# Energy-Efficient Wake-Up Scheduling for Data Collection and Aggregation

Yanwei Wu, *Member*, *IEEE*, Xiang-Yang Li, *Senior Member*, *IEEE*, YunHao Liu, *Senior Member*, *IEEE*, and Wei Lou

**Abstract**—A sensor in wireless sensor networks (WSNs) periodically produces data as it monitors its vicinity. The basic operation in such a network is the systematic gathering (with or without in-network aggregation) and transmitting of sensed data to a base station for further processing. A key challenging question in WSNs is to schedule nodes' activities to reduce energy consumption. In this paper, we focus on designing energy-efficient protocols for low-data-rate WSNs, where sensors consume different energy in different radio states (transmitting, receiving, listening, sleeping, and being idle) and also consume energy for state transition. We use TDMA as the MAC layer protocol and schedule the sensor nodes with consecutive time slots at different radio states while reducing the number of state transitions. We prove that the energy consumption by our scheduling for homogeneous network is at most twice of the optimum and the timespan of our scheduling is at most a constant times of the optimum. The energy consumption by our scheduling for heterogeneous network is at most $\Theta(\log \frac{R_{\max}}{R_{\min}})$ times of the optimum. We also propose effective algorithms to construct data gathering tree such that the energy consumption and the network throughput is within a constant factor of the optimum. Extensive simulation studies show that our algorithms do considerably reduce energy consumption.

**Index Terms**—Energy consumption, MAC, TDMA, scheduling, routing, WSN.

## 1 Introduction

Wireless sensors are often powered by batteries and have limited computing and memory resources. Because of the limitations due to battery life, sensor nodes are built with power conservation in mind, and generally, spend large amounts of time in a low-power "sleep" mode or processing the sensor data. Wireless sensor networks (WSNs) can operate in an event-driven model or regular continuous monitoring model. Here, we focus on a regular continuous monitoring model, where each sensor will monitor its vicinity and periodically sends its collected information to the sink possibly via the relay of other sensors. A key challenging question in WSNs is to schedule nodes' activities to reduce energy consumption.

A tree $T$ rooted at a sink node is called a *data gathering tree*, if every internal node $v$ collects the data from the sensors that are its children, and then, sends the data (possibly with data aggregation) to its parent node. Depending on the application scenario and the computing power of wireless sensors, a wireless sensor could process the data collected from its children sensors, and then, send the processed data to its parent, which is called data aggregation. When a sensor node simply relays the data from its children directly to its parent, this is called *data collection*. Although for simplicity, we assume that there is only one sink for data gathering (i.e., one tree), all our results hold when there are multiple sinks.

Efficient TDMA scheduling has been extensively studied recently for sensor networks. These previous studies did not consider all possible energy consumption by wireless sensors, especially the energy consumed during wasted listening, and the state transitions (e.g., from idle state to listening state, from sleep state to transmitting state, and so on). Typically, a wireless sensor node $v$ will wake up periodically to sense the environment (using a sensing device) and produce new data at rate $\ell(v)$, to process the sensed data (using a computing component), to send some data to the sink node (by switching the radio to transmitting mode), and to receive (by switching the radio to receiving mode) and process data from other wireless sensor nodes (which may involve data aggregation). After these, a sensor node will go to sleep mode again. Notice that the state transition of the *processor*, the *sensor*, and the *radio* costs nonnegligible energy. Waking up a sensor node takes orders of magnitude more time than putting it into sleep mode. Notice that in most applications, the processor and radio run for a brief period of time followed by a sleep cycle. During the sleep, the current consumption by a sensor is in the microamperes as opposed to milliamperes. This results in very low current draw the majority of the time, and short duration spikes while processing, receiving, and transmitting data. This method extends battery life; however, due to the current surges, it reduces specified battery capacity. Thus, given a required number of time slots for transmitting and receiving, a scheduling should reduce the state transitions to increase the lifetime of a sensor. For example, the ATMega 128L processor of Mica mote sensor takes $4$ ms

- *Y. Wu is with the Department of Computer Science, Minnesota State University, Mankato, 273 Wissink Hall, Mankato, MN 56001. E-mail: yanwei.wu@mnsu.edu*
- *X.-Y. Li is with the Department of Computer Science, Illinois Institute of Technology, 10 West 31st Street, Chicago, IL 60616. E-mail: xli@cs.iit.edu.*
- *Y. Liu is with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong. E-mail: liu@cse.ust.hk.*
- *W. Lou is with the Department of Computing, The Hong Kong Polytechnic University, Hong Hum, Kowloon, Hong Kong SAR. E-mail: csweilou@comp.polyu.edu.hk.*

*Manuscript received 21 Feb. 2008; revised 9 Nov. 2008; accepted 9 Feb. 2009; published online 10 Mar. 2009.*
*Recommended for acceptance by D. Gunopulos.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-02-0070.*
*Digital Object Identifier no. 10.1109/TPDS.2009.45.*

for start-up; the RFM radio (used by *Mica*) takes $12\ \mu$s to switch between sending and receiving while the raw bit time is $25\ \mu$s; and the Chipcon radio (used by newer *Mica2* and *Mica2 dot*) takes $250\ \mu$s to switch between sending and receiving while the raw bit time is $26\ \mu$s.

The main contributions of this paper are as follows. To reduce the energy cost, we design data collection and aggregation methods to minimize the wake-up times in a scheduling period, when the number of transmissions and receptions by a sensor node is fixed. Clearly, the best scenario is that a sensor node only wakes up once and finishes all operations continuously. In a scheduling period, any sensor node needs only to wake up at most *twice* in our protocol: once for continuously receiving all packets from its children nodes and once for sending its own data to its parent node. We first consider the homogeneous WSNs in which all wireless sensor nodes have an *identical* interference range $R_I$. We always assume that $R_I(v) \geq (1+\beta)R_T(v)$ for a constant $\beta > 0$ (in practice, $\beta \simeq 1$). Here, different wireless sensor nodes may have *different* communication range $R_T(v)$. We prove that by using our scheduling, the *total* energy consumption per node for a homogeneous network is at most a constant times of the optimum, both for data collection and data aggregation. Both centralized algorithms and communication-efficient distributed algorithms with at most $O(n)$ messages are proposed to find the scheduling.

We then consider WSNs where different wireless sensor nodes may have *different* interference range $R_I(v)$. We propose communication-efficient scheduling protocols that are also energy-efficient: the energy consumption by our scheduling for a heterogeneous network is at most $\Theta(\log \frac{R_{\max}}{R_{\min}})$ times of the optimum. Here, $R_{\max}$ and $R_{\min}$ are the maximum and the minimum interference range of all nodes, respectively. We design both centralized and distributed algorithms for the scheduling in such a heterogeneous network. We conduct extensive simulations to study the performances of our proposed methods, and we found that they perform better than known algorithms in the literature.

The rest of the paper is organized as follows: In Section 2, we present our network model and the problems to be studied. We provide centralized and distributed algorithms for homogeneous network in Section 3, and for heterogeneous network in Section 4. We report our simulation results that compare the performance of our methods with existing methods in Section 6. We review the related work in Section 7, and conclude our paper in Section 8.

## 2 System Model and Assumption

### 2.1 Network System Models

Assume that there is a set $V = \{v_1, v_2, \ldots, v_n\}$ of $n$ sensors deployed in a two-dimensional region. Node $v_0$ is a special computer that serves as the sink node of the network, i.e., all data will be collected and sent to this node. Our results can be easily extended to the scenario when there are multiple sink nodes in the network. For simplicity, we assume that each wireless node $v_i$ will use a fixed power to communicate with its neighboring sensors, and the physical link $v_iv_j$ is reliable if $v_i$ can communicate with $v_j$. With the fixed transmitting power, the network is modeled as a graph $G = (V, E)$, where $E$ is the set of all possible communication links. We further assume that all communication links have the same capacity. Relaxing this assumption will not affect the correctness of our results as will be seen later. We assume that the fixed power transmission by a node $v_i$ will define an *interference range* $R_I(v_i)$ such that the transmission of node $v_i$ will interfere the reception of any node $v_k$ when $\|v_k - v_i\| \leq R_I(v_i)$.

TABLE 1
Energy Cost Symbols

| Symbol | Meaning | Typical Value |
|---|---|---|
| $P_{tx}$ | Energy consumption in transmitting | 60mW |
| $P_{rcv}$ | Energy consumption in receiving | 45mW |
| $P_{lst}$ | Energy consumption in listening | 45mW |
| $P_{slp}$ | Energy consumption in sleeping | 90$\mu$mW |
| $t_p$ | Time needed to poll channel once | 3ms |
| $r_{v_i}$ | Data packets per period by $v_i$ | Varying |
| $L_{data}$ | Data packet length | 36Bytes |
| $t_B$ | Time to transmit or receive a byte | 416E-6s |
| $T$ | a scheduling period (# of time-slots) | 1s to 60s |
| $t_s$ | slot size | 30 ms |
| $P_s$ | # of packets transmitted in a lot | $\sim 10$ |

The energy efficiency is a major design criterion for WSNs. The main energy consumption of a wireless sensor node is typically from the following operations: transmitting a packet, receiving a packet, listening radio signals, sampling the vicinity, reading sample data from the ADC, reading data from the flash, and writing/erasing data in the flash [1], [2]. In this paper, we mainly focus on the energy cost by the radio. The radio is in any of the four states: *transmitting*, *receiving*, *listening*, and *sleeping*, each of which has different energy consumption (energy consumption per unit time) of $P_{tx}, P_{rcv}, P_{lst}$, and $P_{slp}$, respectively. Table 1 summarizes some typical values of the energy cost for different operations. We also consider the energy $E_{A,B}$ consumed by transiting from one state $A$ to another state $B$ for a sensor and other control units. Typically, the time to restart a sensor node from the sleep mode to active mode is about $4$ ms.

