This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Compact Wakeup Scheduling in Wireless Sensor Networks

Junchao Ma and Wei Lou
Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong
{csjma, csweilou}@comp.polyu.edu.hk

***Abstract*—In a traditional wakeup scheduling, sensor nodes start up numerous times to communicate in a period, thus consuming extra energy due to state transitions (e.g. from the sleep state to the active state). In this paper, we address a novel wakeup scheduling problem called compact wakeup scheduling, in which a node needs to wake up only once to communicate bidirectionally with all its neighbors. However, not all communication graphs have valid compact wakeup schedulings, and thus we focus on tree and grid topologies that have valid compact wakeup schedulings. We propose polynomial-time algorithms using the optimum number of time slots in a period for tree and grid topologies.**

I. Introduction

Wakeup scheduling has been widely used in wireless sensor networks (WSNs), for it can reduce the energy wastage caused by the idle listening state. By virtue of it, sensor nodes could operate in a low duty cycle mode, and periodically start up to check the channel for activity.

The previous studies [1], [2] in the wakeup scheduling did not, however, consider all possible energy consumption, especially the energy consumed in the *state transitions*, for example, from the sleep state to the listening state or transmitting state. After such a scheduling, a node may start up numerous times in a period to communicate with its neighbors. The frequent state transitions not only consume extra time, but also consume extra energy. Moreover, the current surges in the state transitions reduces the battery capacity.

In [3], energy efficient centralized and distributed algorithms are proposed to reduce the frequency of state transitions of each node to twice in a data gathering tree: once for receiving data from its children, and once for sending data to its parent. If the network topology is a directed acyclic graph (DAG) where each node v_i has k_i parents, the scheduling in [3] would require v_i to wake up $k_i + 1$ times as the parent nodes are not scheduled together. Moreover, the *two-way* (or bidirectional) communication is not taken into consideration. An interesting problem is to design an efficient scheduling where a node could wake up *only once* and finish all communication tasks with its neighbors consecutively and bidirectionally.

In this paper, we propose *compact wakeup scheduling*, a novel time division multiple access (TDMA) approach to the wakeup scheduling problem, to minimize the frequency of state transitions. Compact wakeup scheduling assigns consecutive time slots to all the links incident to a node v_i so that v_i can start up only once to communicate bidirectionally with all its neighbors in one scheduling period T.

Apart from reducing the transient time and energy cost in the state transitions, compact wakeup scheduling also has other benefits. The network delay, which is a major concern in time-critical monitoring systems like [4], can be reduced. For instance, a sensor may need to wait until all its neighbors wake up so that it can collect the real time data from these neighbors to make the local computation on these data. Furthermore, compact wakeup scheduling can reduce the state transitions of other components in the nodes, such as external memory and sensing devices.

This work is supported in part by grants 1-ZV5N, PolyU 5236/06E and PolyU 5243/08E.

II. Problem Formulation

We assume that a WSN has n static sensor nodes equipped with single omni-directional antennas, and all the nodes have the same communication range. The network is represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and $E = \{e_1, e_2, \cdots, e_m\}$ denotes the set of edges referring to all the communication links.

In TDMA wakeup schedulings, each bidirectional communication link l_{ij} is assigned two time slots: one time slot is that v_i is a transmitter and v_j is a receiver, while the other one is that v_j is a transmitter and v_i is a receiver. In the two time slots, nodes v_i and v_j start up, and switch from the sleep state to the active state. After that, nodes v_i and v_j switch to the sleep state again. We can see that node v_i may start up $2w_i$ times to communicate bidirectionally with its neighbors in a scheduling period T in the worst case, where w_i is the number of neighbors of v_i. To minimize the frequency of state transitions, we propose a new scheduling approach called compact wakeup scheduling.

Definition 1. *Compact wakeup scheduling* is an interference-free wakeup scheduling to assign consecutive time slots to all the links incident to a node v_i, and then v_i needs to start up only once to communicate bidirectionally with all its neighbors. Such a scheduling is said to be *valid* if all the links incident to v_i are assigned consecutive time slots.

