This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks

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

Yanwei Wu, Xiang-Yang Li
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
ywu24@iit.edu, xli@cs.iit.edu

Guihai Chen
State Key Lab of Novel Software Technology
Nanjing University
Nanjing, P. R. China
gchen@nju.edu.cn

***Abstract*—Sleep scheduling is a widely used mechanism in wireless sensor networks (WSNs) to reduce the energy consumption since it can save the energy wastage caused by the idle listening state. In a traditional sleep scheduling, however, sensors have to start up numerous times in a period, and thus consume extra energy due to the state transitions. The objective of this paper is to design an energy efficient sleep scheduling for low data-rate WSNs, where sensors not only consume different amounts of energy in different states (transmit, receive, idle and sleep), but also consume energy for state transitions. We use TDMA as the MAC layer protocol, because it has the advantages of avoiding collisions, idle listening and overhearing. We first propose a novel interference-free TDMA sleep scheduling problem called contiguous link scheduling, which assigns sensors with consecutive time slots to reduce the frequency of state transitions. To tackle this problem, we then present efficient centralized and distributed algorithms that use time slots at most a constant factor of the optimum. The simulation studies corroborate the theoretical results, and show the efficiency of our proposed algorithms.**

Keywords: energy efficient algorithms, sleep scheduling, wireless sensor networks.

I. Introduction

Wireless sensor networks (WSNs) consist of a large number of wireless sensor nodes that organize themselves into multi-hop radio networks. The sensor nodes are typically equipped by power-constrained batteries, which are often difficult and expensive to be replaced once the nodes are deployed. Therefore, it is a critical consideration on reducing the power consumption in the network design.

Previous work [1], [2] has shown that the idle listening state is the major source of energy wastage. In fact, it can consume almost the same amount of energy as required for receiving. Therefore, nodes are generally scheduled to sleep when the radio modules are not in use [3]. After the sleep scheduling, nodes could operate in a low duty cycle mode that they periodically start up to check the channel for activity. Keshavarzian et al. [4] analyzed different sleep scheduling schemes and proposed a scheduling method that can decrease the end-to-end overall delay. This method did not, however, provide an interference-free scheduling, in which every node can start up and transmit or receive its messages without interference during the assigned time slots. One popular approach to avoid interference is to adopt the time division multiple access (TDMA) MAC protocols, which can directly support low duty cycle operations and has the natural advantages of having no contention-introduced overhead and collisions [1]. Moreover, TDMA can guarantee a deterministic delay bound. Thus, we are interested in designing an efficient TDMA sleep scheduling for WSNs.

TDMA protocols divide time into slots, which are allocated to sensor nodes that can turn on the radio during the assigned time slots, and turn off the radio when not transmitting or receiving in the sleep scheduling. In order to be interference-free, a simple approach is to assign each communication link a time slot, and thus, the number of time slots is equal to the number of communication links of the network. This scheme requires much more time slots than necessary, which increases the delay and reduces the channel utilization significantly. This is because multi-hop networks are able to make space reuse in the shared channel, and multiple transmissions can be scheduled in one time slot without any interference. TDMA link scheduling attempts to minimize the number of time slots assigned while producing an interference-free link scheduling, and it has been shown that the problem is NP-complete [5], [6]. Several approximate algorithms have been proposed in the link scheduling problem [7]–[10]. However, if the TDMA link scheduling is used as the startup mechanism in the sleep scheduling, a node may start up numerous times 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 that if the packets are small [11]. Consequently, the transient energy consumption during the startup process can be higher than the energy during the actual transmission. If a sensor node starts up too frequently, it not only needs extra time, but also costs extra energy for the state transition. Therefore, the state transition, e.g., from the sleep state to the active state, should be considered for an energy efficient TDMA sleep scheduling in WSNs.

In this paper, we use a new energy model, where the energy consumption of the state transition is considered. We propose a novel interference-free TDMA sleep scheduling problem called *contiguous link scheduling* to reduce the frequency of state transitions. In the scheduling, links incident to one node are scheduled together to obtain consecutive time slots so that nodes can only start up once to monitor the channel in one scheduling period T. Especially, if the topology is a tree, a node only needs to start up twice in a period, once for receiving data from its children nodes and once for transmitting its data to its parent node.

The main contributions of this paper are summarized as follows: (1) We address the scheduling problem in a new energy model, which is closer to realistic sensors. (2) We propose the contiguous link scheduling problem in WSNs, and prove it to be NP-complete. (3) We present centralized and distributed algorithms that have theoretical performance bound to the optimum of the problem. (4) We develop simulations to show the efficiency of the proposed algorithms.

The remainder of this paper is organized as follows. Section II reviews the related works. Section III describes the system model and then formulates the contiguous link scheduling problem. Section IV presents the centralized algorithm and Section V presents the distributed algorithm for the contiguous link scheduling. Section VI describes and analyzes the simulation results for the proposed algorithms. Section VII concludes the paper.

II. Related Work

Several approximate algorithms have been proposed in the TDMA scheduling problem, including broadcast scheduling [12]–[14] and link scheduling [7]–[10]. Broadcast and link scheduling are time slot assignments to nodes and links, respectively. Ramaswami and Parhi [12] presented an efficient and interference-free centralized and distributed broadcast scheduling in a multi-hop packet radio network. In [13], the broadcast scheduling problem was modeled as a distance-2 coloring problem, and Krumke et al. proposed approximation heuristic algorithms for various geometric graphs. In [14], Ngo et al. presented a centralized genetic-fix algorithm to reduce the search space based on a within-two-hop matrix. However, the performance of broadcast scheduling is worse than link scheduling in WSNs, especially in terms of energy conservation. In the broadcast scheduling, when a node wants to transmit, all the neighbors have to turn on their radio and start up, no matter whether they are the intended receiver or not. In contrast, only the intended receiver needs to start up in the link scheduling. Ramanathan and Lloyd [7] considered both the tree networks and arbitrary networks, and the performance of the proposed algorithms is bounded by the thickness of a network. In [8], Gandham et al. proposed a link scheduling algorithm involving two phases. In the first phase, a valid edge coloring is obtained in a distribution fashion. In the second phase, each color is mapped to a unique time slot, and the hidden terminal and the exposed terminal problems are 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. In [9], Wang et al. proposed both centralized and distributed algorithms with performance guarantee to obtain a good interference-free link scheduling to maximize the throughput of the network. In the algorithms, 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 interference to the already-scheduled sensors. Djukic and Valaee [10] proposed an efficient min-max delay scheduling method to find schedules with minimum round trip delay in the link scheduling. The previous studies in the TDMA scheduling did not consider the energy consumption of radio in the state transition. Compared to broadcast scheduling and link scheduling, the contiguous link scheduling could reduce the frequency that sensor nodes start up, and thus achieve better energy efficiency.

