Distributed Self-Stabilizing Algorithm for Minimum Spanning Tree Construction

Gheorghe Antonoiu[1] and Pradip K. Srimani[1]

Department of Computer Science, Colorado State University, Ft. Collins, CO 80523

Abstract. Minimal Spanning Tree (MST) problem in an arbitrary undirected graph is an important problem in graph theory and has extensive applications. Numerous algorithms are available to compute an MST. Our purpose here is to propose a *self-stabilizing* distributed algorithm for the MST problem and to prove its correctness. The algorithm utilizes an interesting result of [MP88]. We show the correctness of the proposed algorithm by using a new technique involving induction.

1 Introduction

Self-stabilization is a relatively new way of looking at system fault tolerance, especially it provides a "built-in-safeguard" against "transient failures" that might corrupt the data in a distributed system. The concept of self-stabilization was first introduced in [Dij74] and the possibility of using this concept for designing fault tolerant algorithms was first explored in [Lam84].

Recently there has been a spurt of research in designing self-stabilizing distributed graph algorithms for many applications [SS92,BGW89,FD92,ADG92]; a good survey of self-stabilizing algorithms can be found in [Sch93]. One of the most fundamental structures that is very essential in many distributed applications is the *minimum spanning tree* (MST) of a given undirected connected edge-weighted graph. Most of the communication issues in any distributed system including broadcasting, packet routing, resource allocation, deadlock resolution etc. involve maintaining a minimal spanning tree of the underlying symmetric graph of the system. Although there exist a number of self-stabilizing algorithms for the spanning tree problem [CYH91,HC92,SS92,AS95,Agg94], none of those algorithms deals with constructing a MST. Our purpose in this paper is to propose a self-stabilizing distributed algorithm for the MST problem in a symmetric graph and to prove its correctness using induction in an interesting way.

2 Minimal Spanning Tree (MST) of a Graph

Remark 1. If the weights $\{w_{ij}\}$ of a graph are unique (distinct), the graph has a unique MST [HS84].

To design a self-stabilizing algorithm for the MST of a graph, we introduce a new characterization of any path in a given graph.

Definition 2. α-cost of any path from node i to j is defined to be the maximum of the weights of the edges belonging to the path. Ψ_{ij} is defined to be the minimum among the α-cost of all possible paths between the nodes i and j.

Remark 3. We call the path, along which Ψ_{ij} is defined, to be the minimum-α path between nodes i and j; this should not be confused with the traditional shortest path between nodes i and j. The shortest path is defined to be the path of minimum length where the length of a path is the sum of the weights of the edges on the path. Most significant difference between the two metrics, α-cost and length, of a path, assuming nonzero positive edge weights, is that when a path is augmented by an additional edge, length must increase while α-cost may remain constant.

Theorem 4. *Consider a graph G with unique edge weights. An edge e_{ij} is in the unique MST if and only if $\Psi_{ij} = w_{ij}$ [MP88].*

Proof. The proof is by contradiction; for details, see [MP88].

We use Remark 1 and Theorem 4 to develop our algorithm for MST construction. For convenience of description and understanding we first develop a self-stabilizing algorithm for minimum α-cost path to a given reference node r in the graph and then generalize the result to solve the MST problem.

2.1 Minimum α-cost path to a Given Node r

Each node attempts to compute the α-cost of the shortest path (minimum α-cost path) to a given reference node. Call this special node r. Ψ_{ir} denotes the α-cost of the shortest path from node i to node r. Note that for all i, Ψ_{ir} is determined by the topology of the graph and the weights assigned to the edges. Note that $\Psi_{rr} = 0$ (no self-loops). We use the following notations:

- C: An integer constant such that $C \geq n$.
- $\mathcal{N}(x)$: The set of neighbors of node x.
- $L(i)$: The level of node i, the current estimate of the number of edges on the minimum α-cost path.
- $D(i)$: The current estimate of Ψ_{ir} as known at node i.

