This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Contiguous Link Scheduling for Data Aggregation in Wireless Sensor Networks

Junchao Ma, *Student Member, IEEE,* Wei Lou, *Member, IEEE,* and Xiang-Yang Li, *Senior Member, IEEE*

Abstract—Wireless sensor networks (WSNs) consist of a large number of battery-powered wireless sensor nodes, and one key issue in WSNs is to reduce the energy consumption while maintaining the normal functions of WSNs. Data aggregation, as a typical operation in data gathering applications, can cause a lot of energy wastage since sensor nodes, when not receiving data, may keep in the listen state during the data collection process. To save this energy wastage, sleep scheduling algorithms can be used to turn the nodes to the sleep state when their radios are not in use and wake them up when necessary. In this paper, we identify the contiguous link scheduling problem in WSNs, in which each node is assigned consecutive time slots so that the node can wake up only once in a scheduling period to fulfil its data collection task. The objective of the problem is to find an interference-free link scheduling with the minimum number of time slots used. In virtue of the contiguous link scheduling, the energy consumption caused by nodes' state transitions can be reduced. We prove the contiguous link scheduling problem in WSNs to be NP-complete, and then present efficient centralized and distributed algorithms with theoretical performance bounds in both homogeneous and heterogeneous networks. We also conduct simulation experiments that corroborate the theoretical results and demonstrate the efficiency of our proposed algorithms.

Index Terms—Energy efficient algorithms, sleep scheduling, contiguous link scheduling, data aggregation, wireless sensor networks.

1 Introduction

Wireless sensor networks (WSNs) consist of thousands of tiny, inexpensive and battery-powered wireless sensor devices that organize themselves into multihop radio networks. Data aggregation [1] is a typical task in WSN-based data gathering applications, in which an intermediate node could first collect data from its children nodes, process the received data to an aggregated value (e.g., the highest temperature), and then forward the aggregated data to its parents nodes.

As the batteries of most sensor nodes are non-rechargeable, one key issue is to schedule the activities of nodes to reduce the energy consumption. A major source of energy wastage in WSNs is the idle listening state in the radio modules, which in fact consumes almost as much energy as receiving [2]. Therefore, nodes are generally scheduled to sleep when the radio is not in use, and wake up when necessary [3]. By using such a sleep scheduling, nodes could operate in a low-duty-cycle mode, and periodically start up to check the channel for activity.

Effective sleep scheduling methods should allow every node to start up and transmit/receive its messages without interferences. One popular way to achieve this is to adopt the time division multiple access (TDMA) MAC protocols, which can directly support the low-duty-cycle operations and have the natural advantages of having no contention-introduced overhead or collisions [2]. Moreover, the TDMA MAC protocols can guarantee a deterministic delay bound. Thus, we are interested in designing an efficient TDMA sleep scheduling for WSNs.

In the TDMA MAC protocols, time is divided into equal intervals referred to as *time slots*, and a time slot is long enough to transmit one data packet. In order to be interference-free, a simple approach is to assign each communication link a time slot, and then, the number of time slots assigned is equal to the number of communication links of the network. This link scheduling scheme results in many more time slots than necessary, considering that multi-hop networks are able to make space reuse in the shared channel and multiple transmissions can be scheduled in one time slot without any interference. By reducing the time slots assigned, the channel utilization and network throughput can be improved. The TDMA link scheduling aims to minimize the number of time slots assigned while producing an interference-free link scheduling, and it has been proven to be NP-complete [4]. Several approximation algorithms have been proposed for the link scheduling problem [5], [6], [7], [8], [9], [10], [11], [12].

After a TDMA link scheduling, each communication link is assigned a time slot to transmit. To comply with the sleep scheduling, a node needs to start up to transmit or receive in the assigned time slot and turn to sleep after the time slot. This TDMA link scheduling only considers the different energy consumptions of a node in different states (transmit, receive, listen and sleep), but does not take into account the energy consumed by the node's *state transitions*, for example, from the sleep state to the transmit/receive/listen state or vice versa. If a node has multiple neighbors to communicate with, it may start up numerous times in a scheduling period T. If a node starts up frequently, it not only needs extra startup time, but also consumes extra energy for state transitions. Notice that the typical startup time

- *Junchao Ma and Wei Lou are in the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. Email:{csjma, csweilou}@comp.polyu.edu.hk.*
- *Xiang-Yang Li is in the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA. Email: xli@cs.iit.edu.*

Digital Object Indentifier 10.1109/TPDS.2013.296 1045-9219/13/$31.00 © 2013 IEEE

is on the order of milliseconds, while the transmission time may be less than the startup time if the packets are small. Consequently, the transient energy consumption during the startup process can be higher than the energy consumption during the actual transmission. An interesting problem is that whether each node can be scheduled consecutive time slots so that it only needs to start up once to fulfil all its tasks. In virtue of this, each intermediate node can easily collect the data from all its children nodes and then immediately compute the aggregated value in the data aggregation, rather than waiting for a long delay until all the data can be received and computed. Moreover, the energy consumed in the state transitions can be saved.

In this paper, we use a new energy model, where the energy consumption caused by the nodes' state transitions is considered. We identify the contiguous link scheduling problem in WSNs, in which links incident to one node are scheduled together to obtain consecutive time slots so that the node can start up only once to monitor the channel in a scheduling period T. Especially, for the data aggregation, if the network topology is a tree, each intermediate node needs to start up only twice in a period, once for receiving data from its children nodes and once for transmitting its data to its parent node. The objective of the contiguous link scheduling problem in WSNs is to find a link scheduling with the minimum period, i.e. the number of time slots used in a period, by maximizing the spatial reuse of concurrent transmissions without interferences. This can increase the network throughput and reduce the latency of the data aggregation.

The main contributions of this paper are summarized as follows: (1) We address the scheduling problem using a new energy model, which is closer to realistic sensor nodes. (2) We identify the contiguous link scheduling problem in WSNs and prove that the problem is NP-complete. (3) We present centralized and distributed algorithms that have theoretical performance bounds to the optimum in homogeneous and heterogeneous networks. (4) We conduct simulations to show the efficiency of the proposed algorithms in terms of total energy consumption, throughput, and time delay.

This paper has several additional advances compared to the preliminary work [13]: First, the contiguous link scheduling problem in WSNs is extended from homogeneous networks to heterogeneous networks. Second, the complete proof of the NP-completeness of the contiguous link scheduling problem in WSNs is provided. Third, the theoretical bounds of the proposed algorithms are derived along with their complexity analysis. Finally, the performance of the proposed algorithms for heterogeneous networks are evaluated and analyzed.