We use a TDMA for scheduling node activities to reduce the energy consumption. We assume that the time is logically divided into slots with slot size $t_s$, and time slots are synchronized among nodes. A *schedule period* $\mathcal{T}$ is composed of $T$ consecutive time slots. The activities of every node is then repeated with period $T$. Assume that a node $v_i$ will produce $r_{v_i}$ data packets per scheduling period $T$. Notice that typically, the maximum data rate supported by an RF transceiver of a sensor node is 40 kbps for Mica and Mica2, and 250 kbps for Micaz. Thus, the *maximum* data that can be transmitted in a time slot are about 150 Bytes for Mica/Mica2 sensor nodes and about 935 Bytes for Micaz sensor node under an ideal situation. Notice that the default data packet size by TinyOS is 36 Bytes. Thus, in one time slot, a node can transmit multiple data packets under the ideal environment. When a node $v_i$ is transmitting packets to a neighboring node $v_j$, some other neighboring nodes that are in the listening state will also consume energy. Therefore, the total energy consumption upon the scenario that node $v_i$ transmits in $L$ slots, while $k$ neighboring nodes

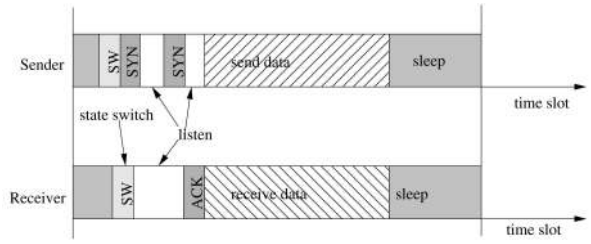



Fig. 1. The synchronization between two nodes.

TABLE 2
Symbol Notations

| Symbol | Meaning |
|---|---|
| $X_{i,S,t}$ | indicator whether node $v_i$ transmitting at time $t$ |
| $X_{i,R,t}$ | indicator whether node $v_i$ receiving at time $t$ |
| $X_{i,P,t}$ | indicator whether node $v_i$ sleeping at time $t$ |
| $X_{i,L,t}$ | indicator whether node $v_i$ listening at time $t$ |
| $E_{P,S}$ | energy consumption from sleeping to transmitting |
| $E_{P,R}$ | energy consumption from sleeping to receiving |
| $E_{P,L}$ | energy consumption from sleeping to listening |

listening is $(P_{tx} + P_{rcv} + k \cdot P_{lst}) \cdot L \cdot t_s$. To minimize the energy consumption, we should schedule the activities of sensor nodes to reduce $k$.

## 2.2 Problem Description

We then describe in detail the problems to be studied in this paper. We divide our studies into two parts: energy-efficient scheduling and data-collection tree construction.

### 2.2.1 Energy-Efficient Scheduling

First, we assume that a tree **T** rooted at the sink is already in place for data collection (or data aggregation). Then, we study how to construct a data collection (or aggregation) tree that can support the highest data rate and reduce the energy consumption. A scheduling of activities for all nodes is to assign each time slot $1 \le t \le T$ to one of the four possible states: transmitting, receiving, listening, and sleeping. Notice that since we use TDMA, no nodes need to be in state *listening* if all nodes are perfectly synchronized. If the synchronization is needed, nodes will also have additional state *listening* so that adjacent nodes can synchronize their activities. See Fig. 1 for an illustration. Here, we assume that the sender node will use a short preamble to synchronize the receiving node (similar to X-MAC [3]). In other words, when a sender wakes up, it will periodically send a message **SYN** (contains its address and the receiver's address, and the time slots needed for sending data) and listen for the **ACK** message from the receiver. When the receiver wakes up (after a state switch time **SW**), it will listen for the message **SYN** and reply a message **ACK** if it gets one completed **SYN** message. After getting the correct **ACK** message, the sender starts sending data. For a node $v_i$, if it is scheduled to transmit at time slot $t$, we denote it as $X_{i,S,t} = 1$; otherwise we denote it as $X_{i,S,t} = 0$. We use variables $X_{i,R,t} \in \{0,1\}, X_{i,P,t} \in \{0,1\}$ and $X_{i,L,t} \in \{0,1\}$ to denote whether the node $v_i$ is scheduled to receive, sleep, or listen at time slot $t$ or not, respectively. We denote energy consumed by state transition as $E_{P,S}, E_{P,R}$, or $E_{P,L}$. See Table 2 for notations used. In practice, the energy consumed from an active state (such as transmitting, receiving, and listening) to an idle state (sleeping or deep sleeping) is often ignored.

Notice that the energy cost by a node $v_i$ in all states is $\sum_{t=1}^{T}(X_{i,S,t} \cdot P_{tx} + X_{i,R,t} \cdot P_{rcv} + X_{i,L,t} \cdot P_{lst} + X_{i,P,t} \cdot P_{slp}) \cdot t_s$; the energy cost for state transitions is $\sum_{t=1}^{T}(X_{i,P,t} \cdot X_{i,S,t+1} \cdot E_{P,S} + X_{i,P,t} \cdot X_{i,R,t+1} \cdot E_{P,R} + X_{i,P,t} \cdot X_{i,L,t+1} \cdot E_{P,L})$, where $T+1$ will be treated as 1. The objective of a schedule $\mathcal{S}$ is to minimize the summation of these two energy costs.

Among numerous schedules for wireless sensor nodes' activities, we consider schedules that satisfy certain feasibility constraints. A schedule $\mathcal{S}$ is called a *valid schedule* (or *feasible schedule*) if it satisfies the following constraints:

1. First of all, the amount of slots assigned to a node $v_i$ for transmitting should be enough. Without loss of generality, we assume that the total number of time slots for transmitting in a scheduling period $T$, based on a data collection tree **T**, required by node $v_i$ is $0 \le w_i \le T$. Here, $w_i$ is computed based on the amount of information (say $W$ packets) received from its children nodes in the data collection tree **T** in a scheduling period and the amount $r_{v_i}$ of data produced by its own sensor in a scheduling period. If node $v_i$ does not have data aggregation ability, $w_i = \lceil \frac{W + r_{v_i}}{P_s} \rceil$. When node $v_i$ has the aggregation ability, $w_i = \lceil \frac{f(W, r_{v_i})}{P_s} \rceil$, where function $f(\cdot,\cdot)$ computes the number of packets needed for aggregated data (generated from the data received from its children and its own data). Notice that in practice, since the wireless channel is not reliable, we may need to adjust $w_i$ as $w_i/p$, where $p$ is the observed link reliability of link $(v_i, v_k)$, where node $v_k$ is the receiving node (typically, the parent node) of node $v_i$. In other words, $w_i/p$ is the expected size of the data to be transferred such that all original data are correctly received. In summary, we need $\sum_{t=1}^{T} X_{i,S,t} \ge w_i/p$.
2. Obviously, a node $v_i$ with children nodes $u_1, u_2, \ldots, u_{d_i}$ should be active for receiving at the time slots when these children nodes send data to $v_i$. In other words, if $X_{j,S,t} = 1$, then $X_{i,R,t} = 1$ when node $v_i$ is the parent node of node $v_j$; equivalently, we need $X_{i,R,t} \ge X_{j,S,t}$ whenever node $v_i$ is parent of node $v_j$ in tree **T**.
3. Further, any node can only be in one of the states, i.e., $X_{i,S,t} + X_{i,R,t} + X_{i,P,t} + X_{i,L,t} = 1$.
4. At last, all transmissions should be interference-free, i.e., $X_{i,S,t} + X_{k,S,t} \le 1$ for any time slot $t$ and any pair of nodes $v_i$ and $v_k$ that will cause interference if they are transmitting simultaneously. Notice that it is also true for $v_k \in I(i)$ if node $v_k$ sends to a node $v_j$ which is in the interference region of node $v_i$. For notational simplicity, we use $I(i)$ to denote the set of nodes that cannot transmit simultaneously with node $v_i$. Thus, we need $X_{i,S,t} + X_{k,S,t} \le 1$ for any node $v_k \in I(i)$.

In a simple event-driven data collection, a sensor, which is triggered by an event, will wake up and monitor its vicinity, and then, produce some sample data. It will then wake up its parent node (called dominator node sometimes) and send

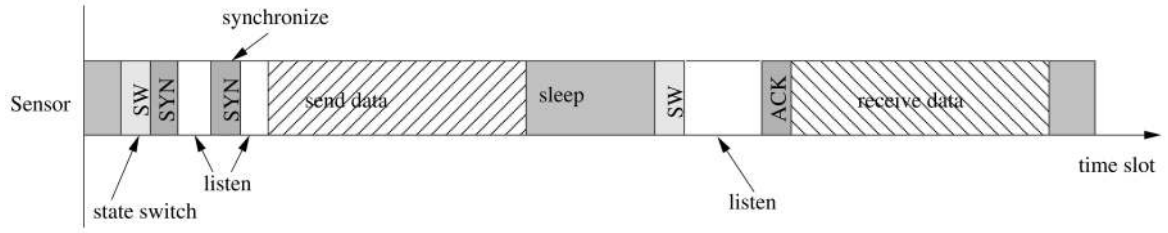



Fig. 2. The time slots for a sensor in one period.

data to it. However, when the dominator node dominates $k$ sensors, it may need to wake up $k$ times to receive all the data form its children nodes in the worst case, which is energy consuming because of multiple state transitions. Our objective is to schedule the activities of sensor nodes to minimize the states transitions (especially from sleeping state to active states), in the meanwhile, the data rate by all sensors is supported. In this paper, we will always consider low-data-rate WSNs where in the majority of time slots, sensor nodes can sleep to save the energy. Notice that in low-data-rate WSNs, each sensor needs to switch from sleeping state to active state *at least once*. Surprisingly, we will design a valid schedule in which any sensor node only needs to wake up *at most twice*: once for receiving data from its children nodes and once for sending its data to its parent node. This also dramatically reduces the cost of the clock synchronization. See Fig. 2 for an illustration of possible schedule for a node.

