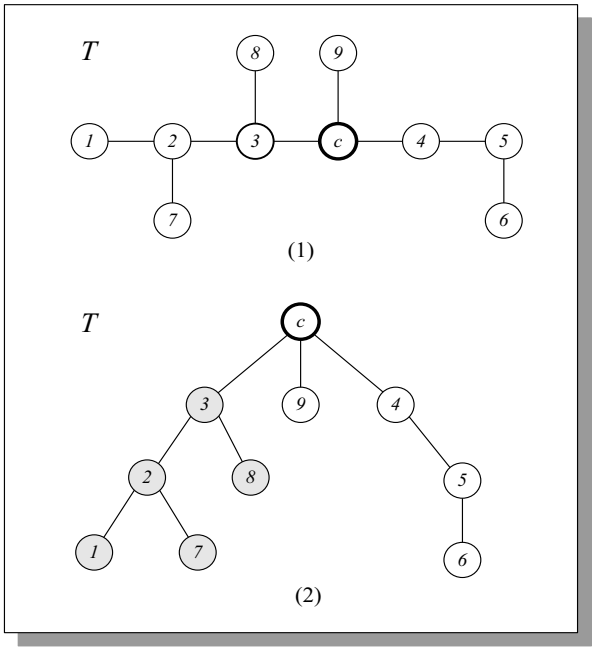

Fig. 3. A tree with a unique center c and the induced rooted tree rooted at c.

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

Proof: The proof is quite easy and is thus omitted. ■

Lemma 4: 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 $H(y_1) = H(y_2) = H(c) - 1$].

Proof: The first part of the claim in the lemma can be easily seen. For the second part, let L be a longest simple path in T. By Lemma 1 in [8], the length of L is even and c is the midpoint of L. Let $L = (x_j, x_{j-1}, \ldots, x_1, c, x'_1, \ldots, x'_{j-1}, x'_j)$. Then one can easily check that $H(x_1) = H(x'_1) = H(c) - 1$. ■

Lemma 5: 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 $H(y_0) = H(c_1) - 1$].

Proof: One can easily see that $\forall y \in N(c_1) - \{c_2\}$, $H(y) \leq H(c_1) - 1$. Let L be a longest simple path in T. By Lemma 1 in [8], the length of L is odd and c_1 and c_2 are the two midpoints of L. Let T_1 and T_2 be as defined in Case 2 of Definition 1. Let $L = (x_j, x_{j-1}, \ldots, x_1, x_0 = c_1, c_2 = x'_0, x'_1, \ldots, x'_{j-1}, x'_j)$. One can check that $H(c_1) = H(c_2) = j$ and $H(x_1) = H(x'_1) = j - 1$. Hence the lemma is proved. ■

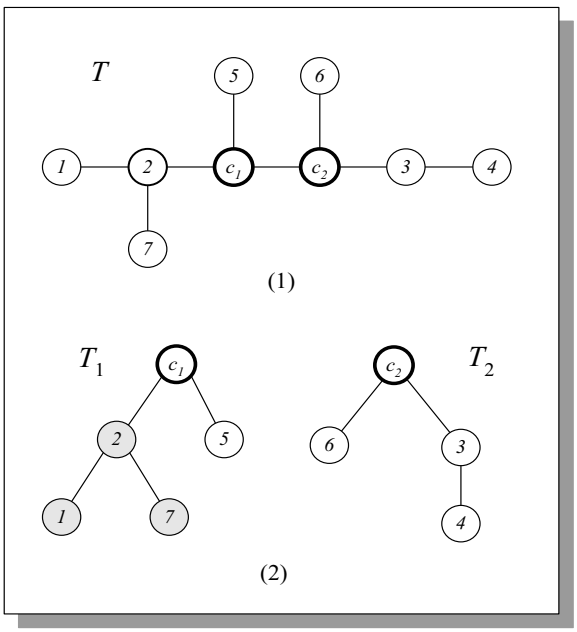

Fig. 4. A tree with two centers c_1 and c_2 and the induced rooted trees rooted at c_1 and c_2.

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 al. in [6] (cf. also [7]) 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 $N = \{0, 1, 2, \ldots\}$. $N_{x,r} = \{r_{yx} \mid y \in N(x)\}$ 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 6 (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 $T(x)$.

