Proceedings of the 5th International Workshop on Distributed Computing (IWDC), 27-30 December 2003, LNCS 2918, pp. 26-32

A Synchronous Self-Stabilizing Minimal Domination Protocol in an Arbitrary Network Graph

Z. Xu, S. T. Hedetniemi, W. Goddard, and P. K. Srimani

Department of Computer Science
Clemson University
Clemson, South Carolina 29634 USA

Abstract. In this paper we propose a new self-stabilizing distributed algorithm for minimal domination protocol in an arbitrary network graph using the synchronous model; the proposed protocol is general in the sense that it can stabilize with every possible minimal dominating set of the graph.

1. Introduction

Most essential services for networked distributed systems (mobile or wired) involve maintaining a global predicate over the entire network (defined by some invariance relation on the global state of the network) by using local knowledge at each participating node. For example, a minimal spanning tree must be maintained to minimize latency and bandwidth requirements of multicast/broadcast messages or to implement echo-based distributed algorithms [8, 9, 1, 3]; a minimal dominating set must be maintained to optimize the number and the locations of the resource centers in a network [14]; an *(r,d)* configuration must be maintained in a network where various resources must be allocated but all nodes have a fixed capacity *r* [10]; a minimal coloring of the nodes must be maintained [15].

In this paper we propose a distributed algorithm to maintain a minimal dominating set in an arbitrary ad hoc network. Our algorithm is fault tolerant (reliable) in the sense that the algorithms can detect occasional link failures and/or new link creations in the network (e.g., due to mobility of the hosts) and can readjust the multi-cast tree. Our approach uses *self-stabilization* [5, 6] to design the fault-tolerant distributed algorithms.

The computation is performed in a distributed manner by using the mechanism of beacon messages. Mobile ad hoc networks use periodic beacon messages (also called ``keep alive'' messages) to inform their neighbors of their continued presence. A node presumes that a neighboring node has moved away unless it receives its beacon message at stipulated interval. This beacon message provides an inexpensive way of periodically exchanging additional information between neighboring nodes. In our algorithm, a node takes action after receiving beacon messages (along with algorithm

related information) from all the neighboring nodes. The most important contribution of the paper involves the analysis of the time complexity of the algorithms in terms of the number of *rounds* needed for the algorithm to stabilize after a topology change, where a round is defined as a period of time in which each node in the system receives beacon messages from all its neighbors. The beacon messages provide information about its neighbor nodes synchronously (at specific time intervals). Thus, the synchronous paradigm used here to analyze the complexity of the self-stabilizing algorithms in ad hoc networks is very different from the traditional paradigm of an adversarial oracle used in proving the convergence and correctness of self-stabilizing distributed algorithms in general. Similar paradigms have been used in [2, 16, 12, 11].

2. System Model

We make the following assumptions about the system. A link-layer protocol at each node *i* maintains the identities of its neighbors in some list *neighbors(i)*. This data link protocol also resolves any contention for the shared medium by supporting logical links between neighbors and ensures that a message sent over a correct (or functioning) logical link is correctly received by the node at the other end. The logical links between two neighboring nodes are assumed to be bounded and FIFO. The link-layer protocol informs the upper layer of any creation/deletion of logical links using the *neighbor discovery protocol* described below.

Each node periodically (at intervals of t_b) broadcasts a *beacon* message. This forms the basis of the *neighbor discovery protocol*. When node *i* receives the beacon signal from node *j* which is not in its neighbors list *neighbors(i),* it adds *j* to its neighbors list, thus establishing link *(i,j)*. For each link *(i,j)*, node *i* maintains a timer t_{ij} for each of its neighbors *j*. If node *i* does not receive a beacon signal from neighbor *j* in time t_b, it assumes that link *(i,j)* is no longer available and removes *j* from its neighbor set. Upon receiving a beacon signal from neighbor j, node i resets its appropriate timer.

When a node *j* sends a beacon message to any of its neighbors, say node *i*, it includes some additional information in the message that is used by node *i* to compute the cost of the link *(i,j)* as well as regarding the state of the node *j*, as used in the algorithm.

