

# Energy-Aware Wake-up Scheduling for Data Collection and Aggregation

Xiang-Yang Li<sup>‡,\*</sup> Yanwei Wu<sup>\*</sup>,

<sup>\*</sup>Department of Computer Science, Illinois Institute of Technology

<sup>‡</sup>Microsoft Research Asia, BeiJing

**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 design both centralized and distributed energy efficient and interference-aware algorithms for homogeneous networks and heterogeneous networks. 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 a constant times that of the optimum and 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 efficient 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 reduce considerable energy consumption.

**Index Terms**—Wireless sensor networks, energy consumption, MAC, scheduling, routing, optimization

## I. INTRODUCTION

The recent advances in micro-sensor technology and low-power analog/digital electronics, have led to the development of distributed, wireless networks of sensor devices. 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. WSNs could operate in an event-driven model or regular continuous monitoring model. In an event-driven model, when some abnormal event happens, the sensor will detect it and send the data to the sink node, possibly via a multiple-hop relays of some other sensors. In a regular continuous monitoring model, each sensor will monitor its vicinity and periodically sends its collected information to the sink possibly via the relay of other sensors.

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, sometimes a wireless sensor will process the data collected from its children sensors and then relay the processed data to its parent. When a sensor node simply relay the data from its children directly to its parent, this is called *data collection*. Although in this paper, 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 (and thus multiple trees) for data gathering.

Efficient TDMA scheduling has been extensively studied recently for wireless networks. These previous studies did not consider all possible energy consumption by wireless sensors, especially the energy consumed during useless listening, and the state transitions (such as from idle state to listening state, from sleep state to transmitting state and so on). Typically, a wireless sensor node will wake up periodically to sense the environment (using sensing device), to process the sensed data (using computing component), to send some data to the sink node (by switching the radio to transmitting mode), 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 much 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 micro-amps as opposed to milli-amps. 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 time-slots for transmitting and receiving, a scheduling should reduce the state transitions to increase the life-time of a sensor. For example, the ATMega 128L processor of Mica mote sensor takes  $4ms$  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  $25\mu s$ ; 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 follows. To reduce the energy cost, we would need to minimize the wake-up times in a scheduling period, when the number of transitions 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 without being idle. We design an energy efficient scheduling in which any sensor node wakes up at most *twice*: 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 every wireless sensor node  $v$  has 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  $R_T(v)$ . Each wireless sensor node  $v$  will periodically monitor its vicinity and produce data at rate  $\ell(v)$ . 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, no matter whether wireless sensor nodes may have the data aggregation ability. Both centralized algorithm and communication efficient distributed algorithm with at most  $O(n)$  messages are proposed to decide 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. The basic idea of our protocol for heterogeneous WSNs is that we put the sensor nodes with similar interference range into a bucket and then schedule the nodes in each bucket respectively. We also conduct extensive simulations to show that the proposed algorithms are energy efficient.

The rest of the paper is organized as follows. In Section II, we present our network model, present the problems to be studied. We provide centralized and distributed algorithm for homogeneous network in Section III and for heterogeneous network in Section IV. We report our simulation results that compare the performance of our methods with existing methods in Section VI. We review the related work in Section VII and conclude our paper in Section VIII.

## II. SYSTEM MODEL AND ASSUMPTION

### A. Network System Models

Assume that there is a set  $V = \{v_1, v_2, \dots, v_n\}$  of  $n$  sensors deployed in a 2-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 simply 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  $v_i v_j$  is reliable if  $v_i$  can communicate with  $v_j$ . With the fixed transmitting power, all the communications are 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, although relaxing this assumption will not affect the

correctness of most 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  $\|v_k - v_i\| \leq R_I(v_i)$ .

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, writing/erasing data in the flash [18], [20]. 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 power consumption (energy consumption per unit time) of  $P_{tx}$ ,  $P_{rcv}$ ,  $P_{lst}$  and  $P_{slp}$ , respectively. Table I 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 *4ms*.

TABLE I  
ENERGY-COST SYMBOLS.

Symbol	Meaning	Typical Value
$P_{tx}$	Power consumption in transmitting	60mW
$P_{rcv}$	Power consumption in receiving	45mW
$P_{lst}$	Power consumption in listening	45mW
$P_{slp}$	Power 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	$\sim 10$

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  $t_s$ . A schedule period  $T$  is composed of  $T$  consecutive time-slots. The activities of every node is then repeated with period  $T$ . For simplicity, we assume that the number of data packets each node will transmit is always a numerable number.

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 is 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 with  $k$  neighboring nodes 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  $k$ .

### B. 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.

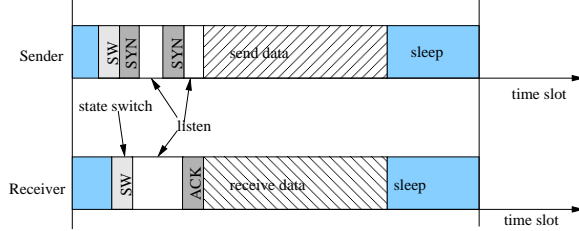


Fig. 1. the synchronization between two nodes.