III. System Model and Problem Formulation

In this section, we present the system model consisting of a network model, an interference model and an energy model, then we formulate the contiguous link scheduling problem and prove it to be NP-complete.

A. System Description

Network Model. We assume that a WSN has n static sensor nodes, which are all equipped with single omni-directional antennas, and there exists a sink node to collect the data from other sensor nodes. With the assumption that all the sensors have the same communication range r, the network can be represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and E 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 transmission range of v_i. In a directed graph, the edge e is called incident from v_i, and incident to v_j.

Two types of network topologies for data collection and aggregation are discussed in this paper, data gathering tree and directed acyclic graph (DAG). A data gathering tree is a tree routed at a sink node, where each intermediate node collects the data from its children nodes and then forwards the data to its parent node. A DAG is a graph with no directed cycles, that is, there is no path that starts and ends at the same vertex. The depth of a vertex in a DAG is the length of the longest path from that vertex to a sink.

Interference Model. In wireless networks, the packets transmitted by a node may be received by all the nodes within its transmission range. Therefore, interferences may occur among these nodes due to the broadcast nature of the wireless medium. There are two types of interferences: primary interference and secondary interference [7]. 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 occurs when a node tuned to a particular transmitter is also within the transmission range of another transmission intended for other nodes. Both primary interference and secondary interference are considered in this paper.

The interference between two links in the network depends on the interference model, and we use the protocol model [15], [16] in this paper. In the protocol model, each node v_i has a

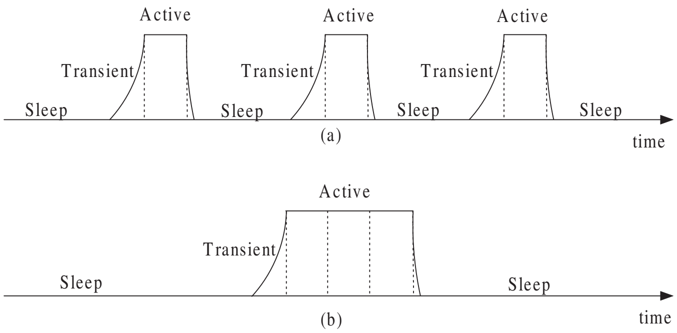


Fig. 1. The energy model: (a) Before active time slots merged, (b) After active time slots merged.

fixed transmission range r and an interference range R, where $R > r$. We denote the ratio between the transmission range and the interference range as $\gamma = \frac{R}{r}$. In practice, $2 \leq \gamma \leq 4$. A transmission from v_i to v_j is successful if any node v_k located within a distance R from v_j is not transmitting.

TABLE I
TIME AND POWER CONSUMPTION IN THE STARTUP PROCESS FOR A MICA2 MOTE WITH A CC1000 TRANSCEIVER

Operation process	**Time**	**Power consumption**
Sleep	—	$90\mu W$
Radio initialization	$0.35ms$	$18mW$
Turn on Radio	$1.50ms$	$3mW$
Switch to RX/TX state	$0.25ms$	$45mW$
Receive 1 byte	$0.416ms$	$45mW$
Transmit 1 byte	$0.416ms$	$60mW$

Energy Model. In B-MAC [17], Polastre et al. presented the energy consumption of sampling the channel in low-power listening (LPL) on a Mica2 mote. 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 receive/transmit state. The startup process is slow due to the feedback loop in the phase-locked loop (PLL) [18], and a typical setting time of the PLL-based frequency synthesizer is on the order of milliseconds. The startup time and energy consumption in the startup process can also be found in the channel polling in [19]. Table I lists the time and power consumption in the startup process for a Mica2 mote with a CC1000 transceiver. We can see that the time to activate a sensor is $2.1ms$, and the energy consumption is about $22\mu J$.

Our energy model is similar to the one used in [20]. In our model, we assume that each node operates in three states: active state (transmit, receive and listen), sleep state, and transient state (state transition). The energy consumption of sensor nodes in the sleep state is much less than the consumption in the active state, and a significant energy saving can be achieved if the sleep state is employed during the periods of inactivity. 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 energy model is illustrated in Fig. 1 (a), and there is a significant energy consumption and time overhead when the sensor's radio powers on and off.

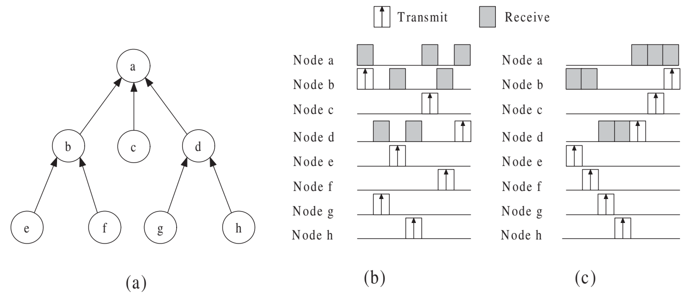


Fig. 2. Link scheduling and contiguous link scheduling: (a) Network topology, (b) Link scheduling, (c) Contiguous link scheduling

Fig. 1 (b) shows that merging the sensor's active time slots together can reduce the startup frequency so as to save both energy and time, which benefits the duty cycle network design.