The remainder of this paper is organized as follows. Section 2 describes the system model and formulates the contiguous link scheduling problem in WSNs. Section 3 presents the centralized algorithms and Section 4 presents the distributed algorithms for the problem. Section 5 gives the performance analysis of our algorithms. Section 6 describes and analyzes the simulation results of the proposed algorithms. Section 7 concludes the paper. Additional sections have been added in the supplementary file, including related work, NP-completeness proof, performance analysis, and more simulation results.

2 System Model and Problem Formulation

In this section, we first present the system model consisting of a network model, an interference model and an energy model, then we formulate the contiguous link scheduling problem in WSNs.

2.1 System Model

Network Model. We assume that a WSN has n static sensor nodes, which are all equipped with single omni-directional antennas, and there exists a sink node to collect the data from other sensor nodes. The network is represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and E denotes the set of edges referred to the communication links. In the network, each node v_i has a transmission range r_i. If there is a link $l_{i,j}$ (i.e. the link from v_i to v_j) in E, node v_j is located within the transmission range of node v_i. Similar to most TDMA protocols, we assume that the time is synchronized in the network and the clock drift of a node can be overlooked.

We consider two types of network topologies in this paper, data gathering tree and directed acyclic graph (DAG), which are commonly used for data aggregation in WSNs. A data gathering tree is a tree rooted at a sink node, where each intermediate node collects the data from its children nodes and then forwards the data to its parent node [14]. A DAG is a graph with no directed cycles, that is, there is no path that starts and ends at the same node. The depth of a node in a DAG is the length of the longest path from that node to the sink [15].

Interference Model. In wireless networks, the packets transmitted by a node may be received by all the nodes within its transmission range due to the broadcast nature of the wireless medium. Therefore, the transmission of one link may interfere with the reception of another link. We consider both primary interference and secondary interference [5] in this paper. The primary interference occurs when a node has more than one communication task in a single time slot. The secondary interference occurs when a node tuned to a particular transmitter is also within the transmission range of another transmission intended for other nodes.

We suppose the interference range of a node v_i is R_i, and the ratio of the interference range to the transmission range is denoted as $\gamma_i = \frac{R_i}{r_i}$. In practice, $2 \leq \gamma_i \leq 4$. We define $\gamma_{max} = \max_{1 \leq i \leq n} \gamma_i$ and $\gamma_{min} = \min_{1 \leq i \leq n} \gamma_i$. The maximum transmission range, the minimum transmission range, the maximum interference range and the minimum interference range in a network are denoted as $r_{max} = \max_{1 \leq i \leq n} r_i$, $r_{min} = \min_{1 \leq i \leq n} r_i$, $R_{max} = \max_{1 \leq i \leq n} R_i$ and $R_{min} = \min_{1 \leq i \leq n} R_i$ respectively. The ratio of the maximum transmission range to the minimum transmission range is denoted as $\sigma = \frac{r_{max}}{r_{min}}$.

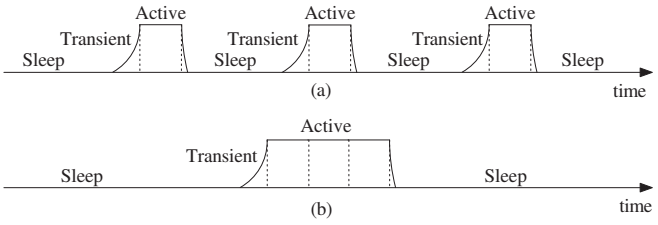


Fig. 1: The energy model: (a) Before active time slots merged, (b) After active time slots merged.

The interference between two links in the network depends on the interference model, and in this paper we use the protocol model [16], [17], in which a transmission from v_i to v_j is considered successful if any node v_k located within a distance R_k from v_j is not transmitting during the same time interval.

Energy Model. In our energy model, we assume that each node operates in three states: active state (transmit, receive and listen), sleep state, and transient state [18], [19]. A node in any active state consumes the power at the same level, compared to the extreme low power consumption in the sleep state.

The transient state of a sensor node comprises two processes: startup (from the sleep state to the active state), and turndown (from the active state to the sleep state). The startup process includes radio initialization, radio and its oscillator startup, and radio switching to receive/transmit state [20], and it is slow due to the feedback loop in the phase-locked loop (PLL), and a typical setting time of the PLL-based frequency synthesizer is on the order of milliseconds. Compared to the startup, the turndown process is rapid enough to be negligible. Therefore, we only consider the startup process in the transient state in this paper.

Our energy model is illustrated in Fig. 1(a), and there is a significant energy consumption and time overhead when the sensor's radio powers on. Fig. 1(b) shows that merging the sensor's active time slots together can reduce the startup frequency so as to save both energy and time.

TABLE 1: Time and power consumption in the startup process for a Tmote sky sensor [21].

Operation process	Time		Power consumption	
Sleep	—	—	P_{sleep}	$0.063mW$
Radio initialization	t_{init}	$0.47ms$	P_{init}	$42mW$
Turn on radio	t_{on}	$1.42ms$	P_{on}	$3mW$
Switch to RX/TX state	t_{sw}	$0.212ms$	P_{sw}	$42mW$
Listen	—	—	P_{ls}	$59.1mW$
Receive 1 byte	t_{rx}	$0.032ms$	P_{rx}	$59.1mW$
Transmit 1 byte	t_{tx}	$0.032ms$	P_{tx}	$52.2mW$

Table 1 lists the typical values of time and power consumption for a Tmote Sky sensor in the startup process. The time to activate a sensor is $t_{init}+t_{on}+t_{sw} \approx 2.1ms$, the energy consumption to activate a sensor is $P_{init}t_{init}+P_{on}t_{on}+P_{sw}t_{sw} \approx 32.9\mu J$, and the energy consumption to transmit a packet (e.g. 36 bytes) is $P_{tx}t_{tx}L_{packet} \approx 60.1\mu J$. We can see that the transient energy consumption during the startup process is over 50% compared to that of transmitting a packet.

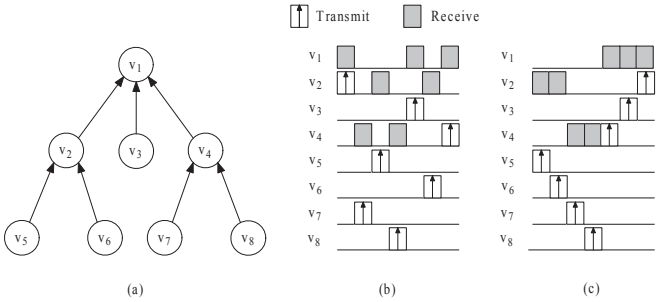


Fig. 2: Link scheduling and contiguous link scheduling: (a) Communication graph, (b) Link scheduling, (c) Contiguous link scheduling.