Proof: Let the legitimate configuration be fixed. Then in the legitimate configuration, $\forall x \in V$, [$\deg(x) =$

$1 \wedge h_{xy} = 0$ for the unique $y \in N(x)$] or [$\deg(x) > 1 \wedge (\forall y \in N(x),\ r_{yx} = h_{yx} \wedge\ h_{xy}\ = 1 + \max N^-_{x,r})$]. Also, let l be the diameter of T and $m = \lfloor \frac{l}{2} \rfloor$.

Claim. $\forall j \in \{0, 1, \ldots, m\}$, [$\forall x \in V$ with $0 \leq H(x) \leq j$ and $\forall y \in N(x)$, $h_{xy} = H(x)$] and [$\forall x \in V$ with $H(x) > j$ and $\forall y \in N(x)$, $h_{xy} \geq j$].

Proof of the Claim. We prove the claim by induction on j. For $j = 0$, it is obvious that if $H(x) = 0$, then x is a leaf node (i.e., $\deg(x) = 1$) and hence $h_{xy} = 0$ for the unique $y \in N(x)$. If $H(x) > 0$, then obviously $h_{xy} \geq 0$ for any $y \in N(x)$. Hence the claim is true for $j = 0$. Assume that for $j = k$ with $0 \leq k < m$, the claim is true, that is, [$\forall x \in V$ with $0 \leq H(x) \leq k$ and $\forall y \in N(x)$, $h_{xy} = H(x)$] and [$\forall x \in V$ with $H(x) > k$ and $\forall y \in N(x)$, $h_{xy} \geq k$]. Let $x \in V$ with $H(x) = k + 1$. By Lemmas 1, 2 and 3, we have that [$\exists y_0 \in N(x)$ such that $H(y_0) \geq k$] and [$\exists y_1 \in N(x) - \{y_0\}$ such that $H(y_1) = k$] and [$\forall y \in N(x) - \{y_0, y_1\}$, $H(y) \leq k$]. Thus, by induction hypothesis, we get $h_{y_0 x} \geq k$, $h_{y_1 x} = k$ and $h_{yx} \leq k$ for any $y \in N(x) - \{y_0, y_1\}$. Since in any legitimate configuration, $N_{x,r} = \{r_{yx} \mid y \in N(x)\} = \{h_{yx} \mid y \in N(x)\}$, we have that $\max N^-_{x,r} = k$ and hence $h_{xy} = 1 + \max N^-_{x,r} = 1 + k$ for any $y \in N(x)$. Thus, we have shown ($\sharp$): $\forall x \in V$ with $H(x) = k+1$ and $\forall y \in N(x)$, $h_{xy} = H(x)$. Arguing analogously as above, we can get ($\sharp\sharp$): $\forall x \in V$ with $H(x) > k+1$ and $\forall y \in N(x)$, $h_{xy} \geq k+1$. Then, the induction hypothesis, together with ($\sharp$) and ($\sharp\sharp$), implies that the above claim is true for $j = k + 1$. Hence the claim is proved.

Setting $j = m$ in the claim, Theorem 1 trivially follows. ■

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 7 (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.

Proof: In the configuration, $\forall x \in V$ with $\deg(x) = 1$, since $H(x) = 0$, we have that $h_{xy}\ = H(x) = 0$ for the unique $y \in N(x)$. Also in the configuration, for any $x \in V$ with $\deg(x) > 1$, let $H(x) = i$. Then by Lemmas 1, 2 and 3, we have that [$\exists y_0 \in N(x)$ such that $H(y_0) \geq i - 1$] and [$\exists y_1 \in N(x) - \{y_0\}$ such that $H(y_1) = i - 1$] and [$\forall y \in N(x) - \{y_0, y_1\}$, $H(y) \leq i - 1$]. Hence $N_{x,r} = \{r_{yx} \mid y \in N(x)\} = \{h_{yx} \mid y \in N(x)\} = \{H(y) \mid y \in N(x)\}$ and thus, $\max N^-_{x,r} = i - 1$. Therefore, $\forall y \in N(x)$, $h_{xy} = H(x) = i = 1 + (i - 1) = 1 + \max N^-_{x,r}$. From above, we see that the configuration 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.