In the compact wakeup scheduling, the two time slots assigned to each bidirectional link l_{ij} are adjacent, and node v_i can finish its bidirectional communication with v_j in consecutive time slots. Fig. 1 (a) shows the given network topology. Fig. 1 (b) shows a wakeup scheduling, in which a node starts up numerous times in a period. Fig. 1 (c) shows a compact

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

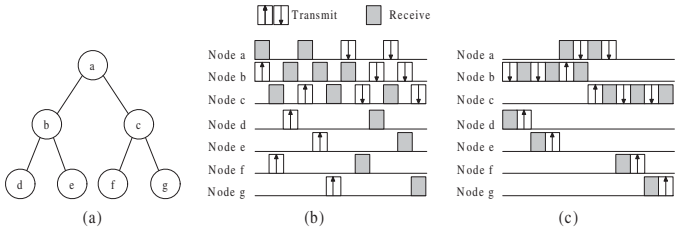


Fig. 1. Wakeup scheduling and compact wakeup scheduling: (a) network topology, (b) wakeup scheduling, (c) compact wakeup scheduling.

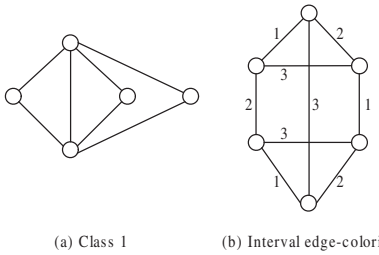


Fig. 2. Class 1 graphs without valid compact wakeup schedulings.

wakeup scheduling, in which a node could start up only once to communicate bidirectionally with its neighbors. Compact wakeup scheduling can reduce the time for a node to collect the data from its neighbors. As shown in Fig. 1 (b) and (c), node c needs to wake up five more times to communicate with all its neighbors without the compact wakeup scheduling.

An edge coloring of graph G is called a *valid coloring* if any two adjacent edges of G are assigned different colors. A valid coloring of G is called an *interval (or consecutive) edge-coloring* if, for each vertex v, the colors of edges incident to v form an integer interval.

Theorem 1. *A communication graph G with a valid compact wakeup scheduling has an interval edge-coloring, and belongs to Class 1 graphs, where the edge chromatic number is equal to the maximum degree* Δ *of graph G.*

Proof: If graph G has a valid compact wakeup scheduling, any node v_i in G can wake up once to communicate with all its neighbors. Each two-way communication link can be colored with one color, and then the links incident to one node are assigned consecutive colors. Thus, graph G has an interval edge-coloring. According to [5], graph with an interval edge-coloring belongs to Class 1 graphs. Therefore, graph G is a Class 1 graph. ■

Unfortunately, the converse proposition is not true. The graph in Fig. 2 (a) belongs to Class 1 graphs, but has no valid interval edge-coloring, and thus it has no valid compact wakeup schedulings. The Class 1 graphs even with valid interval edge-colorings may not have valid compact wakeup schedulings. For example, the graph in Fig. 2 (b) has an interval edge-coloring, but all valid interval edge-colorings could not avoid the hidden terminal problem. Thus, graphs with valid compact wakeup schedulings are a proper subset of graphs with valid interval edge-colorings, and also a proper subset of Class 1 graphs.

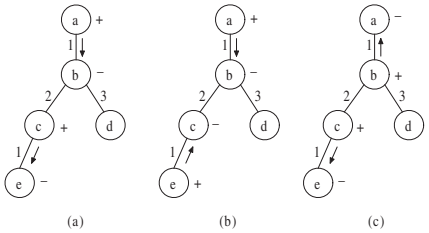


Fig. 3. Compact wakeup scheduling of a tree: (a) hidden terminal problem, (b) avoid the hidden terminal problem, (c) avoid the exposed terminal problem.

Since not all communication graphs have valid compact wakeup schedulings, we will focus on particular graphs, such as tree and grid topologies. Interestingly and surprisingly, we can obtain polynomial-time algorithms using the optimum number of time slots in a period. By minimizing the number of time slots, the overall network throughput can be maximized.

III. Compact Wakeup Scheduling Algorithms

In this section, we propose polynomial-time algorithms to produce valid compact wakeup schedulings for tree and grid topologies, which are commonly used in WSNs.

