Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 525909, 17 pages
doi:10.1155/2012/525909

Research Article

Interference-Free Wakeup Scheduling with Consecutive Constraints in Wireless Sensor Networks

Junchao Ma and Wei Lou

Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong

Correspondence should be addressed to Wei Lou, csweilou@comp.polyu.edu.hk

Received 13 June 2011; Revised 30 September 2011; Accepted 13 October 2011

Academic Editor: Yuhang Yang

Wakeup scheduling has been widely used in wireless sensor networks (WSNs), for it can reduce the energy wastage caused by the idle listening state. In a traditional wakeup scheduling, sensor nodes start up numerous times 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 interference-free 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 it is NP-complete to decide whether a valid compact wakeup scheduling exists for an arbitrary graph. In particular, tree and grid topologies, which are commonly used in WSNs, have valid compact wakeup schedulings. We propose polynomial-time algorithms using the optimum number of time slots in a period for trees and grid graphs. Simulations further validate our theoretical results.

1. Introduction

Wireless sensor networks (WSNs) consist of hundreds to thousands of tiny, inexpensive, and battery-powered wireless sensing devices which organize themselves into multihop radio networks. As the batteries of most sensor nodes are nonrechargeable, one key challenging issue is to schedule the activities of nodes to minimize the energy consumption. The major source of energy wastage [1–3] in WSNs is the idle listening state in the radio modules, which in fact consumes almost as much energy as receiving. Therefore, nodes are generally scheduled to sleep when the radio is not in use [4, 5] and wake up when necessary. By using wakeup scheduling, nodes could operate in a low-duty-cycle mode, and periodically start up to check the channel for activity.

In wireless networks, the packets transmitted by a node may be received by all the nodes within its transmission range due to the broadcast nature of the wireless medium. Therefore, the transmission of one link may interfere with the reception of another link. To avoid the interferences among the communication links, we adopt the time division multiple access (TDMA) MAC protocols, such as TRAMA [6], DCQS [7], and DRAND [8]. TDMA protocols have the natural advantages of having no contention-introduced overhead or collisions [1]. In such protocols, the time is divided into equal intervals referred to as *time slots*. Correspondingly, nodes turn on the radio during the assigned time slots and turn off the radio when they are not transmitting or receiving in the wakeup scheduling. For multiple transmission links can communicate at the same time in wireless networks, several nodes can wake up to transmit their packets simultaneously when they do not interfere with each other. Therefore, we attempt to minimize the number of time slots assigned to each node while guaranteeing interference-free among the communication links.

The previous studies [9, 10] 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. Note that the typical startup time is on the order of milliseconds, while the transmission time may be less than the startup time if the packets are small [11]. Take Tmote Sky [12] as an example, the time and energy consumption to activate a node is about 1.4 ms and 17 μJ, respectively, whereas the time and energy consumption to transmit 1 byte is about 0.032 ms and 1.7 μJ, respectively. If a sensor node starts up too frequently, it not only needs extra time, but also consumes extra energy for state transitions. Moreover, it reduces the battery capacity due to the current surges in the state transitions.

Figure 1 shows the battery voltage of Tmote Sky sensors with different startup frequencies but with the same duty cycle (50%): one starts up every 20 ms, stays in the receive state for 10 ms, and turns to the sleep state for the rest period; the other one starts up every 100 ms, stays in the receive state for 50 ms, and turns to the sleep state for the rest period. We can see that about 8% battery voltage can be saved by reducing the startup frequency from five times to once in every 100 ms. To minimize the energy cost, the state transitions should be considered in the wakeup scheduling design. Unlike the previous work, we are interested in the scheduling with *consecutive constraints*, where all the links incident to a node are assigned consecutive time slots so that each node needs to wake up only once to communicate bidirectionally with its neighbors.

In [13], 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 [13] 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 that in [14], 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. Note that compact wakeup scheduling cannot only reduce the state transitions of transceivers, but also reduce the state transitions of other components in the nodes, such as external memory and sensing devices.

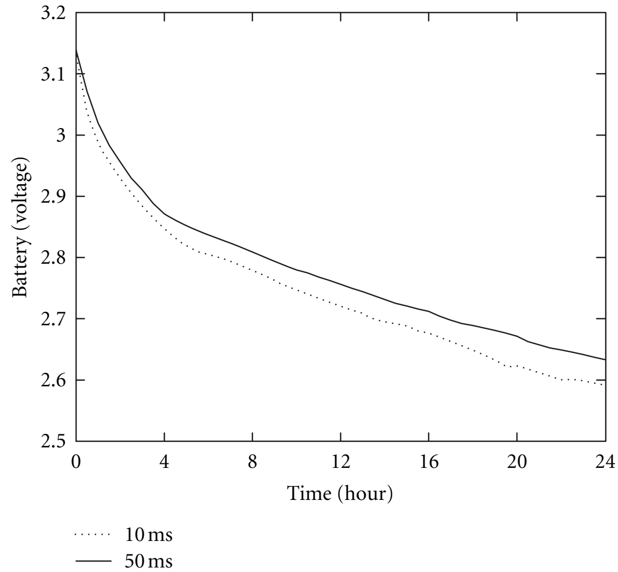
The main contributions of this paper are summarized as follows.

(i) We formulate the compact wakeup scheduling problem in WSNs to minimize the frequency of state transitions, and prove it to be NP-complete.

(ii) We present polynomial-time algorithms using the optimum number of time slots in a period for trees and grid graphs. In grid graphs, we point out all the possible coloring patterns and give the lower bound as well as the upper bound of the compact wakeup scheduling.

(iii) We develop simulations to show the efficiency of compact wakeup scheduling.

(a) Tmote Sky



(b) Battery

Figure 1: (a) Tmote Sky sensor. (b) The battery voltage of Tmotes with different startup frequency. AA Carbon-Zinc batteries are used in the experiment. 10 ms and 50 ms are active time in each period.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the system model and formulates the compact wakeup scheduling problem. Section 4 presents polynomial-time algorithms for trees and grid graphs. Section 5 shows the performance evaluation. Section 6 concludes the paper and provides directions for future research.

2. Related Work

Wakeup scheduling has attracted a lot of interest in WSNs in virtue of its energy efficiency. S-MAC [1] is a contention-based MAC scheme. In S-MAC, nodes periodically sleep and wake up, and each active period is of a fixed size with a variable sleep time. T-MAC [2] improves S-MAC by adopting a dynamic duty cycle, that is, transmitting all messages in bursts and ending the listening period when nothing is heard within a limited time. DW-MAC [4] allows nodes to wake up on demand during the sleep period and ensures that data transmissions do not collide at their intended receivers. TreeMAC [15] is a localized TDMA MAC protocol, which is designed to achieve high throughput and low congestion with low overhead. PW-MAC [16] minimizes the energy consumption by enabling senders to predict the wakeup times of receivers based on asynchronous duty cycling.

Link scheduling is time slot assignments to communication links in TDMA MAC protocols. Ramanathan and Lloyd [17] consider both the tree networks and arbitrary networks, and the performance of the proposed algorithms is bounded by the thickness of a network. In [18], Gandham et al. propose a link scheduling algorithm involving two phases. In the first phase, a valid edge coloring is obtained in a distributed fashion. In the second phase, each color is mapped to a unique time slot, and the hidden terminal problem as well as the exposed terminal problem is avoided by assigning each edge a direction of transmission. The overall scheduling requires at most $2(\Delta + 1)$ time slots when the topologies are acyclic, where Δ is the maximum degree of a graph. In [10], Wang et al. propose a degree-based heuristic algorithm with performance guarantee to obtain a good interference-free link scheduling to maximize the throughput of the network. In the algorithm, the sensors are scheduled individually in a predefined order without consecutive assignment of time slots, and each node is assigned the best possible time slot to transmit or receive without causing interferences to the already-scheduled sensors. In [19], Wu et al. propose efficient centralized and distributed scheduling algorithms that reduce the energy cost of state transitions and also propose an efficient method to construct an energy-efficient data-gathering tree. In [13], Ma et al. address the contiguous link scheduling problem by applying the interval vertex coloring in a merged conflict graph and assigning consecutive time slots to the links incident to one node to achieve better energy efficiency.