B. Problem Formulation

In a TDMA sleep scheduling, each link l_{ij} is assigned a time slot, in which both sender node v_i and receiver node v_j should start up to communicate. After the allocated time slot, nodes v_i and v_j change to sleep. When using the traditional link scheduling algorithms (e.g. [9], called *degree-based heuristic* in this paper) which schedule the communication links one by one, node v_j may start up w_j times to monitor the channel in a period T, where w_j is the number of neighbors. As addressed before, the frequent startup would consume a large amount of extra energy and time. A reasonable design is to assign consecutive time slots to all directed links incident to the same node, and then a node only needs to start up once to receive all the packets from its neighbors. We refer to such an interference-free scheduling as the *contiguous link scheduling*. A contiguous link scheduling is said to be *valid* if all the links incident to one node are assigned consecutive time slots.

Fig. 2 shows a sample of the contiguous link scheduling. In Fig. 2 (a), the given network is a data gathering tree routed at node a, in which any two links interfere with each other. Fig. 2 (b) shows an interference-free link scheduling, where a node starts up numerous times in a period. Fig. 2 (c) shows the contiguous link scheduling that a node can start up only once for receiving data from its neighbors. Note that the contiguous link scheduling can be applied to both a tree topology and a mesh topology, such as DAG. Especially, if the network is a data gathering tree where each node only has one parent, a node just needs to start up twice at most in a period: once for receiving data from its children nodes and once for transmitting its data to its parent node.

An *interval vertex coloring* [21], [22] is an assignment of a set $S(i)$ of w_i consecutive colors to each node v_i in such a way that $S(i) \cap S(j) = \emptyset$ for any two adjacent nodes v_i and v_j. In the following part of this section, we show that a valid contiguous link scheduling can be obtained by the interval vertex coloring in the merged conflict graph, and we prove that the contiguous link scheduling problem is NP-complete.

Merged Conflict Graph. Given an interference model, the interference of the links in the communication graph $G = (V, E)$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

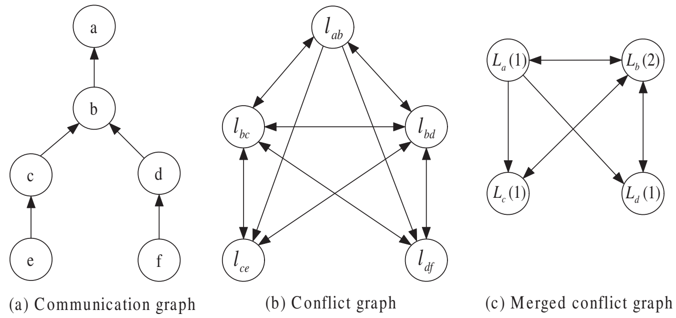


Fig. 3. Communication graph and corresponding conflict graphs

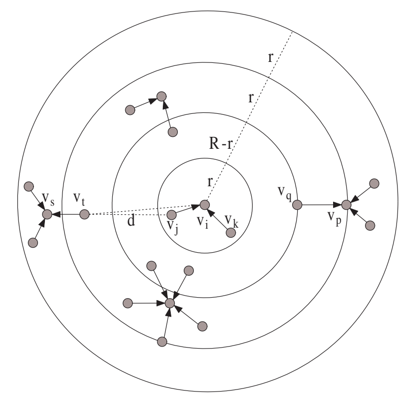


Fig. 4. Interference of links in the protocol model

can be represented as a conflict graph G_c [16]. Corresponding to each directed link between v_i and v_j in G, the conflict graph contains a vertex (denoted by l_{ij}), and there is an edge between vertices l_{ij} and l_{pq} in the conflict graph if l_{ij} interferes with l_{pq} in the network. Notice that the conflict graph in the protocol model is a directed graph, since the links may not interfere with each other. In [9], the link scheduling problem is modeled as the vertex coloring in the conflict graph. Different from the conflict graph, we propose a *merged conflict graph* G_{mc} to model the contiguous link scheduling problem. The w_i directed links incident to the same node in G correspond to a vertex in G_{mc}, and there is an edge between any two vertices in G_{mc} if and only if at least one pair of the corresponding links in G interfere with each other. In G_{mc}, we use $L_{v_i}(w_i)$ to denote the corresponding w_i links incident to node v_i, and w_i is the weight of vertex $L_{v_i}(w_i)$. For example, the links l_{bc} and l_{bd} in Fig. 3 (b) correspond to $L_b(2)$ in Fig. 3 (c). For the sample communication graph shown in Figs 3 (a), if the interference range is two hops, the corresponding conflict graph and merged conflict graph are shown in Figs 3 (b) and 3 (c), respectively. In Fig. 3 (c), we can see that the number of vertices in G_{mc} is equal to the number of receiving nodes in G, and w_i is the number of links directed to the same node. Obviously, we could obtain a valid contiguous link scheduling by the interval vertex coloring in the merged conflict graph.

Theorem 1. *The contiguous link scheduling problem is NP-complete.*

Proof: The problem is clearly in NP since an assignment can be verified in polynomial time.

We transform the interval vertex coloring to the contiguous link scheduling, and the interval vertex coloring problem is proved to be NP-complete [22]. Suppose that each vertex v_i in a weighted graph G_w has a weight of w_i, and the number of adjacent vertices of each vertex v_i is w'_i, which is no less than w_i. We construct a communication graph G' by replacing each vertex v_i in G_w with a node v'_i, and each node v'_i randomly chooses w_i nodes as adjacent nodes (or neighbors) in the corresponding w'_i adjacent vertices in G_w. Then, each node v'_i in G' has w_i incident links. The remaining $w'_i - w_i$ adjacent vertices of v_i in G_w can be seen as interference in the communication graph. This transformation can be performed in polynomial time and we could obtain a contiguous link scheduling in G' by an interval vertex coloring in G_w, therefore, the problem is NP-complete. ■

IV. Centralized Algorithms

In this section, we propose the centralized contiguous link scheduling algorithms, centralized scheduling and centralized scheduling with spatial reuse (recursive backtracking and minimum conflicts heuristic).