2.2 Problem Formulation

In a TDMA sleep scheduling, each link $l_{i,j}$ is assigned a time slot, in which both sender node v_i and receiver node v_j should start up to communicate. After the allocated time slot, nodes v_i and v_j change to the sleep state. When using traditional link scheduling algorithms (e.g. [7], called *degree-based heuristic* in this paper) which schedule the communication links one by one, node v_j may start up w_j times to monitor the channel in a period T, where w_j is the number of directed links incident to node v_j. The frequent startup would consume a large amount of extra energy and time. Therefore, we assign consecutive time slots to all the directed links incident to the same node so that the node can start up only once to receive all the packets from its neighbors.

Definition 1. The *contiguous link scheduling problem in WSNs* is to find an interference-free link scheduling with the minimum period, in which each link is assigned a time slot to transmit and all the links incident to one node are assigned consecutive time slots. A contiguous link scheduling is said to be *valid* if all the links incident to one node are assigned consecutive time slots.

Fig. 2 illustrates the link scheduling and contiguous link scheduling. In Fig. 2(a), the given network is a data gathering tree rooted at node v_1, in which any two links interfere with each other. Fig. 2(b) shows an interference-free link scheduling, where a node starts up numerous times in a period. Fig. 2(c) shows a contiguous link scheduling that a node can start up only once for receiving data from its neighbors. Note that the contiguous link scheduling can be applied not only to trees, but also to other topologies, such as DAGs. In particular, if the network is a data gathering tree where each node has only one parent, a node just needs to start up at most twice in a period, once for receiving data from its children nodes and once for transmitting its data to its parent node.

Given the protocol interference model, the interference of the links in the communication graph $G = (V, E)$ can be represented as a conflict graph G_c [17]. Corresponding to each directed link from node v_i to node v_j in G, the conflict graph G_c contains a vertex $l_{i,j}$. There is an edge between the two vertices in G_c if the corresponding links interfere with each other in G. In [7], the link scheduling problem is modeled as a vertex coloring of the conflict graph.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

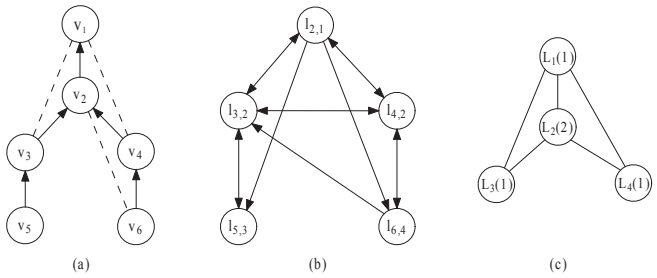


Fig. 3: Conflict graph and merged conflict graph: (a) Communication graph, (b) Conflict graph, (c) Merged conflict graph. Dashed lines represent the interference relationship.

For the sample communication graph shown in Fig. 3(a), where dashed lines represent the interference relationship, the corresponding conflict graph is shown in Fig. 3(b).

To solve the contiguous link scheduling problem in WSNs, we propose a *merged conflict graph* G_{mc} corresponding to a graph G. We formulate the problem as an interval vertex-coloring of the graph G_{mc}, where an *interval vertex-coloring* is an assignment of w_i consecutive colors to w_i nodes satisfying the constraint that no two adjacent nodes share the same color. In G_{mc}, the w_i directed links incident to node v_i in G correspond to a vertex $L_i(w_i)$, and w_i is the weight of vertex $L_i(w_i)$. There is an edge between the two vertices $L_i(w_i)$ and $L_j(w_j)$ in G_{mc} if and only if at least one link incident to node v_i interferes with a link incident to node v_j in G. We can see that G_{mc} is a vertex-weighted graph. For the communication graph in Fig. 3(a), the corresponding merged conflict graph is shown in Fig. 3(c). For example, the two links $l_{3,2}$ and $l_{4,2}$ incident to node v_2 shown in Fig. 3(a) correspond to vertex $L_2(2)$ in Fig. 3(c). Similarly, link $l_{6,4}$ corresponds to vertex $L_4(1)$. There is one edge between $L_2(2)$ and $L_4(1)$ in Fig. 3(c) since $l_{6,4}$ interferes with $l_{3,2}$ and $l_{4,2}$ in Fig. 3(b). From the communication graph in Fig. 3(a) and the merged conflict graph in Fig. 3(c), we can see that the number of vertices in G_{mc} is equal to the number of receiving nodes in G.

The following theorem states the NP-completeness of the contiguous link scheduling problem in WSNs.

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

The complete proof of Theorem 1 is given in the supplementary file. As the contiguous link scheduling problem in WSNs is NP-complete, it is impossible to find a polynomial time algorithm with the optimal solution if $P \neq NP$. In the following two sections, we propose both centralized and distributed approximation algorithms with performance bounds.

3 Centralized Algorithms

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

3.1 Centralized Scheduling

We first propose a *centralized scheduling* algorithm for the contiguous link scheduling problem in WSNs. Instead of scheduling a time slot individually for each communication link, the links incident to a node are scheduled consecutive time slots, and then each node can start up only once to receive all the data from its neighbors. The centralized scheduling algorithm is described in Algorithm 1: We first construct a merged conflict graph G_{mc} based on the communication graph G, and each vertex $L_i(w_i)$ in G_{mc} has a weight of w_i. The scheduling is proceeded in the decreasing order of the weights, i.e., the node which has more incident links is scheduled earlier. In the assignment, each node v_i is assigned the smallest w_i consecutive time slots using the first-fit heuristic, and these time slots are not yet assigned to any node v_j if $L_j(w_j)$ is adjacent to $L_i(w_i)$ in G_{mc}. After that, the w_i time slots are assigned sequentially to the w_i links incident to v_i.

Algorithm 1 Centralized scheduling (Centralized)

Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.
1: Construct the merged conflict graph G_{mc}, and initialize an empty stack S.
2: Push the vertices in G_{mc} in the non-decreasing order of weights to the stack S.
3: **while** S is not empty **do**
4: Pop a vertex $L_i(w_i)$ from S. Assign the smallest w_i consecutive time slots to node v_i, which are not yet assigned to any node v_j if $L_j(w_j)$ is adjacent to $L_i(w_i)$ in G_{mc}.
5: Schedule the w_i time slots sequentially to the w_i links that are incident to node v_i in G.