**Energy-Efficient Scheduling:** First, we assume that a tree  $\mathbf{T}$  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  $1 \leq t \leq T$  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 performed, nodes will also have additional state *listening* so adjacent nodes can synchronize their activities. See Figure 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 [7]). 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  $X_{i,S,t} = 0$ . If the node  $v_i$  is scheduled to receive the data at time-slot  $t$ , we denote it as  $X_{i,R,t} = 1$ ; otherwise we denote it as  $X_{i,R,t} = 0$ . We use variables  $X_{i,P,t} \in \{0, 1\}$  and  $X_{i,L,t} \in \{0, 1\}$  to denote whether the node  $v_i$  is scheduled to sleep or listen at time-slot  $t$  or not. We also denote the energy consumption per time-slot in transmitting, listening, sleeping as  $P_{tx}$ ,  $P_{rcv}$  and  $P_{slp}$  respectively. We use  $E_{P,S}$  to denote the consumed energy when a node switches from state sleeping to state transmitting. Similarly, we use  $E_{P,R}$  to denote the consumed energy for switching from state sleeping to state receiving. 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  $v_i$  at 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 above energy-costs.

Clearly, there are numerous schedules for wireless sensor nodes' activities. We can only 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  $\mathbf{T}$ , required by node  $v_i$  is  $0 \leq w_i \leq T$ . Here  $w_i$  is computed based on the amount of information received from its children nodes in the data collection tree  $\mathbf{T}$  and the amount of data produced by its own sensor. If node  $v_i$  does not have data aggregation ability,  $w_i$  is simply  $\lceil \frac{W+r_{v_i}}{P_s} \rceil$ , where  $W$  is the total number of packets received from its children nodes in a period. When node  $v_i$  has the aggregation ability,  $w_i$  is  $\lceil \frac{f(W, r_{v_i})}{P_s} \rceil$ , 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  $\sum_{t=1}^T X_{i,S,t} \geq w_i/p$ .

2) Obviously, a node  $v_i$  with children nodes  $u_1, u_2, \dots, u_d$  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  $X_{i,R,t} \geq X_{j,S,t}$  whenever node  $v_i$  is parent of node  $v_j$  in tree  $\mathbf{T}$ .

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.*,  $X_{i,S,t} + X_{k,S,t} \leq 1$  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  $X_{i,S,t} + X_{k,S,t} \leq 1$  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 data to it. However, when the dominator node dominates  $k$  sensors, it will wake up for  $k$  times to receive all the data from 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 are 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 need 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 Figure 2 for an illustration of possible schedule for a node.

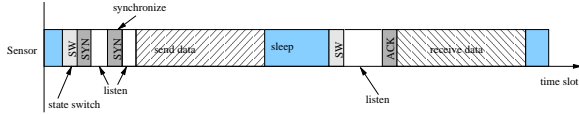


Fig. 2. The time-slots for a sensor in one period.

**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 [2], [16], [31] have been proposed for various interference models.

### III. 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.

#### A. 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.* [31]) 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. The children nodes of any sensor node  $v_i$  constitute a virtual cluster  $\mathbf{C}_i$ . For each cluster  $\mathbf{C}_i$  (formed by all children nodes of a node  $v_i$ ), we define  $\mathbf{W}_i = \sum_{v_j \in \mathbf{C}_i} w_j$  as its *weight*. Here  $\mathbf{C}_i$  denotes the set of children nodes of node  $v_i$  in data gathering tree  $\mathbf{T}$ . Obviously,  $\mathbf{W}_i = \sum_{v_j \in \mathbf{C}_i} w_j$  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 consuming due to the state transition will be definitely saved since each node only need to wake up twice: once for receiving all data from its children and once for transmitting its data to its own parent.

For easy description, we use the following definition of conflicting clusters.

*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.

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; nodes  $u_2$  uses time-slots 3, 4; node  $u_3$  uses time-slots 3, 4; nodes  $u_4$  uses time-slots 1, 2. Based on this observation, our schedule looks like over-pessimistic. However, we can prove that the maximum period required by our schedule is only a constant times of the optimum solution.

**Algorithm:** 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  $v_i$ , if distributed method is required. 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 non-increasing 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  $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_n$  is the sorted receiving-

weights in non-increasing order. Then we schedule the sensors sequentially. For sensor  $v_i$ , we search in the time-slots which have already been scheduled. If there are consecutive  $\mathbf{W}_i$  time-slots which are not occupied by any already-scheduled conflicting cluster  $\mathbf{C}_j$  with  $j < i$ , we schedule cluster  $\mathbf{C}_j$  to these time-slots. Otherwise, we schedule cluster  $\mathbf{C}_i$  to some new time-slots.