A. Trees

To obtain a valid compact wakeup scheduling of a tree, we first obtain an interval edge-coloring of a tree, then try to assign time slot to each edge and ensure interference-free. If graph G is a tree of degree Δ, we could get an interval edge-coloring of G using color $1, \cdots, \Delta$. In the coloring process (lines 1 to 7 in Algorithm 1), when coloring a new uncolored edge, the consecutiveness of edge-coloring remains invariant, and the edges already colored form a consecutively colored subgraph. After all edges are colored, we could get an interval edge-coloring and the total number of colors assigned is Δ.

We now describe how the interval edge-coloring is used to assign time slots to each edge. The idea is to map each color to two consecutive time slots and assign a valid direction of transmission to avoid interferences. In Fig. 3, link l_{ab} and l_{ce} are assigned the same color "1" in the interval edge-coloring, while time slot ts_1 and ts_2 are allocated for color "1". If time slot ts_1 is assigned in the directions of transmission as shown in Fig. 3 (a), the hidden terminal problem would happen because the reception at node v_b is garbled due to the collision of transmission from nodes v_a and v_c. Alternatively, if time slot ts_1 is assigned in the directions of transmission as shown in Fig. 3 (b), the hidden terminal problem could be avoided. Similarly, time slot ts_2 is assigned in the reverse directions of transmission as shown in Fig. 3 (c). Inspired by this, we should determine the directions of transmission along each link carefully to avoid the hidden terminal problem, i.e., determine a node when to transmit and when to receive.

We first introduce how to avoid the hidden terminal problem after a valid coloring is obtained. In this paper, the transmitter is marked with a sign "+" and the receiver is marked with a sign "−". Given a coloring of graph G and a color k, a subgraph $G^k = (V^k, E^k)$ is defined as follows: a) V^k is the set of vertices incident to the edges colored with k. b) E^k is the

Algorithm 1 Compact wakeup scheduling of a tree

Input: A tree $G = (V, E)$.
Output: A valid compact wakeup scheduling.

// Interval edge-coloring of a tree
1: Color any edge with 1.
2: **while** there are uncolored edges **do**
3: Find a uncolored edge e whose end vertex v is adjacent to an already colored edge. Let $\{a, \cdots, b\}$ be the interval of colors assigned to v.
4: **if** $a > 1$ **then**
5: Color edge e with $a - 1$.
6: **else**
7: Color edge e with $b + 1$.
// Time slot assignment
8: Map each color to two consecutive time slots, and use Algorithm 2 to determine a valid direction of transmission assignment to avoid interferences.

Algorithm 2 DFS-based sign assignment algorithm

Input: A subgraph $G^k = (V^k, E^k)$.
Output: A valid direction of transmission assignment.

1: Start by visiting any node in V^k, and assign a sign "+" to it.
2: Initiate a DFS procedure.
3: **while** there are unvisited nodes **do**
4: Let edge e be traversed from a visited node v_i to an unvisited node v_j using the DFS procedure.
5: **if** e is colored with k **then**
6: Assign v_j the sign opposite to v_i.
7: **else**
8: Assign v_j the sign same to v_i.

set of edges with both end vertices in V^k. When a node is assigned a sign "−", the only neighbor assigned a sign "+" in G^k is the neighbor incident to the edge colored with k, and the other neighbors in G^k are individually assigned a sign "−". Then, nodes incident to an edge colored with k always have an opposite sign, and nodes incident to an edge colored with other colors have the same sign. Algorithm 2, based on Depth First Search (DFS), can provide a valid direction of transmission assignment to G^k. Note that the time slot assignment also avoids the exposed terminal problem, as shown in Fig. 3 (c).

We define $\chi_{cw}(G)$ as the minimum number of colors assigned over all valid compact wakeup schedulings of graph G. As any valid coloring in a tree requires at least Δ colors and an interval edge-coloring can be obtained using Δ colors, $\chi_{cw}(G)$ is equal to Δ. Then, the number of time slots assigned in the compact wakeup scheduling is 2Δ, which is the optimum number of time slots. Algorithm 1 describes the compact wakeup scheduling for trees. Both the interval edge-coloring of a tree and the time slot assignment can be obtained using $O(n)$, where n is the number of vertices in a tree. Thus, the algorithm to produce a valid compact wakeup scheduling for trees is polynomial-time.