#### 2.2.2 Data Collection Tree Construction

Previously, we assume that a tree $\mathbf{T}$ is given for the data collection or aggregation. In the literature, a number of trees have been proposed for the data collection or data aggregation for various purposes. In this paper, we will further study how to (approximately) construct an optimum tree $\mathbf{T}$ such that the total energy cost of the optimum activities scheduling based on this tree is the lowest among all trees satisfying the data rate requirements by all nodes. It has been observed in the literature that the largest data rate that can be supported by different network topologies will vary. Thus, our objective is to find a data collection tree $\mathbf{T}$ that should satisfy the data requirements of all nodes, i.e., there exists a *valid scheduling* of node activities. Observe that given a wireless network topology, it is generally NP-hard to find the largest data rate that can be supported. Recently, a number of constant approximation algorithms [4], [5], [6] have been proposed for various interference models.

## 3 Homogeneous Wireless Sensor Networks

In a homogeneous sensor network, every sensor node has the same interference range, while the amount of data (represented by the number of time slots $w_i$) to be transferred by a sensor node $v_i$ to its parent could be different.

### 3.1 Centralized Activity Scheduling

We first study a centralized scheduling of sensor activities to minimize the energy cost. Assume that we are given the data gathering tree $\mathbf{T}$ for the sensor network. Traditionally, the scheduling algorithms (e.g., [6]) often schedule the individual activities for each sensor one by one: assuming that sensors will schedule in a *random order*, each sensor node will find the best time slots for sending its data, and also the best time slots to receive data from each child individually without causing interference to already-scheduled sensors. This schedule is called **schedule by random**. Unfortunately, these scheduling strategies cannot minimize the energy cost for each sensor node: some sensors may need to wake up multiple times in a scheduling period $T$. In this paper, to reduce the energy cost, we will schedule the activities of a subset of sensors in one bundle. Let $\mathcal{C}_{\mathbf{T}}(v_i)$ denote the set of children nodes of node $v_i$ in data gathering tree $\mathbf{T}$. Then we say that $\mathcal{C}_{\mathbf{T}}(v_i)$ constitutes a virtual cluster $\mathbf{C}_i$. For each cluster $\mathbf{C}_i$, we define $\mathbf{W}_i = \sum_{v_j \in \mathcal{C}_{\mathbf{T}}(v_i)} w_j$ as its *weight*. Obviously, $\mathbf{W}_i = \sum_{v_j \in \mathcal{C}_{\mathbf{T}}(v_i)} w_j$ is the total number of time slots that node $v_i$ should wake up to receive the data from its children in the data gathering tree $\mathbf{T}$. Then instead of scheduling the *transmitting* time slots for each individual child node of node $v_i$, we schedule a chunk of consecutive $\mathbf{W}_i$ time slots to the cluster $\mathbf{C}_i$ of these children nodes. Then each child $v_j$ will be assigned a consecutive $w_j$ time slots from this chunk. All the children will send their data in this period and the parent will receive the data at the same time. Thus, the energy consumption due to the state transition will be definitely saved since each node needs only to wake up twice: once for receiving all data from its children and once for transmitting its data to its own parent. To present our methods, we define conflicting clusters as follows.

**Definition 1.** *Two clusters $\mathbf{C}_i$ and $\mathbf{C}_k$ are said to be conflicting with each other if there exists a sensor node $u \in \mathbf{C}_i$ and a sensor node $v \in \mathbf{C}_k$ such that $u$ and $v$ cannot be scheduled simultaneously for transmitting.*

We schedule the clusters in the *decreasing* order of their weight. To schedule a cluster $\mathbf{C}_i$, we use the *first-fit* approach: the chunk of time slots scheduled to sensors in $\mathbf{C}_i$ is the earliest consecutive $\mathbf{W}_i$ time slots such that it will not have any overlap with *already-scheduled time slots* for scheduled conflicting clusters. Thus, our schedule ensures that it will not cause *any* interference for the transmissions of *any* sensors in $\mathbf{C}_i$. We describe the detailed method in Algorithm 1.

**Algorithm 1.** Activity Schedule using Tree $\mathbf{T}$

1: **for** each sensor $v_i$ **do**
2: calculate its receiving-weight $\mathbf{W}_i = \sum_{v_j \in \mathcal{C}_{\mathbf{T}}(v_i)} w_j$.
3: Sort the sensor in nonincreasing order of weight $\mathbf{W}_i$.
4: **for** $i = 1$ to $n$ **do**
5: Assign cluster $\mathbf{C}_i$ (equivalently sensor node $v_i$ for receiving) $\mathbf{W}_i$ earliest available time slots for transmitting that will not overlap with time slots assigned to conflicting clusters.

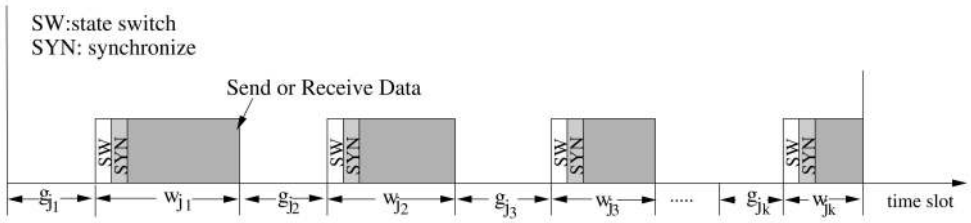



Fig. 3. Illustration of the scheduled slots before scheduling $\mathbf{W}_i$ receiving slots for node $v_i$.

6: Each sensor node $v_j$ in $\mathbf{C}_i$ will be sequentially assigned $w_j$ consecutive time slots for transmitting.

Notice that two conflicting clusters may still be scheduled together. For example, assume that $\mathbf{C}_1 = \{u_1, u_2\}$ and $\mathbf{C}_2 = \{u_3, u_4\}$ and each sensor node needs $w_i = 2$ time slots for transmitting. Further assume that only one pair of nodes $u_1$ and $u_3$ cannot be scheduled simultaneously. Then following schedule is valid: node $u_1$ uses time slots 1, 2; node $u_2$ uses time slots 3, 4; node $u_3$ uses time slots 3, 4; and node $u_4$ uses time slots 1, 2. Based on this observation, our schedule seems to be pessimistic. However, we will prove that the maximum period required by our schedule is only a constant factor of the optimum solution.

### 3.1.1 Distributed Method

For the convenience of designing-efficient distributed activities scheduling method, we associate the cluster $\mathbf{C}_i$ with the parent node $v_i$. In other words, the scheduling for sensors in $\mathbf{C}_i$ will actually be done by node $v_i$ in distributed implementation. We call $\mathbf{W}_i$ as the *receiving weight* (or simply *weight* if it is clear from the context) of node $v_i$. Our **schedule by Cluster** method is based on the first-fit strategy. We sort the sensor nodes according to the nonincreasing receiving weight $\mathbf{W}_i$. If some of the sensors have the same receiving weight, we break the tie by sensors' ID. For simplicity, assume that the sequence $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_n$ is the sorted receiving weights in nonincreasing order. Then we schedule the sensors sequentially. For sensor $v_i$, starting from the time slot 1, we search for consecutive $\mathbf{W}_i$ time slots which are not be occupied by any already-scheduled conflicting cluster $\mathbf{C}_j$ with $j < i$. If found, we schedule cluster $\mathbf{C}_i$ to these time slots. Otherwise, we schedule cluster $\mathbf{C}_i$ to new time slots that are not used by any cluster before.

### 3.1.2 Performance Analysis

The interference-free scheduling is similar to the graph coloring. For a scheduling $\mathcal{S}$, let $T(\mathcal{S})$ be the *span* of time slots used by all nodes in a period. Typically, for a schedule, we start from time slot 1, then $T(\mathcal{S})$ is simply the last time slot that has active node activities (i.e., transmitting or receiving). Notice that $1/T(\mathcal{S})$ is closely related to the maximum data rate that can be supported by the schedule $\mathcal{S}$. We then prove that the timespan $T(\mathcal{S})$ achieved by our method described in Algorithm 1 is at most a constant factor of the optimum. To simplify the proof, we integrate the time for the wake up and clock synchronization to the time slots for transmitting or receiving the data. In other words, all the weights $w_i$ and $\mathbf{W}_i$ also include the wake-up cost and clock-synchronization cost. It is reasonable because the time for the state switching and clock synchronization is much smaller and can typically be done within one time slot. We just add one more time slot for receiving or transmitting.

**Theorem 1.** *The energy consumption for the scheduling derived by Algorithm 1 is at most twice of the optimum.*

**Proof.** For a given data gathering tree **T**, the time slot needed to transmit and receive by an individual node is fixed because it only depends on the tree structure. So, the difference of the energy consumption from different schedules is the cost for the wake up and clock synchronization. In our scheduling, there are at most two state switches for each node. So, the total number of state switches is at most $2n$ times of the energy consumption for a node to switch the state. That is, $E_S \leq 2 \cdot n \cdot E_s$, where $E_S$ is the energy consumption for state switch in our scheduling, $E_s$ is the energy consumption for the state switch of one sensor, and $n$ is the number of sensors. Because there is at least one state switch for each node in any scheduling, $E_S^{opt} \geq n \cdot E_s$, where $E_S^{opt}$ is the optimal energy consumption for state switch. We denote $E_T$ as the total energy consumption in the active states by all nodes by our method, $E$ as the total energy consumption in our scheduling and $E^{opt}$ as the optimal energy consumption. We get $E = E_T + E_S \leq E_T^{opt} + 2E_S^{opt} < 2E_T^{opt} + 2E_S^{opt} = 2E^{opt}$. This finishes the proof. □

**Theorem 2.** *The timespan derived by Algorithm 1 is at most a constant factor of the optimum.*

**Proof.** Consider the sensor node $v_i$ that has the last time slots for receiving in our scheduling (i.e., the cluster $\mathbf{C}_i$ has the last time slots for sending data to $v_i$). Notice that it is not necessary that sensor node $v_i$ has the smallest receiving weight. For sensor $v_i$, we consider the moment when the sensor $v_i$ (equivalently, cluster $\mathbf{C}_i$) is scheduled. At the moment when preparing to schedule for node $v_i$, some sensors are already scheduled and we illustrate the situation of occupied time slots in Fig. 3.