Instead of applying the interval vertex coloring, we apply the interval edge coloring in the compact wakeup scheduling. Interval edge coloring, introduced by Asratian and Kamalian [20] (available in English as [21]), is a special edge coloring in which the colors of edges incident to the same vertex must be contiguous, that is, the colors must be composed of an integer interval. Not every graph has an interval edge coloring, since a graph G with an interval edge coloring belongs to Class 1 graphs where the chromatic number of edge coloring is equal to the maximum degree Δ [21]. Sevastjanov [22] proves that the problem of determining the existence of an interval edge coloring is NP-complete, even for bipartite graphs, and Kubale [23] proves that the interval edge coloring problem with forbidden colors is also NP-complete. Experiments [24] with small and sparse graphs show that the existence of an interval edge-coloring is with high probability. Some examples of graphs with interval edge-colorings are trees, complete bipartite graphs, and grid graphs [20, 21, 25]. Giaro and Kubale give several polynomially solvable graphs in [26].

Compared to the former studies on wakeup scheduling, the compact wakeup scheduling could minimize the energy cost of state transitions, and sensors can start up only once in a period T. Furthermore, the compact wakeup scheduling considers two-way (or bidirectional) communication while our early work [13, 19] only considers one-way communication.

3. Problem Formulation

In this section, we first present the system model then formulate the compact wakeup scheduling problem.

3.1. System Model. We assume that a WSN has n static sensor nodes equipped with single omnidirectional antennas, and all the nodes have the same communication range. The network is represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of nodes and $E = \{e_1, e_2, \ldots, e_m\}$ denotes the set of edges referred to all the communication links. If $\{v_i, v_j\} \subseteq V$, the edge $e = (v_i, v_j) \in E$ if and only if v_j is located within the communication range of v_i. We assume that nodes have the ability of data aggregation and can use one time slot to transmit data in one link.

Each node operates in three states: active state (transmit, receive, and listen), sleep state, and transient state (state transition). The transient state comprises two processes: startup (from the sleep state to the active state) and turndown (from the active state to the sleep state). The startup process from the sleep state to the active state includes radio initialization, radio and its oscillator startup, and the switch of radio to active [27]. The startup process is slow due to the feedback loop in the phase-locked loop (PLL) [28], and a typical setting time of the PLL-based frequency synthesizer is on the order of milliseconds.

We assume that the interference range is equal to the communication range. Two types of interferences, primary interference and secondary interference [17], exist in the network. The primary interference occurs when a node has more than one communication task in a single time slot. Typical examples are sending and receiving at the same time and receiving from two different transmitters. The secondary interference (or called *the hidden terminal problem* [29]) occurs when a node v_i receives packets from a transmitter v_j and v_i is also within the communication range of another transmitter v_k which is intended for other nodes.

3.2. Problem Formulation. 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

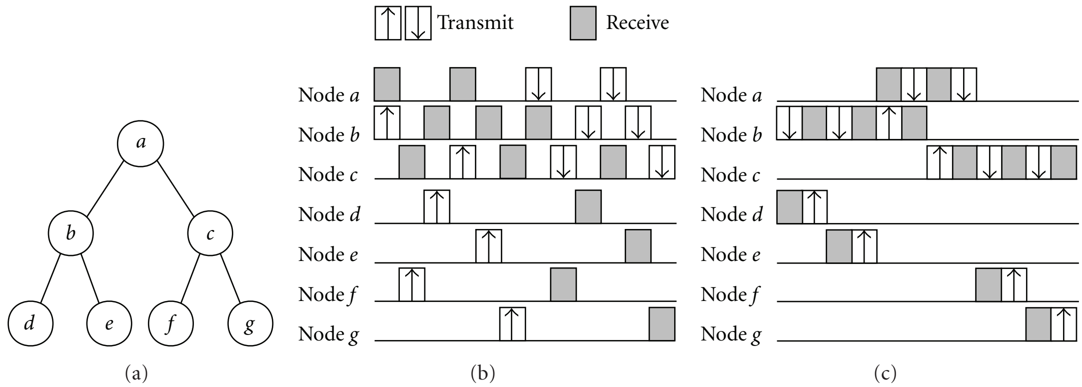


FIGURE 2: Wakeup scheduling and compact wakeup scheduling: (a) network topology, (b) wakeup scheduling, (c) compact wakeup scheduling.

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 aiming 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.

Compact wakeup scheduling attempts to assign consecutive time slots to all the links incident to a node, but it may fail to find such a scheduling. If it succeeds, the scheduling is said to be a *valid* scheduling. If not, the scheduling is said to be not a *valid* scheduling.

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. Figure 2(a) shows the given network topology. Figure 2(b) shows a wakeup scheduling, in which a node starts up numerous times in a period. Figure 2(c) shows a compact 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 Figures 2(b) and 2(c), node c needs 5 more time slots 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 2. *The problem of deciding whether a valid compact wakeup scheduling exists for an arbitrary graph G is NP-complete.*

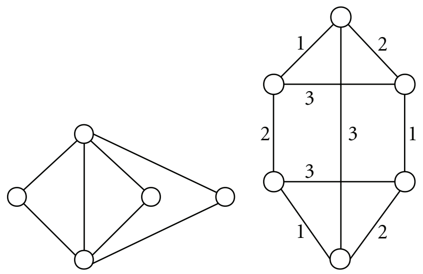
Proof. The compact wakeup scheduling problem is in NP. To verify whether a scheduling is a solution to the compact wakeup scheduling problem, we need to check (i) all the links incident to a node are assigned consecutive time slots; (ii) the scheduling is interference-free. Verifying (i) and (ii) requires $O(n)$ and $O(n^2)$ operations, respectively, where n is the number of nodes. It is clearly that this verification can be done in polynomial time.

To prove that the compact wakeup scheduling problem is NP-hard, we first restate the interval edge-coloring problem with forbidden colors which is NP-complete [21, 23]. "Given a graph G, a forbidding function F which represents the colors that cannot be assigned to each edge e, and an integer k, does there exist an interval edge coloring of G using k colors and avoiding F?" The interference, such as the hidden terminal problem, in the compact wakeup scheduling is a special case of the forbidding function in the interval edge-coloring. Thus, the compact wakeup scheduling is equivalent to the interval edge-coloring with forbidden colors, which is NP-hard. Therefore, the problem is NP-complete. □

Theorem 3. *A communication graph G with a valid compact wakeup scheduling has an interval edge coloring and belongs to Class 1 graphs.*

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 [21], graph with an interval edge coloring belongs to Class 1 graphs *where the edge chromatic number is equal to the maximum degree* Δ *of graph G*. Therefore, graph G is a Class 1 graph. □

Unfortunately, the converse proposition is not true. The graph in Figure 3(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 Figure 3(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



(a) Class 1 (b) Interval edge coloring

Figure 3: Class 1 graphs without valid compact wakeup schedulings.

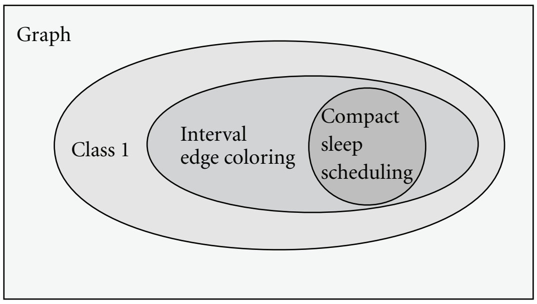


Figure 4: The relationship among Class 1 graphs, graphs with valid interval edge colorings and graphs with valid compact wakeup schedulings.

of graphs with valid interval edge colorings, and also a proper subset of Class 1 graphs, as shown in Figure 4.

Since not all communication graphs have valid compact wakeup schedulings and the problem of deciding whether a valid scheduling exists for an arbitrary graph is NP-complete, 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.

3.3. Direction of Transmission Assignment in WSNs. In the link scheduling in WSNs, each edge in the communication graph has two transmission links: one is upload link, and the other one is downlowd link. We can easily find an edge coloring of a communication graph using $\Delta + 1$ colors [30], but how can this coloring be used to assign time slots to each transmission link? In [18], each color is mapped to two unique time slots and each transmission link is assigned a time slot according to the direction of transmission assignment (i.e., which end node of edge e will transmit or receive). Both the hidden terminal problem and the exposed terminal problem can be avoided. When the topologies are acyclic, the overall scheduling requires at most $2(\Delta + 1)$ time slots, where Δ is the maximum degree of a graph. When the topologies have cycles, additional time slots may be needed.

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 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 1, based on Depth First Search (DFS), can provide a *valid* direction of transmission assignment to each edge in G^k after a valid edge coloring is obtained in acyclic topologies. Such an assignment enables one-way communication. We can reverse the direction of transmission assignment along each edge to support bidirectional communication, and then each edge is assigned two time slots.