B. Grid Graphs

A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$) is a square lattice graph composed of $\mathcal{V}\mathcal{H}$ vertices. The grid graph has $\mathcal{H}$ vertical paths and $\mathcal{V}$ horizontal paths, where each vertical path consists of $\mathcal{V}$ vertices and each horizontal path consists of $\mathcal{H}$ vertices.

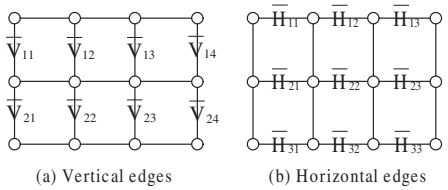


Fig. 4. Vertical and horizontal edges in grid graphs.

Definition 2. In a $\mathcal{V} \times \mathcal{H}$ grid graph, $\overline{V}_{ij}$ ($1 \le i \le \mathcal{V} - 1,\ 1 \le j \le \mathcal{H}$) denotes the i^{th} vertical edge in the j^{th} vertical path, and $\overline{H}_{ij}$ ($1 \le i \le \mathcal{V},\ 1 \le j \le \mathcal{H} - 1$) denotes the j^{th} horizontal edge in the i^{th} horizontal path.

Sample grids with labeled vertical and horizontal edges are illustrated in Fig. 4 (a) and (b). $\overline{V}_{ij}$ is called *parallel* to $\overline{V}_{mn}$ if $i = m$, $\overline{H}_{ij}$ is called *parallel* to $\overline{H}_{mn}$ if $j = n$. For example, $\overline{H}_{11}$, $\overline{H}_{21}$ and $\overline{H}_{31}$ are parallel in Fig. 4 (b).

Grid graphs can be consecutively colored with Δ colors, and one interval edge-coloring approach is given below: For a $\mathcal{V} \times \mathcal{H}$ grid graph, let c be a consecutive coloring of each horizontal path with colors 2 and 3. For each $i = 1, 2, \cdots, \mathcal{V}$, we color the edges of i^{th} horizontal path according to c. Let $\{a, \cdots, b\}$ be an interval of colors assigned at each vertex in the corresponding horizontal path, then edge $\overline{V}_{1j}$ is colored with $a - 1$, $\overline{V}_{2j}$ with $b + 1$, $\overline{V}_{3j}$ with $a - 1$, and so forth, where $1 \le j \le \mathcal{H}$. By repeating this for all edges, we could obtain a interval edge-coloring of G, and a sample of the edge-coloring is shown in Fig. 5 (a).

A valid direction of transmission assignment can be obtained to avoid the hidden terminal problem using Algorithm 2 in acyclic subgraphs G^k. But grid graphs contain cycles, a valid assignment does not exist if we use the interval edge-coloring approach above. For example, this edge-coloring cannot avoid the hidden terminal problem as shown in Fig. 5 (b). Interestingly, Gandham et al. [6] prove that all the nodes in a cycle

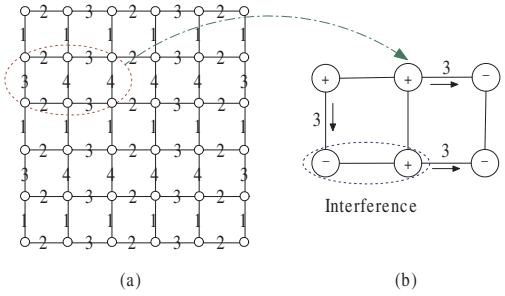


Fig. 5. An interval edge-coloring of a grid graph: (a) interval edge-coloring, (b) hidden terminal problem.

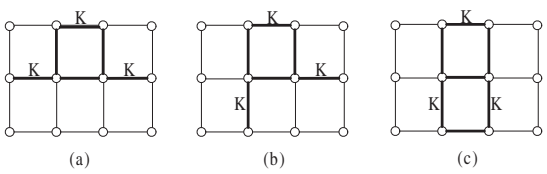


Fig. 6. Invalid colorings in the compact wakeup scheduling of grid graphs.

of G^k can be given a valid sign "+" or "−" if and only if there are an even number of edges with color k in the cycle.