Thus each node i maintains two data structures $L(i)$ and $D(i)$ and they determine the *local state* of node i. We assume that $0 \leq L(i) \leq C$; we do not need to consider level values beyond that (even after perturbation), as we can always assume each processor is capable of doing a modulo $(C+1)$ operation and always keeps the remainder as its level value. The variable $D(i)$ assume an an arbitrary value between 0 and some large positive number which we shall call MAX (determined by the length of the registers holding these variables).

Definition 5. For any arbitrary node x, the ordered pair, $S(x) = (D(x), L(x))$ defines the local state of the node x at any given point of time. The vector of all the node states define the global state of the system.

We introduce a total ordering relation between any two arbitrary local states.

Definition 6. Given two local states $S = (D, L)$ and $S' = (D', L')$, S is less than S' or $S < S'$, iff $(D < D') \vee ((D = D') \wedge (L < L'))$, i.e., state tuples are lexicographically ordered.

Definition 7. In any system state, for any arbitrary node x, we define $\mathcal{N}_C(x) = \{y | y \in \mathcal{N}(x), L(y) < C\}$, to be the set of its neighbors with level value $< C$.

Definition 8. In any system state, for any arbitrary node x, $\mathcal{N}_C(x) \neq \emptyset$, we define the following: (1) $\delta_{min}(x) = \min_{y \in \mathcal{N}_C(x)} \{\max\{w_{xy}, D(y)\}\}$; (2) $\Delta_{min}(x) = \{y | (y \in \mathcal{N}_C(x)) \wedge (\max\{w_{xy}, D(y)\} = \delta_{min}(x)\}$; (3) $L_{min}(x) = \min\{L(y) | y \in \Delta_{min}(x)\}$.

We make the following immediate observations:

(a) If the set $\mathcal{N}_C(x)$ for any node x is empty, all neighbors of node x has a level equal to C. The parameters $\delta_{min}(x)$, $\Delta_{min}(x)$ and $L_{min}(x)$ are undefined indicating that the estimates at each neighbor of node x is wrong.
(b) $\delta_{min}(x)$ of any node x is a refined estimate of Ψ_{xr} based on the estimates at the neighbors of node x. $\delta_{min}(x)$ is defined when $\mathcal{N}_C(x) \neq \emptyset$.
(c) The set $\Delta_{min}(x)$ denotes the neighbors y of node x such that $\max\{w_{xy}, D(y)\} = \delta_{min}(x)$. The set $\Delta_{min}(x)$ is defined and nonempty when $\mathcal{N}_C(x) \neq \emptyset$.
(d) $L_{min}(x)$ indicates the minimum of the level values of the nodes in the set $\Delta_{min}(x)$. The parameter $L_{min}(x)$ is defined when $\mathcal{N}_C(x) \neq \emptyset$.

Our objective is to design an algorithm to compute the minimum α-cost of each node to the reference node r, i.e., when the algorithm stabilizes, we will have $D(x) = \Psi_{xr}$ at each node x. Each node x looks at its own state $S(x)$ (the pair $(D(x), L(x))$) and the states of its neighbors and takes action by changing its own level and cost estimate. Our algorithm has a single rule for all the nodes in the graph (actually, the reference node take different action than all other nodes). The rule at node x is as follows:

$$
(\mathbf{R}) \quad \begin{cases} \textbf{if } (x = r) \wedge (L(x) \neq 0 \vee D(x) \neq 0) \textbf{ then } L(x) = 0 \ \ \& \ \ D(x) = 0; \\ \textbf{else if } (\mathcal{N}_C(x) = \emptyset) \wedge (D(x) \neq MAX \vee L(x) \neq C) \\ \qquad\qquad \textbf{then } D(x) = MAX \ \ \& \ \ L(x) = C \\ \quad \textbf{else if } (L(x) \neq L_{min}(x) + 1) \vee (D(x) \neq \delta_{min}(x)) \\ \qquad\qquad \textbf{then } L(x) = L_{min}(x) + 1, \ \ \& \ \ D(x) = \delta_{min}(x); \end{cases}
$$

Remark 9. The reference node r is **privileged** if $D(r) \neq 0$ or $L(r) \neq 0$. The reference node may be privileged in an illegitimate state, but once it takes an action, it becomes un privileged and can never be privileged again.