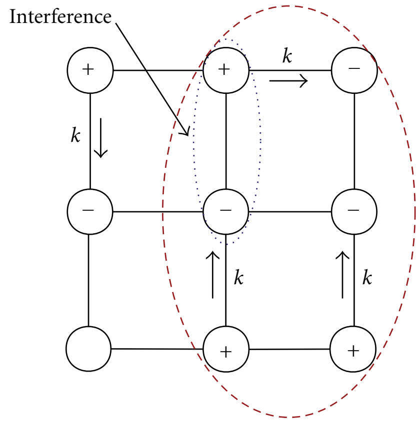


Figure 5: Cycle that has odd number of edges with color k cannot be assigned a valid direction of transmission.

Gandham et al. [18] prove that a valid direction of transmission assignment exists in acyclic topologies (e.g., tree graphs). If a valid edge coloring is obtained in the topologies which are not acyclic (e.g., grid graphs), a valid direction of transmission assignment may not exist due to the hidden terminal problem, as shown in Figure 5. Interestingly, Gandham et al. [18] also prove that all the nodes in a cycle 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.

4. 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 [31–35].

4.1. 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 slots to each edge and make it interference-free.

If graph G is a tree of degree Δ, we could get an interval edge coloring with Δ colors for G using Algorithm 2 [26]: we first color any edge with 1, then find an uncolored

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 Depth First Search (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.
(9) **end if**
(10) **end while**

Algorithm 1: DFS-based sign assignment algorithm [18].

Input: A tree $G = (V, E)$.
Output: A valid interval edge-coloring with Δ colors.
(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 $\{a, \dots, b\}$ 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 $b + 1$.
(8) **end if**
(9) **end while**

Algorithm 2: Interval edge coloring of a tree [26].

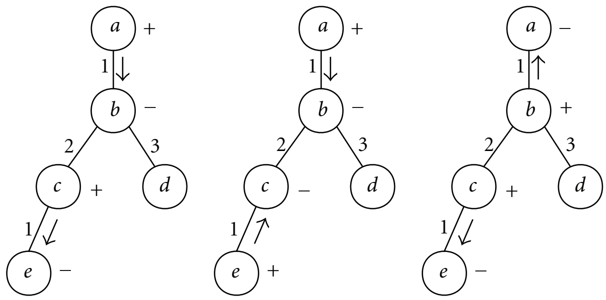
Input: A tree $G = (V, E)$.
Output: A valid compact wakeup scheduling.
(1) Use Algorithm 2 to obtain a valid interval edge-coloring with Δ colors for G.
(2) **for** $k = 1$ to Δ **do**
(3) Map color k to two consecutive time slots $\{2k - 1, 2k\}$.
(4) Use Algorithm 1 to determine a valid direction of transmission assignment for time slot $2k - 1$.
(5) Reverse the direction of transmission along each edge to obtain the other assignment for time slot $2k$.
(6) **end for**

Algorithm 3: Compact wakeup scheduling of a tree.

edge e adjacent to an already colored edge, and assign e with a consecutive color until all the edges are colored. In the coloring process, 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 in Algorithm 3. The idea is to map color k to two consecutive time slots $\{2k - 1, 2k\}$, and use Algorithm 1 to determine a valid direction of transmission assignment for time slot $2k-1$, and then reverse the direction of transmission along each edge to obtain the other assignment for time slot $2k$.

In Figure 6, links 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 Figure 6(a),



(a) (b) (c)

Figure 6: Compact wakeup scheduling of a tree: (a) hidden terminal problem, (b) avoid the hidden terminal problem, (c) avoid the exposed terminal problem.

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 Figure 6(b), the hidden terminal problem could be avoided. Similarly, time slot ts_2 is assigned in the reverse directions of transmission as shown in Figure 6(c). Inspired by this, we should determine the directions of transmission along each link carefully to avoid the hidden terminal problem, that is, determine a node when to transmit and when to receive.

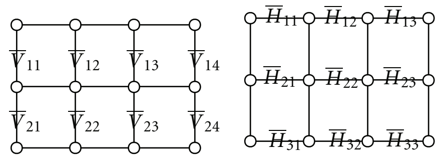
A tree does not have any cycles, and thus it is always possible to obtain a valid compact wakeup scheduling. Algorithm 1, based on Depth First Search (DFS), can provide a valid direction of transmission assignment to G^k. Note that the time slot assignment also avoids the exposed terminal problem [29], as shown in Figure 6(c).

Definition 4. The span of a valid compact wakeup scheduling of graph G is the number of colors assigned. The minimum and maximum span over all valid compact wakeup schedulings of G are denoted by $\chi_{\mathrm{cw}}(G)$ and $\zeta_{\mathrm{cw}}(G)$, respectively.

As any valid coloring in a tree requires at least Δ colors and an interval edge coloring can be obtained using Δ colors, $\chi_{\mathrm{cw}}(G)$ 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 3 describes the compact wakeup scheduling for trees. Both the interval edge coloring of a tree and the time slot assignments 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.

4.2. Grid Graphs. A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$) 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.

Definition 5. In a $\mathcal{V} \times \mathcal{H}$ grid graph, $\overline{V}_{ij}$ ($1 \leq i \leq \mathcal{V}-1$, $1 \leq j \leq \mathcal{H}$) denotes the ith vertical edge in the jth vertical path, and $\overline{H}_{ij}$ ($1 \leq i \leq \mathcal{V}$, $1 \leq j \leq \mathcal{H}-1$) denotes the jth horizontal edge in the ith horizontal path.



(a) Vertical edges (b) Horizontal edges

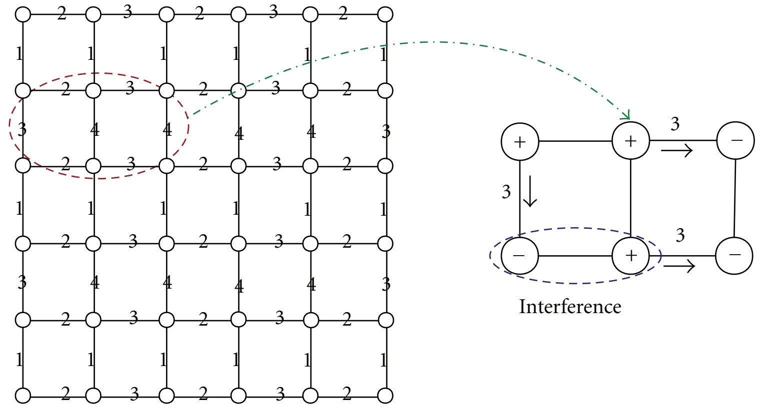
Figure 7: Vertical and horizontal edges in grid graphs.

Sample grids with labeled vertical and horizontal edges are illustrated in Figures 7(a) and 7(b). $\overline{V}_{ij}$ is called *parallel* to $\overline{V}_{mn}$ if $i = m$, and $\overline{H}_{ij}$ 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 Figure 7(b).

Grid graphs can be consecutively colored with Δ colors, and one interval edge-coloring approach is given as follows. 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 $i = 1, 2, \ldots, \mathcal{V}$, we color the edges of ith horizontal path according to c. Let $\{a, \ldots, b\}$ 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 $1 \leq j \leq \mathcal{H}$. By repeating this for all edges, we could obtain an interval edge coloring of G, and a sample of the edge coloring is shown in Figure 8(a).

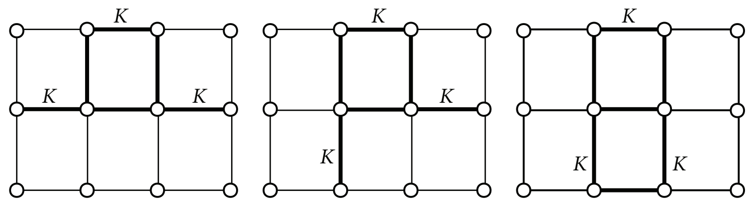
A valid direction of transmission assignment can be obtained to avoid the hidden terminal problem using Algorithm 1 in acyclic subgraphs G^k. But grid graphs contain cycles, and a valid assignment does not exist if we use the interval edge coloring approach above. For example, this edge coloring cannot avoid the hidden terminal problem as shown in Figure 8(b). Interestingly, Gandham et al. [18] prove that all the nodes in a cycle 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 Figure 8(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. Our key results for grid graphs are summarized below.



(a) (b)

Figure 8: An interval edge coloring of a grid graph: (a) interval edge coloring, (b) hidden terminal problem.



(a) (b) (c)

Figure 9: Invalid colorings in the compact wakeup scheduling.

(1) We obtain an interval edge coloring according to the parity of $\mathcal{V}$ and $\mathcal{H}$.

 (i) If both $\mathcal{V}$ and $\mathcal{H}$ are even, $\chi_{\text{cw}}(G) = 4$.

 (ii) If one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd, $\chi_{\text{cw}}(G) = 5$.

 (iii) If both $\mathcal{V}$ and $\mathcal{H}$ are odd, $\chi_{\text{cw}}(G) = 6$.