The topology of the ad-hoc network is modeled by a (undirected) graph $G = (V,E)$, where *V* is the set of nodes and *E* is the set of links between neighboring nodes. We assume that the links between two adjacent nodes are always bidirectional. Since the nodes are mobile, the network topology changes with time. We assume that no node leaves the system and no new node joins the system; we also assume that transient link failures are handled by the link-layer protocol by using time-outs, retransmissions, and per-hop acknowledgments. Thus, the network graph has always the same node set but different edge sets. Further, we assume that the network topology remains connected. These assumptions hold in mobile ad hoc networks in

which the movement of nodes is coordinated to ensure that the topology does not get disconnected. We also assume that each node is assigned a unique ID.

3. Minimal Dominating Set

Given an undirected graph G= (V, E), a *dominating set* S is defined to be a subset of vertices such that $\forall v \in V\text{-}S: \mathcal{N}(i) \cap S \neq \emptyset$ and a dominating set S is called *minimal* iff there does not exist another dominating set S′ such that $S' \subset S$. Note that $\mathcal{N}(i)$ and $\mathcal{M}[i]$ respectively represent the open and the closed neighborhoods of the node i. In this section we present a synchronous model, self-stabilizing protocol for finding a minimal dominating set. Figure 1 shows the pseudo-code of the protocol that is executed at each node i, where $1 \leq i \leq n$ (we assume nodes are numbered 1 through n). Each node i has two local variables: x(i), a Boolean flag (the value x(i)=0 indicates that $i \notin S$ while the value x(i) = 1indicates that $i \in S$), and a pointer variable P(i). Note that *P(i)=i* indicates that node i is currently not dominated; P(i)=null indicates that the node i is dominated at least twice, i.e., $|\mathcal{N}(i) \cap S| \geq 2$ and P(i)=j indicates that node i is dominated only by node j, i.e., $|\mathcal{N}(i) \cap S| = \{j\}$.

R1: ***if*** $(|\mathcal{N}[i] \cap S| = 1) \wedge (P(i) \notin \mathcal{N}[i] \cap S)$
  then $P(i) := j \in N[i] \cap S$
R2: ***if*** $(|\mathcal{N}[i] \cap S| = 0)$
  then if $P(i) \neq i$
    then $P(i) = i$
  else if $(i < \min(\{j \mid j \in \mathcal{N}(i), P(j)=j\}))$
    then $x(i):=1, P(i):=i$
  else $P(i):=i$
R3: ***if*** $(|\mathcal{N}[i] \cap S| > 1) \wedge (x(i)=0) \wedge (P(i) \neq null)$
  then $P(i):= null$
R4: ***if*** $(|\mathcal{N}[i] \cap S| > 1) \wedge (x(i)=1) \wedge (\forall j \in \mathcal{N}(i)\text{-}S, P(j)=null)$
  then $x(i):=0$, *and* $P(i) = \begin{cases} S \cap \mathcal{N}(i), \text{ if } |S \cap \mathcal{N}(i)| = 1 \\ \text{null}, \quad \text{otherwise} \end{cases}$

Figure 1: Minimal Domination Protocol

4. Correctness Proof

Theorem: When the protocol stabilizes (terminates), the set $S = \{i \mid x(i) = 1\}$ gives a minimal dominating set of the network graph.

Proof: (a) Assume that the set S is not dominating when the protocol has terminated. Then, $\exists i \in V$, $S \cap \mathcal{N}[i] = \phi$. Then, we have x(i)=0 and P(i)=i (by R2). Also, $\exists j \in S \cap \mathcal{N}(i)$, $P(j)=j \wedge j<i$. If $|S \cap \mathcal{N}[j]| = 1$, then by R1, node j has to move in the next round, a contradiction. If $|S \cap \mathcal{N}[j]| > 1$, then by R3, j will move, again a contradiction. Hence, we have $|S \cap \mathcal{N}[j]| = 0$. Since node j cannot make a move, then $\exists k \in S \cap \mathcal{N}(j)$, $P(k)=k \wedge k<j$. Repeating the argument and noting there is only finitely many nodes, we reach a vertex v, where R2 will apply. This is contradiction to the hypothesis that the protocol is terminated. Therefore, S is dominating.