III. An Illustration

Figure 5 illustrates the distributed system that assumes the read/write separate atomicity and is equipped with the proposed algorithm. An execution of the algorithm in the system is given in Table 1. In each configuration shown in Table 1, the shaded part indicates the execution of a single atomic step (or, a move) by the unique processor selected by the central daemon. Note that the system reaches the legitimate configuration at Configuration 43.

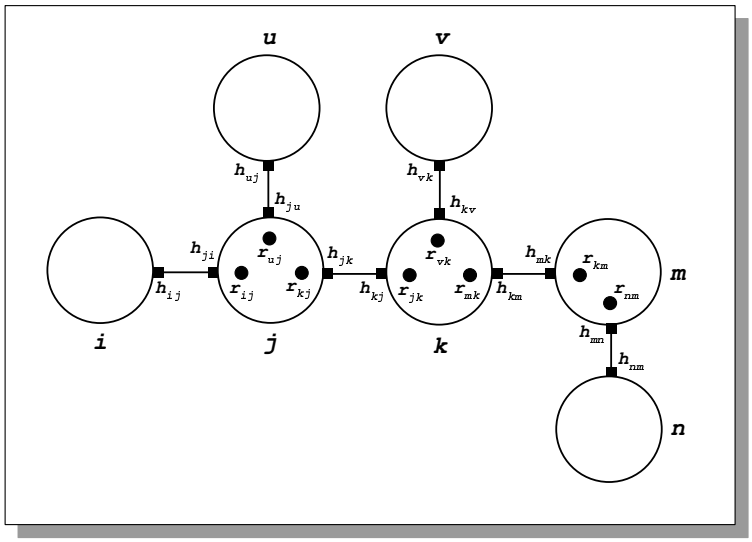

Fig. 5. The structure of a system that assumes the read/write separate atomicity and is equipped with the proposed algorithm.

IV. Correctness Proof

We now give the correctness proof in the following theorem. To facilitate the presentation in the following proof, we define a terminology. 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 above algorithm. Note that due to the content of the above algorithm and the way the algorithm is executed by processors in the system (as mentioned earlier), the moves designated by the central demon in any execution of the algorithm has to obey certain restrictions. For instance, in any execution of the algorithm, each non-leaf processor makes read action infinitely often. For another instance, in any execution, if after having completed a full round of reading all its neighbors, a non-leaf processor finds itself able to make a write action (i.e., it finds that the guard condition of statement

06 in the algorithm evaluates to true), then the very write action has to follow as the next move by the processor. In other words, if we view an execution of the algorithm in the system as an infinite sequence $(C_0, m_1, C_1, m_2, \ldots)$ in which $\forall i$, the configuration C_{i+1} is obtained from the configuration C_i after a unique processor in the system makes the move m_{i+1}, then the sequence cannot be arbitrary and without any restriction. To prove that the algorithm is self-stabilizing, we are required to show that for any such execution sequence $(C_0, m_1, C_1, m_2, \ldots)$ which obeys the restrictions induced from the content of the algorithm and the way the algorithm is executed, there exists a natural number p such that $\forall i \geq p$, C_i is the legitimate configuration. 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. To illustrate how to use these notations, for instance, if a processor x executes a write action "*write* $(h_{xy} = 1 + \max N^-_{x,r})$" at a time instant t, then, since $h_{xy}(t^+) \neq h_{xy}(t^-)$, $h_{xy}(t)$ is undefined. On the other hand, if a processor x executes a read action "*read* $(r_{yx} := h_{yx})$" at t, then either $r_{yx}(t^+) = r_{yx}(t^-)$ or $r_{yx}(t^+) \neq r_{yx}(t^-)$ is possible. In the former case, $r_{yx}(t)$ is defined whereas in the latter case, $r_{yx}(t)$ is undefined. Note also that due to the difference in computational model, the proof to be given in the following comes with quite different flavor from those proofs in [1][8] which study self-stabilizing systems that use the computational model of Dijkstra type. Moreover, some parts in the following proof may seem unnecessarily complicated, but they are really indispensable for the rigor of the proof, as we have checked very carefully. Also be advised that although we have abused some notations in the following proof, for the sake of simplicity in presentation, we have done so with great care lest it should cause any confusion to the readers.