Example 1. The sample network, as shown in Fig. 4(a), is a directed acyclic graph (DAG) where dashed lines represent the interference relationship. The corresponding merged conflict graph of the network is shown in Fig. 4(b). When using the centralized scheduling algorithm (Algorithm 1), all the nodes that have incident links (i.e., nodes v_1, v_2, v_5, v_7 and v_9) will be scheduled according to their weights. Since the weights of nodes v_1, v_2, v_5, v_7 and v_9 are 4, 1, 1, 4 and 5 respectively, the scheduling order is v_9, v_1, v_7, v_2 and v_5. As $L_1(4)$, $L_7(4)$ and $L_9(5)$ are pair-wise adjacent in the merged conflict graph G_{mc} as shown in Fig. 4(b), v_9 is assigned consecutive time slots from t_1 to t_5, v_1 is assigned consecutive time slots from t_6 to t_9, and v_7 is assigned consecutive time slots from t_{10} to t_{13}. For v_2 and v_5, they are assigned time slots t_1 and t_{10}, respectively. After the scheduling, the total number of time slots assigned is 13. The time slot assigned to each link is indicated in Fig. 4(a).

3.2 Centralized Scheduling with Spatial Reuse

The contiguous link scheduling allows the links incident to v_i and v_j to have some time slots overlapped when $L_i(w_i)$ and $L_j(w_j)$ are adjacent in G_{mc}, as long as the same time slots assigned to different links do not cause interferences.

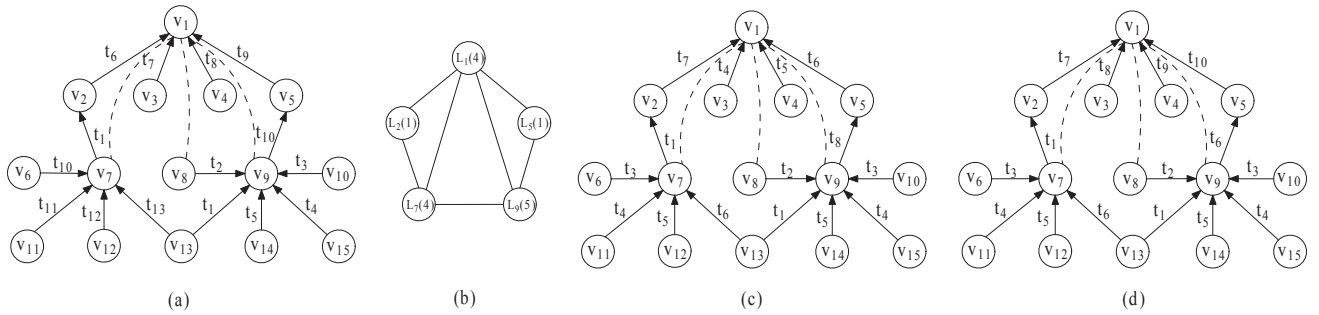


Fig. 4: Contiguous link scheduling: (a) Centralized scheduling, (b) Merged conflict graph, (c) Centralized scheduling with spatial reuse, (d) Distributed scheduling with efficient delay. t_1, t_2, $\cdots$, t_{13} are assigned time slots, and dashed lines represent the interference relationship.

A sample is illustrated in Fig. 4(c). In G_{mc}, $L_1(4)$ is adjacent to $L_7(4)$ as shown in Fig. 4(b) because $l_{2,1}$ interferes with $l_{6,7}$. But $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ do not interfere with $l_{6,7}$, $l_{11,7}$, $l_{12,7}$ and $l_{13,7}$, so v_1 can still be assigned time slots from t_4 to t_6 when v_7 are assigned time slots from t_3 to t_6. By this way, we could get another valid contiguous link scheduling using 8 time slots (as shown in Fig. 4(c)), which is fewer than that in the centralized scheduling. Based on this observation, we propose a *centralized scheduling with spatial reuse* to enhance the centralized scheduling algorithm.

Algorithm 2 describes the centralized scheduling with spatial reuse. It calls either the *recursive backtracking* algorithm (Algorithm 3) or *minimum conflict heuristic* algorithm (Algorithm 4) in its process. The minimum conflict heuristic algorithm is designed to improve the time complexity of the recursive backtracking. The centralized scheduling with spatial reuse works as follows: It first constructs a merged conflict graph G_{mc} based on the communication graph G, and construct a so-called interference matrix (see in Definition 2) for each node v_i. It then assigns time slots to nodes in the decreasing order of their weights. It assigns the smallest w_i available consecutive time slots to node v_i using either the recursive backtracking or the minimum conflict heuristic. After the links incident to node v_i are assigned time slots, v_i will broadcast the information to all the nodes whose incident links interfere with the links incident to v_i. Then each node in the network would know the time slots that its incident links could not use.

We introduce the *interference matrix* to indicate whether a link can use a time slot or not, which is defined as follows:

Definition 2. An *interference matrix* of node v_i is a $t \times w_i$ matrix $M = (m_{j,k})_{t \times w_i}$ $(1 \le j \le t, 1 \le k \le w_i)$ that indicates whether a time slot could be assigned to a link l_k incident to v_i without interferences, where

$$m_{j,k} = \begin{cases} - & : \textit{link } l_k \textit{ cannot use time slot } t_j \textit{ (interference)}; \\ 0 & : \textit{link } l_k \textit{ can use time slot } t_j \textit{ (interference-free)}; \\ 1 & : \textit{link } l_k \textit{ selects time slot } t_j \textit{ (selection)}. \end{cases}$$

Here, "–" denotes that assigning the link to this time slot will interfere with the already-scheduled links, "0" denotes that assigning the link to this time slot will be interference-free, and "1" denotes that the link has been assigned this time slot. Note that in the interference matrix, the number of columns w_i is equal to the number of links incident to node v_i, and the number of rows t is the number of time slots already assigned in the scheduling.

Algorithm 2 Centralized scheduling with spatial reuse

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

1: Construct the merged conflict graph G_{mc}, construct an interference matrix for each node v_i, and initialize an empty stack S.
2: Push the vertices in G_{mc} in the non-decreasing order of weights to the stack S.
3: **while** S is not empty **do**
4: Pop a vertex $L_i(w_i)$ from S. Assign the smallest w_i consecutive time slots to node v_i, using recursive backtracking (Algorithm 3) or minimum conflict heuristic (Algorithm 4).
5: Schedule the w_i time slots sequentially to the w_i links that are incident to node v_i in G.
6: v_i broadcasts the time slot assignment to the nodes whose incident links interfere with the links incident to v_i, then the informed nodes update their interference matrices.