(b) Assume that S is dominating, but not minimal when the protocol terminates. Then $\exists i \in S$, such that $S' = S-\{i\}$ is a dominating set. Therefore, $\forall j \in \mathcal{N}[i]$, $\exists k \in S-\{i\}$, $k \in \mathcal{N}[j]$. If x(j)=0, then by R1 and R3, P(j) is either k or null. So R4 must apply on node i, a contradiction. Thus, S is minimal dominating set. ❐

5. Convergence and Time Complexity

Lemma 1: If x(i) changes from 0 to 1 in a round, then any node $j \in N(i)$ cannot change its x(j) value in the same round

Proof: If x(j)=0, then by R2, only the smaller node between i and j is able to move; If x(j)=1, then by R2, node i can't move. ❐

Lemma 2: If a node i changes its x-value from 0 to 1, then x(i) will not change again.

Proof: If x(i) changes from 0 to 1, then by R2, all nodes in the neighborhood $\mathcal{N}(i)$ should have x(j)=0. And by Lemma 1, they will stay at x(j)=0 after mode i moves. These neighbor nodes have at least one node i in the neighborhood that is in S (i.e., x(i)=1), so they won't go into S unless node i goes out. But by R4, i will never go out of S unless it is adjacent to some nodes in S. ❐

Theorem 2: The protocol will terminate in at most 4n rounds starting from any arbitrary illegitimate state.

Proof: By lemma 2, each node will change its change x value at most twice. Therefore, there can be at most 2n changes of x values on all nodes in all the time. In our synchronous model, there can be at most 2n steps which contain changes of x value. Note that if there is no change in x value of any node in a round, then the move involves only changes in pointer values. Since the change in any pointer value is determined only by x values, there can be at most one step which does not contain changes of x value. Therefore, the upper bound of execution time in synchronized model is $4n + 1$ rounds. ❐

Example: Consider a network graph of 7 nodes in Figure 2; each node is numbered from 1 through 7. We use shaded circle to represent node in S (i.e. x-value equals 1), and un-shaded circle to represent node not in S (i.e. x-value equals 0). The arrows on the edge represent the pointers from one node to another. If an arrow (i, j) is drawn, then P(i)=j. If a node i is pointing to it self, then P(i)=i. A zigzag arrow represents a null pointer.

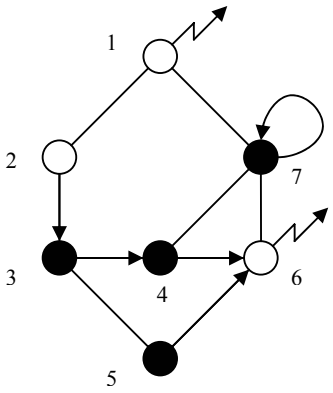

Figure 2(a): An arbitrary initial state: nodes 1, 4, 5, and 7 are privileged to move.

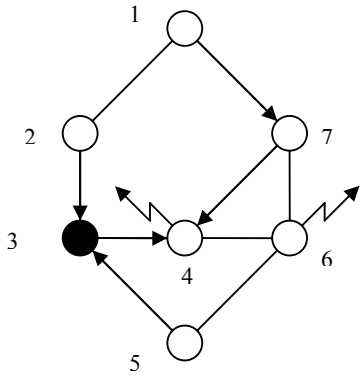

Figure 2(b): The system state after one round; nodes 1, 3, 4, 6 and 7 privileged

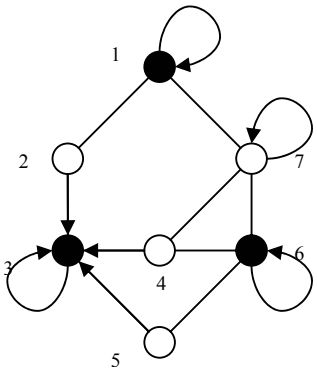

Figure 2(c): system state after 2 rounds; nodes 1, 3, and 4 are privileged.