---

**Algorithm 1** Centralized Activity Schedule using **T**


---

- 1: **for** each sensor  $v_i$  **do**
  - 2:   calculate its receiving-weight  $\mathbf{W}_i = \sum_{v_j \in \mathcal{C}_T(v_i)} w_j$ .
  - 3:   Sort the sensor in non-increasing 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.
  - 6:   Each sensor node  $v_j$  in  $\mathbf{C}_i$  will be sequentially assigned  $w_j$  consecutive time-slots for transmitting.
- 

**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 time-span  $T(\mathcal{S})$  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-slots 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 2 state switching for each node. So the total state switching 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 switching in our scheduling and  $E_s$  is the energy consumption for the state switching of one sensor and  $n$  is the number of sensors. Because there is at least one state switching 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 switching. 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 time-span 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 Figure 3. In the figure,  $\mathbf{W}_{j_1}, \mathbf{W}_{j_2}, \dots, \mathbf{W}_{j_k}$

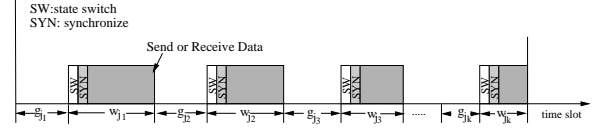


Fig. 3. Illustration of the scheduled slots before scheduling  $\mathbf{W}_i$  receiving slots for node  $v_i$ .

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 non-conflicting 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  $\mathbf{W}_{j_1}, \mathbf{W}_{j_2}, \dots, \mathbf{W}_{j_k}$  is no smaller than the time-slots  $\mathbf{W}_i$  required by sensor  $v_i$  for receiving since we scheduled the sensors in the decreasing order of the receiving-weight  $\mathbf{W}_i$ . In addition, in the figure  $g_{j_1}, g_{j_2}, \dots, g_{j_k}$  represent the time-slots occupied by some other sensors, which do *not* conflict with cluster  $\mathbf{C}_i$ . Notice that some of  $g_{j_1}, g_{j_2}, \dots, g_{j_k}$  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

$$\begin{aligned}
 T_i &= \sum_{l=1}^k g_{i_l} + \sum_{l=1}^k \mathbf{W}_{i_l} + \mathbf{W}_i \\
 &\leq k \cdot \mathbf{W}_i + \sum_{l=1}^k \mathbf{W}_{i_l} + \mathbf{W}_i \\
 &\leq 2(\sum_{l=1}^k \mathbf{W}_{i_l} + \mathbf{W}_i)
 \end{aligned}$$

Recall that we assumed that sensor node  $v_i$  has the largest time-slots for receiving. According to the above result, we get the time-span  $T$  used by our schedule satisfies  $T \leq 2(\sum_{l=1}^k \mathbf{W}_{i_l} + \mathbf{W}_i)$ . We denote  $T_{opt}$  as the smallest time-span by any schedule. Consider the time-slots needed by an optimum schedule for the following clusters  $\mathbf{C}_i, \mathbf{C}_{j_1}, \mathbf{C}_{j_2}, \dots, \mathbf{C}_{j_k}$ . Here  $\mathbf{C}_{j_1}, \mathbf{C}_{j_2}, \dots, \mathbf{C}_{j_k}$  are *all* clusters that conflict with cluster  $\mathbf{C}_i$  and has a larger 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 depending on the interference model and will be specified later. Combining with  $T \leq 2(\sum_{l=1}^k \mathbf{W}_{j_l} + \mathbf{W}_i)$ , we get  $T \leq 2\alpha \cdot T_{opt}$ .

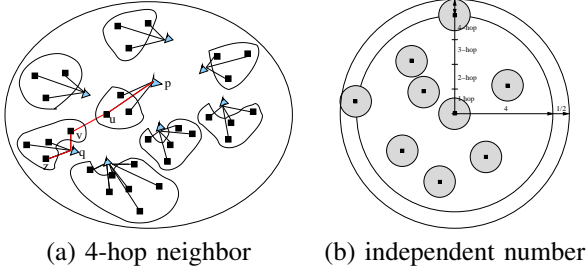


Fig. 4. Illustration of 4-hop distance

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  $\mathbf{C}_{j_l}$ . Figure 4 (a) 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 one-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 Figure 4 (b). The total time-slots 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 such nodes at any timeslot is at most  $\frac{(3\frac{1}{2})^2}{(\frac{\beta}{1+\beta})^2} = \frac{49(1+\beta)^2}{\beta^2}$ . Let  $\alpha = \frac{49(1+\beta)^2}{\beta^2}$ . Thus, the total time-slots 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 time-span 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  $\mathbf{r} = \langle r_{v_1}, r_{v_2}, \dots, r_{v_{n-1}}, r_{v_n} \rangle$  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  $\mathbf{r}' = \langle r'_{v_1}, r'_{v_2}, \dots, r'_{v_{n-1}}, r'_{v_n} \rangle$  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.

**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  $v$  is 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 time-slots smaller, we use time-slots from later to earlier. In other words, the cluster with the largest weight, say  $\mathbf{W}_1$ , will be scheduled in time-slots  $[T - \mathbf{W}_1 + 1, T]$ , where  $T$  is the scheduling period. 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**<sup>1</sup>: The first case is that 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. In this case,  $\mathbf{C}_i$  will be scheduled in time-slots  $[t_i - \mathbf{W}_i, t_i - 1]$ .
- 2) **Exist Big Gaps**: The second case is that 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

<sup>1</sup>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  $\mathbf{C}_i$ .

schedule the time-slots for transmitting by nodes in  $C_i$  in the latest gap, *i.e.*, closest to time  $T$ .