A. Centralized Scheduling

We first study a centralized algorithm to the contiguous link scheduling problem. In stead of scheduling a time slot individually for each communication link, each node v_i will be assigned w_i consecutive time slots, where w_i is the number of links incident to v_i. Each node can only start up once to receive all the data from its neighbors. In the centralized scheduling, we color the nodes in the decreasing order of their weight. Since each node v_i is assigned the smallest w_i consecutive time slots which are not assigned to other nodes that interfere with v_i in the merged conflict graph G_{mc}, our scheduling algorithm is interference-free. The centralized scheduling is described in Algorithm 1.

We assume $R/r = \gamma$, where r and R are the transmission and interference range of node v_i, respectively. We use $D(v_i, x)$ to denote the disk centered at node v_i with radius x, and $\| L(v_i) - L(v_j) \|$ to denote the distance between nodes v_i and v_j.

Lemma 1. *Let $I(e)$ be the links that interfere with a link e in the corresponding weighted conflict graph G_{mc} under the protocol model, then at least $|I(e)|/C_1$ time slots are needed to schedule all links in $I(e)$, where $C_1 = \frac{9(\gamma+1)^2}{(\gamma-1)^2}$ and $|I(e)|$ is the number of links in $I(e)$.*

Proof: If a node v_j can communicate with node v_i, then node v_j must be in $D(v_i, r)$. Fig. 4 shows that, if a link incident to v_p interferes with l_{ji} in G_{mc}, v_p must be in $D(v_i, R + r)$, and the link must be in $D(v_i, R + 2r)$, because there is at least one adjacent node of v_p in $D(v_i, R)$ to interfere with v_i, such as v_q. We observe that the distance between two nodes transmitting simultaneously without interference in G_{mc} should be at least $R - r$. For example, if l_{ji} and l_{ts} are interference-free, the distance between v_i and v_t is larger

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Algorithm 1 Centralized scheduling (Centralized)

Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.
1: Construct the merged conflict graph G_{mc}, and initialize an empty stack S.
2: Push the vertices in G_{mc} in the non-decreasing order of weight w_i to the stack S.
3: **while** S is not empty **do**
4: Pop the vertices in S sequentially and assign each vertex $L_{v_i}(w_i)$ the smallest w_i consecutive time slots for node v_i to receive, which are not yet assigned to any of its neighbors in G_{mc}.
5: Schedule the w_i time slots for transmitting sequentially for each adjacent node of node v_i in G.

than R, and the distance between node v_j and v_t should be $d \geqslant \| L(v_i) - L(v_t) \| - \| L(v_i) - L(v_j) \| \geqslant R - r$. There are at most $C_1 = \frac{\pi[R+2r+0.5(R-r)]^2}{\pi[0.5(R-r)]^2} = \frac{9(\gamma+1)^2}{(\gamma-1)^2}$ links with radius $(R-r)/2$ that can be placed in $D(v_i, R+2r)$. Thus, there exists a link set with the size at least $|I(e)|/C_1$ such that each pair of links in the set interferes with each other. Therefore, at least $|I(e)|/C_1$ time slots are needed to schedule all links in $I(e)$. ■

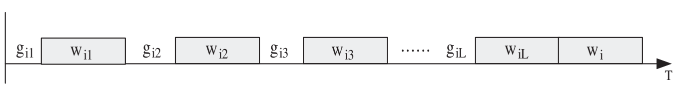


Fig. 5. The time slots that conflict with v_i in G_{mc} when v_i is scheduled

Theorem 2. *The number of time slots used by Algorithm 1 is at most a constant factor of the optimum.*

Proof: Suppose that v_i is the sensor node to be scheduled in the last w_i time slots, and all the other sensors have already been scheduled, as shown in Fig. 5. In the figure, w_{i_l} $(1 \leqslant l \leqslant L)$ is the number of consecutive time slots occupied by the links incident to node v_{i_l} that interfere with the links incident to v_i in the merged conflict graph G_{mc}, and $g_{i_1}, g_{i_2}, \cdots, g_{i_L}$ represent the gaps occupied by other non-conflicting links. The links are scheduled in the non-increasing order of weight, so w_{i_l} $(1 \leqslant l \leqslant L)$ is not smaller than w_i. Since each vertex v_i is assigned the smallest w_i consecutive time slots, which do not interfere with v_i, g_{i_l} $(1 \leqslant l \leqslant L)$ is smaller than w_i. The total number of time slots used by our scheduling is

$$\begin{aligned} T = \sum_{l=1}^{L} g_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i &< L \cdot w_i + \sum_{l=1}^{L} w_{i_l} + w_i \\ &\leqslant 2(\sum_{l=1}^{L} w_{i_l} + w_i) \end{aligned}$$

The number of links that interfere with the links incident to v_i in G_{mc} is $\sum_{l=1}^{L} w_{i_l} + w_i$. We denote T_{opt} as the minimum number of time slots used by any scheduling, i.e., the minimum scheduling period. From Lemma 1, we know that $T_{opt} \geqslant \frac{\sum_{l=1}^{L} w_{i_l} + w_i}{C_1}$. Then, we get $T \leqslant 2C_1 \cdot T_{opt} = C \cdot T_{opt}$, where $C = 2C_1$. ■

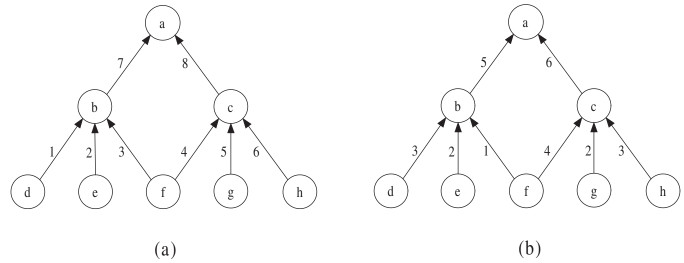


Fig. 6. Contiguous link scheduling: (a) Centralized scheduling, (b) Centralized scheduling with spatial reuse

B. Centralized Scheduling with Spatial Reuse

In the contiguous link scheduling, it is possible to assign the vertices $L_{v_i}(w_i)$ and $L_{v_j}(w_j)$ the same time slots when $L_{v_i}(w_i)$ interferes with $L_{v_j}(w_j)$ in G_{mc}, because not every link in $L_{v_i}(w_i)$ interferes with every link in $L_{v_j}(w_j)$. A sample is illustrated in Fig. 6. The given network is a directed acyclic graph (DAG), and the interference range of each node is two hops. Since $L_a(2)$, $L_b(3)$ and $L_c(3)$ interfere with each other in the merged conflict graph, and the time slots assigned using Algorithm 1 are shown in Fig. 6 (a). We could also get a valid contiguous link scheduling shown in Fig. 6 (b), which uses fewer time slots. Based on the observation, we propose the centralized scheduling with spatial reuse to improve the centralized scheduling by re-arranging the time slots in an interference matrix.