If the edges colored with "3" in the cycle of Fig. 5 (b) are assigned with other colors, the consecutiveness of the colors assigned to the edges incident to one node cannot be held. Our solution for a grid graph first considers the property of the hidden terminal problem in the grid graph, and then deals with the consecutiveness of the edge-coloring.

Definition 3. In a grid graph, the maximum degree of vertices is $\Delta = 4$. The vertices of degree 4 are *inner vertices*, the vertices of degree 2 or 3 are *boundary vertices*, the edges incident to at least one inner vertex are *inner edges*, and the edges incident to two boundary vertices are *boundary edges*.

Definition 4. In the compact wakeup scheduling of a grid graph, the colors assigned to the inner edges incident to an inner vertex form an interval of 4 integers. When the total number of colors assigned is less than 8, certain color must appear in one of the inner edges and this color is referred to as a *critical color*.

In grid graphs, if the total number of colors assigned is M $(4 \le M \le 7)$, then the number of critical colors is $8 - M$. For example, if $M = 4$, the set of critical colors is $\{1, 2, 3, 4\}$; if $M = 5$, the set of critical colors is $\{2, 3, 4\}$; if $M = 6$, the set of critical colors is $\{3, 4\}$.

Lemma 1. *If K is a critical color assigned to an inner edge e incident to two inner vertices in the compact wakeup scheduling of a grid graph, the inner edges parallel to e are all colored with K.*

Proof: Without loss of generality, we assume that an inner horizontal edge $\overline{H}_{ij}$ $(2 \le i \le \mathcal{V}-1,\ 2 \le j \le \mathcal{H}-2)$ is colored with K in a $\mathcal{V} \times \mathcal{H}$ grid graph. The cases of colorings shown in Fig. 6 would lead to odd number of edges with color K in subgraph G^K (see the thick lines in Fig. 6), and no feasible direction of transmission can be obtained. Since K is a critical color, $\overline{H}_{(i+1)j}$ $(i + 1 \le \mathcal{V} - 1)$ must be colored with K. By applying recursion, the horizontal edges $\overline{H}_{mj}$ $(2 \le m \le \mathcal{V} - 1)$ are in a parallel pattern, as shown in Fig. 7 (a). ■

Lemma 2. *If K is a critical color, the inner edges colored with K are in an interlined pattern.*

Proof: Without loss of generality, we assume an inner horizontal edge $\overline{H}_{ij}$ $(2 \le i \le \mathcal{V}-1,\ 2 \le j \le \mathcal{H}-2)$ is colored with K in a $\mathcal{V} \times \mathcal{H}$ grid graph. According to Lemma 1, the horizontal edges $\overline{H}_{mj}$ $(2 \le m \le \mathcal{V}-1)$ are colored with K. Let $k = j+2$ $(k \le \mathcal{H}-2)$, $\overline{H}_{3k}$ must be colored with K, since K is a critical color and $\overline{H}_{3(k-1)}$, $\overline{V}_{2k}$ as well as $\overline{V}_{3k}$ cannot be colored with K. According to Lemma 1, the horizontal edges $\overline{H}_{mk}$ $(2 \le m \le \mathcal{V}-1)$ are colored with K, and the result still holds when $k = j - 2$ $(k \ge 2)$. By applying recursion, the horizontal edges $\overline{H}_{mk}$ $(k = j \pm 2n,\ n \in N,\ 2 \le m \le \mathcal{V}-1,\ 2 \le k \le \mathcal{H}-2)$ are colored with K. If $k = 1$ $(or\ \mathcal{H}-1)$ and $k = j \pm 2n,\ n \in N$, the horizontal edges $\overline{H}_{mk}$ $(3 \le m \le \mathcal{V}-2)$ are colored with K, since K is a critical color. Hence, the inner edges colored with a critical color are in an interlined pattern, as shown in Fig. 7 (b). ■

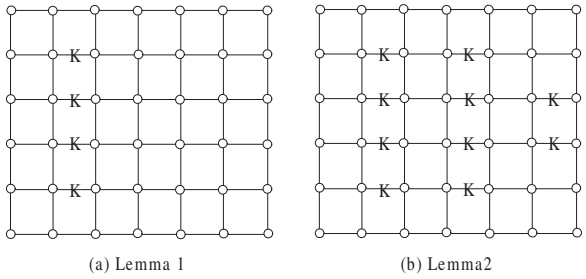


Fig. 7. Coloring pattern in the compact wakeup scheduling of grid graphs.