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

Example 2. In Fig. 4(c), nodes v_7 and v_9 have already been scheduled time slots $\{t_3, t_4, t_5, t_6\}$ and $\{t_1, t_2, t_3, t_4, t_5\}$ respectively. Node v_1 that has 4 incident links $\{l_{2,1}, l_{3,1}, l_{4,1}, l_{5,1}\}$ is to be scheduled. The time slots used by the links that interfere with links $l_{2,1}$, $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ are $\{t_2, t_3, t_4, t_5, t_6\}$, $\{t_2\}$, $\{t_2\}$ and $\{t_1, t_2, t_3, t_4, t_5\}$, respectively. The interference matrix of node v_1 is shown as M_0 and one interference submatrix of M_0 is shown as M'_0.

$$M_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \begin{array}{c} \begin{array}{cccc} l_{2,1} & l_{3,1} & l_{4,1} & l_{5,1} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & - \\ - & - & - & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix} \end{array}, \quad M'_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} l_{2,1} & l_{3,1} & l_{4,1} & l_{5,1} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & - \\ - & - & - & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \end{pmatrix} \end{array}.$$

An assignment in the interference matrix $M = (m_{j,k})_{t \times w_i}$ of node v_i is said to be *valid* if (1) there is one and only

one "1" in each row and each column, and (2) there are w_i consecutive rows that have "1" in each row in the matrix. The links incident to v_i could be scheduled w_i consecutive time slots by obtaining a valid assignment in the interference matrix.

In order to reduce the time slots used in the scheduling, we propose the recursive backtracking algorithm, as shown in Algorithm 3. The algorithm first finds the smallest w_i consecutive rows that have at least one "0" in each row in the interference matrix M, and constructs an interference submatrix M' which consists of the w_i rows. Then it starts the first selection in the first row, and the second selection in the second row without interferences to the selection in the first row. The algorithm continues the selection in the next row until a valid assignment is found. During the process of each selection in a row, there may be several candidates "0". In these candidates, the selected one is referred to as *predecessor*, and the candidates "0" in the next row are referred to as *successors* of the predecessor. If one successor fails in the selection, it then executes the *backtracking procedure*: the algorithm checks whether the next successor of the predecessor satisfies the condition that there is only one "1" in each column. If the successors are exhausted, the algorithm backtracks to the previous predecessor and tries the next successor of the previous predecessor. If there are no more predecessors, the algorithm adds a new time slot interference-free to all the links incident to v_i, and the corresponding interference matrix adds a zero row vector $(0)_{1\times w_i}$ in the last row.

Algorithm 3 Recursive backtracking

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

1: Construct an interference submatrix M' consisting of the smallest w_i consecutive rows that have at least one "0" in each row.
2: **while** a valid assignment is not obtained **do**
3: Start the first selection in the first row.
4: Continue the selection in the next row satisfying the condition that there is only one "1" in each column, until a valid assignment is obtained.
5: **if** a selection fails to obtain a valid assignment **then**
6: Execute the backtracking procedure.
7: **if** the recursive backtracking fails **then**
8: Update M' by deleting the first row of M' and adding a zero row vector $(0)_{1\times w_i}$ in the last row of M'.

Example 3. We first construct an interference submatrix M'_1 of interference matrix M_0. We then select $m_{1,2}$ and $m_{2,3}$ in the first two rows in M'_1 (marked as "①" in the matrix), but it fails in the third row as shown in M'_2. Then we use the backtracking procedure, and select $m_{1,3}$ and $m_{2,2}$ in the first two rows in M'_1, but it fails again as shown in M'_3. As all the predecessors and successors are exhausted, we delete the first row and add a zero vector $(0)_{1\times w_i}$ as the last row to construct a new interference submatrix M'_4. Finally, we can get a valid assignment shown in M'_5, i.e., links $l_{2,1}$, $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ are assigned time slots t_7, t_4, t_5 and t_6, respectively, as shown in Fig. 4(c).

$$M'_1 = \begin{pmatrix} - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix}, \quad M'_2 = \begin{pmatrix} - & ① & 0 & - \\ - & 0 & ① & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix},$$

$$M'_3 = \begin{pmatrix} - & 0 & ① & - \\ - & ① & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix}, \quad M'_4 = \begin{pmatrix} - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$M'_5 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & 1 & - \\ - & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

The recursive backtracking algorithm is a brute-force search algorithm, and the time complexity can be $O(w_i!)$ in the worst case. To reduce the complexity, we present a fast algorithm called minimum conflicts heuristic that tries to minimize the number of conflicts in a so-called *conflict matrix*, which is defined as follows:

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

The minimum conflicts heuristic is a local search algorithm in the constraint satisfaction problem [22], whose main idea is to assign each variable a value in the initial state and then change one variable at a time by selecting the value that results in the minimum number of conflicts with other variables.

Algorithm 4 describes the minimum conflicts heuristic algorithm. It first constructs an interference submatrix M' which consists of w_i consecutive rows, and then starts with a random initial configuration in M', e.g., with one selection per column. The algorithm then uses a heuristic to determine how to reduce the conflicts by moving the selection "1" with the largest number of conflicts to the position in the same column where the number of conflicts is minimum in the conflict matrix. It continues to reduce the conflicts until there is no conflict or the initial configuration fails. If the initial configuration fails, it will add a new time slot and continue.

