Fast Abstract: Self-Stabilizing Deterministic TDMA for Sensor Networks*

Mahesh Arumugam and Sandeep S. Kulkarni

Michigan State University, East Lansing MI 48824
Email: {arumugam,sandeep}@cse.msu.edu

Abstract. In this paper, we present a self-stabilizing, deterministic algorithm for TDMA in sensor networks where a sensor is aware of only its neighbors. We derive this algorithm by systematically reusing a distance 2 coloring algorithm.

Keywords: Self-stabilization, TDMA, Determinism, Sensor networks

1 Introduction

An algorithm for time division multiple access (TDMA) is desirable in sensor networks for energy management. TDMA assigns communication slots to each sensor such that a sensor can turn its radio off in the slots not assigned to itself and its neighbors. Moreover, TDMA guarantees reliable, collision-free communication. Also, TDMA is desirable in transforming existing distributed algorithms (in read/write model) into a computation model (write all with collision or WAC model) that is consistent with sensor networks [1]. Another important requirement is *self-stabilization* [2]. Existing solutions for self-stabilizing TDMA are either randomized [3, 4] or assume that the topology is fixed [5]. We are not aware of self-stabilizing deterministic TDMA algorithm for sensor networks. With this motivation, we present a self-stabilizing deterministic TDMA algorithm.

2 TDMA Algorithm

First, we obtain distance 2 vertex coloring for $G = (V, E)$, where V is the set of all sensors deployed and E is communication topology. Once distance 2 coloring is obtained, TDMA slots can be computed.

Distance 2 coloring. Each sensor maintains two public variables *color*, the color of the sensor, *nbrClr*, a vector containing $\langle id, c\rangle$ elements, where c is the color assigned to neighbor id. Also, each sensor maintains a private variable *dist2Clr*, which is a vector similar to *nbrClr* containing the colors assigned to the sensors at distance 2. Initially, the base station circulates a token for obtaining distance 2 coloring. Whenever a sensor (say, j) receives the token, j computes *used.j*, which denotes the colors used in its distance 2 neighborhood. If *nbrClr.j* (or *dist2Clr.j*) contains $\langle l, undefined\rangle$, l did not receive the token yet and, hence, *color.l* is not assigned. Sensor j chooses a color such that it does not conflict with the colors in *used.j*, reports its color to its distance 2 neighbors (using the primitive *report_distance_2_nbrs*), and forwards the token to one of its neighbors according to a token circulation algorithm. (We discuss the implementation of the primitive later in this section.)

Theorem 2.1 The algorithm satisfies the specification of distance 2 coloring. □

TDMA slots. The color of the sensor identifies the initial TDMA slot. If d is the maximum degree of the communication graph G, the TDMA period, $P = d^2 + 1$ suffices. Now, a sensor can start transmitting application messages in the TDMA slots when all the sensors in its distance 2 neighborhood are colored.

Theorem 2.2 The above TDMA algorithm guarantees collision-freedom □

Primitive: *report_distance_2_nbrs*. Whenever j decides its color, it updates *nbrClr* value of its distance 1 neighbors and *dist2Clr* value of its distance 2 neighbors. Sensor j sends a broadcast message with its color and a schedule for its distance 1 neighbors. The distance 1 neighbors of j update their *nbrClr* values. Based on the schedule in the message, each neighbor broadcasts their *nbrClr* vectors. If a distance 1 neighbor (say, l) of j is already colored, the schedule requires l to broadcast *nbrClr.l* in its TDMA slot. Otherwise, the schedule specifies the slot that l should

* This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF Equipment Grant EIA-0130724, and a grant from Michigan State University.

use such that it does not interfere with the slots already assigned to j's distance 2 neighborhood. If there exists a sensor k such that $distance_G(l,k) \leq 2$, then k will not transmit in its TDMA slots, as l is not yet colored. A sensor (say, m) updates *dist2Clr.m* with $\langle j, color.j \rangle$ iff $(m \neq j) \wedge (j \notin N.m)$. This schedule guarantees collision-free update of *color.j* at distance 2 neighborhood of j and requires at most $d+1$ update messages.

Adding stabilization. The token may be lost due to: (1) clock drift, (2) corruption of *nbrClr* values, and/or (3) corruption of token message. To deal with this problem, the base station initiates token circulation once every *token circulation period*, P_{tc} slots. The value of P_{tc} is chosen such that the time taken for circulation is at most P_{tc} (which depends on $|E|$ in G). When the base station initiates a token circulation, it sets a timeout. If the base station receives the token back within P_{tc} slots, it resets the timeout. Otherwise, the base station concludes that the token is lost in the network. To recover from this problem, the base station initiates a recovery by sending a *recovery token*. Similarly, whenever a sensor (say, $j \neq r$) forwards the token, it expects to receive the token in the subsequent round within P_{tc} slots. If j observes that the token is lost, it sets *nbrClr.j* and *dist2Clr.j* to undefined. Further, j stops transmitting in its current TDMA slots.

The base station waits until its distance 3 neighborhood stops transmitting before initiating recovery. This ensures that the primitive *report_distance_2_nbrs* can update the distance 2 neighbors successfully. Let T_{rt} be the time required for sensors in the distance 3 neighborhood of the base station to stop transmitting (which is bounded by P_{tc}). After T_{rt} time, the base station recomputes its color, reports its color to its distance 2 neighbors, and forwards the *recovery token*. When a sensor (say, j) receives the *recovery token*, it waits until the sensors in its distance 3 neighborhood have stopped transmitting. Then, j follows the above TDMA algorithm to recompute its color and TDMA slots. Thus, we have

Theorem 2.3 Starting from arbitrary initial states, the TDMA algorithm recovers to states from where collision-free communication among sensors is restored. □

Time complexity for recovery. Suppose $T_{rt} = P_{tc}$, i.e., the base station waits for one token circulation period before forwarding the *recovery token*. Now, when the base station forwards the *recovery token*, all the sensors in the network would have stopped transmitting. Further, whenever a sensor receives the token, it can report its color without waiting for additional time. To compute time for recovery, observe that it takes at most one token circulation for the base station to detect token loss, one token circulation for the sensors to stop and wait for recovery, and at most one token circulation for the network to resume normal operation. Thus, the time required for the network to self-stabilize is at most $2 * P_{tc}+$ time taken for token circulation, which is bounded by $3 * P_{tc}$.

3 Conclusion

In this paper, we presented a self-stabilizing deterministic TDMA algorithm for sensor networks. While we expect that the degree to be small, the number of colors required is at most d times the optimal, where d is the maximum degree in G.

References

1. S. S. Kulkarni and M. Arumugam. Transformations for write-all-with-collision model. *Conference on Principles of Distributed Systems (OPODIS)*, 2003.
2. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11), 1974.
3. T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. *Algorithmic Aspects of Wireless Sensor Networks*, 2004.
4. C. Busch, M. M-Ismail, F. Sivrikaya, and B. Yener. Contention-free MAC protocols for wireless sensor networks. *Conference on Distributed Computing (DISC)*, 2004.
5. S. S. Kulkarni and M. Arumugam. SS-TDMA: A self-stabilizing MAC for sensor networks. In *Sensor Network Operations*. IEEE Press, 2005, to appear.