Theorem 8 (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 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 $H(x) = 0$ (i.e., $\deg(x) = 1$), there exists a $t_0(x) > 0$ such that $\forall t > t_0(x)$, $h_{xy}(t) = 0$ for $y \in N(x)$. Let $t_0 = \max_{H(x)=0} t_0(x)$. 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 $\forall y \in N(x)$, $h_{xy}(t^+) \geq 0$]. 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.

Subclaim 2. If $x \in V$ with $H(x) = k+1$ and $t_1(x)$ is as in Subclaim 1, then $\forall y \in N(x)$, $\exists \bar{t}_y > t_1(x)$ such that $\forall t > \bar{t}_y$, $h_{xy}(t) = H(x)$.

Proof of Subclaim 2. Let $t'(x) > t_1(x)$ 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 $h_{xy}(t'(x)) = 1 + \max N^-_{x,r}(t'(x))$ (otherwise, x will execute statement 06 in the algorithm to change the value of h_{xy} after $t'(x)$, a contradiction.) Hence, $h_{xy}(t'(x)) = 1 + k$, by Subclaim 1. Therefore, we have (♯): $\forall t > t_1(x)$, $h_{xy}(t) = h_{xy}(t'(x)) = 1 + k = H(x)$. (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 (♯♯): $\forall t > \bar{t}_y$, $h_{xy}(t) = 1 + k = H(x)$. From (♯) and (♯♯) above, Subclaim 2 follows.

By letting $t^*(x) = \max\limits_{y \in N(x)} \bar{t}_y$, we have that $\forall t > t^*(x)$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$. Then, by letting $t^* = \max\limits_{H(x)=k+1} t^*(x)$, we have that $t^* > t_k$ and $\forall t > t^*$, [$\forall x \in V$ with $H(x) = k+1$ and $\forall y \in N(x)$, $h_{xy} = H(x)$]. Thus, we have obtained

Subclaim 3. $\exists t^* > t_k$ such that $\forall t > t^*$, [$\forall x \in V$ with $H(x) = k+1$ and $\forall y \in N(x)$, $h_{xy}(t) = H(x)$.

Then, arguing analogously as from Subclaim 1 till Subclaim 3, we can also prove

Subclaim 4. $\exists \tilde{t} > t_k$ such that $\forall t > \tilde{t}$, [$\forall x \in V$ with $H(x) > k+1$ and $\forall y \in N(x)$, $h_{xy}(t^+) \geq k+1$].

Finally, by letting $t_{k+1} = \max\{t^*, \tilde{t}\}$, we have that $t_{k+1} > t_k$ and $\forall t > t_{k+1}$, [$\forall x \in V$ 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 $\deg(x) > 1$ and $\forall y \in N(x)$, $r_{yx} = h_{yx}$. Therefore, after t_m^*, [$\forall x \in V$ and $\forall y \in N(x)$, $h_{xy} = H(x)$] and [$\forall x \in V$ with $\deg(x) > 1$ and $\forall y \in N(x)$, $r_{yx} = h_{yx}$], that is, the system is in the legitimate configuration. Hence, the proof is completed. ■

V. 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 [1] and $H(x)$ defined in this paper are actually the same. Thus, by Theorem 4.4 in [1], 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. 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).

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

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

[4] 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).

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

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

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

[8] 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).