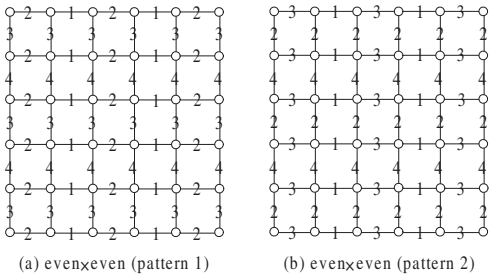


Fig. 8. The coloring patterns in the compact wakeup scheduling of grid graphs when both $\mathcal{V}$ and $\mathcal{H}$ are even.

Theorem 2. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are even) can be consecutively colored with 4 colors in the compact wakeup scheduling, and $\chi_{cw}(G) = 4$.*

Proof: Fig. 8 (a) and (b) show two possible colorings in the compact wakeup scheduling, if both $\mathcal{V}$ and $\mathcal{H}$ are even. Since the edges with the same color are in a parallel and interlined pattern, there are an even number of edges with color k $(1 \le k \le 4)$ in a cycle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. Therefore, $\chi_{cw}(G) = 4$. ■

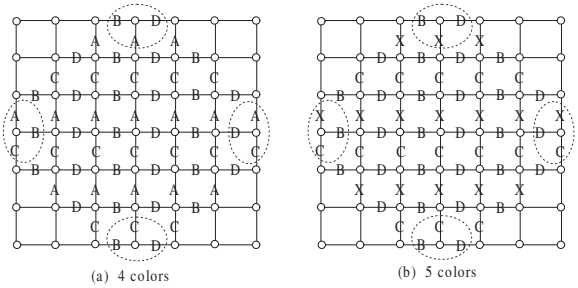
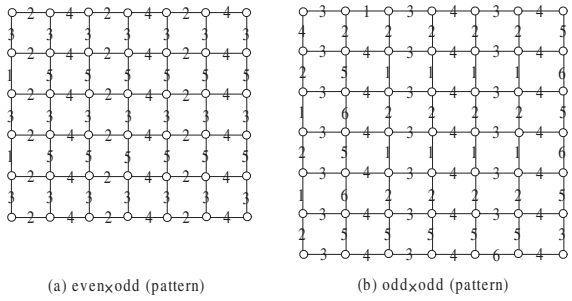


Fig. 9. The colorings in the compact wakeup scheduling using 4 and 5 colors of grid graphs when both $\mathcal{V}$ and $\mathcal{H}$ are odd.



(a) even×odd (pattern) (b) odd×odd (pattern)

Fig. 10. The coloring patterns in the compact wakeup scheduling of grid graphs: (a) One of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. (b) Both $\mathcal{V}$ and $\mathcal{H}$ are odd.

Lemma 3. *A $\mathcal{V} \times \mathcal{H}$ grid graph($3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) cannot be consecutively colored with 4 or 5 colors in the compact wakeup scheduling.*

Proof: 1) If the grid could be consecutively colored with 4 colors $A, B, C, D \in \{1, 2, 3, 4\}$, the four colors are all critical colors. An inner vertex has 4 incident inner edges, the inner edges are colored with A, B, C, D, respectively. According to Lemma 1 and Lemma 2, A, B, C, D are all in a parallel and interlined pattern, and the coloring is shown in Fig. 9 (a). As the colors assigned to the edges incident to the vertices in the dashed circles must be consecutive, the color sets $\{A, B, C\}$, $\{B, C, D\}$, $\{A, C, D\}$ and $\{A, B, D\}$ must consist of three consecutive numbers. However, $\{1, 2, 3\}$ and $\{2, 3, 4\}$ are the only two possible cases with three consecutive numbers, which leads to a contradiction. Note that $\{4, 1, 2\}$ is not consecutive, as nodes operate in a low duty cycle mode and then a gap exists between 4 and 1.

