This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Transmission Scheduling in Sensor Networks via Directed Edge Coloring

Maggie Cheng Li Yin
Department of Computer Science
University of Missouri
Rolla, MO 65401
{chengm,lyh38}@umr.edu

Abstract— **This paper presents a transmission scheduling scheme in sensor networks. Each node is assigned a list of time slots to use for unicast and broadcast communication. The algorithm employs edge coloring on a directed graph for transmission scheduling. It is different from previous works that use vertex coloring of a graph for node scheduling, or those that use edge coloring of undirected graphs for link scheduling. The proposed algorithm uses the least number of time slots compared to its counterparts and it avoids both the hidden terminal problem and the exposed terminal problem in both unicast and broadcast communication.**

Index Terms— **MAC, TDMA, sensor networks, edge coloring, vertex coloring**

I. Introduction

Wireless sensor networks have been envisioned to have wide applications in security surveillance, health care, and habitat monitoring. Sensor networks usually consist of a collection of small nodes powered by battery. Each of these sensor nodes has a limited transmission range and sensing range, but together they achieve a high level sensing and communication task. In such wireless sensor networks, multiple hop transmission is indispensable.

A MAC layer scheme plays an important role in improving the timeliness of data dissemination and data gathering as well as energy efficiency, which are all important performance measures of sensor networks. In the past, random schemes inherited from wireless ad hoc networks have been adopted for sensor networks. However, a random scheme could waste energy due to collisions. If CSMA/CA is used, it could waste energy because the radio keeps listening to the channel. Yet it still fails to provide a bounded access delay. By contrast, TDMA can easily determine when to listen to the channel and when to transmit, therefore it can completely avoid collisions, provide a bounded access delay, and achieve high energy efficiency.

Despite the overhead in setting up a fixed time slot assignment, TDMA is still a promising solution for sensor networks, because the time slot assignment is done once and used forever (or for a relatively longer period of time). TDMA schemes are suitable for networks that have a small or moderate number of nodes, and can be easily extended to large scale sensor networks by using a distributed solution that compromises optimality, or a clustered solution in which the task of computing a TDMA schedule is done by multiple cluster heads rather than by one central node, and the scale of scheduling is reduced to a small cluster rather than the entire network.

Transmission scheduling involves assigning time slots to nodes or links. To assign each node a unique slot certainly can avoid collisions, but it consumes too many slots; to assign each directional link a unique slot can also avoid collisions, but it has the same drawback of consuming too many slots. To efficiently use the time slots, vertex coloring of graphs for node scheduling or edge coloring for link scheduling is used. The purpose of computing a schedule via graph coloring is to reduce the number of slots used. The total number of slots used is an indication of access delay, because it determines the turn around time for each transmitter.

The objective of vertex coloring is to assign colors to vertices such that adjacent vertices get different colors and the total number of colors used in minimum. Different graph models have been proposed to avoid the hidden problem in wireless networks, but they cannot solve the exposed terminal problem. The inherent problem with the vertex coloring approach is that nodes that are connected by an edge cannot use the same color, which gives rise to the exposed terminal problem.

Edge coloring is to assign colors to edges such that edges incident on the same vertex must use different colors and the total number of colors used is minimum. Extensions from the classical edge coloring problem have been proposed, such as distance-k edge coloring, which requires edges at distance $< k$ use different colors, i.e., for two edges e and e' with the same color, the minimum path length between them is at least k. An invariant result from this approach is that edges that share a vertex cannot use the same color, no matter what value of k is used. This causes difficulty for broadcast traffic, because a node would need to send to each individual destination in different slots.

Vertex coloring is good for broadcast scheduling and edge coloring is good for unicast scheduling. In sensor networks, the dominant traffic includes both broadcast and unicast, and the two traffic modes are often interleaved. For example, the

The work is supported in part by National Science Foundation under grant CCF-0514940.

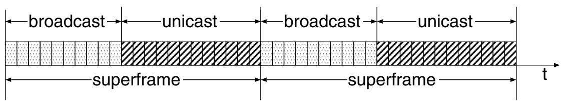

Fig. 1. A super frame that separates unicast traffic from broadcast traffic

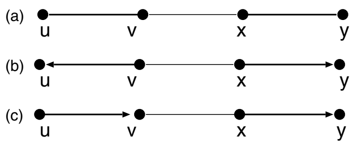

Fig. 2. Edge coloring on an undirected graph

base station needs to disseminate a query to all nodes through broadcast, and each sensor node needs to report its sensory data upstream to the base station through unicast. It is not feasible to keep two schedules and make nodes switch from one schedule to another immediately. Alternatively, to use a superframe that reserves a fixed number of slots for unicast and broadcast separately will end up using too many slots (Fig. 1).