Remark 10. Any other node x, with $\mathcal{N}_C(x) = \emptyset$ is **privileged** if $(D(x) \neq MAX \vee L(x) \neq C)$; any node x, with $\mathcal{N}_C(x) \neq \emptyset$ is **privileged** if $L(x) \neq L_{min}(x) + 1 \vee D(x) \neq \delta_{min}(x)$. Note that any node x, $x \neq r$, is privileged and takes action, it becomes un privileged, but can be privileged again later (only after at least one move by one of its neighbors).

Remark 11. Given any arbitrary initial system state, the number of all possible distinct local states that any node can have subsequently is finite (L values can range over $0 \cdots C-1$ and the D values can range over the edge weights and the initial D values at the nodes). Thus, the number of all possible global system states is also finite.

Definition 12. Any global system state, when no node is privileged, is called a legitimate state; any other state is illegitimate.

Remark 13. In a legitimate state, $L(r) = D(r) = 0$.

Lemma 14. *In a legitimate state, any node x, $x \neq r$, with $L(x) < C$ has $\mathcal{N}_C(x) \neq \emptyset$ and has at least one neighbor y such that $L(y) = L(x) - 1$.*

Proof. For any un privileged node x, with $L(x) < C$, we have $L(x) = L_{min}(x)+1$ and since $L(x) < C$, we get $L_{min}(x) < C \Rightarrow \mathcal{N}_C(x) \neq \emptyset$. We also have that $L_{min}(x) = L(x) - 1$ and since $L(x) < C$, there exists at least one neighbor y of node x such that $L(y) = L(x) - 1$.

Lemma 15. *In a legitimate state, when no node is privileged, for any arbitrary node x, $L(x) < C$.*

Proof. In a legitimate state, the reference node r has $L(r) = 0$. Assume that a node x has $L(x) = C$; since x is un privileged, $\mathcal{N}_C(x) = \emptyset$. Consider the subset of nodes in graph G with level C. This subset forms a subgraph G' of G. Since G is connected and $r \notin G'$, there must be at least one node $y \in G'$ such that $\mathcal{N}_C(y) \neq \emptyset$ and since this y is unprivileged, there exists a node z such that $L(z) = L(y) - 1 = C - 1$. Then, by repeated application of the Lemma 14, there must be at least one node each with level values $C-1, C-2, \cdots, 0$. This is a contradiction since $C \geq n$ where n is the number of nodes in the graph.

Corollary 16. *For some integer m, $m < C$, (m denotes the highest level of a node in a legitimate state), the set of nodes in the graph is given by $\bigcup_{0 \leq k \leq m} R(k)$, where $R(k)$ is the set of nodes with level k.*

Lemma 17. *In a legitimate state, (1) $R(0) = \{r\}$; (2) for each node $x \in R(k)$, $1 \leq k \leq m$, there exists a node $y \in R(k-1)$ such that $D(x) = \max\{D(y), w_{xy}\}$.*

Proof. (1) Clearly, $R(0)$ contains the reference node r since in a legitimate state $L(r) = 0$. Assume $R(0)$ contains another node x. Since x is not privileged and is not the reference node, $L(x) = L_{min}(x) + 1$ and since levels cannot be negative, $L(x) > 0$; thus, $R(0)$ cannot contain x.

(2) Since node x is not privileged, $L_{min}(x) = L(x) - 1$ and $D(x) = \delta_{min}(x)$, i.e., there exists a node y, such that $L(y) = L(x) - 1$ (thus, $y \in R(k-1)$) and $\delta_{min}(x) = \max\{D(y), w_{xy}\}$.