In the figure, $\mathbf{W}_{j_1}, \mathbf{W}_{j_2}, \ldots, \mathbf{W}_{j_k}$ represent the consecutive time slots occupied by the clusters which conflict with cluster $\mathbf{C}_i$ and are scheduled before cluster $\mathbf{C}_i$ (and equivalently, sensor $v_i$). For time slots occupied by other nonconflicting clusters, we denote them as *gaps*, which could be assigned to cluster $\mathbf{C}_i$ (and sensor $v_i$). Notice that some of the time slots assigned to clusters conflicting with $\mathbf{C}_i$ may overlap, i.e., one chunk of consecutive time slots $\mathbf{W}_{j_a}$ with $1 \leq a \leq k$ may denote the time slots assigned to several clusters conflicting with $\mathbf{C}_i$. So, every chunk of time slots $\mathbf{W}_{j_1}, \mathbf{W}_{j_2}, \ldots, \mathbf{W}_{j_k}$ is no smaller than the time slots $\mathbf{W}_i$ required by sensor $v_i$ for

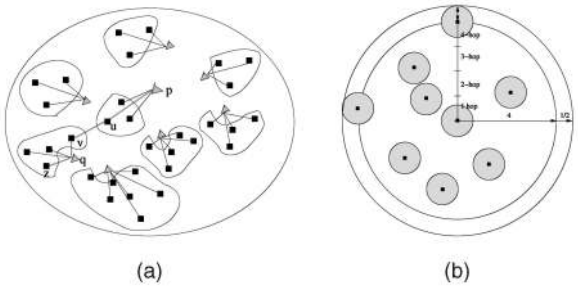



Fig. 4. Illustration of 4-hop distance. (a) 4-hop neighbor. (b) Independent number.

receiving since we scheduled the sensors in the decreasing order of the receiving weight $\mathbf{W}_i$. In addition, in the figure, $g_{j_1}, g_{j_2}, \ldots, g_{j_k}$ represent the time slots occupied by some other sensors, which do *not* conflict with cluster $\mathbf{C}_i$. Notice that some of $g_{j_1}, g_{j_2}, \ldots, g_{j_k}$ may be empty slots.

**Case 1.** $\exists l, \mathbf{W}_i \leq g_{i_l}, 1 \leq i_l \leq i_k$. In this case, $v_i$ can be scheduled inside time slots $g_{i_l}$ because the sensors scheduled there do not interfere with cluster $\mathbf{C}_i$ (and also $v_i$ for receiving data). That is, $v_i$ can be scheduled before $v_{i_l}$, which contradicts the assumption that $v_i$ has the *largest* time slots for receiving. In other words, this case is impossible.

**Case 2.** $\mathbf{W}_i > g_{i_l}, \forall 1 \leq i_l \leq i_k$. The total time slots used by our scheduling after $v_i$ is scheduled are

$$T_i = \sum_{l=1}^{k} g_{i_l} + \sum_{l=1}^{k} \mathbf{W}_{i_l} + \mathbf{W}_i$$
$$\leq k \cdot \mathbf{W}_i + \sum_{l=1}^{k} \mathbf{W}_{i_l} + \mathbf{W}_i \leq 2\left(\sum_{l=1}^{k} \mathbf{W}_{i_l} + \mathbf{W}_i\right).$$

Recall that we assumed that sensor node $v_i$ has the largest time slots for receiving. According to the above result, the timespan $T$ used by our schedule satisfies $T \leq 2(\sum_{l=1}^{n_k} \mathbf{W}_{n_l} + \mathbf{W}_n)$. We denote $T_{opt}$ as the smallest timespan by any schedule. In an optimum schedule, consider the time slots needed for the following clusters $\mathbf{C}_i, \mathbf{C}_{j_1}, \mathbf{C}_{j_2}, \ldots, \mathbf{C}_{j_k}$. Here, $\mathbf{C}_{j_1}, \mathbf{C}_{j_2}, \ldots, \mathbf{C}_{j_k}$ are *all* clusters that conflict with cluster $\mathbf{C}_i$ and have a weight $\mathbf{W}_{j_l} \geq \mathbf{W}_i$. We will show that

$$T_{opt} \geq \frac{\sum_{l=1}^{k} \mathbf{W}_{j_l} + \mathbf{W}_i}{\alpha},$$

where $\alpha$ is a constant (called the independence number) depending on the interference model and will to be specified later. Combining with $T \leq 2(\sum_{l=1}^{k} \mathbf{W}_{j_l} + \mathbf{W}_i)$, we get $T \leq 2\alpha \cdot T_{opt}$.

We then prove a constant value $\alpha$. Let $p = v_i$ be the parent node of all sensor nodes in cluster $\mathbf{C}_i$. First of all, we will show that every sensor node $z$ from some cluster conflicting with $\mathbf{C}_i$ is at most 4-hops away from node $p$. Assume that node $z$ is from a conflicting cluster $\mathbf{C}_{j_l}$. Fig. 4a illustrates the case. Assume that the cluster $\mathbf{C}_{j_l}$ conflicts with cluster $\mathbf{C}_i$ because a node $u \in \mathbf{C}_i$ and a node $v \in \mathbf{C}_{j_l}$ cannot be scheduled for transmitting simultaneously. Let $q$ be the receiving node (i.e., parent node) of sensor node $v$. Then, we either have $q$ is inside the interference region of node $u$ or $p$ is inside the interference region of node $v$. Since we typically assumed that the interference region of a node is all nodes within 2-hop communication distance, we can conclude that node $q$ is within 3-hop distance from node $p$. For node $z$, it is 1-hop communication neighbor of node $q$. Thus, every node $z$ is at most 4-hops away from node $p$.

Notice that either $\|p - v\| \leq R_I(v)$ or $\|q - u\| \leq R_I(u)$. Additionally, $\|q - v\| \leq R_T(q) \leq R_I(q), \|q - z\| \leq R_T(q) \leq R_I(q), \|p - u\| \leq R_T(p) \leq R_I(p)$, since the interference range of any node is at least its communication range. Recall that we assumed that all nodes have the same interference range $R_I$. Consequently, the above analysis also shows that node $z$ is within the distance at most $3R_I$ from node $p$. That is, the sensors from conflicting clusters $\mathbf{C}_{j_l}$ $(1 \leq l \leq k)$ can only be distributed inside the circle with the radius $3R_I$ as in Fig. 4b. The total time slot required by all these sensors for sending data is $\sum_{l=1}^{k} \mathbf{W}_{j_l} + \mathbf{W}_i$. Observe that for an interference model, two nodes transmitting simultaneously must be a certain distance away from each other. For example, for the transmitter interference model (TxIM), the separation distance is about the interference range $R_I$. For the fixed power protocol model (fPrIM), let $d$ be the distance between two transmitting nodes $u$ and $v$ and $p$ the receiving node of $u$'s transmission. Then, $\|v - p\| \leq \|u - v\| + \|u - p\| \leq d + R_T(u)$, which is less than $R_I(u)$ if $d \leq R_I(u) - R_T(u)$. In other words, the transmission by node $v$ will interfere the receiving of node $p$. Recall that we assumed $R_I = R_I(u) \geq (1 + \beta)R_T(u)$ for every node $u$ for some constant $\beta \simeq 1$. Consequently, the distance between two nodes transmitting simultaneously should be at least $R_I(u) - R_T(u) \geq \frac{\beta}{1+\beta} \cdot R_I$. Then, for any scheduling for sensor nodes in clusters $\mathbf{C}_i, \mathbf{C}_{j_l}$ $(1 \leq l \leq k)$, the simultaneous transmitting sensor nodes should be at least a distance $\frac{\beta}{1+\beta} \cdot R_I$ away from each other. Thus, the number of simultaneously transmitting nodes at any time slot is at most

$$\frac{\left(3 + \frac{1}{2}\right)^2}{\left(\frac{\beta}{2(1+\beta)}\right)^2} = \frac{49(1+\beta)^2}{\beta^2}.$$

Let

$$\alpha = \frac{49(1+\beta)^2}{\beta^2}.$$

Thus, the total time slot needed for scheduling the sensors in clusters $\mathbf{C}_i, \mathbf{C}_{j_l}$ $(1 \leq l \leq k)$ is at least

$$\frac{\sum_{l=1}^{k} \mathbf{W}_{j_l} + \mathbf{W}_i}{\alpha}.$$

Thus, the timespan of the schedule produced by Algorithm 1 is at most $2\alpha$ times of the optimum. □

Based on the above results, we formally define what is a *low-data-rate* sensor network. Given a data collection tree $\mathbf{T}$ and the data rate $r_{v_i}$ for every sensor $v_i$, the data rate vector $\mathbf{r} = \langle r_{v_1}, r_{v_2}, \ldots, r_{v_{n-1}}, r_{v_n} \rangle$ is called *schedulable* under tree $\mathbf{T}$ if

there is a schedule of node activities such that the data produced by all sensors will be received by the sink node within a finite time. A data rate $\mathbf{r}' = \langle r'_{v_1}, r'_{v_2}, \ldots, r'_{v_{n-1}}, r'_{v_n} \rangle$ is called *low-data-rate* if $2\alpha \cdot \mathbf{r}'$ is schedulable. Notice that based on Theorem 2, a low-data-rate sensor network can always be scheduled by Algorithm 1.

#### 3.1.3 More Discussions

Besides reducing the energy consumption and increasing network throughput, another important issue for the data collection in WSNs is to reduce the delay. In a schedule of the data collection, if the time slots for transmitting data by a node $v$ are later than the time slots for transmitting of its parent node, say $p(v)$, then the data collected by $v$ can only be sent by $p(v)$ in next round, which will cause a possible large delay. Thus, in our scheduling algorithm, to reduce the delay, we will adopt the following changes, which will not affect the correctness of Theorem 1 and Theorem 2. Instead of scheduling using the available earliest time slots, we use the latest available time slots. Assume that we start from time slot $T$ initially. The cluster with the largest weight, say $\mathbf{W}_1$, will be scheduled in time slots $[T - \mathbf{W}_1 + 1, T]$. When we want to schedule a cluster $\mathbf{C}_i$ with weight $\mathbf{W}_i$, let $t_i$ be the *earliest* time slot that has been used by some scheduled cluster conflicting with $\mathbf{C}_i$. There are two cases here.