Currently, all existing work on edge coloring is to assign colors to undirected edges on an undirected graph. Using this approach for transmission scheduling in sensor networks, each undirected edge will be mapped to two directional links, and each color will be mapped to two time slots. The direction of transmission is considered after edge coloring is completed, which often yields unsatisfactory result. For the example in Figure 2.(a), to assign edges (u,v) and (x,y) different colors can lead to the exposed terminal problem (Figure 2.(b)), which is manifested as using more slots than necessary in this context, but to assign them the same color can lead to the hidden terminal problem (Figure 2.(c)). The inherent problem of edge coloring on an undirected graph is that each edge is undirected therefore during edge coloring phase the direction of transmission is not considered; adding the directions later won't change the result of edge coloring.

The above observations motivate a different graph coloring scheme that is not confined by the vertex coloring and classical edge coloring and can meet the unique communication needs of sensor networks. In order to (1) use minimum slots in order to reduce the turn around time of each node, (2) accommodate both unicast and broadcast traffic, and (3) avoid both the hidden terminal problem and the exposed terminal problem in all traffic, neither vertex coloring nor edge coloring of an undirected graph is good enough.

We therefore propose to assign time slots to directional links via edge coloring of a directed graph. The number of colors needed is the same as the number of time slots needed. By contrast, in a two-phase approach that first computes a proper edge coloring of an undirected graph then generates 2 time slots for each color and maps time slots to directional links, the number of time slots needed is twice the number of colors needed. Since it is different from classical undirected edge coloring, new rules for time slot reuse are defined in this paper.

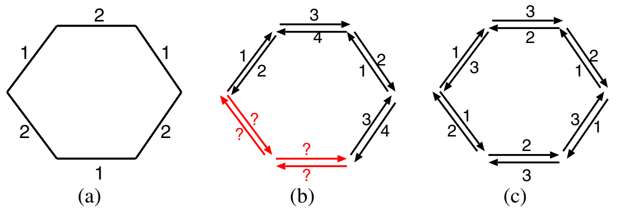

Fig. 3. (a) For a cyclic graph, edge coloring of the undirected graph results in an odd number of edges with the same color on a cycle; (b) Based on the result from (a), there is no proper edge orientation that produces an interference-free transmission schedule; (c) Using edge coloring on a directed graph can directly find a transmission schedule.

Edge coloring on a directed graph also successfully solves another problem that the previous approach can not solve—When there are an odd number of edges with the same color on a cycle, previous two-phase approach will fail to find an edge orientation. Fig. 3 shows the detail. The proposed algorithm Edge Coloring on Directed Graphs(ECDiG) is evaluated through extensive simulation. The results show that this algorithm uses less time slots than the previous approaches that use edge coloring of undirected graphs. By using ECDiG algorithm, nodes use a consistent schedule for both broadcast and unicast, and most importantly it completely avoids the hidden terminal problem and the exposed terminal problem. Finally, needless to say, it can handle transmission schedule for networks with unidirectional links where the transmission power of nodes at the two ends of a link is not the same, which is very possible in wireless networks.

The rest of the paper is organized as following. Section II surveys the most related work on transmission scheduling via edge coloring and vertex coloring. Section III presents the directed edge coloring algorithm. Section IV provides results compared with the OEC scheme from [1]. Finally, section V ends the paper with plans for future work.

II. Related Work

Channel assignment problem has a similar combinatorial structure to the graph coloring problem, i.e., to have transmitters that are sufficiently apart use the same channel. Spatial reuse can maximize channel utilization. One of the objectives of channel assignment is to maximize the number of users sharing the same channel simultaneously or to use as little as possible channel resource to support a certain group of users. Channel assignment could be time slots assignment (for TDMA), frequency assignment (for FDMA), or code assignment (for CDMA). In [2], a unified framework was introduced to identify atomic constraints underlying most assignment problems. Here we discusses the channel assignment problem in the context of TDMA. Typically, edge coloring is used for link scheduling and vertex coloring is used for node scheduling.