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

Another thing that we can make further improvement (although it is at most a constant time factor) is to reduce the time-span of the resulting schedule. Remember that we say that two clusters  $C_i$  and  $C_j$  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 overlap. Thus, we can modify our scheduling further to reduce the time-span of the schedule: we try to schedule  $C_i$  to latest consecutive  $W_i$  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**.

### B. Distributed Activity Scheduling

For sensor networks, distributed scheduling of activities is often preferred over centralized scheduling. We then design a distributed activities scheduling algorithm. The method is essentially same as the centralized scheduling with the following differences. We replace the global sorting of the weight of clusters with the local sorting: a cluster can schedule its transmitting activities only if all *conflicting* clusters with *larger* weight has been scheduled. Secondly, for easy maintenance, each sensor node  $v_i$  will be responsible for the scheduling of transmitting activities of its children nodes  $C_i$  (equivalently, the receiving activity of itself). To inform potentially conflicting clusters about its own weight  $W_i$ , sensor node  $v_i$  will first multicast a *request* message (including its weight  $W_i$ ) to its  $k$ -hop neighbors to request for scheduling. Notice, based on previous analysis, we will choose  $k = 4$  in our algorithm. If the sensor does not have the largest weight, it will wait till it has the largest weight among its  $k$ -hop non-scheduled 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 $T$ by $v_i$

---

- 1: The sensor  $v_i$  first computes  $W_i$  based on the data rates from its children in  $C_i$ . It sends a *request* message to all sensors that are within a constant  $k$ -hops to request for scheduling. This can be done using a simple flooding with  $TTL = k$ .
  - 2: **if**  $W_i \geq W_j$  for each non-scheduled  $k$ -hop neighbor  $v_j$  **then**
  - 3:   Schedule  $v_i$  earliest  $W_i$  available time-slots for receiving data and sequentially schedule the time-slots for transmitting for every sensor in  $C_i$ .
  - 4:   Send a *scheduled* message to all its unscheduled  $k$ -hop neighbors.
- 

*Theorem 3:* The time-span 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 messages used in Algorithm 2 are linear.

*Proof:* Suppose the maximal  $k$ -hop neighbors are  $\Delta_k$ , which is a number often much smaller than  $n$ . The total number of the *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 [11] 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 sequentially. 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  $n, n-1, n-2, \dots, 1$ . 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.

## IV. 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 time-span used by a schedule. To schedule the activities of sensors for heterogeneous sensors, we will firstly 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 non-increasing 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 time-span 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.

*Theorem 5:* The time-span 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

---

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


---

- 1: **for** each sensor  $v_i$  **do**
  - 2:   Insert the sensor into bucket  $[2^{i-1}R_{\min}, 2^iR_{\min})$  in non-increasing order according to the weight  $\mathbf{W}_i$  if its interference radius is within  $[2^{i-1}R_{\min}, 2^iR_{\min})$ .
  - 3: **for** each bucket  $b_i$  **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.
- 

$\Theta(\log \frac{R_{\max}}{R_{\min}})$  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 also can prove that Algorithm 1 produces a schedule whose time-span is at most  $\Theta(\rho_k^2)$  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 heterogeneous network. The scheduling is similar to the centralized algorithm except that the sensors will find a schedule by collecting the information within  $k$ -hop firstly. 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.

---

**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  $[2^{l-1}R_{\min}, 2^lR_{\min})$  run the distributed scheduling.
- 

Similar to Theorem 5, the next theorem is straightforward.

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

#### V. FORMATION OF DATA GATHERING TREE

Obviously, the performance of our scheduling algorithm depending 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 time-span of a scheduling, the shortest-hop path tree may be not 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 time-span of the best scheduling  $\mathcal{S}_2$  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, 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 **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 time-span 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 aggregation tree for homogeneous networks. Our tree will be based on a connected dominating set (CDS). A number of message efficient distributed methods have been proposed for constructing CDS, *e.g.* [3]. In this kind of CDS structure, the total number of neighbors in the backbone of any node is bounded by a constant. Given the CDS constructed as that in [3], we then construct a BFS tree on top of this CDS, and 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 result proved in [3],  $h$  satisfies that  $h \leq 3 \cdot h_0$ , 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 least energy consumption for transmitting, receiving, and switching states. Then  $E_T^{CDS} \leq 3 \cdot E_T^{BFS} \leq 3 \cdot E_T^{OPT}$ , where  $E_T^H$  is the energy consumption in active states by a tree  $H$ . Similarly, let  $E_S^{H,S}$  be the energy consumption for state switching in a structure  $H$  using a schedule  $S$ .

A node wakes up from the sleeping state to a active state at most twice in our scheduling: one from the sleeping state to transmitting state and the other one from the sleeping state to 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 the state. 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,  $E^{CDS} = E_T^{CDS} + E_S^{CDS} \leq 3 \cdot E_T^{OPT} + 2 \cdot E_S^{OPT} \leq 3E^{OPT}$ . ■

*Theorem 8:* The time-span 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 independent 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 time-span is tightly related to  $W_i$  of every node  $v_i$ . We leave it as a future work to design a data collection tree that has constant approximation ratios on both the total energy consumption and the time-span needed for schedules.