Definition 1. An *interference matrix* of node v_i is an $m \times n$ matrix $M = (m_{j,k})_{m \times n}$ $(1 \leqslant j \leqslant m, 1 \leqslant k \leqslant n)$ that shows whether a time slot could be assigned to a link incident to v_i without interference. In the matrix M,

$$m_{j,k} = \begin{cases} - : link\ l_k\ could\ not\ use\ time\ slot\ t_j\ (interference) \\ 0 : link\ l_k\ could\ use\ time\ slot\ t_j\ (interference\text{-}free) \\ 1 : link\ l_k\ selects\ time\ slot\ t_j\ (selection) \end{cases}$$

where "–" denotes interference with the already-scheduled links, "0" denotes interference-free and "1" denotes the selected (or assigned) time slot for the link (selection).

Note that the number of columns n is equal to the number of links incident to node v_i, and the number of rows m is the number of time slots assigned in the scheduling. A sample of the interference matrix is shown below. In the matrix M_0, node v_i has 5 incident links $\{l_1, l_2, l_3, l_4, l_5\}$, and the time slots which have been used by other links that interfere with links l_1, l_2, l_3, l_4, l_5 are $\{t_1, t_2, t_6\}$, $\{t_2, t_4, t_7\}$, $\{t_1, t_2, t_5\}$, $\{t_2, t_3, t_6\}$ and $\{t_2\}$, respectively.

$$M_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{array} \begin{array}{c} \begin{array}{ccccc} l_1 & l_2 & l_3 & l_4 & l_5 \end{array} \\ \begin{pmatrix} - & 0 & - & 0 & 0 \\ - & - & - & - & - \\ 0 & 0 & 0 & - & 0 \\ 0 & - & 0 & 0 & 0 \\ 0 & 0 & - & 0 & 0 \\ - & 0 & 0 & - & 0 \\ 0 & - & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Definition 2. An assignment in the interference matrix $M = (m_{j,k})_{m\times n}$ of node v_i is said to be *valid* if there is only one "1" in each row and each column, and there are n consecutive rows that have "1" in each row in the matrix. The links incident to v_i could be scheduled consecutive time slots by obtaining a valid assignment in the interference matrix.

Definition 3. An *interference submatrix* is an $n \times n$ matrix $M' = (m_{j,k})_{n\times n}$, which consists of n consecutive rows in the interference matrix M.

Algorithm 2 Centralized scheduling with spatial reuse

Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.

Construct the merged conflict graph G_{mc}, construct an interference matrix for each vertex $L_{v_i}(w_i)$, and initialize an empty stack S.
Push the vertices in G_{mc} in the non-decreasing order of weight w_i to the stack S.
while S is not empty **do**
    Pop the vertices in S sequentially and assign each vertex $L_{v_i}(w_i)$ the smallest w_i consecutive time slots, using Algorithm 3 or Algorithm 4 in the interference matrix M to find a solution.
    v_i broadcasts the time slot assignments to the nodes whose incident links interfere with the links incident to v_i, then the nodes update their interference matrixes.

Algorithm 2 describes the centralized scheduling with spatial reuse. In order to reduce the time slots assigned, the *recursive backtracking* algorithm is used to make sure that a node is assigned the smallest available consecutive time slots. In the algorithm, when a node v_i is assigned time slots, it will broadcast the information to all the nodes whose incident links interfere with the links incident to v_i. Then each node in the network would have an interference matrix to indicate the time slots that its incident links could not use. The algorithm first finds the smallest n consecutive rows that have "0" in each row in the interference matrix M, and constructs an interference submatrix M' which consists of the n rows. Then it starts the first selection in the first row, and the second selection in the second row without interference to the selection in the first row. The algorithm continues the selection to the next row until a valid assignment is found. During the process of each selection in a row, there may be several candidates "0", and the selected candidate is referred to as *predecessor*, and the candidates "0" in the next row are referred to as *successors* of the predecessor. If a successor fails in the selection, it then executes the *backtracking procedure*: the algorithm checks whether the next successor of the predecessor satisfies the condition that there is only one "1" in each column. If the successors are exhausted, the algorithm backtracks to the previous predecessor and tries the next successor of the previous predecessor. If there are no more predecessors, the algorithm adds a new time slot interference-free to all the links incident to v_i, and the corresponding interference matrix adds a zero row vector $(0)_{1\times n}$ in the last row. The details of the recursive backtracking algorithm are shown in Algorithm 3.

Algorithm 3 Recursive backtracking

Input: An interference matrix $M = (m_{j,k})_{m\times n}$.
Output: A valid assignment in M.

Construct an interference submatrix M' consisting of the smallest n consecutive rows that have "0" in each row.
Start the first selection in the first row.
Continue the selection in the next row satisfying the condition that there is only one "1" in each column, until a valid assignment is obtained.
if a selection fails to obtain a valid assignment **then**
    Execute the backtracking procedure.
if the recursive backtracking fails **then**
    Add a zero row vector in the last row of M, and delete the first row in the n consecutive rows. Update M, and repeat the recursive backtracking in M.

The recursive backtracking algorithm is a brute-force search algorithm, which is too complex for sensor networks. Therefore, we present a fast algorithm called *minimum conflicts heuristic*, as shown in Algorithm 4.