(2) We point out all the possible coloring patterns.

(3) We give the upper bound of the compact wakeup scheduling.

Definition 6. 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 7. 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*.

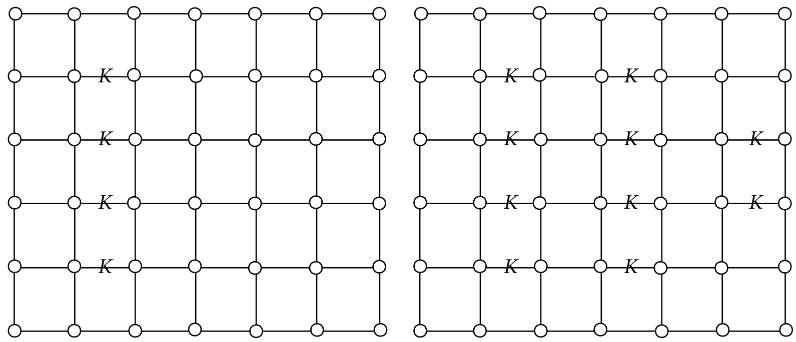
In grid graphs, if the total number of colors assigned is M $(4 \le M \le 7)$, then the number of critical colors is $8 - M$. For example, if $M = 4$, the set of critical colors is $\{1, 2, 3, 4\}$; if $M = 5$, the set of critical colors is $\{2, 3, 4\}$; if $M = 6$, the set of critical colors is $\{3,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

Lemma 8. *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 Figure 9 would lead to odd number of edges with color K in subgraph G^K (see the thick lines in Figure 9), 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 Figure 10(a). □

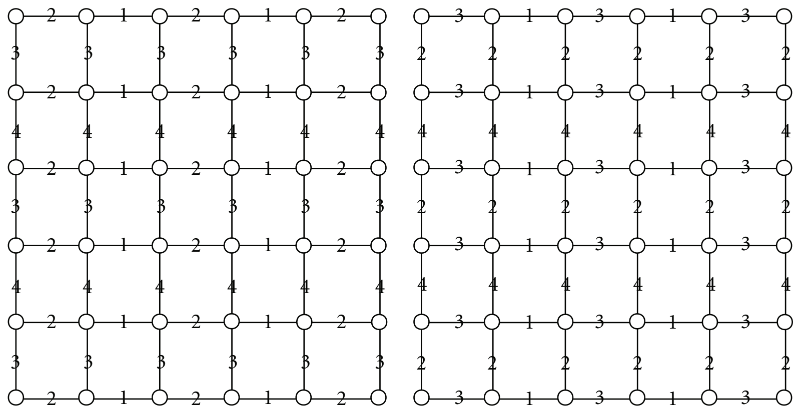
Lemma 9. *If K is a critical color, the inner edges colored with K are in an interlined pattern.*

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 8, 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 8, 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



(a) Lemma 8 (b) Lemma 9

Figure 10: Coloring pattern in the compact wakeup scheduling.



(a) Even × even (pattern 1) (b) Even × even (pattern 2)

Figure 11: The coloring patterns in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are even).

$\overline{H}_{mk}$ $(3 \leq m \leq \mathcal{V} - 2)$ 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 Figure 10(b). □

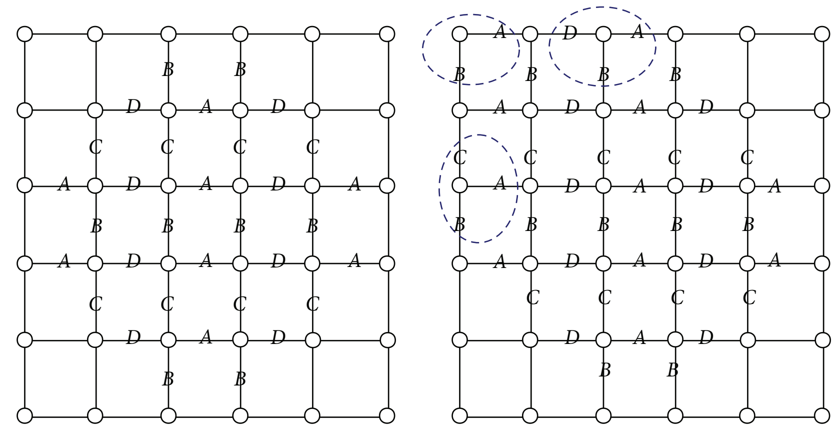
Theorem 10. *A $\mathcal{V} \times \mathcal{H}$ grid graph $(3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are even) can be consecutively colored with 4 colors in the compact wakeup scheduling, and the possible colorings must be the patterns as shown in Figure 11, and $\chi_{\text{cw}}(G) = 4$.*

Proof. Figures 11(a) and 11(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 $(1 \leq k \leq 4)$ in a cycle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. Therefore, $\chi_{\text{cw}}(G) = 4$.

As $\chi_{\text{cw}}(G)$ is equal to 4, we assume the four critical colors are A, B, C and D. According to Lemmas 8 and 9, A, B, C, and D are all in a parallel and interlined pattern shown as status 1 in Figure 12(a). To avoid the hidden terminal problem, $\overline{H}_{13}$ cannot be colored with D or C and can only be colored with A. Then $\overline{H}_{12}$ must be colored with D. Similarly, $\overline{V}_{21}$ and $\overline{V}_{31}$ are colored with C and B, respectively. Then $\overline{H}_{11}$, $\overline{V}_{11}$, $\overline{H}_{21}$, and $\overline{V}_{12}$ are colored with A, B, A, and B, respectively. We can color other edges in a similar way. In status 2 shown in Figure 12(b), we can see the color sets $\{A, B\}$, $\{A, B, D\}$, and $\{A, B, C\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Therefore, $\{A, B, C\}$ and $\{A, B, D\}$ belong to $\{1, 2, 3\}$ and $\{2, 3, 4\}$, and $\{A, B\}$ belongs to $\{2, 3\}$. For C and D are symmetrical, we can get $C = 4$ and $D = 1$. For the case that $A = 2$ and $B = 3$, the coloring pattern is Figure 11(a). For the case that $A = 3$ and $B = 2$, the coloring pattern is Figure 11(b). □

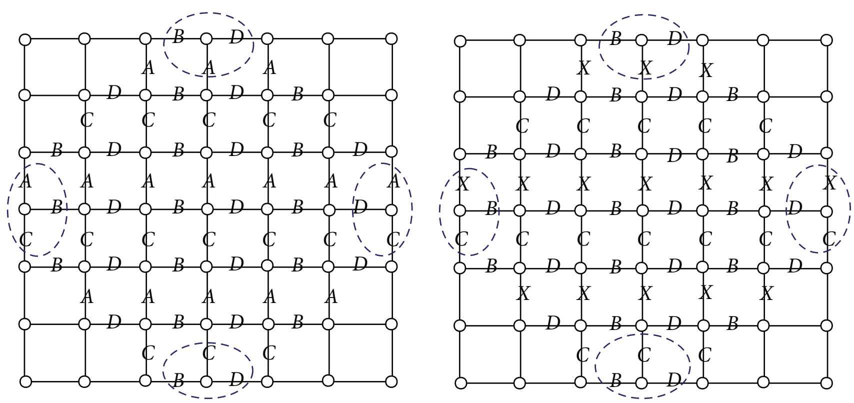
Lemma 11. *A $\mathcal{V} \times \mathcal{H}$ grid graph $(3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) cannot be consecutively colored with 4 or 5 colors in the compact wakeup scheduling.*

Proof. (1) If the grid could be consecutively colored with 4 colors A, B, C, and D, the four colors belonging to $\{1, 2, 3, 4\}$ are all critical colors. For an inner vertex has 4 incident inner edges, the inner edges are colored with A, B, C, and D, respectively. According to Lemmas 8 and 9, A, B, C and D are all in a parallel and interlined pattern, and the coloring is shown in Figure 13(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



(a) Even × even (status 1) (b) Even × even (status 2)

Figure 12: The coloring in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are even).



(a) 4 colors (b) 5 colors

Figure 13: The colorings in the compact wakeup scheduling using 4 and 5 colors (both $\mathcal{V}$ and $\mathcal{H}$ are odd).

cases with three consecutive numbers, which leads to a contradiction.

(2) If the grid could be consecutively colored with 5 colors, B, C and D belonging to $\{2,3,4\}$ are critical colors and the noncritical 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, and D, respectively. According to Lemmas 8 and 9, B, C, and D are all in a parallel and interlined pattern, and the coloring is shown in Figure 13(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.