## VI. PERFORMANCE EVALUATION

In this section, we will exam the time-span, the energy consumption and the average data delay with various number of nodes, various data rate or various tree structures both in homogeneous networks and heterogeneous networks. We will compare the following schedule methods: the schedule by cluster, the schedule by cluster with squeeze, and the schedule by random. As we mentioned in the former section, the performance may be improved by adjusting the sensors in two interference clusters. For two interference clusters, especially the ones which interfere each other only due to the interference of a few sensors in the clusters, the interference between them may be easily avoided just by not overlapping the time-slots of the interference sensors. Therefore, the two clusters could be scheduled in some overlapped time-slots even there are potential interferences. We call it *schedule by cluster and squeeze*. Otherwise, if we randomly choose a sensor to schedule until all the sensors in the network are scheduled, we call it *schedule by random*. We will compare these three methods both in data collection model and data aggregation model. We also form different underlying tree structures to study these methods. All simulation results reported here are the average values of 50 runs of different instances. We have found that under our scheduling, majority nodes only need to wake up once, while many nodes need to wake up numerous times (up to several hundred times, which is proportional to the total time-slots required by this node) under the schedule by random method.

### A. Impact of Data Rate

In this simulation, we use different data rates to study how the data rate can affect the time-span, energy consumption

and data delay. We firstly construct a random sensor network by randomly dispersing 300 sensors in a square area of  $500 \times 500$  square meters. The sensor network is connected with high probability because the network is generated under the constraint  $n\pi r^2 \geq c \log n$ , where  $n$  is the number of nodes,  $r$  is the transmission radius and  $c$  is some constant. We set the transmission radius as 75 meter and the interference radius is thus  $2 \times 75$  meter. Then we construct different tree structures as the topology of the sensor network. Based on the topology, a sink node (with a fixed position) will collect data from the other sensors in the network. We simulate in the sensor network with 300 nodes when data rate of links varies from 5 packets per slot to 10 packets per slot.

Figure 5 (for the data collection) and Figure 6 (for the data aggregation) illustrate that time-span, energy consumption and average data delay will decrease when the data rate increases. It is obvious due to the reduction of time-slots needed for each node. Among different structures, we have found that SPT has the smallest time-span, energy consumption, and data delay, while MST is always the worst with the largest among all three. We have also found that the schedule by cluster and squeeze method produces the least energy consumption. However, it does not always produce the smallest time-span, which may due to the reason that the aggressively scheduling earlier clusters actually reduces the number of gaps that can be used by later clusters, which have to use *new* time-slots afterwards. although, given the fixed network, the time-slots required by each node will *not* increase, the energy consumption is not guaranteed to decrease when the link capacity (called data rate here) increases. This is because the sorting of the number of slots required by all nodes could change (which implies different schedules).

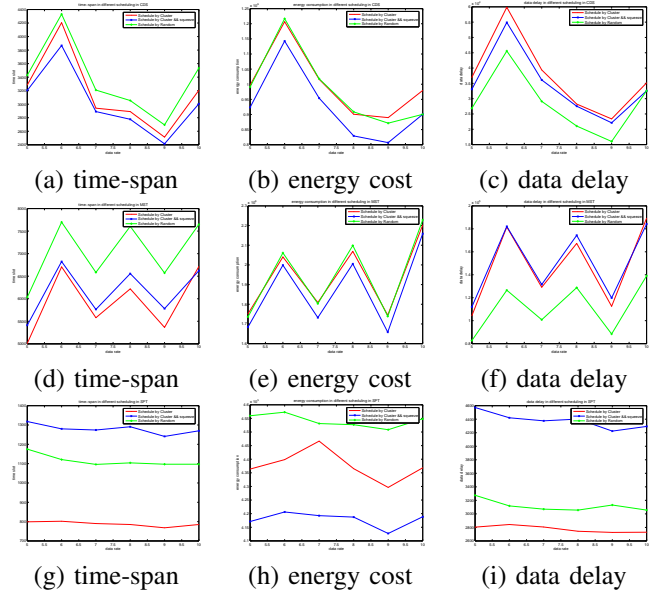


Fig. 5. Impact on the time-span, energy cost and data delay with various data rates in homogeneous networks by CDS tree (1st row), MST tree (2nd row) and SPT tree (3rd row).

## B. Impact of Nodes

In this simulation, we use different number of nodes to study how the number of nodes can affect the time-span, energy consumption and data delay when the nodes vary from 200 to 400 in the sensor network. We set the transmission radius as 75 meter and the interference radius is thus  $2 \times 75$  meter. We also use 5 packets per slot as the data rate.