Definition 4. A *conflict matrix* is an $n\times n$ matrix $M_C = (c_{j,k})_{n\times n}$ that describes the number of conflicts in the interference submatrix $M' = (m_{j,k})_{n\times n}$, and each element $c_{j,k}$ in the matrix is the number of conflicts, which is the sum of the selections "1" in the row and column that include $c_{j,k}$. That is, $c_{j,k} = \sum_{l=1}^{n} m_{l,k} + \sum_{l=1}^{n} m_{j,l} - m_{j,k}$, if $m_{j,k} \neq -$.

The minimum conflicts heuristic algorithm first constructs an interference submatrix M' which consists of n consecutive rows, and then starts with a random initial configuration in M', e.g., with one selection per column or per row. One initial configuration of matrix M_0 is shown in the matrix M'_1. The corresponding conflict matrix of matrix M'_1 is shown in the matrix M_{C_1}. The algorithm then uses a heuristic to determine how to reduce the interference by moving the selection "1" with the largest number of conflicts to the position in the same column where the number of conflicts is minimum. The first step of improvement is shown in the matrix M'_2, and the corresponding conflict matrix of matrix M'_2 is M_{C_2}. It continues to reduce the interference until there is no interference or the initial configuration fails. If the initial configuration fails, it will add a new time slot and continue. Though the algorithm converges much faster, the minimum conflicts heuristic may get stuck on a local optimum, thus, it does not guarantee a solution to find the consecutive time slots which actually exist.

$$M'_1 = \begin{pmatrix} 1 & 0 & 1 & - & 1 \\ 0 & - & 0 & 1 & 0 \\ 0 & 1 & - & 0 & 0 \\ - & 0 & 0 & - & 0 \\ 0 & - & 0 & 0 & 0 \end{pmatrix}, M_{C_1} = \begin{pmatrix} 3 & 4 & 3 & - & 3 \\ 2 & - & 2 & 1 & 2 \\ 2 & 1 & - & 2 & 2 \\ - & 1 & 1 & - & 1 \\ 1 & - & 1 & 1 & 1 \end{pmatrix}$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

$$M_2' = \begin{pmatrix} 0 & 0 & 1 & - & 1 \\ 0 & - & 0 & 1 & 0 \\ 0 & 1 & - & 0 & 0 \\ - & 0 & 0 & - & 0 \\ 1 & - & 0 & 0 & 0 \end{pmatrix}, M_{C_2} = \begin{pmatrix} 3 & 3 & 2 & - & 2 \\ 2 & - & 2 & 1 & 2 \\ 2 & 1 & - & 2 & 2 \\ - & 1 & 1 & - & 1 \\ 1 & - & 2 & 2 & 2 \end{pmatrix}$$

Algorithm 4 Minimum conflicts heuristic

Input: An interference matrix $M = (m_{j,k})_{m\times n}$.
Output: A valid assignment in M.

1: Construct an interference submatrix M' consisting of the smallest n consecutive rows that have "0" in each row.
2: Initialize the matrix M' with a random configuration.
3: Count the number of conflicts in M', and obtain the conflict matrix M_C.
4: Use a heuristic to reduce the interference, moving the selection "1" with the largest number of conflicts to the position in the same column, where the number of conflicts is minimum.
5: **if** the minimum conflicts heuristic fails **then**
6: Add a zero row vector in the last row of M, and delete the first row in the n consecutive rows. Update M, and repeat the minimum conflicts heuristic in M.

As Algorithm 2 reduces the number of time slots compared to Algorithm 1, it also follows a constant bound performance guarantee of Algorithm 1. Note that two vertices that conflict with each other in the merged conflict graph may overlap some time slots in Algorithm 2.

Corollary 1. *The number of time slots used by Algorithm 2 is at most a constant factor of the optimum.*

V. Distributed Algorithms

Wireless sensor networks are self-organized and distributed, centralized algorithms could not be used without a predefined leader. Therefore, it is necessary to design efficient distributed algorithms. In this section, we propose two distributed algorithms, distributed scheduling and distributed scheduling with efficient delay.

A. Distributed Scheduling

In the distributed scheduling, we use a random order rather than a global decreasing order of the weight, and we assume that there is a contention-based MAC (e.g. S-MAC [1]) available for a node to compete the channel and to obtain an interference-free contiguous link scheduling. The distributed scheduling is simple and efficient so that each sensor node can run the scheduling without extra computation. The distributed scheduling is shown in Algorithm 5.

Theorem 3. *The number of time slots used by Algorithm 5 is at most a constant factor of the optimum.*

Proof: Suppose that node v_i is scheduled in the last w_i time slots, and all the other nodes have already been scheduled. We denote $K = \frac{w_i}{w_{min}}$, where $w_{min} = \min_{1\leqslant l\leqslant L} w_{i_l}$.

Algorithm 5 Distributed scheduling (Distributed)

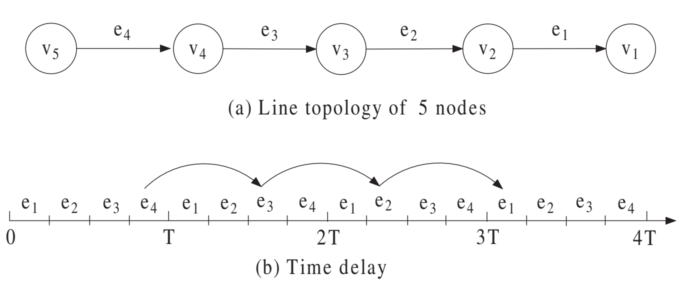
1: Each node v_i monitors and competes for the channel.
2: **if** node v_i obtains the channel **then**
3: v_i assigns the smallest w_i consecutive time slots sequentially to its incident links which do not interfere with the links that have already been scheduled.
4: v_i broadcasts the information to the nodes that are in the interference range of v_i, and these nodes could not transmit in the time slots due to the interference.
5: **else**
6: v_i waits for a random time, then goes to step 1.