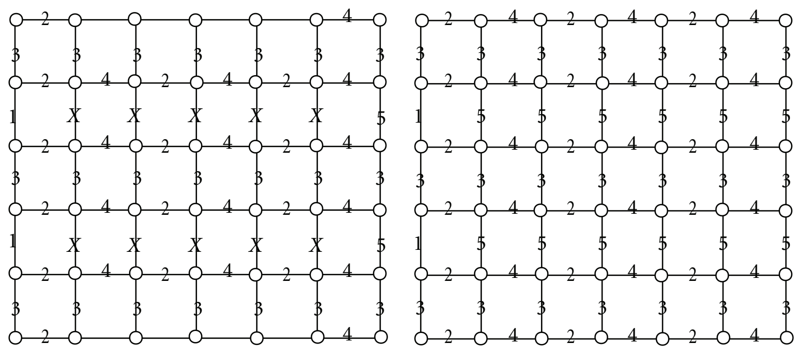
Lemma 12. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$, one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd) cannot be consecutively colored with 4 colors in the compact wakeup scheduling.*

Theorem 13. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$, 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 the possible coloring must be the pattern as shown in Figure 14(a), and $\chi_{\text{cw}}(G) = 5$.*

Proof. Figure 14(b) shows a possible coloring in the compact wakeup scheduling by determining the colors for the rest uncolored edges in Figure 14(a), if one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. Since the edges with the same color are in a parallel and interlined pattern, there are an even number of edges with color k ($1 \le k \le 5$) in a cycle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. By combining with Lemma 12, $\chi_{\text{cw}}(G) = 5$.

Since $\chi_{\text{cw}}(G)$ is equal to 5, we assume B, C, and D belonging to $\{2,3,4\}$ are critical colors and the noncritical color 1 or 5 is denoted by X. As an inner vertex has 3 incident critical inner edges, Figures 15(a), 15(c), and 15(d) are the possible coloring patterns.

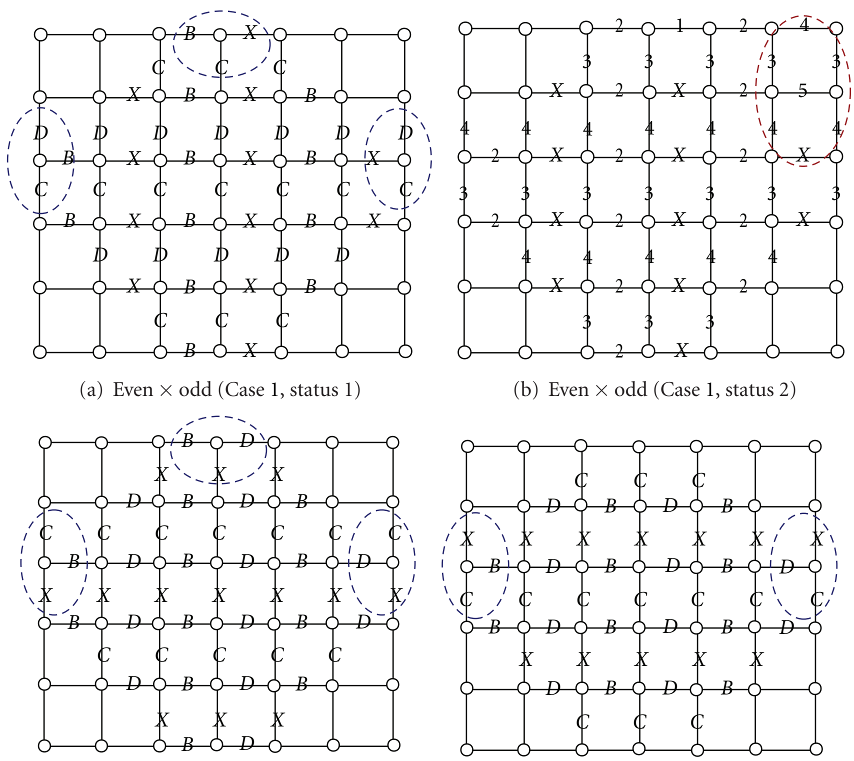
Case 1. In status 1 shown in Figure 15(a), $\overline{H}_{14}$ cannot be colored with B, C, or D and can only be colored with X.



(a) Even × odd (general pattern)

(b) Even × odd (pattern)

Figure 14: The general coloring pattern and coloring pattern in the compact wakeup scheduling (one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. The uncolored edges depend on the edges colored with X, and $X = 1$ or 5).



(a) Even × odd (Case 1, status 1)

(b) Even × odd (Case 1, status 2)

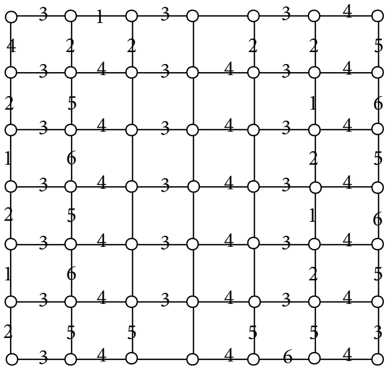
(c) Even × odd (Case 2)

(d) Even × odd (Case 3)

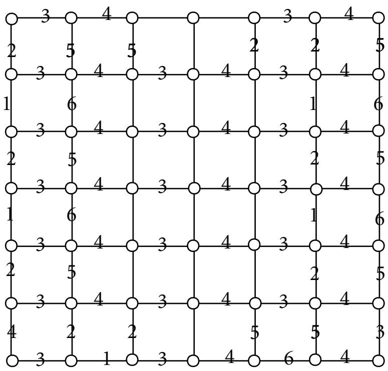
Figure 15: The colorings in the compact wakeup scheduling (one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd).

Then $\overline{H}_{13}$ must be colored with B. Similarly, $\overline{V}_{21}$, $\overline{V}_{31}$, $\overline{V}_{27}$, $\overline{V}_{37}$ are colored with D, C, D, and C, respectively. We can see that the color sets $\{B, C, X\}$, $\{B, C, D\}$, and $\{C, D, X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Then, $\{B, C, X\}$ and $\{C, D, X\}$ belong to $\{1, 2, 3\}$ and $\{3, 4, 5\}$. Then, we can get $C = 3$. If $B = 2$ and $D = 4$, $\overline{H}_{15}$, $\overline{V}_{16}$, $\overline{H}_{26}$, and $\overline{V}_{17}$ are colored with 2, 3, 5, and 3, respectively, shown as the status 2 in Figure 15(b). Then $\overline{H}_{16}$ can only be colored with 4, which leads to interferences in the dashed circle. Similarly, if $B = 4$ and $D = 2$, we cannot get an interference-free scheduling either. Hence, the coloring pattern in Figure 15(a) is not valid.

Case 2. In Figure 15(c), $\overline{H}_{14}$ cannot be colored with B, C, or X and can only be colored with D. Then $\overline{H}_{13}$ must be colored with B. Similarly, $\overline{V}_{21}$, $\overline{V}_{31}$, $\overline{V}_{27}$, and $\overline{V}_{37}$ are colored with C, X, C, and X, respectively. We can see that the color sets $\{B, D, X\}$, $\{B, C, X\}$, and $\{C, D, X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Moreover, $B, C, D \in \{2, 3, 4\}$ are consecutive.



(a) Odd × odd (general pattern 1)



(b) Odd × odd (general pattern 2)

Figure 16: The general coloring patterns in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are odd. The uncolored edges have various alternatives).

However, $\{1,2,3\}$, $\{2,3,4\}$, and $\{3,4,5\}$ are the only three possible cases with three consecutive numbers. Hence, the coloring pattern in Figure 15(c) is not valid.

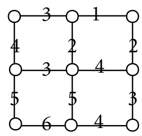
Case 3. In Figure 15(d), $\overline{V}_{21}$ cannot be colored with B, C, or D and can only be colored with X. Then $\overline{V}_{31}$ must be colored with C. Similarly, $\overline{V}_{27}$ and $\overline{V}_{37}$ are colored with X and C, respectively. We can see that the color sets $\{B,C,X\}$ and $\{C,D,X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Then $C = 3$. For B and D are symmetrical, we can get $B = 2$ and $D = 4$. By assigning the possible colors in other edges, the coloring pattern in Figure 14(a) is obtained. □

Theorem 14. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) can be consecutively colored with 6 colors in the compact wakeup scheduling, and the possible colorings must be the patterns as shown in Figure 16, and $\chi_{\mathrm{cw}}(G) = 6$.*

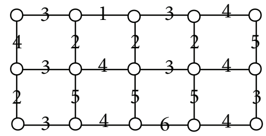
Proof. Figures 18(a) and 18(b) show two possible colorings in the compact wakeup scheduling by determining the colors for the rest uncolored edges in Figures 16(a) and 16(b), if both $\mathcal{V}$ and $\mathcal{H}$ are odd. Particularly, if $\mathcal{V} = 3$, the possible colorings are shown in Figures 17(a) and 17(b). Since there are even number of edges with color k ($1 \le k \le 6$) in a circle in the subgraph G^k, then the scheduling could avoid the hidden terminal problem. According to Lemma 11, $\chi'(G) = 6$.