We have found that, both in the data collection model and the data aggregation model, the time-span increases with the number of nodes increases, which is also true for the average data delay. This is because the number of interference clusters increases when the number of nodes increases. In addition, adding nodes also increases the total data traffic in the data collection model so that it will lengthen the time-span and the average data delay. Although the energy consumption in both models will increase when the number of nodes increases, as shown in Figure 7 and Figure 8, the energy consumption of the root may not increase in the data aggregation model, because the time-slots needed for the root may not change. Figure 7 and Figure 8 also show that our schedule by cluster method consumes less energy than the other two while it results in longer time-span and average data delay in either CDS, MST or SPT, which is because we use more strict interference definition in our schedule by cluster method.

## C. Impact of Heterogeneous Nodes

In this simulation, we use the ratio of the maximal interference radius to the minimal interference radius in the network to study how the difference of this ratio can affect the time-span, energy consumption and data delay when the ratio varies from 1 to 4 in the sensor network with 300 nodes and 5 packets per slot.

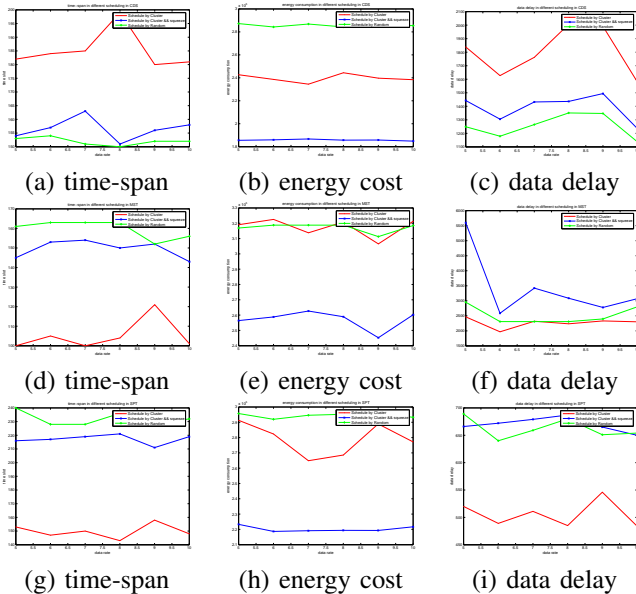


Fig. 6. Impact on the time-span, energy cost and data delay with various data rates in homogeneous networks with the data aggregation by CDS tree (1st row), MST tree (2nd row) and SPT tree (3rd row).

Figure 9 and Figure 10 show that the time-span, energy consumption and average data delay increase when the ratio of  $\frac{R_{max}}{R_{min}}$  increases. It is reasonable 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 distributed into different buckets so that they should be scheduled sequentially. While the energy consumption does not vary significantly when the ratio of  $\frac{R_{max}}{R_{min}}$  changes, it more depends on the tree structure.

## VII. RELATED WORK

A number of MAC protocols have been proposed for WSNs. Polastre *et al.* proposed a MAC protocol, called B-MAC [26], which is used as the default MAC for Mica2. Buettner *et al.* proposed a new approach to low power listening called X-MAC [7], which employs a short preamble to further reduce energy consumption and to reduce latency. In [30], 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* [1] 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 [5], [6], [17], [27], [28]. To save energy consumption, the wake-up scheduling has been widely used. Keshavarzian *et al.* [13] 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 [8], [31]. TDMA-based scheduling algorithms that minimize

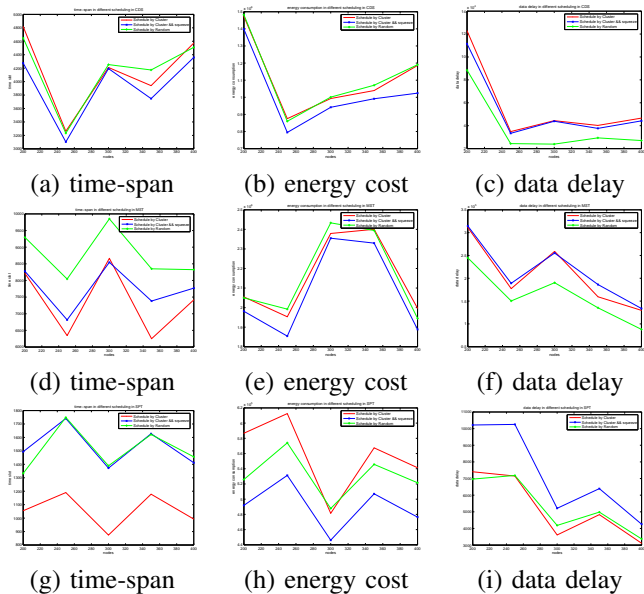


Fig. 7. Impact on the time-span, energy cost and data delay with various nodes in homogeneous networks by CDS tree (1st row), MST tree (2nd row) and SPT tree (3rd row).