Theorem 18. *In a legitimate state, when no node is privileged, for any arbitrary node x, we have $D(x) = \Psi_{xr}$.*

Proof. Consider any path $r = y_0, y_1, \ldots, y_\ell = x$ from the reference node r to any arbitrary node x. The α-cost of this path is given by $w = \max\{w(y_i, y_{i+1}) | i = 0, \cdots, \ell - 1\}$. Also, since no node is privileged, $D(y_0) = 0$, and for all i, $i = 1, \cdots, \ell$, $D(y_i) = \delta_{min}(y_i) \leq \max\{D(y_{i-1}, w(y_{i-1}, y_i)\} \leq w$. Thus, we have proved that for any arbitrary node x, $D(x) \leq \Psi_{xr}$.

To prove $D(x) \geq \Psi_{xr}$, we use induction. Clearly the claim holds for the node r in $R(0)$. Assume the claim hold for nodes in $R(k)$. Consider any arbitrary node x in $R(k+1)$. By Lemma 17, there exists a node y in $R(k)$ such that $D(x) = \max\{D(y), w_{xy}\}$. Since $D(y) = \Psi_{yr}$ (i.e., there exists a path from node y to node r with α-cost $D(y)$), there is a path from node x to r with cost $D(x)$, i.e., $D(x) \geq \Psi_{xr}$.

Next, we need to prove that the system converges to a legitimate state after a finite number of moves starting from any arbitrary initial illegitimate state. We need some more definitions.

Definition 19. In any illegitimate state, a **forcing node** of any privileged node x ($x \neq r$), is defined to be

$$\begin{cases} \text{node } x & \text{if } \mathcal{N}_C(x) = \emptyset \\ \text{a node } y | y \in \Delta_{min}(x) \wedge L(y) = L_{min}(x) & \text{otherwise} \end{cases}$$

Remark 20. The reference node r, when it is privileged, does not have any forcing node. Also, for any other node x, the forcing node may not be unique, i.e., the set $\{y | y \in \Delta_{min}(x) \wedge L(y) = L_{min}(x)\}$ may have more than one node. But, the new state of a node after the move is the same irrespective of the choice of the forcing node.

Lemma 21. *When a privileged node x takes action, the new state of node x is greater than the state of its forcing node (in the previous system state).*

Proof. If $\mathcal{N}_C(x) = \emptyset$ and x is privileged, node x is its own forcing node, $S(x) < (MAX, C)$ and the new state after the move $S'(x) = (MAX, C)$ and hence $S'(x) > S(x)$. If $\mathcal{N}_C(x) \neq \emptyset$, the forcing node $y \in \mathcal{N}_C(x)$ has $L(y) < C$ ($y \in L_{min}(x)$ and after the move, $D'(x) \geq \max\{w_{xy}, D(y)\} \geq D(y)$ and $L'(x) = L(y) + 1 > L(y)$; hence, $S'(x) > S(y)$.

Let A be a subset of the node set V of the graph not including the reference node r. The following definitions are based on such a set A.

Definition 22. For any given A, the set of nodes in A that have an edge to some node in $V - A$ is called the **border set** of A and is denoted by B_A.

Remark 23. For a given graph and a given set A, the set B_A is always non null since $r \notin A$ and the graph is connected.

Definition 24. For a given A, and a system state, the minimum value of the local states $S(x)$ for all $x \in A$ is called the **minimum value** of A and is denoted by $Min(A)$.

Remark 25. The quantity $Min(A)$ is an ordered pair of estimate values and levels (just like local states of nodes) and hence can be compared by the total ordering of Definition 6. Also, note that $Min(A)$ is a function of the given set A and a given global system state.

Lemma 26. *For a given A and a given global system state with its $Min(A) = c$, $Min(A)$ can decrease at a subsequent system state only after a node $x \in B_A$ makes a move with a forcing node in $\{V - A\}$ such that after the move $S(x) < c$.*

Proof. Since no node in $\{A - B_A\}$ has any neighbor outside of A and since the new state of a node making a move is greater than its forcing node (Lemma 21), to lower the value of $Min(A)$, a node $x \in B_A$ must make a move with a forcing node in $\{V - A\}$ such that after the move $S(x) < c$.