1. **No big gap.**[1] In this case, we cannot find consecutive $\mathbf{W}_i$ time slots in the scheduled period $[t_i, T]$ that will *not* cause conflict with already scheduled clusters. Then, $\mathbf{C}_i$ will be scheduled in time slots $[t_i - \mathbf{W}_i, t_i - 1]$.
2. **Existence of big gaps.** In the second case, we can find such consecutive $\mathbf{W}_i$ time slots in $[t_i, T]$. In this case, let $v_i$ be the parent node of all nodes in $\mathbf{C}_i$ in $\mathbf{T}$. Notice that for the data collection, the transmitting time of $v_i$ must have been scheduled already since the cluster containing $v_i$ will have larger weight than $\mathbf{W}_i$. Let $Y_i$ be the starting time slot of transmitting by $v_i$. Then we will schedule the time slots for transmitting by nodes in $\mathbf{C}_i$ (equivalently, the time slots for receiving by $v_i$) ahead of $Y_i$, say the latest gap before $Y_i$. If no such gap exists, we will just schedule the time slots for transmitting by nodes in $\mathbf{C}_i$ in the latest gap, i.e., closest to time $T$.

In our performance evaluation, the *schedule by cluster* method will include such enhancement.

We can make further improvement (although its improvement is at most a constant factor) is to reduce the timespan of the resulting schedule. Remember that we say that two clusters $\mathbf{C}_i$ and $\mathbf{C}_j$ conflict with each other if there is *any pair* of nodes $u$ and $v$ (one from each cluster) such that $u$ and $v$ cannot be scheduled simultaneously. Notice that in our scheduling, conflicting clusters will *never* have overlap time slots. In practice, these two clusters may still be able to scheduled using overlap time slots. Thus, we can modify our scheduling to further reduce the timespan of the schedule: we try to schedule $\mathbf{C}_i$ to latest consecutive $\mathbf{W}_i$ time slots that will not cause any interference. Notice that the techniques used to reduce the delay still apply. In our performance evaluation, this method is called **Schedule by Cluster and Squeeze**.

In sensor networks, a sensor may be depleted or destroyed. Our scheduling is adaptive in this scenario as long as the network is still connected. After removing depleted or destroyed nodes, we just need to update the data collection tree using nearby nodes locally, and then, using the distributed scheduling in the following section.

### 3.2 Distributed Activity Scheduling

We then design a distributed activities scheduling algorithm. An obvious difficulty of the distributed implementation is to sort nodes' weights. We use *local sorting* in distributed algorithm by replacing the global sorting in centralized algorithm: a cluster can schedule its transmitting activities if all *conflicting* clusters with *larger* weight have been scheduled. Second, for easy maintenance, each sensor node $v_i$ will be responsible for the scheduling of transmitting activities of its children nodes $\mathbf{C}_i$ (equivalently, the receiving activity of itself). To inform potentially conflicting clusters about its own weight $\mathbf{W}_i$, sensor node $v_i$ will first multicast a *request* message (including its weight $\mathbf{W}_i$) to its $k$-hop neighbors to request for scheduling. Notice, based on previous analysis, that we will choose $k = 4$ in our algorithm. If a sensor does not have the largest weight, it will wait till it has the largest weight among its $k$-hop nonscheduled neighbors. The sensor finds the earliest available time slots and marks itself as scheduled. Then the sensor sends *scheduled* message to its unscheduled neighbors with its reserved time-slots information included. Algorithm 2 illustrates our method.

**Algorithm 2.** Distributed Activity Scheduling using $\mathbf{T}$ by $v_i$

1: The sensor $v_i$ first computes $\mathbf{W}_i$ based on the data rates from its children in $\mathbf{C}_i$. It sends a message *request* to all sensors that are within a constant $k$-hops to request for scheduling. This can be done using a simple flooding with time-to-live $TTL = k$.
2: **if** $\mathbf{W}_i \geq \mathbf{W}_j$ for each nonscheduled $k$-hop neighbor $v_j$ **then**
3: Schedule $v_i$ earliest $\mathbf{W}_i$ available time slots for receiving data and sequentially schedule the time slots for transmitting for every sensor in $\mathbf{C}_i$.
4: Send a message *IamScheduled* (with its own schedule) to all its unscheduled $k$-hop neighbors and mark itself scheduled.

**Theorem 3.** *The timespan of the schedule produced by Algorithm* 2 *is at most a constant factor of the optimum.*

**Proof.** In Algorithm 2, when we schedule the receiving activities of a sensor node (and thus, the transmitting activities of all its children nodes), we consider all possible sensor nodes that could conflict with its receiving activities. Similar to the proof of Theorem 2, we can prove the correctness of this theorem. □

**Theorem 4.** *The number of messages used in Algorithm* 2 *is* $O(n)$.

**Proof.** Suppose the maximum number of $k$-hop neighbors is $\Delta_k$, which is often much smaller than $n$. The total number

1. A gap is maximal consecutive time slots in $[t_i, T]$ that can be used by current to-be-scheduled cluster $\mathbf{C}_i$, i.e., its size is at least $\mathbf{W}_i$. Often many gaps (and at most $i - 1$ gaps) exist for $\mathbf{C}_i$.

of *request* messages is at most $\Delta_k n$. In the worst case, every sensor sends a *request* message to its neighbors, which are at most $\Delta_k$ such relaying messages. Each cluster will send its *scheduled* message to its unscheduled neighbors. The total number of scheduling messages is at most $\Delta_k n$. So, the total messages used in the Algorithm 2 are $\Theta(\Delta_k n)$, which is linear when $\Delta_k$ is constant.

When $\Delta_k$ is not a constant, we can apply the method proposed in [7] to do the message relay based on a connected dominating set of the network. That method can guarantee that the number of nodes used to relay a message from a node $v_i$ to its $k$-hop neighborhood is only a constant. □

We note that the worst case time complexity of Algorithm 2 could be as large as $O(n)$. In the worst case, the sensor will be scheduled sequently. For example, the sensors are distributed in a line with increasing weight from left to right and only adjacent sensors form communication links. The scheduling order will be $n, n-1, n-2, \ldots, 1$. Sensor $i$ will wait until sensor $i+1$ is scheduled, sensor $i+1$ will wait until sensor $i+2$ is scheduled, and so on. Then sensor 1 will wait until all the other sensors are scheduled.

## 4 Heterogeneous Wireless Sensor Networks

In a heterogeneous WSN, we can show that the method developed for a homogeneous WSN cannot guarantee a constant approximation ratio on the timespan used by a schedule. To schedule the activities of sensors for heterogeneous sensors, we will first divide the sensors into buckets according to their interference radii: the $i$th bucket contains all sensors which have the interference radius within

$$[2^{i-1}R_{\min}, 2^i R_{\min}).$$

Here, $R_{\min}$ is the minimum interference radius of the sensor in the network. In each bucket, we schedule the sensors according to the first-fit approach developed in Algorithm 1: we sort the nodes in nonincreasing order according to the receiving weight and assign each node a chunk of the earliest consecutive time slots without causing interference to already-scheduled nodes. Then the scheduling in each bucket has a timespan at most a constant factor of the optimum. We schedule the buckets in sequential order and the time slots used by consecutive buckets are concatenated together. We describe our method in detail in Algorithm 3.

**Algorithm 3.** Centralized Activity Schedule in Heterogeneous Networks using **T**

1: **for** each sensor $v_i$ **do**
2: Insert the sensor into bucket $[2^{i-1}R_{\min}, 2^i R_{\min})$ in nonincreasing order according to the weight $\mathbf{W}_i$ if its interference radius is within $[2^{i-1}R_{\min}, 2^i R_{\min})$.
3: **for** each bucket $b_l$ **do**
4: **for** every cluster $\mathbf{C}_i$ (equivalently sensor node $v_i$ for receiving) in the bucket **do**
5: Assign cluster $\mathbf{C}_i$ a chunk of $\mathbf{W}_i$ earliest available time slots for transmitting which will not overlap with time slots assigned to conflicting clusters.
6: Each sensor node $v_j$ in $\mathbf{C}_i$ will be sequentially assigned $w_j$ consecutive time slots for transmitting.
7: assign $v_i$ a chunk of $\mathbf{W}_i$ earliest available time slots.

**Theorem 5.** *The timespan of the schedule derived by Algorithm 3 is at most* $\Theta(\log \frac{R_{\max}}{R_{\min}})$ *of the optimum.*

The proof is omitted due to space limit. Notice that this is only the theoretical bound. In practice, we expect that this $\Theta(\log \frac{R_{\max}}{R_{\min}})$ ratio to be actually $\Theta(1)$ since our schedule actually will overlap the time slots used by different buckets and $T_i$ could be much smaller than $T_{opt}$. Actually, without using the bucket idea, we can also prove that Algorithm 1 produces a schedule whose timespan is at most $\Theta(\rho_3^2)$ times of the optimum, where $\rho_k = \max \rho_k(v)$ and $\rho_k(v)$ is the ratio of the interference range $R_I(v)$ over the smallest interference range among all nodes in $N_k(v)$. Notice that $\rho_k \leq \frac{R_{\max}}{R_{\min}}$. The basic idea is to show that the number of independent nodes among $N_k(v)$ is at most $\Theta(\rho_k^2)$; thus, the number of simultaneous transmissions for nodes in $N_3(v)$ by any schedule is at most $\Theta(\rho_k^2)$. The rest of the proof is similar to Theorem 2 and is omitted here.

We then design a distributed algorithm for a heterogenous network. The scheduling is similar to the centralized algorithm except that the sensors will find a schedule by collecting the information within $k$-hop first. Based on the $k$-hop information, the sensor chooses a bucket to which it belongs and inserts itself into the bucket according to the data rate. Each sensor will then be scheduled an interference-free time slots after all the sensors with larger weights are scheduled.

Similar to Theorem 5, the next theorem is straightforward.