Since the grid graph can be consecutively colored with 6 colors, 3 and 4 are critical colors. For an inner vertex has two incident critical inner edges, Figures 19(a) and 19(c) are the possible coloring patterns.

Case 1. In Figure 19(a), $\overline{V}_{21}$, $\overline{V}_{31}$, and $\overline{H}_{31}$ cannot be colored with 3, but can only be colored with $\{4,5,6\}$. For $\overline{V}_{31}$ and $\overline{H}_{31}$ cannot be colored with 4, $\overline{V}_{21}$ must be colored with 4. Similarly, $\overline{H}_{12}$ must be colored with 3. Then $\overline{V}_{11}$, $\overline{V}_{21}$, and $\overline{H}_{21}$ must be colored with $\{4,5,6\}$, and $\overline{H}_{11}$, $\overline{H}_{12}$, and $\overline{V}_{12}$ must be colored with $\{1,2,3\}$. For $\overline{V}_{12}$, $\overline{V}_{22}$, $\overline{H}_{21}$, and $\overline{H}_{22}$ are consecutively colored, $\overline{V}_{12}$ is colored with 2 and $\overline{H}_{21}$ is colored with 5. Then, $\overline{V}_{11}$ is colored with 6 and $\overline{H}_{11}$ is colored with 1, which leads to an inconsecutive coloring, as shown in Figure 19(b). Hence, Figure 19(a) is not a possible coloring.

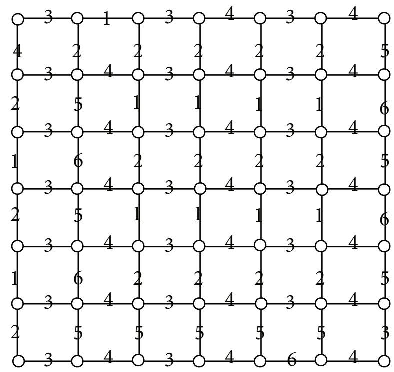


(a) 3 × 3

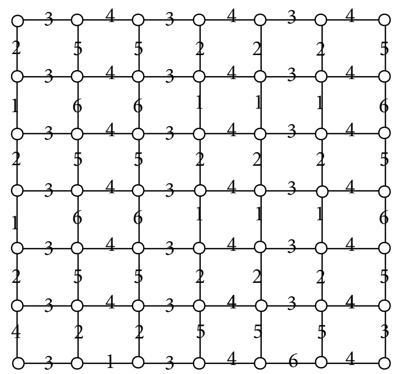


(b) 3 × 5

Figure 17: The coloring patterns in the compact wakeup scheduling ($\mathcal{V} = 3$ and $\mathcal{H}$ is odd).

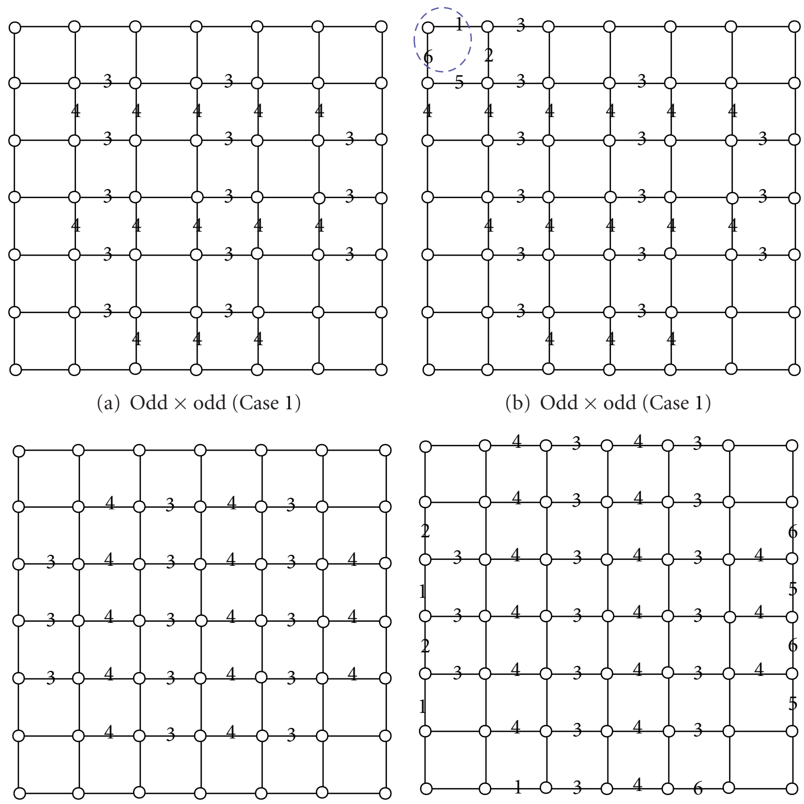


(a) Odd × odd (pattern 1)



(b) Odd × odd (pattern 2)

Figure 18: The coloring patterns in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are odd).



(a) Odd × odd (Case 1)

(b) Odd × odd (Case 1)

(c) Odd × odd (Case 2)

(d) Odd × odd (Case 2)

Figure 19: The colorings in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are odd).

Case 2. In Figure 19(c), $\overline{V}_{21}$, $\overline{V}_{23}$, and $\overline{H}_{31}$ cannot be colored with 4, but can only be colored with $\{1,2,3\}$; $\overline{V}_{27}$, $\overline{V}_{37}$, and $\overline{H}_{36}$ cannot be colored with 3, but can only be $\{4,5,6\}$. According to the symmetrical property, we suppose $\overline{V}_{27}$ and $\overline{V}_{37}$ are colored with 6 and 5, respectively. If $\overline{V}_{21}$ is colored with 2 and $\overline{V}_{31}$ is colored with 1, we can get Figure 19(d). By assigning the possible colors in other edges, the coloring pattern in Figure 16(a) is obtained. If $\overline{V}_{21}$ is colored with 1 and $\overline{V}_{31}$ is colored with 2, we can get the coloring pattern in Figure 16(b).

Hence, the possible colorings must be the patterns as shown in Figures 16(a) and 16(b), and $\chi'(G) = 6$. □

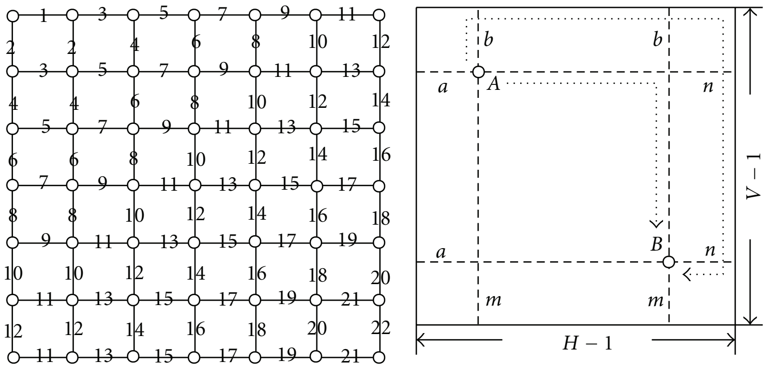
Theorem 15. *In the compact wakeup scheduling of a $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \le \mathcal{V} \le \mathcal{H}$), $2\mathcal{V} + 2\mathcal{H} - 6 \le \zeta_{cw}(G) \le (1/6)(13\mathcal{V} + 13\mathcal{H} - 8)$.*

Proof. Lower bound: we can get a valid consecutive edge coloring with a valid direction of transmission assignment in the compact wakeup scheduling using $2\mathcal{V} + 2\mathcal{H} - 6$ colors. For example, the number of colors assigned is $22 = 2 \times 7 + 2 \times 7 - 6$ in a 7×7 grid as shown in Figure 20(a).