Our approach to prove the convergence of the algorithm is to prove that the assumption of an infinite sequence of moves leads to a contradiction. Let us consider one such infinite sequence of moves starting from a given illegitimate state without reaching the legitimate state. We can divide the set of nodes, V, in two subsets: A, the set of nodes each of which makes an infinite number of moves in the sequence and $\{V - A\}$, the set of nodes each of which makes finitely many moves in the sequence. The reference node r cannot belong to the set A since it can make at best only one move (see Remark 9). Starting from any illegitimate state, after a finite number of moves, all nodes not in set A will stop making moves (from the assumption). Let t_1 denotes this point in time. Let the minimum value of A at t_1 be $Min_1(A)$. The following lemmas are based on such an assumed infinite sequence, the set A and the time instant t_1.

Lemma 27. *Consider an arbitrary system state (after t_1) with $Min(A) = c$. If there exists a node $x \in B_A$ such that $S(x) = c$ and x is un privileged, then x can be privileged again in a subsequent system state only when $Min(A)$ becomes less than c.*

Proof. We need to consider two cases:

(1) $S(x) = (MAX, C)$; since x is the minimal node, each node in A has the state (MAX, C); no node in $A - B_A$ can be privileged; only a node $z \in B_A$ can be privileged and can make a move due to a forcing node in $\{V - A\}$ and after the move, $S(z) < (MAX, C)$.

(2) $S(x) < (MAX, C)$; since x is un privileged, $\mathcal{N}_C(x) \neq \emptyset$ and there exists a neighbor y of x such that $\max(w_{xy}, D(y)) = D(x)$ and $L(y) = L(x) - 1$. Since x is a minimal node in A, the node y is in $\{V - A\}$ and hence node y does not make a move. Since y does not make any move, by the construction of the algorithm (and the definitions of Δ_{min} and L_{min}), in order that node x be privileged again, another neighbor z of x must acquire a state $S'(z) < S(x)$ in a subsequent system state. Since nodes in $\{V - A\}$ do not make any move, $z \in A$ and hence $Min(A)$ is now less than c.

Lemma 28. *If in any system state (after t_1) the subset B_A does not contain any minimal node of A, then it will do so in finitely many moves.*

Proof. The value of $Min(A)$ can possibly be lowered only by a move of a node in B_A with a forcing node in $\{V - A\}$ (see Lemma 21). We now consider two cases:

(1) When a node in B_A makes a move with a forcing node in $\{V - A\}$ such that $Min(A)$ is lowered, the node (in B_A) making the move becomes the minimal node of A;

(2) otherwise, by assumption each node in A makes infinitely many moves. Let t_2 be the time when each node has made at least one move. If B_A does not still contain any minimal node, then $Min_2(A) > Min_1(A)$ by Lemma 21. Since the number of all possible local states is finite, repeating the argument the proof follows.

Theorem 29. *Starting from any illegitimate state, the system reaches the legitimate state in a finite number of moves, irrespective of the order in which the nodes make their moves and the number of nodes that move at any instant.*

Proof. Suppose otherwise. Since each node in A is to make infinitely many moves (the number of all possible local states is finite), and a node making a move becomes un privileged (until one of its neighbors makes a move; see Remark 10), in light of Lemmas 27 and 28, we must have a infinite sequence $Min_1(A) > Min_2(A) > \cdots >$, which is a contradiction.

Corollary 30. *In the sequence of state transitions from the initial global illegitimate state to the final global legitimate state, no illegitimate system state is repeated.*

Proof. The proof follows from the previous lemma. If it were possible to reach the same global illegitimate state in a finite number of moves, then it is possible that the same sequence of moves repeat indefinitely and the system never reaches a legitimate state in a finite number of moves.