**Theorem 6.** *The timespan of the schedule derived by Algorithm 4 is at most* $\Theta(\log \frac{R_{\max}}{R_{\min}})$ *of the optimum.*

**Algorithm 4.** Distributed Activity Scheduling in Heterogeneous Networks using **T** by $v_i$

1: The sensor $v_i$ first computes $\mathbf{W}_i$ based on the data rates from its children in $\mathbf{C}_i$. It collects information of sensors that are within a constant $k$-hops, using a simple flooding with TTL $k$.
2: All sensors with transmission range in the same bucket $[2^{l-1}R_{\min}, 2^l R_{\min})$ run the distributed scheduling.

## 5 Formation of Data Gathering Tree

So far, we always assume that we already have a data gathering tree **T** in hand for the data collection or data aggregation. Obviously, the performance of our scheduling algorithm depends on the underlying data gathering tree **T** used. If we only consider the total energy consumption by *all* nodes in the network for the data collection (i.e., without the data aggregation), the energy cost of a node $v$ only depends on the *total* data rate of all nodes that are descendant of $v$ in the tree **T**. In other words, given a node $u$, the total energy cost used by all nodes to relay the data amount ($r_u$ units) by node $u$ is $h \cdot r_u \cdot E$, where $h$ is the height of node $u$ in the tree **T** and $E$ is the energy used to transmit a unit amount of data. Thus, the tree that will minimize the energy cost for data collection is the shortest hop path tree SPT (or equivalently, breadth-first search tree BFS rooted at the sink node).

On the other hand, to reduce the timespan of a scheduling, the shortest hop path tree may not be always

the best choice. The throughput achieved by the tree SPT may also not be the optimum. We thus would like to design a data gathering tree such that the total energy cost incurred by the data collection of a best scheduling $\mathcal{S}_1$ on this tree is within a constant times of the optimum, and the timespan of the best scheduling $\mathcal{S}_1$ over this tree is also within a constant factor of the optimum. In the literature, a number of data collection trees have been proposed such as the local minimum spanning tree and the minimum spanning tree. However, none of these structures seems to achieve both properties.

Notice that the requirement of the constant approximation ratio on the total energy cost is equivalent to the requirement that the height of every node $v$ in the data collection tree $\mathbf{T}$ is at most a constant factor of that in SPT, i.e., its shortest-hop distance to the sink node. The requirement for constant approximation ratio on the timespan is equivalent to the requirement that the weighted chromatic number is a constant factor of the optimum.

We first consider the case of the data collection tree for homogeneous networks. Our tree will be based on a connected dominating set (CDS). A number of efficient methods have been proposed for constructing the CDS tree, e.g., [8].

We briefly review the method in [8]. First, we construct BFS tree, where the distance of the node is called its level. Then we construct CDS. Initially, we set all nodes white, and then, set the sink black (dominator) and its neighbors gray (dominatees). For each level with nondecreasing order, if there are white nodes left, we choose the one with the smallest ID as dominator, set it black, and set its white neighbors gray as dominatees. If the dominator is not connected to CDS tree, we add its parent as connector. The resulting tree is called *CDS tree*. We then prove the following property of the CDS tree.

**Theorem 7.** *The energy consumption in the CDS tree is a constant factor of that by the optimal tree.*

**Proof.** Given a node $v$, we denote the hop distance from the node $v$ to the sink in the CDS tree as $h$. From the construction, we know $h \leq 2 \times h_0$, where $h_0$ is the hop distance from $v$ to the sink in overall BFS tree. To save energy, we just consider transmitting state and receiving state as the active states in our scheduling. Let $E_T^{CDS}$ be the energy consumed for transmitting/receiving by the CDS tree. Let $OPT$ be the best tree that has the lease energy consumption for transmitting, receiving, and switching states. Then, $E_T^{CDS} \leq 2 \cdot E_T^{BFS} \leq 2 \cdot E_T^{OPT}$, where $E_T^H$ is the energy consumption in active states by a tree $H$. Similarly, let $E_S^{H,\mathcal{S}}$ be the energy consumption for switching state in a structure $H$ using a schedule $\mathcal{S}$.

A node switches between sleeping state and active state at most twice in our scheduling: one between the sleeping state and transmitting state, and the other one between the sleeping state and receiving state. Obviously, the switching cost on CDS by our protocol is at most $E_S^{CDS} \leq 2 \cdot n \cdot E_s$, where $E_s$ is the energy consumption for a node to switch from sleeping state to active state and switch back. Note that $E_S^{OPT} \geq n \cdot E_s$ since one node should wake up at least once to either transmit or receive the data. Thus, $E^{CDS} = E_T^{CDS} + E_S^{CDS} \leq 2 \cdot E_T^{OPT} + 2 \cdot E_S^{OPT} \leq 2E^{OPT}$. □

**Theorem 8.** *The timespan in the CDS tree is a constant factor of that in the optimal tree.*

**Proof.** We schedule the nodes in two phases: first schedule all dominatee nodes using a simple greedy approach, and then, schedule the dominators. Let $T'$ be the time slots needed for scheduling the dominatees (sensors which are not in the backbone). To schedule the sensors in the backbone, we only need a constant number (denoted by $C$) time slots, because the nodal degree of every node is constant in the backbone. Therefore, the total time slots for the CDS tree is $T = T' + C$. While $T_{opt} \geq T'_{opt} \geq \frac{T'}{\alpha}$, where $\alpha$ is the independence number from Theorem 2, $T \leq \alpha \cdot T_{opt} + C \leq (\alpha + C) \cdot T_{opt}$. □

For the case of the data collection, we need to balance the time slots required by different nodes. The timespan is tightly related to $\mathbf{W}_i$ of every node $v_i$. It is also not difficult to prove that the CDS tree has constant approximation ratios on both the total energy consumption and the timespan needed for schedules. The proofs are similar, and thus, omitted here.

## 6 Performance Evaluation

We conduct extensive simulations to compare the performances of our proposed methods with some methods in the literature. In the results reported here, we did not compare our methods with a naive method schedule by random since our simulation results have shown that our method outperforms this method significantly. We will examine the energy consumption and the maximum time delay with various number of nodes, and at various data rates based on various tree structures in homogeneous networks. Here, the maximum time delay is the maximum time taken by the last data transmitted to the sink. In all scenarios, we compare our cluster-based method with the node-based method provided in [9]. In both methods, we implemented both centralized scheduling and distributed scheduling. Since the distributed scheduling is essentially same as centralized scheduling, we will only report the results of distributed scheduling here. We form different underlying tree structures to study these two methods. We have found that under our scheduling, nodes need to wake up at most twice to transmit or receive, while many nodes need to wake up numerous times (up to several 100 times, which is proportional to the total time slots required by this node) under the schedule by node-based method. At last, we also simulate the impact of various $\frac{R_{max}}{R_{min}}$ ratios on timespan in heterogeneous networks. And we compare the heterogeneous method with buckets with the cluster-based one in homogeneous network. All the following figures are simulations conducted in data collection network. The simulation results for data aggregation are similar and thus omitted.

### 6.1 Impact of Data Rate

In this simulation, we use different data rates to study how the data rate can affect energy consumption and data delay.

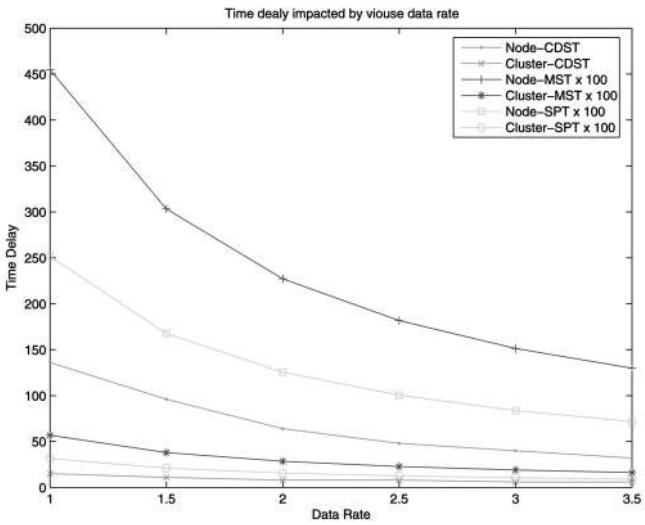



Fig. 5. Impact on the data delay with various data rates in homogeneous networks using CDS tree, MST tree, and SPT tree.

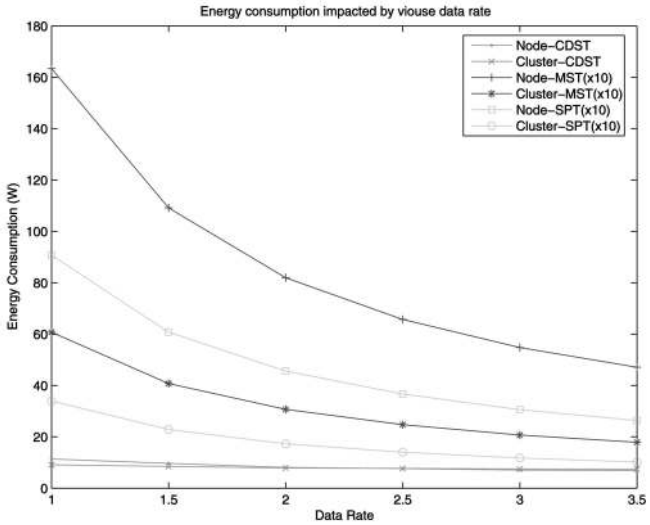



Fig. 6. Impact on the energy cost with various data rates in homogeneous networks using CDS tree, MST tree, and SPT tree.

We first construct a random sensor network by randomly placing 32 sensors in a square area of $5 \times 5$ square meters. We set the transmission radius as 1 meter and the interference radius as 2 meters. Then we construct different tree structures as the topology of the sensor network, CDS tree (CDST) [8] based on BFS tree, minimum spanning tree (MST) and the shortest path tree (SPT). Based on the topology, a sink node (with a fixed position) will collect data from the other sensors in the network. We simulate the sensor networks when data rate of links varies from 1 packets per slot to 3.5 packets per slot.