Upper bound: for a consecutive edge coloring in the compact wakeup scheduling of a grid graph G, the difference in colors of edges incident to a node v cannot exceed $\deg(v_i) - 1$. Suppose that $v_1, v_2, \ldots, v_m$ is the vertex sequence of a path connecting edges with extremal colors, we could get $\zeta_{cw}(G) \le 1 + \sum_{i=1}^{m}(\deg(v_i) - 1)$. We suppose vertices A and B are on the path connecting edges with minimum and maximum colors, respectively, as shown in Figure 20(b). We assume vertex A is on the common point of $\overline{H}_{(b+1)(a+1)}$ and $\overline{V}_{(b+1)(a+1)}$, and vertex B is on the common point of $\overline{H}_{(\mathcal{V}-m)(\mathcal{H}-1-n)}$ and $\overline{V}_{(\mathcal{V}-1-m)(\mathcal{H}-n)}$. We can get $\zeta_{cw}(G) \le 1 + 3(\mathcal{H} - 1 - a - n + 1) + 3(\mathcal{V} - 1 - b - m) = 3(\mathcal{V} + \mathcal{H} - a - b - m - n) - 2$ using route 1. We have also known $\zeta_{cw}(G) \ge 2\mathcal{V} + 2\mathcal{H} - 6$. $3(\mathcal{V} + \mathcal{H} - a - b - m - n) - 2$ should be no less than $2\mathcal{V} + 2\mathcal{H} - 6$. Otherwise, $2\mathcal{V} + 2\mathcal{H} - 6$ should also be the upper bound. Then we get $\mathcal{V} + \mathcal{H} + 4 \ge 3(a + b + m + n)$. Without loss of generality, we assume $a + m \ge b + n$. Then, $b + n \le (1/6)(\mathcal{V} + \mathcal{H} + 4)$. We can also get $\zeta_{cw}(G) \le 1 + 3b + 2(\mathcal{H} - a - 1) + 1 + 2(\mathcal{V} - m - 1) + 3n = 2(\mathcal{V} + \mathcal{H} - a - m) + 3b + 3n - 2$ using route 2. Then, $\zeta_{cw}(G) = 2(\mathcal{V} + \mathcal{H}) + 3(b + n) - 2(a + m) - 2 \le 2(\mathcal{V} + \mathcal{H}) + b + n - 2 \le (1/6)(13\mathcal{V} + 13\mathcal{H} - 8)$.

Thus, $\zeta_{cw}(G)$ is bounded by $2\mathcal{V} + 2\mathcal{H} - 6$ and $(1/6)(13\mathcal{V} + 13\mathcal{H} - 8)$. □

According to Theorems 10, 13, and 14, the number of time slots assigned is optimum. If both $\mathcal{V}$ and $\mathcal{H}$ are even, the number of time slots assigned in a period is $4 \times 2 = 8$ in a $\mathcal{V} \times \mathcal{H}$ grid graph. If one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd, the number of time slots is $5 \times 2 = 10$. If both $\mathcal{V}$ and $\mathcal{H}$ are odd, the number of time slots is $6 \times 2 = 12$. Algorithms 4 and 5 describe the interval edge coloring and



(a) (b)

Figure 20: $\zeta_{cw}(G)$ in the compact wakeup scheduling: (a) lower bound of $\zeta_{cw}(G)$, (b) upper bound of $\zeta_{cw}(G)$.

Input: A $\mathcal{V} \times \mathcal{H}$ grid graph G ($3 \le \mathcal{V} \le \mathcal{H}$).
Output: A valid interval edge-coloring with $\chi_{cw}(G)$ colors.
(1) Decide the parity of $\mathcal{V}$ and $\mathcal{H}$ (even or odd).
(2) **if** both $\mathcal{V}$ and $\mathcal{H}$ are even **then**
(3) Color G using the pattern in Figure 11.
(4) **else if** one of $\mathcal{V}$ and $\mathcal{H}$ is even and one is odd **then**
(5) Color G using the pattern in Figure 14(b).
(6) **else**
(7) Color G using the pattern in Figure 18.
(8) **end if**

Algorithm 4: Interval edge coloring of a grid graph.

compact wakeup scheduling of a grid graph, respectively. The complexity of the compact wakeup scheduling of a grid graph is $O(n)$.

5. Performance Evaluation

In this section, we study the performance of the compact wakeup scheduling of trees and grid graphs, and we also compare our algorithms with the *degree-based heuristic* in [10] and the *contiguous link scheduling* in [13]. The performance metrics used in the evaluation are the transient energy consumption and the waiting period. The total energy consumption is an important metric in WSNs, but the energy consumption except the transient energy consumption is the same among the three schemes under identical traffic conditions, so we will only focus on the transient energy consumption. The waiting period is defined as the total time a node stays in the waiting status from the first neighbor waking up to the last neighbor waking up as the node waits for gathering the information from all its neighbors. The waiting period reflects the extra delay caused by the node if it stays in the sleep state for the wakeup of neighbors.

We adopt the following parameters in our simulation: the transient energy to activate a sensor is 17 μJ [12], a time slot is 0.1 second, a scheduling period T is 10 seconds (=100 time slots), and the network operating time is 1 day. In the tree construction of n nodes, the number of children nodes of each sensor is randomly set from 1 to 4. The root node first determines its children nodes, and then each child node determines its children nodes, and so on until the total number of nodes in the tree reaches n. In the tree construction, we vary n from 20 to 120 in steps of 20, 10 trees are generated, and the average performance over all these trees is reported. For the grid graph, we use square grid graphs, where $\mathcal{V} = \mathcal{H}$. In the grid graph construction, we vary $\mathcal{V}$ from 2 to 12 in steps of 2.

Figure 21 shows the total transient energy consumption of the following schemes: degree-based heuristic (degree-based), contiguous link scheduling (contiguous), and compact wakeup scheduling (compact). In both the tree and grid topologies, the transient energy consumption increases as the number of nodes increases. The energy consumption in the compact wakeup scheduling is the smallest among the three schemes, for the frequency of state transitions is minimized in the scheduling. As shown in Figure 21(a), compact wakeup scheduling reduces the energy consumption significantly by approximately 50% as compared to that in the degree-based heuristic and about 35% as compared to that in the contiguous link scheduling.

Figure 22 shows the total waiting period increases as the number of nodes increases in the degree-based heuristic and contiguous link scheduling, while the waiting period is zero in the compact wakeup scheduling. With smaller waiting periods, it would be faster for nodes to gather the information from their neighbors, thus reducing network delay.

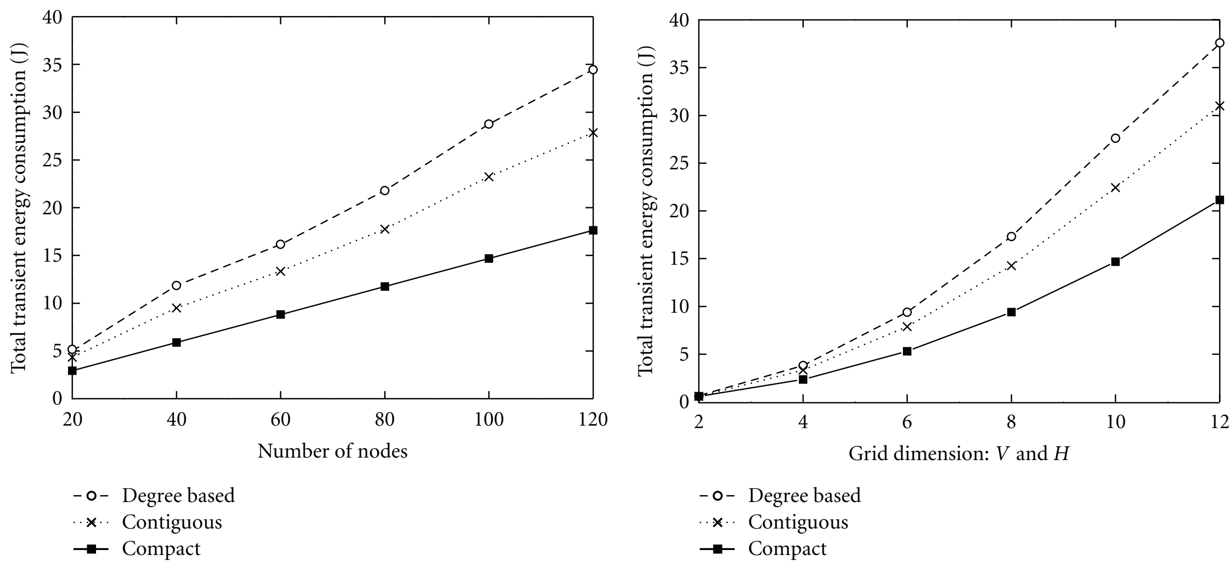
We summarize observations from the simulation results as follows. (1) The waiting period of trees and grid graphs with valid compact wakeup scheduling is zero. (2) Compact wakeup scheduling can significantly reduce network delay and energy consumption.