A. Edge Coloring Approach

Edge coloring of undirected graphs has long been used for link scheduling in wireless networks. This approach typically involves two phases: in the first phase, a proper edge coloring is computed to make sure that edges incident on the same vertex get different colors; in the second phase, each time slot is mapped to a unique link with a direction of transmission. If t colors are required for a proper edge coloring of the graph, $2t$ time slots are needed to schedule all unicast transmissions. [3], [1] belong to this category. Some variations of this approach use distance-k edge coloring in the first phase with $k > 1$. A popular one is distance-2 edge coloring [4], which requires no two edges of the same color be incident on the same vertex or neighbors.

Vizing's theorem states that a graph can be edge-colored in either δ or $\delta+1$ colors, where δ is the maximum degree of the graph. A constructive proof of the theorem can find a proper edge coloring of the graph [5], which by itself is a centralized edge coloring algorithm using at most $\delta + 1$ colors. In [3], a distributed version of the algorithm is proposed that colors a graph with at most $\delta+1$ colors. The link scheduling will need $2(\delta + 1)$ time slots correspondingly. But the algorithm in [3] requires even number of edges with the same color on a cycle. For cycles with an odd number of edges with the same color, the second phase will fail to find a valid orientation of edges.

To find the optimal edge coloring of an undirected graph is NP-complete [6]. Distributed edge coloring algorithms with a performance bound includes [7], [8]. The best result is obtained in [9] that generates a valid edge coloring using $(1 + \epsilon)\delta$ colors in $O(polylog(n))$ time. Herman et al. [1] expanded it to oriented edge coloring, which includes the edge coloring algorithm in [9] as the first step, and then re-colors edges randomly with probability p, and then assigns orientations to all edges. Compared with [3], Herman et al. improved the time complexity from linear to $O(polylog(n))$, but the problem with odd cycles remains, even though the re-coloring process in the second phase re-color some edges with a probability p, there is no guarantee that cycles with an odd number of edges of the same color will be changed.

B. Vertex Coloring Approach

Vertex coloring has been used for node scheduling in telecommunication systems[10], [11], [12], [13]. In TDMA systems, the optimization objective is to allow nodes to transmit without generating interference to each other while using a minimum number of time slots. Different classes of graphs have been used to model wireless networks. In containment graphs, two nodes must be assigned different time slots if one is in the transmission range of the other, thus only the primary interference is considered when the intended receiver is transmitting at the same time. However, this gives rise to the hidden terminal problem due to the secondary collision— when the intended receiver is in the transmission range of two active transmitters. Intersection graphs avoid the hidden terminal problem at the cost of more slots— In intersection graphs, two nodes must be assigned different time slots if their coverage area overlaps with each other regardless of the locations of the receivers. This model can be justified in cellular systems where channel assignment to base stations only needs to consider the coverage area of the base stations without considering the location of mobile users. In ad hoc networks where every node's location matters, the intersection graphs can not be used. Actually a better solution can be derived from the containment graphs. Suppose the containment graph is G, the proper vertex coloring of G^2 can avoid the hidden terminal problem with less time slots, where G^2 is a graph in which the vertex set is the same, and the edge set includes the original edge set and the edges that connect pairs of nodes within 2-hops. A proper vertex coloring of G^2 is also called distance-2 vertex coloring of G. In [14] a new graph model is proposed that extends the distance-2 vertex coloring to handle scenarios where the service area of a node is smaller than the interference area. In this case, a node s is able to interfere with another node t's reception even if t is not in s's service area and therefore can not be the intended receiver of s.

III. Edge Coloring of Directed Graphs

Instead of using a two-phase approach that first computes a proper edge coloring of an undirected graph, then generates $2t$ time slots for t colors and assigns time slots to directed edges, we propose to build the directed graph first and then assign colors to directed edges directly based on the following rules:

1) Edges that are conflicting with each other must use different colors;
2) Edges that are not conflicting with each other can use the same or different colors.

The following section describes the criteria for determining if conflict exists between two directional edges.

A. Conflict Check Criteria

The conflict check procedure should eliminate the hidden terminal problem and at the same time avoid the exposed terminal problem. This motivates a different definition of "proper edge coloring" of directed graphs from the classical edge coloring— In classical edge coloring, edges are undirected and two edges are considered conflicting with each other if they are incident on the same vertex.