Example 4. We first construct the interference submatrix M'_1 of interference matrix M_0, and one initial configuration of matrix M'_1 is shown as the matrix M'_6, and the corresponding conflict matrix of matrix M'_6 is shown as the matrix M_{C_1}. For example, $c_{1,3} = m_{1,2} + m_{2,3} + m_{3,3} = 3$, $c_{2,3} = m_{2,3} + m_{3,3} = 2$. As the largest number of conflicts among the selections in M_{C_1} is $m_{2,3}$ and none of the numbers of conflicts in the third column is smaller than $m_{2,3}$, the initial configuration M'_6 fails. Then we add a new time slot, and get the interference submatrix M'_4. One initial configuration of matrix M'_4 is shown as the matrix M'_7, and the corresponding conflict matrix of matrix M'_7 is shown as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Algorithm 4 Minimum conflicts heuristic

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

1: Construct an interference submatrix M' consisting of the smallest w_i consecutive rows that have at least one "0" in each row.
2: **while** a valid assignment is not obtained **do**
3: Initialize the matrix M' with a random configuration.
4: Count the number of conflicts in M', and obtain the conflict matrix M_C.
5: Use a heuristic to reduce the interference until a valid assignment is obtained, moving the selection "1" with the largest number of conflicts to the position in the same column, where the number of conflicts is minimum.
6: **if** the minimum conflicts heuristic fails **then**
7: Update M' by deleting the first row of M' and adding a zero row vector $(0)_{1 \times w_i}$ in the last row of M'.

the matrix M_{C_2}. By moving the selection from $m_{4,4}$ to $m_{3,4}$ in M'_7 according to the minimum conflicts heuristic, we can obtain a valid assignment which is same with M'_5 in the recursive backtracking.

$$M'_6 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & \textcircled{1} & - \\ - & 0 & 1 & - \\ - & 0 & 0 & 1 \end{pmatrix}, \quad M_{C_1} = \begin{pmatrix} - & 1 & 3 & - \\ - & 2 & \textcircled{2} & - \\ - & 2 & 2 & - \\ - & 2 & 3 & 1 \end{pmatrix},$$

$$M'_7 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & 1 & - \\ - & 0 & 0 & \textcircled{0} \\ 1 & 0 & 0 & \textcircled{1} \end{pmatrix}, \quad M_{C_2} = \begin{pmatrix} - & 1 & 2 & - \\ - & 2 & 1 & - \\ - & 1 & 1 & 1 \\ 2 & 3 & 3 & \textcircled{2} \end{pmatrix}.$$

4 Distributed Algorithms

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

4.1 Distributed Scheduling

In the distributed scheduling, we use a random order rather than a global decreasing order of the weights, and we assume that there is a contention-based MAC (e.g. B-MAC [20]) available for a node to compete for the channel and to obtain an interference-free contiguous link scheduling. The distributed scheduling is simple and efficient so that each sensor node can run the scheduling with less computation. The distributed scheduling is shown in Algorithm 5.

In the distributed scheduling algorithm, a node v_i first competes to obtain the channel, then assigns the smallest consecutive time slots for its incident links without interferences using the first-fit heuristic same as the centralized

Algorithm 5 Distributed scheduling (Distributed)

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

1: **while** a valid contiguous link scheduling is not obtained **do**
2: Each node v_i that is not yet scheduled monitors and competes for the channel.
3: **if** node v_i obtains the channel **then**
4: v_i assigns the smallest w_i consecutive time slots sequentially to its incident links which do not interfere with the links that have already been scheduled. Suppose link $l_{j,i}$ incident to v_i is assigned time slot t_j.
5: v_i broadcasts the assignment information to the nodes that are in the interference range of v_i, and these nodes could not transmit in the w_i time slots due to the interferences.
6: Each node v_j adjacent to v_i broadcasts the assignment information to the nodes that are in the interference range of v_j, and these nodes could not receive in time slot t_j due to the interferences.
7: **else**
8: v_i waits for a random time.

scheduling. After that, v_i and its neighbors should notify the nodes in their interference ranges the time slots they could not use. For node v_i, it broadcasts the assignment information to the nodes that are in the interference range of v_i, and these nodes could not transmit in these w_i time slots because their transmissions will interfere v_i's packet reception in these w_i time slots. For each link $l_{j,i}$ incident to v_i, time slot t_j is assigned. Node v_j then broadcasts the assignment information to the nodes that are in its interference range, and these nodes could not use time slot t_j to receive packets because their packet receiving will be interfered by v_j's packet transmitting in time slot t_j.

4.2 Distributed Scheduling with Efficient Delay

In the TDMA sleep scheduling, a node stays in the sleep state for most time, and periodically starts up to check for activity. As a forwarding node has to wait until its next-hop neighbor starts up and is ready to receive, the message delivery delay will increase. When packets are forwarded from an incoming link to an outgoing link, they could only be forwarded to the outgoing link in the next period T if the incoming link is scheduled to be active after the outgoing link. This kind of delay will accumulate at every hop in the network, which may lead to a long latency. A sample network is illustrated in Fig. 5(a). As a line topology, the data is transmitted from v_5 to v_1 along the line. If links e_1, e_2, e_3, e_4 are assigned time slots t_1, t_2, t_3, t_4 respectively (Schedule 1), the time delay for a packet transmission from v_5 to v_1 is almost $3T$, as shown in Fig. 5(b). However, if links e_1, e_2, e_3, e_4 are assigned time slots t_4, t_3, t_2, t_1 respectively (Schedule 2), v_5 could transmit the data to v_1 in one period T. In order to reduce this delay,

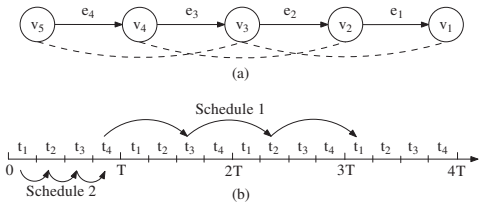


Fig. 5: Distributed scheduling with efficient delay: (a) Line topology of 5 nodes, (b) Scheduling delay. Dashed lines represent the interference relationship.

we schedule the links from bottom to top, that is, a node with a higher depth should be scheduled earlier. Hence, a node v_i can only be scheduled until all the children nodes of v_i are already scheduled. The algorithm is described in Algorithm 6.

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

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

1: **while** a valid contiguous link scheduling is not obtained **do**
2: Each node v_i that is not yet scheduled monitors and competes for the channel if all the children nodes of v_i are already scheduled.
3: **if** node v_i obtains the channel **then**
4: v_i assigns the smallest w_i consecutive time slots sequentially to its incident links which do not interfere with the links that have already been scheduled. Suppose link $l_{j,i}$ incident to v_i is assigned time slot t_j.
5: v_i broadcasts the assignment information to the nodes that are in the interference range of v_i, and these nodes could not transmit in the w_i time slots due to the interferences.
6: Each node v_j adjacent to v_i broadcasts the assignment information to the nodes that are in the interference range of v_j, and these nodes could not receive in time slot t_j due to the interferences.
7: **else**
8: v_i waits for a random time.