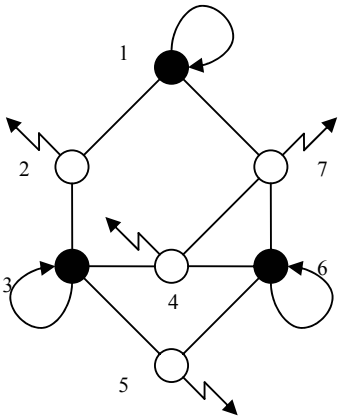

Figure 2(d): The system state after 3 Rounds; protocol terminated

6. Conclusion

We have proposed a self-stabilizing distributed algorithm for maintaining a minimal dominating set in a network graph. The algorithm (protocol) stabilizes in O(n) rounds in the synchronous model which can be used for ad hoc networks. The proposed protocol is general in the sense that the protocol can stabilize with every possible minimal dominating set of the graph. Consider an arbitrary minimal dominating set S of the graph G and then consider the following global system state: if $i \in S$, then x(i)=1, if $i \notin S$, then x(i)=0, if ($|S \cap N[i]| > 1$) then P(i) = null, and if ($|S \cap N[i]| = 1$), then $P(i)=j \in S \cap N[i]$. It is easy to see that the protocol is stable in this system state. The significance of this "completeness" of the protocol is that if the system is initialized to any minimal dominating set with the correct pointer settings, including minimal dominating sets that are not independent, then it will remain stable. While the protocol proposed in [17] can only stabilize with an independent set, the protocol proposed in this paper is capable of being stable with *any* minimal dominating set. The importance is that for some graphs no dominating set of smallest cardinality is independent. For example, consider the graph G formed by taking two stars $K_{\{1,n\}}$, and joining their centers by an edge. For this graph, the algorithm of [17] will stabilize with a set S having at least n+1 nodes, but the proposed algorithm can stabilize with a set S having the minimum cardinality of two (adjacent) nodes.

7. Acknowledgement

The work is supported by an NSF Award ANI-0218495.

7. References

[1] H. Abu-Amara, B.Coan, S.Dolev, A. Kanevsky, and J. L. Welch, "Self-stabilizing topology maintenance protocols for high-speed networks", *IEEE/ACM Transactions on Networking*, 4(6):902 –912, 1996.

[2] Y. Afek and S. Dolev, "Local stabilizer", In *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems*, pages 74 – 84, 1997.

[3] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*, McGraw Hill, 1998.

[4] J. Beauquier, A. K. Datta, M.Gradinariu, and F. Magniette, "Self-stabilizing local mutual exclusion and daemon refinement". In *DISC00 Distributed Computing 14th International Symposium, Springer LNCS:1914*, pages 223 – 237, 2000.

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

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

[7] S. Dolev. Self-Stabilization. MIT Press, 2000.

[8] S. Dolev, D. K. Pradhan, and J. L. Welch., "Modified tree structure for location management in mobile environments", *Computer Communications*, 19:335--345, 1996.

[9] S. Dolev and J. L. Welch, "Crash resilient communication in dynamic networks", *IEEE Transactions on Computers*, 46:14 – 26, 1997.

[10] S. Fujita, T. Kameda, and M. Yamashita, "A resource assignment problem on graphs", In *Proceedings of the 6th International Symposium on Algorithms and Computation*, pages 418 – 427, Cairns, Australia, December 1995.

[11] S. K. S Gupta, A.Bouabdallah, and P. K.Srimani., "Self-stabilizing protocol for shortest path tree for multi-cast routing in mobile networks", In *Euro-Par'00 Parallel Processing, Proceedings LNCS: 1900*, pages 600 – 604, 2000.

[12] S. K. S. Gupta and P.K. Srimani, "Using self-stabilization to design adaptive multicast protocol for mobile ad hoc networks", Proceedings of the *DIMACS Workshop on Mobile Networks and Computing*, pages 67 – 84, Rutgers University, NJ, 1999.

[13] S. Hsu and S.T. Huang, "A self-stabilizing algorithm for maximal matching", *Information Processing. Letters*, 43:77 – 81, 1992.

[14] T.W. Haynes, S.T.Hedetniemi, and P.J. Slater. *Fundamentals of Domination in Graphs,* Marcel Dekker, 1998.

[15] S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, "Fault tolerant distributed coloring algorithms that stabilize in linear time", Proceedings of the *IPDPS-2002 Workshop on Advances in Parallel and Distributed Computational Models*, pages 1 – 5, 2002.

[16] S. Shukla, D. Rosenkrantz, and S. Ravi, "Developing self-stabilizing coloring algorithms via systematic randomization", In *Proceedings of International. Workshop on Parallel Processing*, pages 668 – 673, 1994.

[17] 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", Proceedings of the Fifth IPDPS Workshop on Advances in Parallel and Distributed Computational Models, Nice, France, April 22-26, 2003