2.2 The MST algorithm

We can now generalize the algorithm in the previous section to compute the minimum α-cost paths to all nodes and thereby compute the MST of the graph. Instead of the simple local variable $D(i)$, each node i now maintains a local array $D_i[1..n]$ and instead of the simple local variable $L(i)$, each node i now maintains a local array $L_i[1..n]$. The value of $D_i[j]$, for all $i, j \in V$, at any system state gives the cost of the minimum α-cost path from node i to j in that system state. Similarly, the value of $L_i[j]$ is the value of the level of node i with respect to the implicit tree rooted at node j. The contents of the arrays $D_i[\ ]$ and $L_i[\ ]$ denote the local state of the node i and the union of all local states defines the global system state. Ψ_{ij} denotes the cost of the minimum α-cost path from node i to node j for all i and j. Note that $\Psi_{ii} = 0$ for all i. Each node behaves as a special (reference) node when it attempts to compute the α-cost to itself; it unconditionally sets that value to 0. The data structure Ω_i at each node i keeps track of the MST edges incident on node i

We now present the self-stabilizing algorithm to compute the MST. Every node in the system has the same uniform rule. The rule at node i is as follows:

$$
(\mathbf{R}) \quad \begin{cases} \forall j = 1, \cdots, n \ \ \textbf{do} \\ \textbf{if } ((j = i) \wedge (D_i(j) \neq 0) \vee (L_i(j) \neq 0)) \textbf{ then } L_i(j) = 0 \ \ \& \ \ D_i(j) = 0; \\ \textbf{else if } ((j \neq i) \wedge (\mathcal{N}_C(i) = \emptyset) \wedge (D_i(j) \neq MAX \vee L_i(j) \neq C) \\ \qquad\qquad \textbf{then } D_i(j) = MAX \ \ \& \ \ L_i(j) = C \\ \quad \textbf{else if } ((j \neq i) \wedge ((L_i(j) \neq L_{min}(j) + 1) \vee (D_i(j) \neq \delta_{min}(j))) \\ \qquad\qquad \textbf{then } L_i(j) = L_{min}(j) + 1 \ \ \& \ \ D_i(j) = \delta_{min}(j) \ \ \& \\ \qquad\qquad \Omega_i = \{k | k \in \mathcal{N}(i) \wedge w_{ik} = D_i(k)\}; \end{cases}
$$

3 Conclusion

We have proposed a self stabilizing algorithm for MST computation in a arbitrary undirected graph; each edge of the graph is assigned an unique non zero weight. When the algorithm terminates (in finite time), each node knows which of its incident edges belong to the MST of the graph.

References

[ADG92] A. Arora, S. Dolev, and M. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters*, 1(1):11–18, 1992.

[Agg94] S. Aggrawal. Time optimal self-stabilizing spanning tree algorithms. Technical Report MIT/LCS/TR-632, Massachusetts Institute of Technology, May 1994.

[AS95] G. Antonoiu and P. K. Srimani. A self-stabilizing distributed algorithm to construct an arbitrary spanning tree of a connected graph. *Computers Mathematics and Applications*, 30(9):1–7, September 1995.

[BGW89] G. M. Brown, M. G. Gouda, and C. L. Wu. Token systems that self-stabilize. *IEEE Trans. Comput.*, 38(6):845–852, June 1989.

[CYH91] N. S. Chen, H. P. Yu, and S. T. Huang. A self-stabilizing algorithm for constructing spanning trees. *Inf. Processing Letters*, 39(3):14–151, 1991.

[Dij74] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, November 1974.

[FD92] M. Flatebo and A. K. Datta. Two-State self-Stabilizing algorithms. In *Proceedings of the IPPS-92*, California, June 1992.

[HC92] S.T. Huang and N.-S. Chen. A self-stabilizing algorithm for constructing breadth first trees. *Inf. Processing Letters*, 41:109–117, January 1992.

[HS84] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, 1984.

[Lam84] L. Lamport. Solved problems, unsolved problems, and non-problems in concurrency. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, pages 1–11, 1984.

[MP88] B. M. Maggs and S. A. Plotkin. Minimum-cost spanning tree as a path finding problem. *Information Processing Letters*, 26:291–293, January 1988.

[Sch93] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, March 1993.

[SS92] S. Sur and P. K. Srimani. A self-stabilizing distributed algorithm to construct BFS spanning tress of a symmetric graph. *Parallel Processing Letters*, 2(2,3):171–180, September 1992.