If two edges satisfy one of the following two conditions, they are considered conflicting with each other (see Fig. 4.(a) and (d)).

1) Two edges share the same vertex and at least one of them is an in-edge of this vertex (Fig. 4.(a)) ; or
2) Two disjointed edges are conflicting if there is a third edge that shares the head with one edge and shares the tail with the other (Fig. 4.(d)).

Note that two edges are not conflicting if they only share the tail, as shown in Fig. 4.(c); and the disjointed edges (highlighted) in (b) are not conflicting because the transmission on one edge won't be heard by the destination of the other. In the following algorithm, procedure $conflict(c, e)$ will return TRUE if edge e cannot be colored in color c because there exists an edge f of color c that has a conflict with e.

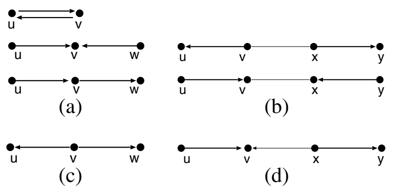

Fig. 4. Conflict check criteria

In addition, regardless of the direction of edges, we define distance-2 edges and distance-1 edges as follows: if two edges are incident on the same vertex, we call them distance-1 edges; if two disjointed edges are incident on neighbors, we call them distance-2 edges. Conflicting edges could be distance-2 edges or distance-1 edges as shown in Fig. 4.(a, d); non-conflicting edges could also be distance-2 edges or distance-1 edges as shown in Fig. 4.(b, c).

B. Algorithm Description

In the following edge coloring algorithm ECDiG, the directed graph $G(V, E)$ is the graph model of the network. A directed edge $(u, v) \in E$ represents the one way transmission from node u to node v.

```
ECDiG($G(V, E)$)
  let $S = E$
  initialize each edge $e \in S$ as unvisited and uncolored, set $tag(e) = \phi$
  **while** $S \neq \phi$ **do**
    randomly select an edge $r$ from $S$
    let $c$ be the smallest color not included in $tag(r)$
    visited($r$)=TRUE
    BuildTree($r$, $c$)
    TraverseTree($r$, $c$)
    **for** each edge $e$ in $S$ **do**
      visited($e$)=FALSE
    **end for**
  **end while**
END of ECDiG
```

```
BuildTree($r$, $c$)
  color($r$)=$c$, colored($r$)=TRUE, $S = S \setminus \{r\}$
  **for** each distance-2 edge $e$ of $r$ **do**
    **if** visited($e$)=FALSE **then**
      visited($e$)=TRUE
      **if** conflict($c$, $e$)=FALSE **then**
        add $e$ to the children list of $r$
      **else**
        $tag(e) = tag(e) \cup \{c\}$
      **end if**
    **end if**
  **end for**
  **for** each child $g$ of $r$ **do**
    BuildTree($g$, $c$)
  **end for**
END of BuildTree
```

```
TraverseTree($r$, $c$)
  **for** each distance-1 edge $e$ of $r$ **do**
    **if** visited($e$)=FALSE **then**
      visited($e$)=TRUE
      **if** conflict($c$, $e$)=FALSE **then**
        color($e$)=$c$, colored($e$)=TRUE, $S = S \setminus \{e\}$
      **else**
        $tag(e) = tag(e) \cup \{c\}$
      **end if**
    **end if**
  **end for**
  **for** each child $g$ of $r$ **do**
    TraverseTree($g$, $c$)
  **end for**
END of TraverseTree
```

The time complexity of ECDiG is $O(m\dot{\Delta})$, where $\dot{\Delta}$ is the maximum number of edges within any edge's distance-2 neighborhood, and m is the number of edges.

C. Walk-through Example

A walk-through example using ECDiG is shown in Fig. 5. ECDiG algorithm yields 5 colors, exactly the same as the optimal solution. The two-phase algorithms from [3] and [1] that use edge coloring on an undirected graph first and then map time slots $2i - 1$ and $2i$ to the pair of edges with color i won't find a solution for this example due to the reason stated in section I (Fig 3).