Example 5. A sample of the distributed scheduling with efficient delay is as shown in Fig. 4(d). As the scheduling is from bottom to top, the scheduling order is v_9, v_7, v_2, v_5 and v_1. After the scheduling, v_9 is assigned consecutive time slots from t_1 to t_5, v_7 is assigned consecutive time slots from t_3 to t_6, v_2 is assigned time slot t_1, v_5 is assigned time slot t_6 and v_1 is assigned consecutive time slots from t_7 to t_{10}. After the scheduling, the total number of time slots assigned is 10. The time slot assigned to each link is indicated in Fig. 4(d).

5 Performance Analysis

In this section, we analyze the theoretical performance of our proposed algorithms. We first present the approximation ratios of these algorithms as theorems, and then show their time and message complexities. The details of the performance analysis, including all the proofs and complexity analysis of the algorithms, are given in the supplementary file.

Theorem 2. *The number of time slots used by the centralized scheduling algorithm is at most a constant factor of the optimum.*

Theorem 3. *The number of time slots used by the centralized scheduling with spatial reuse algorithm is at most a constant factor of the optimum.*

Theorem 4. *The number of time slots used by the distributed scheduling algorithm is at most $\Theta(K)$ times of the optimum, where $K = \frac{w_{max}}{w_{min}}$, $w_{max} = \max_{1 \le l \le L} w_{i_l}$ and $w_{min} = \min_{1 \le l \le L} w_{i_l}$.*

Theorem 5. *The number of time slots used by the distributed scheduling with efficient delay algorithm is at most $\Theta(K)$ times of the optimum.*

The time and message complexities of our algorithms are shown in Table 2. Here, n denotes the number of nodes in the network, Δ denotes the maximum number of links incident to a node in the communication graph G, Δ' denotes the maximum degree in the corresponding merged conflict graph G_{mc}, and ρ denotes the maximum number of k-hop neighbors of a node in the network, where $k = \lceil \gamma_{max} \rceil$.

TABLE 2: Time and message complexities of our algorithms.

Algorithm	Complexity	
	Time	Message
Centralized	$O(n^2)$	—
Recursive backtracking	$O(n^2 + n\Delta! + n\Delta'\Delta^2)$	—
Minimum conflict heuristic	$O(n^2 + n\Delta'\Delta^2)$	—
Distributed	$O(\Delta n)$	$O(\Delta \rho n)$
Distributed-delay	$O(\Delta n)$	$O(\Delta \rho n)$

6 Simulation Results

In this section, we evaluate the performance of our algorithms using a simulator built in C++. The algorithms compared in the simulation are the proposed centralized and distributed algorithms (centralized, recursive backtracking, minimum conflicts heuristic, distributed, distributed-delay), and the *degree-based heuristic* [7] where the contiguous link scheduling is not used and the communication links are scheduled one by one. The performance metrics used in the evaluation are total energy consumption, throughput, and time delay. We show the simulation results of our algorithms in heterogeneous networks here. The performance comparison of these algorithms in homogeneous networks is given in the supplementary file.

We adopt the time and power consumption in Table 1 in the simulation. We assume the packet size is 36 bytes, the data rate is 250kbps, a time slot is 4 *ms*. For each algorithm, the network operates 10000 scheduling periods. We construct a breadth first search (BFS) tree and a directed acyclic graph (DAG) rooted at the sink node as the topologies of the network.

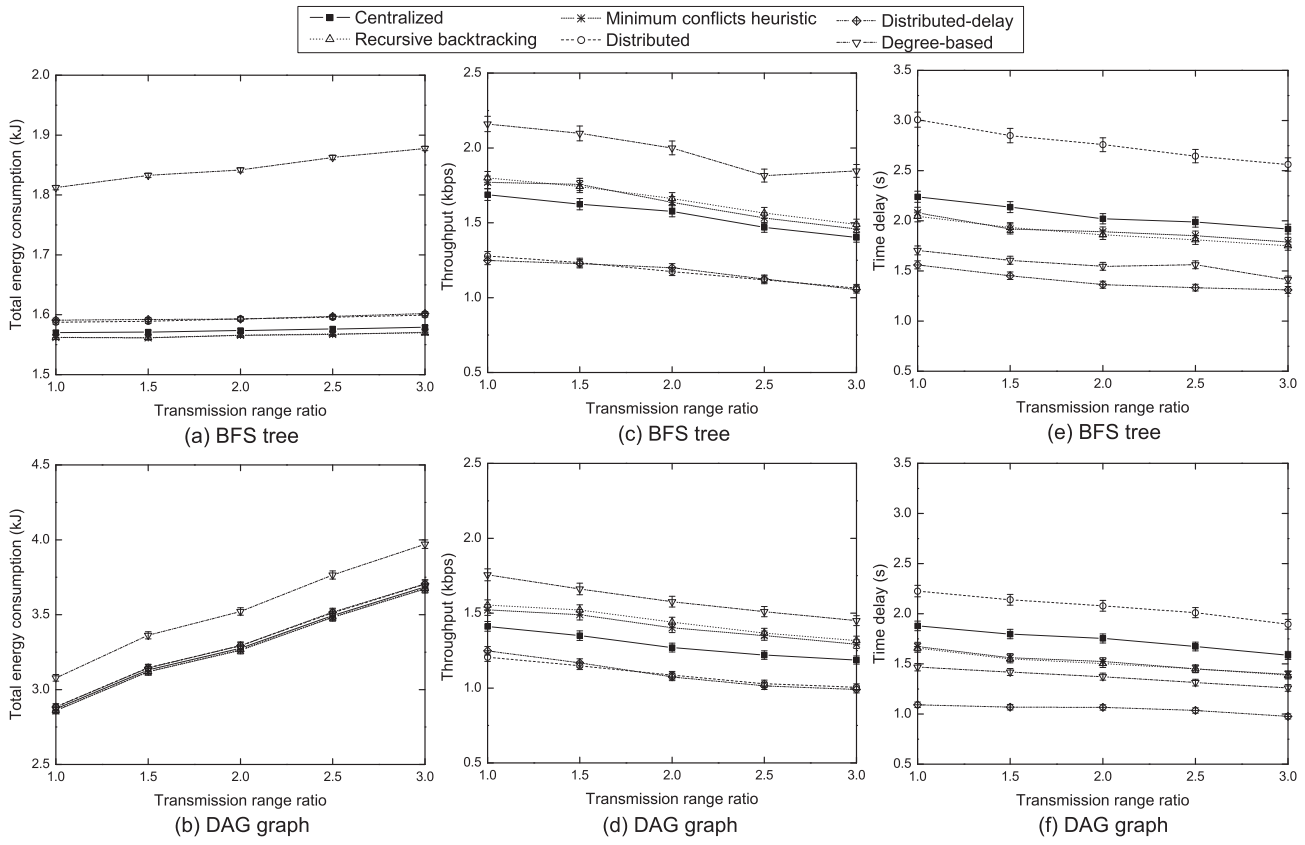


Fig. 6: Performance evaluation in heterogeneous networks.