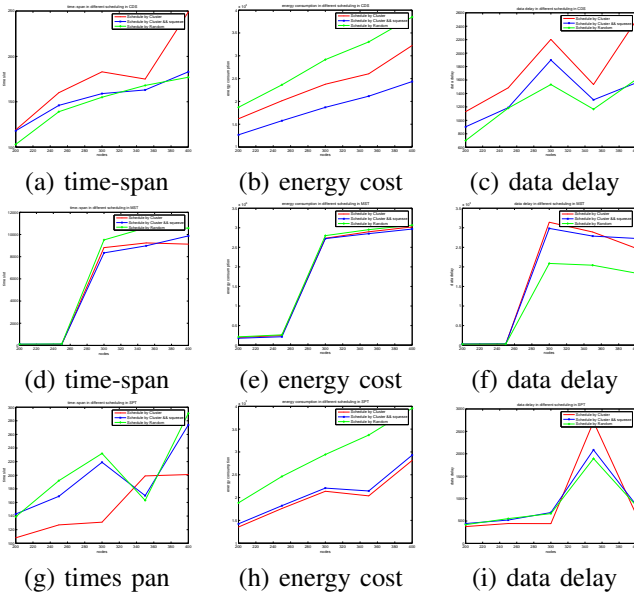


Fig. 8. Impact on the time-span, energy cost and data delay with various nodes in homogeneous networks with the data aggregation by CDS tree (1st row), MST tree (2nd row) and SPT tree (3rd row).

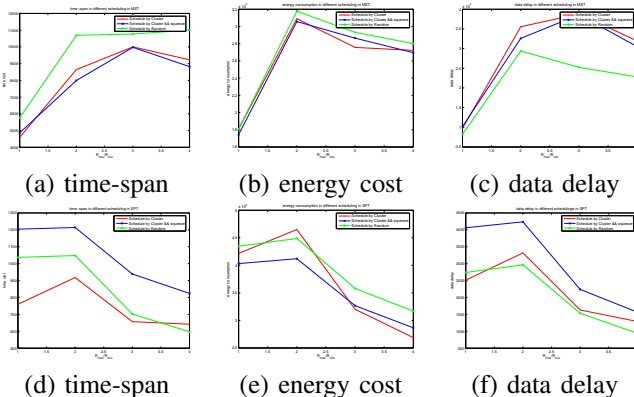


Fig. 9. Impact on the time-span, energy cost and data delay with various  $\frac{R_{max}}{R_{min}}$  ratios in heterogeneous networks by MST tree (1st row) and SPT tree (2nd row).

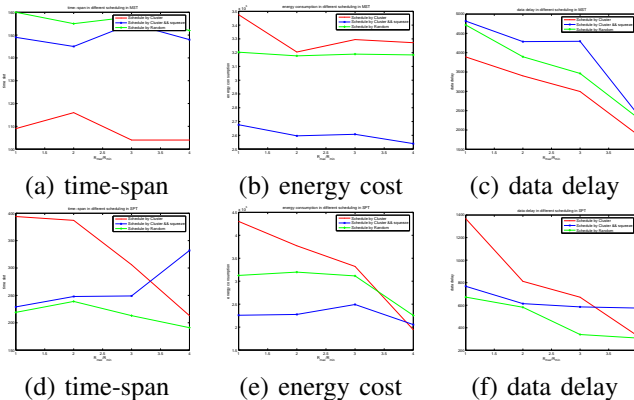


Fig. 10. Impact on the time-span, energy cost and data delay with various  $\frac{R_{max}}{R_{min}}$  ratios in heterogeneous networks with data aggregation by MST tree (1st row) and SPT tree (2nd row).

the number of time slots or the message delay are proved NP-complete [4], [9], [10]. Approximate algorithms have been therefore proposed, including both link scheduling [8], [10], [17], [31] and broadcast scheduling [14], [21], [29]. 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 [24] proposed a randomized distributed edge coloring method that uses at most  $2\Delta + 1$  colors. Ramanathan [27] proposed a unified framework for TDMA, FDMA and CDMA based multi-hop wireless networks. They also proposed a time-slot assignment to edges; the number of time-slots required is at most  $O(\theta)$  times the optimum, where  $\theta$  is the thickness of a graph. Gandham *et al.* [10] 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 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. Krumke *et al.* [14] proposed efficient approximation algorithms for the distance-2 vertex coloring problem for various geometric graphs including  $(r, s)$ -civilized graphs, planar graphs, graphs with bounded genus, etc. In [15], 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. Several distributed algorithms that use  $O(\Delta)$  colors have been proposed in literatures. A  $(\Delta + 1)$ -coloring can be computed in time  $O(\log n + \Delta)$  [23] or  $O(\Delta \log n)$  [12]. In [19], Maraco *et al.* proposed a distributed algorithm that computed an  $O(\Delta)$ -coloring in time  $O(\log n)$ . All of the above distributed algorithms do not take the interference into account and is based on the message passing model [25], which implies that the actual time used in a wireless environment could be much larger [22]. Recently, Moscibroda *et al.* [22] proposed an  $O(\Delta)$  distributed coloring method with time-complexity  $O(\Delta \log n)$ . Wang *et al.* [31] also proposed both centralized and distributed interference-aware link scheduling algorithms with performance guarantee to maximize the throughput of the network.

## VIII. 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 only needs to wake up 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 time-span 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 tradeoffs among these potentially conflicting objectives?