Case 1: If K is a constant, the number of time slots used is at most a constant factor of the optimum, since

$$\begin{aligned} T &= \sum_{l=1}^{L} g_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i \qquad < \sum_{l=1}^{L} w_i + \sum_{l=1}^{L} w_{i_l} + w_i \\ &\leqslant \sum_{l=1}^{L} K\cdot w_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i \leqslant (K+1)(\sum_{l=1}^{L} w_{i_l} + w_i) \\ &\leqslant (K+1)C_1 \cdot T_{opt} \end{aligned}$$

Case 2: If K is not a constant, we could divide the time slots assigned before v_i is scheduled into several consecutive groups, and $k_g = \frac{w_{g_i}}{w_{g_{min}}}$ is a constant in each group, where w_{g_i} is the number of time slots assigned to the last node in group g, and $w_{g_{min}}$ is the minimum number of time slots assigned to a node in group g. Let $T_{g_{opt}}$ be the optimum number of time slots in group g, and it must be that $T_{g_{opt}} \leqslant T_{opt}$. Then the number of time slots used in each group is $T_g \leqslant (k_g+1)C_1 \cdot T_{g_{opt}} \leqslant (k_g+1)C_1 \cdot T_{opt}$. Thus, the total number of time slots used is $T \leqslant (K_g+1)N_gC_1 \cdot T_{opt}$, where N_g is the number of groups, and K_g is the largest k_g among the groups. Note that the weight of the nodes in the latter group is always higher order infinite compared to the nodes in the former group, i.e., the ratio of the weight of any node in the former group to the weight of any node in the latter group is 0. Otherwise, we could merge the two groups into a new group g', and the ratio between $w_{g'_i}$ and $w_{g'_{min}}$ in group g' is still a constant. Since the scheduling is in a random order, the number of groups N_g is finite. We define $C = (K_g+1)N_gC_1$, then we get $T \leqslant C \cdot T_{opt}$. ∎

B. Distributed Scheduling with Efficient Delay



(a) Line topology of 5 nodes

(b) Time delay

Fig. 7. TDMA scheduling delay

In the TDMA sleep scheduling, a node stays in the sleep state for most time, and periodically starts up to check for activity. As a forwarding node has to wait until its next-hop neighbor starts up and is ready to receive, the message delivery delay will increase. When packets are forwarded from an incoming link to an outgoing link, they could only be forwarded to the outgoing link in the next period T if the incoming link is scheduled to be active after the outgoing link. This kind of delay will accumulate at every hop in the network, which may lead to a long latency. A sample is illustrated in Fig. 7. The network topology is a line as shown in Fig. 7 (a), and the transmission sequence is $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4$ as shown in Fig. 7 (b). Under this situation, the time delay for a packet transmitting from v_5 to v_1 is almost $3T$. However, if the transmission is $e_4 \rightarrow e_3 \rightarrow e_2 \rightarrow e_1$, v_5 could transmit the data to v_1 in one period T. In order to reduce this delay, we schedule the links from bottom to top, that is, a node with higher depth should be scheduled earlier. Hence, a node v_i can only be scheduled until all the children nodes of v_i are already scheduled. The algorithm is described in Algorithm 6.

Algorithm 6 Distributed scheduling with efficient delay (Distributed-delay)

1: Each node v_i monitors and competes for the channel if all the children nodes of v_i are already scheduled.
2: **if** node v_i obtains the channel **then**
3: v_i assigns the smallest w_i consecutive time slots sequentially to its incident links which do not interfere with the links that have already been scheduled.
4: v_i broadcasts the information to the nodes that are in the interference range of v_i, and these nodes could not transmit in the time slots due to the interference.
5: **else**
6: v_i waits for a random time, then goes to step 1.

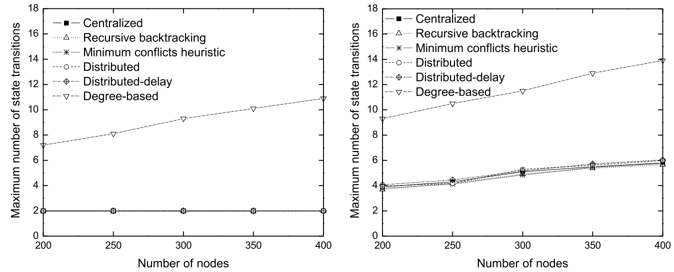
As the links are still scheduled in a random order, the algorithm follows a constant bound performance guarantee of Algorithm 5, that is, the number of time slots used by Algorithm 6 is at most a constant factor of the optimum.

Corollary 2. *The number of time slots used by Algorithm 6 is at most a constant factor of the optimum.*

VI. Simulation Results

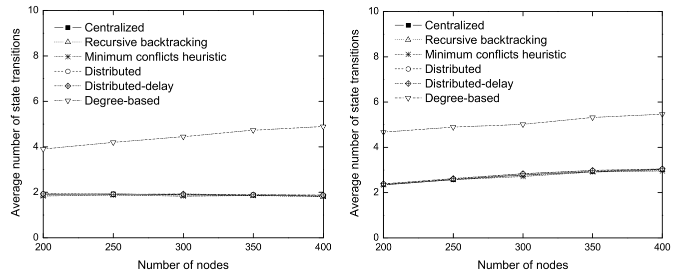
In this section, we study the average-case performance of the proposed centralized and distributed algorithms for the contiguous link scheduling using a simulator built in C++, and we also compare our algorithms with the *degree-based heuristic* in [9]. The performance metrics used in the evaluation are the number of state transitions, the number of time slots assigned, and time delay.

In the simulations, nodes with a transmission range of $15m$ and an interference range of $30m$ are deployed in a square area of $100m \times 100m$. We test the networks when the number of nodes varies from 200 to 400 in steps of 50. We construct a breadth first search (BFS) tree and a directed acyclic graph



(a) BFS tree (b) DAG graph

Fig. 8. Maximum number of state transitions



(a) BFS tree (b) DAG graph

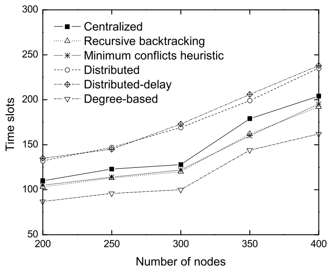
Fig. 9. Average number of state transitions

(DAG) rooted at the sink node as the topologies of the network. For each case, 50 networks are randomly generated, and the average performance over all of these randomly sampled networks is reported.