Figs. 5 and 6 show that energy consumption and maximum delay will decrease when the data rate increases, but the reduction is slower than the data rate increasing. Among different structures, we found that CDS tree has the smallest timespan, and therefore, the smallest energy consumption and data delay, while MST is always the worst with the largest among all trees. This is because CDS tree structure has the shortest hops from all other nodes to the sink. In MST, there is always long chains in the structure, which expands the timespan.

We also found that the schedule by our cluster-based method performs much better than the node-based method. In cluster-based method, to avoid potentially delaying the transmission to the next scheduling period, we intentionally schedule the sending time slots of a parent node after its children nodes' sending time slots. That is, the transmissions are always within one scheduling period. While in node-based method [9], the maximum delay for the data to its parent node may take several time periods because the node can only process one unit of (packets/data rate) per time slot, while it may have more data and need several time slots. In addition, if the parent node is scheduled before the child node, the data will be delayed at least one more time period. As expected, the energy consumption in cluster-based method is also much better than node-based method, as shown in Fig. 6. The energy consumption is dramatically reduced when scheduling the node to wake up at most twice in a time period.

### 6.2 Impact of Number of Nodes

In this simulation, we study how the number of nodes can affect the data delay and energy consumption. We vary the number of nodes from 29 to 39 in the sensor network. We set the transmission radius as 1 meter and the interference radius as 2 meters. We also use 1 packet per slot as the data rate.

We found that the maximum time delay increases with the number of nodes increases, which is also true for the energy consumption. This is because the number of interference clusters increases when the number of nodes increases. In addition, adding nodes also increases the total traffic load in the data collection model so that it will lengthen the timespan, and thus, the maximum delay and energy consumption. Figs. 7 and 8 also show that our cluster-based schedule consumes less energy and produces smaller time delay than node-based method in either CDS, MST, or SPT. This is because we use less state switches and assign the child node time slots earlier than its parent node.

### 6.3 Impact of Heterogeneous Nodes

We then study the impact of interference model on the network performances, especially the ratio of the maximal interference radius to the minimal interference radius. We also compare the heterogeneous method using buckets with the cluster-based method without using buckets. We vary the ratio $\log \frac{\max}{\min}$ from 1 to 2.1 in a sensor network with 64 nodes in a square region with side length 6 meters and data rate as 1 packet per slot.

Fig. 9 shows that the trend of timespan increases when the ratio of $\frac{R_{max}}{R_{min}}$ increases under CDS tree structure. This is because the number of buckets increases when the ratio of $\frac{R_{max}}{R_{min}}$ increases and some sensors, which can be scheduled with overlapped time slots, may be assigned into different buckets so that they will be scheduled sequentially. While the energy consumption does not vary significantly when the ratio of $\frac{R_{max}}{R_{min}}$ changes, it depends on the tree structure. From Fig. 9, we also observe that the timespan in the cluster-based method is higher than the method using buckets.

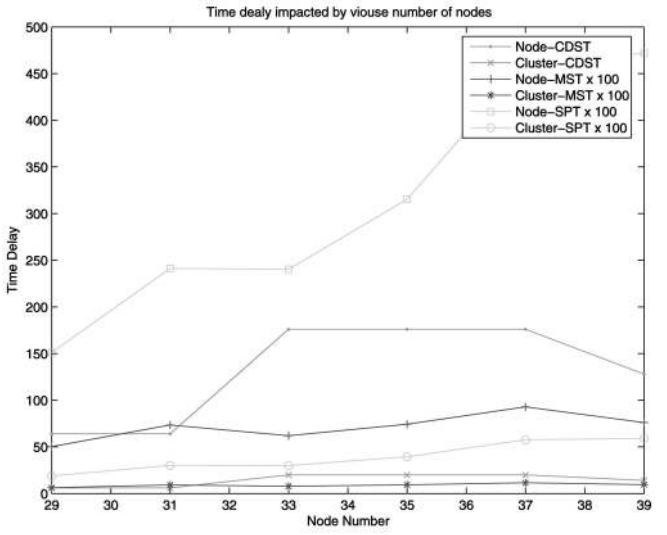



Fig. 7. Impact on the data delay with various number of nodes in homogeneous networks using CDS tree, MST tree, and SPT tree.

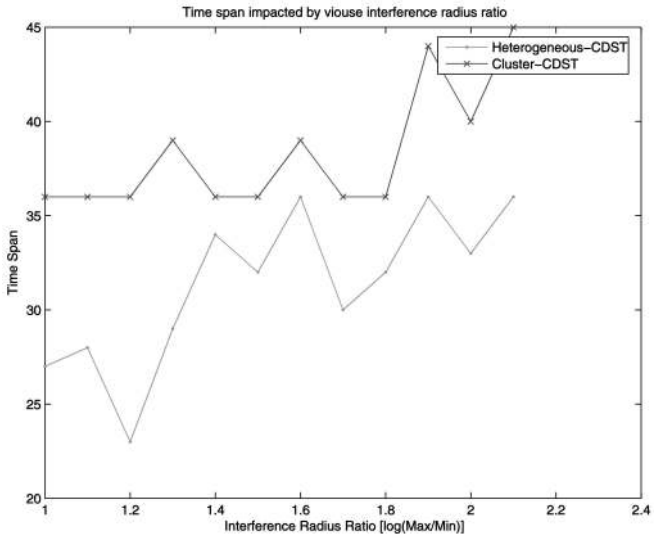



Fig. 9. Impact on the timespan with various $\frac{R_{max}}{R_{min}}$ ratios in heterogeneous networks using CDS tree.

## 7 Related Work

A number of MAC protocols have been proposed for WSNs. Polastre et al. proposed an MAC protocol, called B-MAC [10], which is used as the default MAC for Mica2. Buettner et al. proposed a new approach to low-power listening called X-MAC [3], which employs a short preamble to further reduce energy consumption and reduce latency. In [11], Rhee et al. presented the design, implementation, and performance evaluation of a hybrid MAC protocol, called **Z-MAC**, for WSNs that combines the strengths of TDMA and CSMA while offsetting their weaknesses. Ahn et al. proposed a new MAC protocol called *Funneling-MAC* [12] that uses the CSMA MAC protocol for nodes far away from the sink and uses the TDMA protocol for nodes that are close to the sink.

Scheduling has been studied extensively [9], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. Arisha et al. [20] proposed an energy-aware TDMA-based scheduling. But they did not consider parallel transmission in the network. Thus, the proposed scheduling could have a large delay. Ergen and Varaiya [9] proposed a TDMA scheduling algorithm to improve the delay in the network. However, they did not consider the energy consumption. Moreover, they did not consider the weight of node and the transmission order of the parent node and child node. In [23], Solis and Obraczka studied the trade-off of energy efficiency and delay for data aggregation. In [24], Dai et al. proposed the algorithm to save energy by reducing redundant data. In [25], Rajagopalan and Varshney gave a survey for data aggregation techniques.

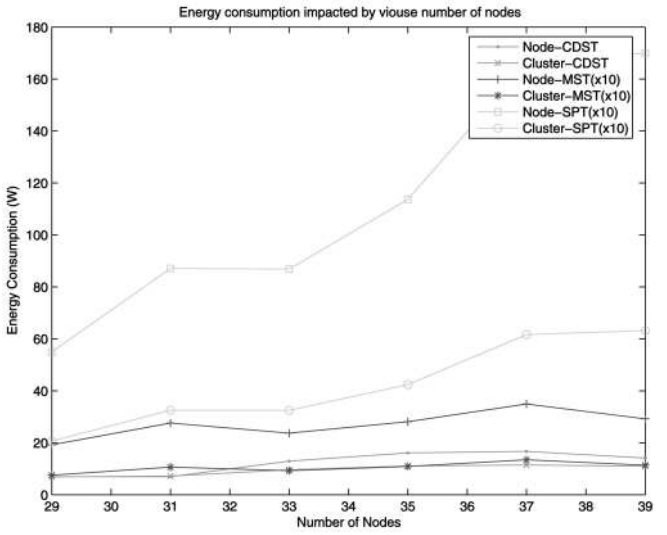



Fig. 8. Impact on the energy cost with various number of nodes in homogeneous networks using CDS tree, MST tree, and SPT tree.

To save energy consumption, the wake-up scheduling has been widely used. Keshavarzian et al. [26] analyzed different wake-up scheduling schemes and proposed a new scheduling method that can decrease the end-to-end overall delay. However, they did not consider the time-slot assignment problem to avoid interference. TDMA-based wake-up scheduling can provide both energy-efficient and conflict-free channel access [6], [27]. TDMA-based scheduling algorithms that minimize the number of time slots or the message delay are proved NP-complete [28], [29]. Approximate algorithms have been therefore proposed, including both link scheduling [6], [16], [27] and broadcast scheduling [30], [31], [32]. Link scheduling and broadcast scheduling are time-slot assignments to links and nodes, which can be reduced to different coloring problems: *edge coloring* and *vertex coloring*. Panconesi and Srinivasan [33] proposed a randomized distributed edge coloring method that uses at most $2\Delta + 1$ colors. Ramanathan [15] proposed a unified framework for TDMA-, FDMA-, and CDMA-based multihop wireless networks. Krumke et al. [31] proposed the efficient approximation algorithms for the distance-2 vertex coloring problem for various geometric graphs. In [34], Kumar et al. studied packet scheduling under RTS/CTS interference model and gave polylogarithmic/constant factor approximation algorithms for various families of disk graphs and randomized near-optimal approximation

algorithms for general graphs. Recently, Moscibroda and Wattenhofer [35] proposed an $O(\Delta)$ distributed coloring method with time complexity $O(\Delta \log n)$. Wang et al. [6] also proposed both centralized and distributed interference-aware link scheduling algorithms to maximize the throughput of the network.