6. Conclusion and Future Work

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

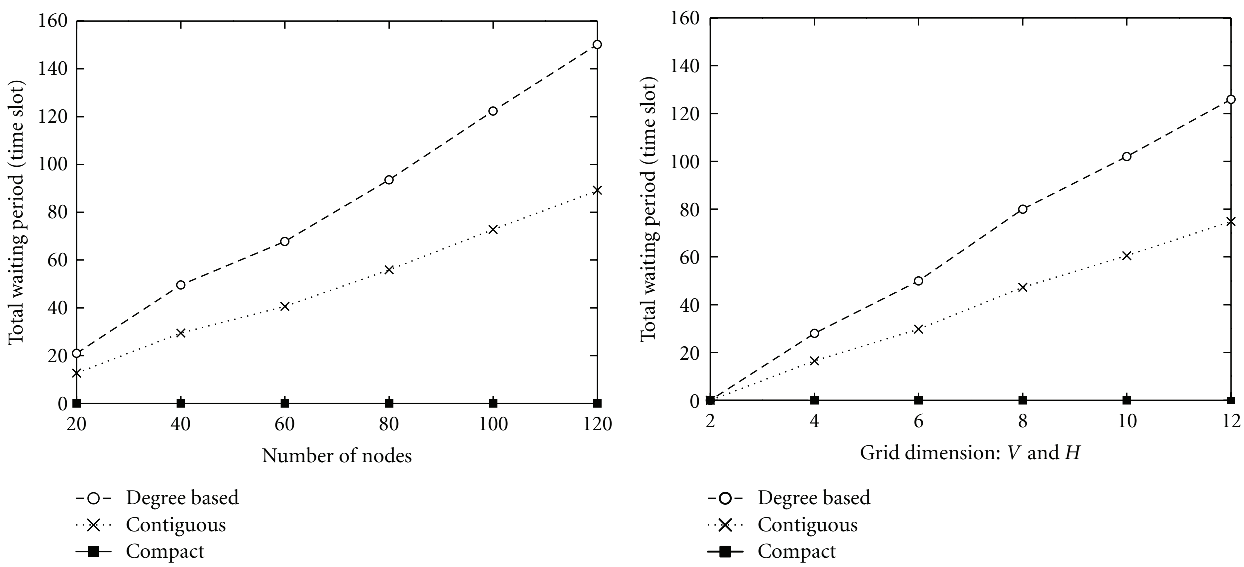
Input: A $\mathcal{V} \times \mathcal{H}$ grid graph G $(3 \le \mathcal{V} \le \mathcal{H})$.
Output: A valid compact wakeup scheduling.
(1) Use **Algorithm** 4 to obtain a valid interval edge-coloring with $\chi_{cw}(G)$ colors for G.
(2) **for** $k = 1$ to $\chi_{cw}(G)$ **do**
(3) Map color k to two consecutive time slots $\{2k-1, 2k\}$.
(4) Use **Algorithm** 1 to determine a valid direction of transmission assignment for time slot $2k-1$.
(5) Reverse the direction of transmission along each edge to obtain the other assignment for time slot $2k$.
(6) **end for**

Algorithm 5: Compact wakeup scheduling of a grid graph.



(a) Tree (b) Grid Graph

Figure 21: Transient energy consumption.



(a) Tree (b) Grid Graph

Figure 22: Waiting period.

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 grid graphs, we point out all the possible coloring patterns and give the lower bound as well as the upper bound of the compact wakeup scheduling. In the process of time slot assignments, both the hidden terminal and exposed terminal problems can be avoided. The simulation results corroborate the theoretical analysis and show the efficiency of compact wakeup scheduling.

In our future work, we will consider the heterogeneous network model and try to obtain efficient algorithms for other kinds of network topologies with valid compact wakeup schedulings. Another challenging topic is to find the scheduling with the minimum waiting period if a valid compact wakeup scheduling does not exist for a given topology.

Acknowledgments

This work is supported in part by Grants PolyU 5236/06E, PolyU 5243/08E, A-PJ16, and 1-ZV5N.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM '02)*, June 2002.

[2] T. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *:Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys '03)*, November 2003.

[3] M. A. Ameen, S. M. R. Islam, and K. Kwak, "Energy saving mechanisms for MAC protocols in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2010, Article ID 163413, 16 pages, 2010.

[4] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '08)*, pp. 53–62, May 2008.

[5] I. Bekmezci and F. Alagoz, "Energy efficient, delay sensitive, fault tolerant wireless sensor network for military monitoring," *International Journal of Distributed Sensor Networks*, vol. 5, no. 6, pp. 729–747, 2009.

[6] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 181–192, November 2003.

[7] O. Chipara, C. Lu, and J. Stankovic, "Dynamic conflict-free query scheduling for wireless sensor networks," in *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP '06)*, pp. 321–331, November 2006.

[8] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '06)*, pp. 190–201, May 2006.

[9] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, (MOBIHOC '06)*, pp. 322–333, May 2006.

[10] W. Wang, Y. Wang, X. Y. Li, W. Z. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MOBICOM '06)*, pp. 262–273, September 2006.

[11] A. Wang, S. Cho, C. Sodini, and A. Chandrakasan, "Energy efficient modulation and MAC for asymmetric RF microsensor systems," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '01)*, 2001.

[12] Sentilla Corporation, "Tmote Sky Datasheet," http://www.sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf.

[13] J. Ma, W. Lou, Y. Wu, X. Y. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *Proceedings of the 28th Conference on Computer Communications (INFOCOM '09)*, pp. 630–638, April 2009.

[14] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 69–78, April 2007.

[15] W. Song, R. Huang, B. Shirazi, and B. LaHusen, "TreeMAC: localized TDMA MAC protocol for real-time high-datarate sensor networks," in *Proceedings of the IEEE Pervasive Computing and Communication Conference (PerCom '09)*, 2009.

[16] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: an energy-efficient predictive-wakeup MAC protocol for wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '11)*, 2011.

[17] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166–177, 1993.

[18] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: distributed edge coloring revisited," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '05)*, 2005.

[19] Y. Wu, X. Y. Li, Y. Liu, and W. Lou, "Energy-efficient wakeup scheduling for data collection and aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 275–287, 2009.

[20] A. S. Asratian and R. R. Kamalian, "Interval coloring of the edges of a graph," *Applied Mathematics*, vol. 5, pp. 25–34, 1987 (Russian).

[21] 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.

[22] S. V. Sevastjanov, "On interval colorability of a bipartite graph," *Metody Diskretnogo Analiza*, vol. 50, pp. 61–72, 1990.

[23] M. Kubale, "Interval edge coloring of a graph with forbidden colors," *Discrete Mathematics*, vol. 121, no. 1–3, pp. 135–143, 1993.

[24] K. Giaro, *Compact task scheduling on dedicated processors with no waiting periods*, Ph.D. dissertation, Technical University of Gdansk, ETI Faculty, Gdansk, Poland, 1999.

[25] K. Giaro and M. Kubale, “Consecutive edge-colorings of complete and incomplete cartesian products of graphs,” *Congressus Numerantium*, vol. 128, pp. 143–149, 1997.

[26] K. Giaro and M. Kubale, “Compact scheduling of zero-one time operations in multi-stage systems,” *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 95–103, 2004.

[27] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys ’04)*, pp. 95–107, November 2004.

[28] R. E. Best, *Phase-Locked Loops: Design, Simulation and Applications*, McGraw-Hill, New York, NY, USA, 2003.

[29] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: a media access protocol for wireless LAN’s,” in *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM ’94)*, 1994.

[30] J. Misra and D. Gries, “A constructive proof of Vizing’s theorem,” *Information Processing Letters*, vol. 41, no. 3, pp. 131–133, 1992.

[31] S. Shakkottai, R. Srikant, and N. Shroff, “Unreliable sensor grids: coverage, connectivity and diameter,” in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM ’03)*, pp. 1073–1083, April 2003.

[32] V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, “Design of surveillance sensor grids with a lifetime constraint,” in *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN ’04)*, 2004.

[33] C. Zhang, J. Kurose, Y. Liu, D. Towsley, and M. Zink, “A distributed algorithm for joint sensing and routing in wireless networks with non-steerable Directional Antennas,” in *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP ’06)*, pp. 218–227, November 2006.

[34] D. Musiani, K. Lin, and T. S. Rosing, “An active sensing platform for wireless structural health monitoring,” in *Proceedings of the 6th IEEE International Symposium on Information Processing in Sensor Networks (IPSN ’07)*, pp. 390–399, April 2007.

[35] S. Bapat, V. Kulathumani, and A. Arora, “Analyzing the yield of exscal, a large-scale wireless sensor network experiment,” in *Proceedings of the 13th IEEE International Conference on Network Protocols (ICNP ’05)*, pp. 53–62, November 2005.