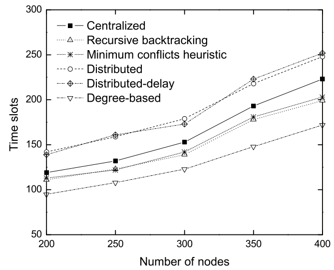
Fig. 8 and Fig. 9 show the maximum and average number of state transitions of the following schemes: centralized, recursive backtracking, minimum conflicts heuristic, distributed, distributed-delay, and degree-based heuristic (degree-based). In the BFS tree model, the maximum number of state transitions is two in the schemes of the contiguous link scheduling, while many nodes need to start up numerous times under the degree-based scheme, which is proportional to the total time slots required by this node. In the DAG model, the number of state transitions of the contiguous link scheduling schemes is much less than the number of the degree-based scheme, and the transient energy cost can be reduced.

The average number of time slots assigned in the scheduling is shown in Fig. 10. In both the BFS tree and DAG model, the number of time slots assigned increases as the number of nodes increases, for the number of interference links increases when the number of nodes increases. In the contiguous link scheduling, the links incident to one node are scheduled together to obtain consecutive time slots to avoid frequent state transitions, and several gaps are formed among the assigned time slots (seen in Fig. 5), which decreases the channel utilization and requires more time slots. Fig. 10 shows that the overhead is not high, and the recursive backtracking scheduling scheme has performance comparable to the degree-based scheme. Although the minimum conflicts heuristic may get stuck on a local optimum, it almost has the same performance compared to recursive backtracking. If the centralized scheduling with spatial reuse is not used, the results would be a little worse,

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

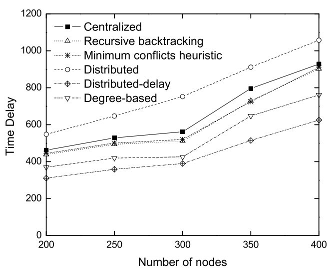


(a) BFS tree

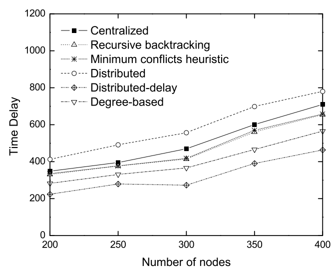


(b) DAG graph

Fig. 10. Average number of time slots assigned



(a) BFS tree



(b) DAG graph

Fig. 11. Average time delay

as shown in the centralized scheduling. The two distributed algorithms have the worst performance, due to the fact that they do not have the global information. The number of time slots in the distributed algorithms is about 1.5 times of the number of time slots in the recursive backtracking scheduling.

Fig. 11 shows the average time delay, and the delay increases as the number of nodes increases in both BFS tree model and DAG model. The distributed scheduling with efficient delay scheme has the best performance, for the links are scheduled from the bottom to the top in the scheduling, which is helpful to reduce the delay.

We summarize observations from the simulation results as follows: (1) The centralized and distributed algorithms proposed can reduce the number of state transitions, and thus achieve better energy efficiency. If the topology is a tree, the nodes can only start up twice in a period. (2) Our proposed distributed algorithms can achieve performance comparable to the centralized algorithms. (3) The distributed scheduling with efficient delay scheme can reduce the network delay.

VII. Conclusion

In this paper, we propose a new interference-free TDMA sleep scheduling problem in WSNs, called contiguous link scheduling. In the scheduling, a sensor node only starts up once to receive all the data from its neighbors, and thus can reduce the energy cost and time overhead in the state transition. Especially, if the topology is a tree, the nodes can only start up twice in one scheduling period. We also propose centralized and distributed algorithms that use time slots at most a constant factor of the optimum. The simulation results corroborate the theoretical analysis, and show the efficiency of our algorithms in terms of the number of state transitions, the number of time slots assigned, and time delay.

Acknowledgment

This work was supported in part by grants A-PH12, PolyU 5236/06E, PolyU 5232/07E, PolyU 5243/08E, HKUST 6169/07E, HKBU 2104/06E, NSF CNS-0832120, NSF CCF-0515088, National Basic Research Program of China (2006CB30300), National High Technology Research and Development Program of China (2007AA01Z180), and China NSF grants (60573131, 60673154, 60721002, 60828003).

References

[1] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2002.

[2] T. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[3] 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 *Proc. of ACM MobiHoc*, 2008.

[4] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. of ACM MobiHoc*, 2006.

[5] E. Arikan, "Some complexity results about packet radio networks," *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 681–685, 1984.

[6] A. Ephremedis and T. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 456–460, 1990.

[7] 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.

[8] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proc. of IEEE INFOCOM*, 2005.

[9] 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.

[10] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proc. of IEEE INFOCOM*, 2007.

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

[12] R. Ramaswami and K. Parhi, "Distributed scheduling of broadcasts in a radio network," in *Proc. of IEEE INFOCOM*, 1989.

[13] S. Krumke, M. Marathe, and S. Ravi, "Models and approximation algorithms for channel assignment in radio networks," *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

[14] C. Y. Ngo and V. O. K. Li, "Centralized broadcast scheduling in packet radio networks via genetic-fix algorithms," *IEEE Transactions on Communications*, vol. 51, no. 9, pp. 1439–1441, 2003.

[15] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[16] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proc. of ACM MobiCom*, 2003.

[17] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[18] R. E. Best, *Phase-locked Loops: Design, Simulation and Applications*. McGraw-Hill, 2003.

[19] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proc. of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[20] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-constrained modulation optimization," *IEEE Transactions on Wireless Communications*, vol. 4, no. 5, pp. 2349–2360, 2005.

[21] D. de Werra and A. Hertz, "Consecutive colorings of graphs," *Mathematical Methods of Operations Research*, vol. 32, no. 1, pp. 1432–2994, 1988.

[22] M. Kubale, "Interval vertex-coloring of a graph with forbidden colors," *Discrete Mathematics*, vol. 74, no. 1-2, pp. 125–136, 1989.