2) If the grid could be consecutively colored with 5 colors, $B, C, D \in \{2, 3, 4\}$ are critical colors and the non-critical color 1 or 5 is denoted by X. For an inner vertex has 3 incident critical inner edges, the inner edges are colored with B, C, D, respectively. According to Lemma 1 and Lemma 2, B, C, D are all in a parallel and interlined pattern, and the coloring is shown in Fig. 9 (b). Since the colors assigned to the edges incident to the vertices in the dashed circles must be consecutive, the color sets $\{B, C, X\}$, $\{B, C, D\}$, $\{C, D, X\}$ and $\{B, D, X\}$ must consist of three consecutive numbers. However, $\{1, 2, 3\}$, $\{2, 3, 4\}$ and $\{3, 4, 5\}$ are the only three possible cases with three consecutive numbers, which leads to a contradiction. ■

Similarly, we could get the following lemma.

Lemma 4. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd) cannot be consecutively colored with 4 colors in the compact wakeup scheduling.*

Theorem 3. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd) can be consecutively colored with 5 colors in the compact wakeup scheduling, and $\chi_{cw}(G) = 5$.*

Theorem 4. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) can be consecutively colored with 6 colors in the compact wakeup scheduling, and $\chi_{cw}(G) = 6$.*

Algorithm 3 Compact wakeup scheduling of a grid graph

Input: A $\mathcal{V} \times \mathcal{H}$ grid graph G ($3 \leq \mathcal{V} \leq \mathcal{H}$).
Output: A valid compact wakeup scheduling.
// Interval edge-coloring of a grid
Decide the parity of $\mathcal{V}$ and $\mathcal{H}$ (even or odd).
if both $\mathcal{V}$ and $\mathcal{H}$ are even **then**
  color G using the pattern in Fig. 8.
else if one of $\mathcal{V}$ and $\mathcal{H}$ is even and one is odd **then**
  color G using the pattern in Fig. 10 (a).
else
  color G using the pattern in Fig. 10 (b).
// Time slot assignment
Map color k to two consecutive time slots $\{2k - 1, 2k\}$, and use Algorithm 2 to determine a valid direction of transmission assignment to avoid interferences.

The proofs of Theorems 4 and 5 are omitted due to space limitations. We can also obtain all the possible coloring patterns in the compact wakeup scheduling for grid graphs. Fig. 10 (a) shows a possible coloring in the compact wakeup scheduling, if one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. Fig. 10 (b) shows a possible coloring in the compact sleep scheduling, if both $\mathcal{V}$ and $\mathcal{H}$ are odd. According to Theorem 3, 4 and 5, the number of time slots assigned is optimum. Algorithm 3 describes the compact wakeup scheduling of a grid graph and its complexity is $O(n)$.

IV. Conclusion

In this paper, we address a new interference-free TDMA wakeup scheduling problem in WSNs, called compact wakeup scheduling. In the scheduling, a node needs to wake up only once to communicate bidirectionally with all its neighbors, thus reducing the time overhead and energy cost in the state transitions. We propose polynomial-time algorithms to achieve the optimum number of time slots assigned in a period for trees and grid graphs. In the process of time slot assignments, both the hidden terminal and exposed terminal problems can be avoided.

References

[1] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. of ACM MobiHoc*, 2006.
[2] W. Wang, Y. Wang, X. Y. Li, W. Z. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *Proc. of ACM MobiCom*, 2006.
[3] J. Ma, W. Lou, Y. Wu, X. Y. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2009.
[4] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *Proc. of ACM/IEEE IPSN*, 2007.
[5] A. S. Asratian and R. R. Kamalian, "Investigation on interval edge-colorings of graphs," *Journal of Combinatorial Theory, Series B*, vol. 62, no. 1, pp. 34–43, 1994.
[6] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proc. of IEEE INFOCOM*, 2005.