D. Remarks on Jointed Non-conflicting Edges

The result from the ECDiG algorithm is that each directed edge will receive a color (or a time slot). Each node will receive a collection of time slots that it can use for transmission. When multiple out-edges of a node v are assigned the same color, e.g., in Fig. 6, edges (v, b), (v, c) and (v, d) are assigned color #2, it means for unicast data, node v will use slot #2 from different cycles to transmit to each of them, because the data packet for each destination is different; but for broadcast data, node v will use slot #2 only once to reach node b, c, and d.

IV. Simulation Results

A. Transmission Scheduling in Sensor Networks

We applied the edge coloring algorithm ECDiG on sensor networks for active transmission scheduling. The first experiment is for link scheduling given unicast traffic. The sensor networks are randomly deployed in a square area of 1000×1000 with node transmission range 150. Each node randomly selects a neighbor as its destination, so the total number of active transmissions is equal to the total number of nodes. Each node has an out-degree 1 and an in-degree between 0 to Δ, where Δ is the maximum number of active transmissions toward a single destination. Experiments are done on sensor networks with $100 \sim 800$ nodes. For each size, 100 sensor networks are randomly generated and the results from the 100

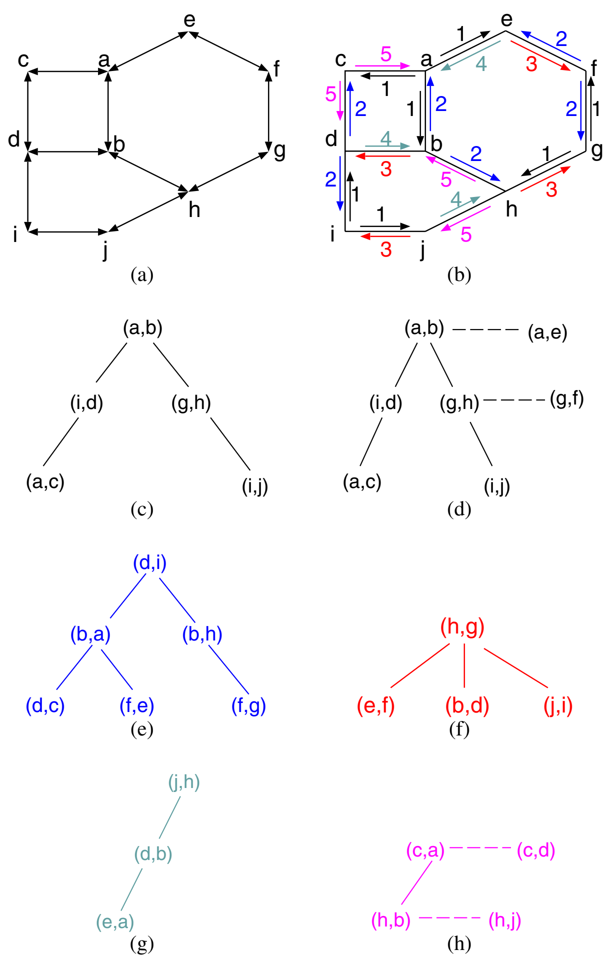

Fig. 5. Example Implementation of ECDiG. Distance-1 edges are connected by dashed lines. (a)given graph G; (b)final color assignment; (c)at the end of BuildTree for c=1; (d)at the end of TraverseTree for c=1;(e)monochromatic tree for c=2;(f)monochromatic tree for c=3; (g)monochromatic tree for c=4; (h)monochromatic tree for c=5.

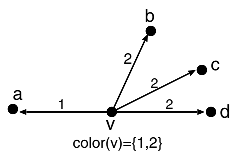

Fig. 6. Node v sends to node a in slot #1; sends to b, c and d in slot #2.

runs are averaged. The results shown in Fig. 7 indicate that the ECDiG algorithm uses much more time slots when node density increases even though the Δ remains almost the same, which is mainly due to the conflicts shown in Fig. 4.(d).

The second experiment is for transmission scheduling for both unicast and broadcast traffic. A tree [1] rooted at the base station is used for data dissemination and data gathering. Edges pointing toward the base station are used for data gathering, and edges pointing away from the base station are used for data dissemination. The scheduling algorithm needs to assign every directional edge a valid time slot. Sensor networks are randomly deployed as described above. In the following tables, h(c) is the height of the monochromatic tree in ECDiG, and d(c) is the diameter of the monochromatic component in OEC. G is the shortest path tree of the original disk graph that models the sensor network, D(G) is the diameter of G, Δ(G) is the maximum degree of G, and V(G) is the number of nodes in G. The numbers of time slots required by ECDiG and OEC algorithms are compared: in average, OEC uses 30% $\sim$ 63% more time slots than ECDiG. As to the running time, OEC is dependent on the diameter of the monochromatic component, which corresponds to the height of the monochromatic tree in ECDiG. A comparison on h(c) and d(c) indicates that OEC loses to ECDiG by 42% $\sim$ 74% .