#### REFERENCES

- [1] AHN, G.-S., MILUZZO, E., CAMPBELL, A. T., HONG, S. G., AND CUOMO, F. Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks. In *In ACM SenSys* (2006).
- [2] ALICHERY, M., BHATIA, R., AND LI, L. E. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *MobiCom* (2005), pp. 58–72.
- [3] ALZOUBI, K., LI, X.-Y., WANG, Y., WAN, P.-J., AND FRIEDER, O. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Processing* (2002).
- [4] ARIKAN, E. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory* 1984, 30(4), 681C685.
- [5] BAO, L., AND GARCIA-LUNA-ACEVES, J. Transmission scheduling in ad hoc networks with directional antennas. In *MobiCom '02* (2002), pp. 48–58.
- [6] BAO, L., AND GARCIA-LUNA-ACEVES, J. J. Channel access scheduling in ad hoc networks with unidirectional links. In *DIALM '01*, (2001), pp. 9–18.
- [7] BUETTNER, M., YEE, G., ANDERSON, E., AND HAN, R. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. In *ACM SenSys* (2006).
- [8] DJUKIC, P., AND VALAEE, S. Link scheduling for minimum delay in spatial re-use tdma. In *IEEE Infocom* (2007).
- [9] EPHREMEDIS, A., AND TRUONG, T. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications* 1990, 38(4), 456C460.
- [10] GANDHAM, S., DAWANDE, M., AND PRAKASH, R. Link scheduling in sensor networks: Distributed edge coloring revisited. In *IEEE Infocom* (2005).
- [11] CĂLINESCU, G. Computing 2-hop neighborhoods in ad hoc wireless networks. In *AdHoc-Now 03* (2003).
- [12] GOLDBERG, A., PLOTKIN, S., AND SHANNON, G. Parallel symmetry-breaking in sparse graphs. In *ACM STOC '87*, pp. 315–324.
- [13] KESHAVARZIAN, A., LEE, H., AND VENKATRAMAN, L. Wakeup scheduling in wireless sensor networks. In *ACM MobiHoc* (2006).
- [14] KRUMKE, S., MARATHE, M., AND RAVI, S. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks* 7 (2001), 567–574.
- [15] KUMAR, V. S. A., MARATHE, M. V., PARTHASARATHY, S., AND SRINIVASAN, A. End-to-end packet-scheduling in wireless ad-hoc networks. In *ACM SODA '04*, pp. 1021–1030.
- [16] KUMAR, V. S. A., MARATHE, M. V., PARTHASARATHY, S., AND SRINIVASAN, A. Algorithmic aspects of capacity in wireless networks. *SIGMETRICS Perform. Eval. Rev.* 33, 1 (2005), 133–144.
- [17] LIU, R., AND LLOYD, E. L. A distributed protocol for adaptive link scheduling in ad-hoc networks. In *IASTED WOC* (2001).
- [18] MAINWARING, A., CULLER, D., POLASTRE, J., SZEWCZYK, R., AND ANDERSON, J. Wireless sensor networks for habitat monitoring. In *ACM WSNA* (2002), pp. 88–97.
- [19] MARCO, G. D., AND PELC, A. Fast distributed graph coloring with  $(\Delta+1)$  colors. In *ACM SODA* (2001), pp. 630–635.
- [20] MILLER, M. J., AND VAIDYA, N. Efficient bounds for the stable set, vertex cover, and set packing problem. In *IEEE WCNC* (2004), vol. 4, pp. 2335–2340.
- [21] NGO, C. Y., AND LI, V. O. K. Centralized broadcast scheduling in packet radio networks via genetic-fix algorithms. *IEEE Transactions on Communications* 2003, 51(9), 1439C1441.
- [22] MOSCIBRODA, T., AND WATTENHOFER, R. Coloring unstructured radio networks. In *ACM SPAA* (2005), pp. 39–48.
- [23] PANCONESI, A., AND RIZZI, R. Some simple distributed algorithms for sparse networks. *Distributed Computing* 14, 2 (2001), 97–100.
- [24] PANCONESI, A., AND SRINIVASAN, A. Improved distributed algorithms for coloring and network decomposition problems. In *ACM STOC* (1992), pp. 581–592.
- [25] PELEG, D. *Distributed computing: a locality-sensitive approach*. ACM STOC, 2000.
- [26] POLASTRE, J., HILL, J., AND CULLER, D. Versatile low power media access for wireless sensor networks. In *ACM SenSys* (2004).
- [27] RAMANATHAN, S. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Network* 5, 2 (1999), 81–94.
- [28] RAMANATHAN, S., AND LLOYD, E. L. Scheduling algorithms for multi-hop radio networks. In *ACM SIGCOMM* (1992), pp. 211–222.
- [29] RAMASWAMI, E., AND PARHI, K. Distributed scheduling of broadcasts in a radio network. In *IEEE Infocom* (1989).
- [30] RHEE, I., WARRIER, A., AIA, M., AND MIN, J. Zmac: a hybrid mac for wireless sensor networks. In *ACM SenSys* (2005).
- [31] WANG, W., WANG, Y., LI, X.-Y., SONG, W.-Z., AND FRIEDER, O. Efficient interference aware tdma link scheduling for static wireless mesh networks. In *ACM MobiCom* (2006).