Configuration	i	j						k						m				n	u	v
number	h_{ij}	r_{ij}	r_{uj}	r_{kj}	h_{ji}	h_{ju}	h_{jk}	r_{jk}	r_{vk}	r_{mk}	h_{kj}	h_{kv}	h_{km}	r_{km}	r_{nm}	h_{mk}	h_{mn}	h_{nm}	h_{uj}	h_{vk}
0	3	0	3	2	1	4	5	2	1	4	3	0	2	3	5	6	2	1	4	7
1	3	3	3	2	1	4	5	2	1	4	3	0	2	3	5	6	2	1	4	7
2	0	3	3	2	1	4	5	2	1	4	3	0	2	3	5	6	2	1	4	7
3	0	3	3	2	1	4	5	2	1	4	3	0	2	3	5	6	2	1	0	7
4	0	3	0	2	1	4	5	2	1	4	3	0	2	3	5	6	2	1	0	7
5	0	3	0	3	1	4	5	2	1	4	3	0	2	3	5	6	2	1	0	7
6	0	3	0	3	1	4	5	5	1	4	3	0	2	3	5	6	2	1	0	7
7	0	3	0	3	1	4	5	5	7	4	3	0	2	3	5	6	2	1	0	7
8	0	3	0	3	1	4	5	5	7	4	3	0	2	3	5	6	2	1	0	0
9	0	3	0	3	1	4	5	5	7	6	3	0	2	3	5	6	2	1	0	0
10	0	3	0	3	1	4	5	5	7	6	3	0	2	2	5	6	2	1	0	0
11	0	3	0	3	1	4	5	5	7	6	7	0	2	2	5	6	2	1	0	0
12	0	3	0	3	1	4	5	5	7	6	7	0	2	2	5	6	2	0	0	0
13	0	3	0	3	1	4	5	5	7	6	7	0	2	2	0	6	2	0	0	0
14	0	3	0	3	4	4	5	5	7	6	7	0	2	2	0	6	2	0	0	0
15	0	3	0	3	4	4	5	5	7	6	7	0	2	2	0	6	2	0	0	0
16	0	3	0	3	4	4	5	5	7	6	7	7	2	2	0	6	2	0	0	0
17	0	3	0	3	4	4	4	5	7	6	7	7	2	2	0	6	2	0	0	0
18	0	3	0	3	4	4	4	5	7	6	7	7	7	2	0	6	2	0	0	0
19	0	3	0	3	4	4	4	5	7	6	7	7	7	2	0	1	2	0	0	0
20	0	0	0	3	4	4	4	5	7	6	7	7	7	2	0	1	2	0	0	0
21	0	0	0	3	4	4	4	5	7	6	7	7	7	2	0	1	2	0	0	0
22	0	0	0	3	4	4	4	4	7	6	7	7	7	2	0	1	2	0	0	0
23	0	0	0	3	4	4	4	4	7	6	7	7	7	2	0	1	1	0	0	0
24	0	0	0	3	4	4	4	4	0	6	7	7	7	2	0	1	1	0	0	0
25	0	0	0	7	4	4	4	4	0	6	7	7	7	2	0	1	1	0	0	0
26	0	0	0	7	4	4	4	4	0	1	7	7	7	2	0	1	1	0	0	0
27	0	0	0	7	4	4	4	4	0	1	7	7	7	7	0	1	1	0	0	0
28	0	0	0	7	4	4	4	4	0	1	7	7	7	7	0	1	1	0	0	0
29	0	0	0	7	1	4	4	4	0	1	7	7	7	7	0	1	1	0	0	0
30	0	0	0	7	1	1	4	4	0	1	7	7	7	7	0	1	1	0	0	0
31	0	0	0	7	1	1	4	4	0	1	2	7	7	7	0	1	1	0	0	0
32	0	0	0	7	1	1	4	4	0	1	2	7	7	7	0	1	1	0	0	0
33	0	0	0	7	1	1	4	4	0	1	2	2	7	7	0	1	1	0	0	0
34	0	0	0	7	1	1	1	4	0	1	2	2	7	7	0	1	1	0	0	0
35	0	0	0	7	1	1	1	4	0	1	2	2	2	7	0	1	1	0	0	0
36	0	0	0	7	1	1	1	4	0	1	2	2	2	7	0	1	1	0	0	0
37	0	0	0	7	1	1	1	4	0	1	2	2	2	2	0	1	1	0	0	0
38	0	0	0	7	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
39	0	0	0	7	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
40	0	0	0	7	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
41	0	0	0	7	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
42	0	0	0	7	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
43	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
44	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
45	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
46	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
47	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
48	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
49	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
50	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
51	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0
52	0	0	0	2	1	1	1	1	0	1	2	2	2	2	0	1	1	0	0	0

Figure 6: An example which illustrates the execution of the algorithm