[1]It could be a shortest path tree, or a diffusion tree, etc. We chose to use the shortest path tree in this experiment.

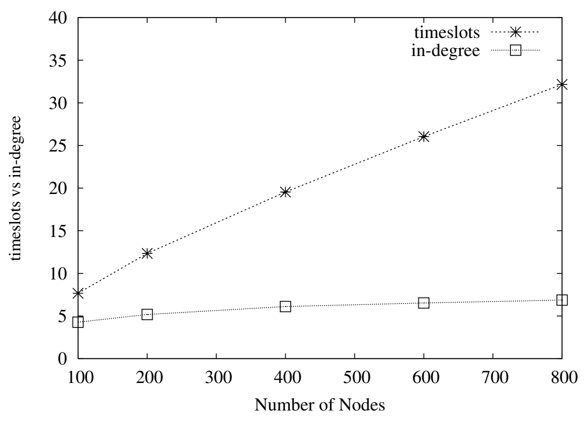

Fig. 7. Node Scheduling in Sensor Networks

Comparing the time slots used in this experiment and those in the first experiment, the results suggest that joint scheduling for unicast and broadcast saves time slots compared to the superframe approach that reserves a fixed number of slots for unicast traffic and a fixed number of slots for broadcast traffic, in which the total number of slots required is the sum of the two (Fig. 1).

TABLE I
SPANNING TREE

ECDiG					OEC	
slots	h(c)	D(G)	Δ(G)	V(G)	slots	d(c)
12.44	7.45	17.78	8.07	100	16.14	10.60
20.03	6.91	15.89	14.32	200	28.64	10.77
34.05	6.38	14.72	25.74	400	51.48	10.78
47.94	6.31	14.58	37.91	600	75.82	11.00
59.14	6.53	14.79	48.15	800	96.30	11.37

In addition, we also observed the list of time slots each node receives. For all network sizes, the average list size is 1.2 $\sim$ 1.4, and the standard deviation is 0.57 $\sim$ 0.63, which indicates that, compared to a vertex coloring approach that assigns each node exactly one slot, our approach does not require significantly more slots for broadcast traffic.

B. Graph Coloring of Tree Structures

In order to further compare the channel efficiency of ECDiG algorithm with OEC, we tried graph coloring on acyclic graphs. Since OEC does not guarantee solution on cyclic graphs, we can only use tree structures. We choose three types of trees—random trees, caterpillar trees and bush trees. These trees are generated using *Mathematica* in the same way as in [1].

We compare maximum height of the monochromatic trees in ECDiG and the maximum diameter of the monochromatic components in OEC, and the number of time slots needed for each algorithm. In ECDiG, the slot number is the color number; in OEC, the slot number is twice the color number. This comparison gives a fair evaluation of the two algorithms in terms of channel efficiency and running time. The results in tables II, III, and IV show that ECDiG uses less time slots and shorter running time than OEC does for every type of trees and in any size.

TABLE II
BUSH TREE

ECDiG					OEC	
slots	h(c)	D(G)	Δ(G)	V(G)	slots	d(c)
6.08	7.05	12.55	4.01	62.20	8.02	9.93
6.66	13.60	21.11	4.65	184.50	9.30	18.64
6.96	18.25	27.23	4.92	315.60	9.84	24.73
7.21	20.10	29.64	5.18	420.00	10.36	27.22
7.39	26.89	38.07	5.35	654.50	10.70	35.81
7.69	32.01	44.71	5.60	922.00	11.20	42.47
7.71	41.95	56.33	5.68	1406.00	11.36	54.04