## 8 Conclusion

In this paper, we propose efficient centralized and distributed scheduling algorithms that not only remove the unnecessary listening cost, but also reduce the energy cost for state switching and clock synchronization. In our protocol, every node needs only to wake up at most twice in one scheduling period: one for receiving data from its children and one for sending data to its parent. We have also proposed an efficient method to construct energy-efficient data gathering tree, whose energy cost and timespan of the scheduling are both within constants times of the optimum. Our extensive simulation results have verified our theoretical statements. An interesting question left for the future research is to design efficient data collection or data aggregation tree and nodes' activity scheduling algorithm such that the data delay, the number of messages needed for collection (or aggregation), and the energy cost are all almost optimum. If this is impossible, then what are the best trade-offs among these potentially conflicting objectives?

## Acknowledgments

The research of authors is partially supported by US National Science Foundation (NSF) CNS-0832120, National Natural Science Foundation of China under Grant No. 60828003, the Natural Science Foundation of Zhejiang Province under Grant No. Z1080979, the National Basic Research Program of China (973 Program) under grant No. 2006CB30300 and No. 2010CB328100, the National High Tech. Research and Development Program of China (863 Program) under grant No. 2007AA01Z180, the Hong Kong CERG under Grant PolyU-5232/07E, and Hong Kong RGC HKUST 6169/07.

## References

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* pp. 88-97, 2002.

[2] M.J. Miller and N.H. Vaidya, "Efficient Bounds for the Stable Set, Vertex Cover, and Set Packing Problem," *Proc. Wireless Comm. and Networking Conf.,* vol. 4, pp. 2335-2340, 2004.

[3] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-mac: A Short Preamble mac Protocol for Duty-Cycledwireless Sensor Networks," *Proc. First ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2006.

[4] M. Alicherry, R. Bhatia, and L.E. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks," *Proc. ACM MobiCom,* pp. 58-72, 2005.

[5] V.S.A. Kumar, M.V. Marathe, S. Parthasarathy, and A. Srinivasan, "Algorithmic Aspects of Capacity in Wireless Networks," *SIGMETRICS Performance Evaluation Rev.,* vol. 33, no. 1, pp. 133-144, 2005.

[6] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, "Efficient Interference Aware tdma Link Scheduling for Static Wireless Mesh Networks," *Proc. ACM MobiCom,* 2006.

[7] G. Călinescu, "Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks," *Proc. Int'l Conf. AD-HOC Networks and Wireless (AdHoc-Now '03),* 2003.

[8] P.-J. Wan, K.M. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2002.

[9] S.C. Ergen and P. Varaiya, "TDMA Scheduling Algorithms for Sensor Networks," technical report, Univ. of California, Berkley, 2005.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2004.

[11] I. Rhee, A. Warrier, M. Aia, and J. Min, "ZMAC: A Hybrid MAC for Wireless Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2005.

[12] G.-S. Ahn, E. Miluzzo, A.T. Campbell, S.G. Hong, and F. Cuomo, "Funneling-mac: A Localized, Sink-Oriented MAC for Boosting Fidelity in Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2006.

[13] L. Bao and J.J. Garcia-Luna-Aceves, "Channel Access Scheduling in Ad Hoc Networks with Unidirectional Links," *Proc. Fifth Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (DIALM '01),* pp. 9-18, 2001.

[14] S. Ramanathan and E.L. Lloyd, "Scheduling Algorithms for Multi-Hop Radio Networks," *Proc. ACM SIGCOMM,* pp. 211-222, 1992.

[15] S. Ramanathan, "A Unified Framework and Algorithm for Channel Assignment in Wireless Networks," *Wireless Network,* vol. 5, no. 2, pp. 81-94, 1999.

[16] R. Liu and E.L. Lloyd, "A Distributed Protocol for Adaptive Link Scheduling in Ad-Hoc Networks," *Proc. IASTED Int'l Conf. Wireless and Optical Comm. (WOC),* 2001.

[17] L. Bao and J. Garcia-Luna-Aceves, "Transmission Scheduling in Ad Hoc Networks with Directional Antennas," *Proc. ACM MobiCom,* pp. 48-58, 2002.

[18] X. Yu, S. Mehrotra, and N. Venkatasubramanian, "Sensor Scheduling for Aggregate Monitoring in Wireless Sensor Networks," *Proc. Int'l Conf. Scientific and Statistical Database Management,* p. 24, 2007.

[19] S.S. Kulkarni and M. Arumugam, "Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks," *Int'l J. Distributed Sensor Networks,* vol. 2, pp. 55-78, 2006.

[20] K.A. Arisha, M.A. Youssef, and M.F. Younis, "Energy-Aware TDMA-Based MAC for Sensor Networks," *Proc. IEEE Workshop Integrated Management of Power Aware Comm., Computer and Networking,* 2002.

[21] S. Cui, R. Madan, A. Goldsmith, and S. Lall, "Energy-Delay Tradeoffs for Data Collection in tdma-Based Sensor Networks," *Proc. 40th Ann. IEEE Int'l Conf. Comm.,* 2005.

[22] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DeSync: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks," *Proc. Sixth Int'l Conf. Information Processing in Sensor Networks (IPSN),* pp. 11-20, 2007.

[23] I. Solis and K. Obraczka, "In-Network Aggregation Trade Offs for Data Collection in Wireless Sensor Networks," *Int'l J. Sensor Networks,* vol. 1, pp. 200-212, 2006.

[24] X. Dai, F. Xia, Z. Wang, and Y. Sun, "An Energy-Efficient In-Network Aggregation Query Algorithm for Wireless Sensor Networks," *Proc. Int'l Conf. Innovative Computing, Information and Control,* vol. 3, pp. 255-258, 2006.

[25] R. Rajagopalan and P. Varshney, "Data-Aggregation Techniques in Sensor Networks: A Survey," *IEEE Comm. Surveys and Tutorials,* vol. 8, no. 4, pp. 48-63, Quarter 2006.

[26] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup Scheduling in Wireless Sensor Networks," *Proc. ACM MobiHoc,* pp. 322-333, 2006.

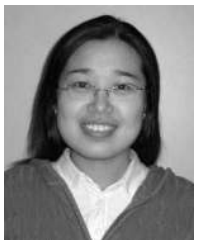
[27] P. Djukic and S. Valaee, "Link Scheduling for Minimum Delay in Spatial Re-Use tdma," *Proc. IEEE INFOCOM,* pp. 28-36, May 2007.

[28] E. Arikan, "Some Complexity Results about Packet Radio Networks," *IEEE Trans. Information Theory,* vol. IT-30, no. 4, pp. 681-685, July 1984.

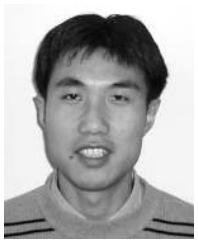
[29] A. Ephremedis and T. Truong, "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Trans. Comm.,* vol. 38, no. 4, pp. 456-460, Apr. 1990.

[30] R. Ramaswami and K.K. Parhi, "Distributed Scheduling of Broadcasts in a Radio Network," *Proc. IEEE INFOCOM,* pp. 497-504, 1989.

[31] S. Krumke, M. Marathe, and S. Ravi, "Models and Approximation Algorithms for Channel Assignment in Radio Networks," *ACM Wireless Networks,* vol. 7, no. 6, pp. 575-584, 2001.
[32] C.Y. Ngo and V.O.K. Li, "Centralized Broadcast Scheduling in Packet Radio Networks via Genetic-Fix Algorithms," *IEEE Trans. Comm.,* vol. 51, no. 9, pp. 1439-1441, Sept. 2003.
[33] A. Panconesi and A. Srinivasan, "Improved Distributed Algorithms for Coloring and Network Decomposition Problems," *Proc. ACM Symp. Theory of Computing,* pp. 581-592, 1992.
[34] A. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan, "End-to-End Packet-Scheduling in Wireless Ad-Hoc Networks," *Proc. ACM Symp. Discrete Algorithms (SODA),* pp. 1021-1030, 2004.
[35] T. Moscibroda and R. Wattenhofer, "Coloring Unstructured Radio Networks," *Proc. 17th Ann. ACM Symp. Parallelism in Algorithms and Architectures (SPAA '05),* pp. 39-48, 2005.



**Yanwei Wu** received the BEng and ME degrees from Tianjin University, PR China, in 1998 and 2003, respectively. She received PhD degree of computer science from the Illinois Institute of Technology in 2009. She is currently an assistant professor of computer science at Minnesota State University, Mankato. Her research interests span wireless networks, game-theoretical study of networks, optimization in mesh network, energy efficiency, and security in wireless network. She also researched on agent-based modeling as a research aid at Argonne National Laboratory in 2007. She is a member of the IEEE.



**Xiang-Yang Li** received the BEng degree in computer science and the Bachelor's degree in business management from Tsinghua University, PR China, in 1995, and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign, in 2000 and 2001, respectively. He has been an associate professor since 2006 and was an assistant professor of computer science in the Illinois Institute of Technology from 2000 to 2006. He was a visiting professor of Microsoft Research Asia from May 2007 to August 2008. His research interests span wireless ad hoc and sensor networks, noncooperative computing, computational geometry, and algorithms. He was a guest editor of special issues for *ACM Mobile Networks and Applications* and the *IEEE Journal on Selected Areas in Communications*. He is a senior member of the IEEE.

**YunHao Liu** received the BS degree from the Automation Department, Tsinghua University, China, in 1995, the MA degree from Beijing Foreign Studies University, China, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University in 2003 and 2004, respectively. He is now an associate professor in the Department of Computer Science and Engineering at Hong Kong University of Science and Technology. His research interests include sensor networking, pervasive computing, and peer-to-peer computing. He is a senior member of the IEEE and the IEEE Computer Society.

**Wei Lou** received the BE degree in electrical engineering from Tsinghua University, China, in 1995, the ME degree in telecommunications from Beijing University of Posts and Telecommunications, China, in 1998, and the PhD degree in computer engineering from Florida Atlantic University, in 2004. He is currently an assistant professor in the Department of Computing, The Hong Kong Polytechnic University, HKSAR, China. His current research interests are in the areas of mobile ad hoc and sensor networks, peer-to-peer networks, and mobile computing. He has worked intensively on designing, analyzing and evaluating practical algorithms with the theoretical basis, as well as building prototype systems. His research work is supported by several Hong Kong GRF grants and Hong Kong Polytechnic University ICRG grants.