TABLE III
CATERPILLAR TREE

ECDiG					OEC	
slots	h(c)	D(G)	Δ(G)	V(G)	slots	d(c)
8.07	12.68	21.00	6.01	100	12.02	15.74
9.15	23.38	35.00	7.02	200	14.04	28.53
10.13	41.51	58.00	8.00	400	16.00	48.08
10.19	64.61	87.00	8.02	600	16.04	74.18
11.15	78.51	101.00	9.02	800	18.04	88.47
11.11	99.96	126.00	9.02	1000	18.04	111.33
13.13	122.03	151.00	11.01	1500	22.02	133.97

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a new transmission scheduling algorithm in sensor networks. The algorithm uses edge coloring on a directed graph to compute the time slots that a node can use when transmitting to each of its neighbors and when broadcasting to all neighbors. The resulting assignment can guarantee there is no hidden terminal problem and exposed terminal problem any time in any communication modes and the total time slots used is the least so far, which implies low access delay (or turn around time) and high channel utilization.

TABLE IV
RANDOM TREE

ECDiG					OEC	
slots	h(c)	D(G)	Δ(G)	V(G)	slots	d(c)
7.26	12.16	27.64	5.18	100	10.36	15.48
7.67	18.45	42.71	5.60	200	11.20	24.66
8.11	27.94	62.60	6.01	400	12.02	36.84
8.35	33.78	78.11	6.30	600	12.60	45.88
8.37	38.05	87.95	6.28	800	12.56	52.04
8.65	43.81	102.81	6.56	1000	13.12	61.24
8.85	52.28	107.21	6.77	1500	13.54	73.12

In the future, we intend to design an efficient distributed version of ECDiG algorithm and develop it into a full-fledged MAC protocol that supports TDMA on top of a random access network.

REFERENCES

[1] T. Herman, S. Pemmaraju, and I. Pirwani, “Oriented edge colorings and link scheduling in sensor networks”, in *First International Conference on Communication System Software and Middleware, 2006 (Comsware 2006)*, 2006, pp. 1– 6.

[2] S. Ramanathan, “A unified framework and algorithm for channel assignment in wireless networks”, *Wireless Networks*, vol. 5, no. 2, pp. 81–94, 1999.

[3] S. Gandham, M. Dawande, and R. Prakash, “Link scheduling in sensor networks: distributed edge coloring revisited”, in *INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005, vol. 4, pp. 2492 – 2501.

[4] Mohammad Mahdian, “On the computational complexity of strong edge coloring”, *Discrete Appl. Math.*, vol. 118, no. 3, pp. 239–248, 2002.

[5] J. Misra and D. Gries, “A constructive proof of vizing’s theorem”, *Inf. Proc. Lett.*, vol. 41, pp. 131–133, 1992.

[6] Ian Holyer, “The np-completeness of edge-colouring”, *SIAM J. COMPUT*, vol. 10, no. 4, pp. 718–720, November 1981.

[7] Alessandro Panconesi and Aravind Srinivasan, “Improved distributed algorithms for coloring and network decomposition problems”, in *STOC ’92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, New York, NY, USA, 1992, pp. 581–592, ACM Press.

[8] A. Panconesi and A. Srinivasan, “Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds”, *SIAM Journal on Computing*, vol. 26, pp. 350–368, 1997.

[9] David A. Grable and Alessandro Panconesi, “Nearly optimal distributed edge colouring in o(log log n) rounds”, in *SODA ’97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, 1997, pp. 278–285, Society for Industrial and Applied Mathematics.

[10] R. Ramaswami and K.K. Parhi, “Distributed scheduling of broadcasts in a radio network”, in *INFOCOM 89*, April 23-27, 1989, vol. 2, pp. 497–504.

[11] Sven O. Krumke, Madhav V. Marathe, and S. S. Ravi, “Models and approximation algorithms for channel assignment in radio networks”, *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

[12] S. Ramanathan and Errol L. Lloyd, “Scheduling algorithms for multi-hop radio networks”, in *SIGCOMM ’92: Conference proceedings on Communications architectures & protocols*, New York, NY, USA, 1992, pp. 211–222, ACM Press.

[13] Arunabha Sen and Mark L. Huson, “A new model for scheduling packet radio networks”, *Wireless Networks*, vol. 3, no. 1, pp. 71–82, 1997.

[14] M. X. Cheng, S. C. Huang, X. Huang, and W. Wu, “New graph model for channel assignment in ad hoc wireless networks”, *IEE Proceedings – Communications*, vol. 152, no. 6, pp. 1039–1046, December 2005.