For the heterogeneous networks, we randomly deploy 300 nodes in a square area of $100m \times 100m$, and we vary the transmission range ratio of the maximal transmission range to the minimal transmission range, $\sigma = \frac{r_{max}}{r_{min}}$, from 1 to 3 with a step of 0.5. The transmission range of each node follows a uniform distribution on the interval $[r_{min}, r_{max}]$ with an average of $15m$, and the interference range of each node is twice the transmission range. For each transmission range ratio, 50 network topologies are generated, and the average performances are reported. For each data point, we also draw its 90% confidence interval.

Figs. 6 (a) and (b) show the total energy consumption in heterogeneous networks. In both the BFS tree and DAG topologies, the total energy consumption in the contiguous link scheduling schemes is much less than in the degree-based scheme. The total energy consumption does not vary significantly as the transmission range ratio σ changes in the BFS tree, while the total energy consumption increases as the transmission range ratio σ increases in the DAG topology.

Figs. 6 (c) and (d) show the average throughput in heterogeneous networks. In both the BFS tree and DAG topologies, the throughput decreases as the transmission range ratio σ increases. This indicates that the heterogeneous nodes are detrimental to the contiguous link scheduling.

In the contiguous link scheduling, the links incident to one node are scheduled together to obtain consecutive time slots to avoid frequent state transitions, and several gaps are formed among the assigned time slots, which decreases the channel utilization and requires more time slots and hence decreases throughput. Figs. 6 (c) and (d) show that the overhead is not high, and the recursive backtracking scheduling scheme has performance comparable to the degree-based scheme. Although the minimum conflicts heuristic may get stuck on a local optimum, it almost has the same performance compared to recursive backtracking. If the centralized scheduling with spatial reuse is not used, the results would be a little worse, as shown in the centralized scheduling. The two distributed algorithms have the worst performance, due to the fact that they do not have the global information.

Figs. 6 (e) and (f) show the average time delay in heterogeneous networks. As the transmission range ratio σ increases, the average time delay decreases in both the BFS tree and DAG topologies because the depth of the network topology reduces as σ increases. In heterogeneous networks, the distributed scheduling with efficient delay scheme has the best performance.

We summarize observations from the simulation results as follows: (1) The proposed centralized and distributed algorithms can achieve better energy efficiency due to the reduction of the state transitions. (2) Our proposed distributed algorithms can achieve performance comparable to the centralized algorithms in both homogeneous and heterogeneous networks. (3) The distributed scheduling with efficient delay scheme can reduce the network delay. (4) In heterogeneous networks, the throughput decreases as the transmission range ratio σ increases, and the average time delay decreases as σ increases.

7 Conclusion

In this paper, we identify the contiguous link scheduling problem in WSNs, in which a sensor node starts up only once to receive all the data from its neighbors, and thus can reduce the energy consumption and time overhead in the state transitions. Especially, if the topology is a tree, each node can start up only twice in one scheduling period. We also propose centralized and distributed algorithms with theoretical performance bounds to the optimum in both homogeneous and heterogeneous networks. The simulation results corroborate the theoretical analysis, and show the efficiency of our algorithms in terms of total energy consumption, throughput, and time delay.

Acknowledgments

This work was supported in part by grants from Hong Kong RGC (PolyU 523606, PolyU 523207, PolyU 521312), Hong Kong PolyU (A-PL84, 1-VZ5N), and National Natural Science Foundation of China (No. 61272463).

References

[1] K. Kalpakis, K. Dasgupta, and P.Namjoshi, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

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

[3] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008.

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

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

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

[7] W. Wang, Y. Wang, X. Y. Li, W. Z. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2006.

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

[9] P. J. Wan, C. Ma, Z. Wang, B. Xu, M. Li, and X. Jia, "Weighted wireless link scheduling without information of positions and interference/communication radii," in *Proc. of IEEE INFOCOM*, 2011.

[10] Y. Zhou, X. Y. Li, M. Liu, Z. Li, S. Tang, X. Mao, and Q. Huang, "Distributed link scheduling for throughput maximization under physical interference model," in *Proc. of IEEE INFOCOM Mini-conference*, 2012.

[11] X. Xu, X. Y. Li, P. J. Wan, and M. Song, "Efficient aggregation scheduling in multihop wireless sensor networks with SINR constraints," *IEEE Transactions on Mobile Computing*, 2012.

[12] T. Alsulaiman, S. K. Prasad, and A. Zelikovsky, "Distributed algorithms for TDMA link scheduling in sensor networks," *International Journal of Networking and Computing*, vol. 3, no. 1, pp. 55–74, 2013.

[13] J. Ma, W. Lou, Y. Wu, X. Y. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2009.

[14] N. Thepvilojanapong, Y. Tobe, and K. Sezaki, "On the construction of efficient data gathering tree in wireless sensor networks," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.

[15] Q. Lampin, D. Barthel, and F. Valois, "Efficient route redundancy in DAG-based wireless sensor networks," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2010.

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

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

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

[19] Y. Wu, X. Y. Li, Y. Liu, and W. Lou, "Energy-efficient wake-up scheduling for data collection and aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 275–287, 2010.

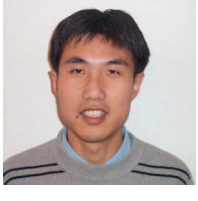
[20] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[21] S. Croce, F. Marcelloni, and M. Vecchio, "Reducing power consumption in wireless sensor networks using a novel approach to data aggregation," *The Computer Journal*, vol. 51, no. 2, pp. 227–239, 2008.

[22] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd Edtion, 2003.

Junchao Ma is a PhD student in Department of Computing at The Hong Kong Polytechnic University. He received B.Eng in Automation from Zhejiang University, China in 2006 and M.Eng. in Department of Electronic and Information Engineering from The Hong Kong Polytechnic University, Hong Kong in 2007. His research interests are broadly in the fields of wireless ad hoc and sensor networks, such as sleep scheduling, routing, topology control and privacy.

Dr. Wei Lou is currently an assistant professor in the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. He received the B.E. degree in Electrical Engineering from Tsinghua University, China in 1995, the M.E. degree in Telecommunications from Beijing University of Posts and Telecommunications, China in 1998, and the Ph.D. degree in Computer Engineering from Florida Atlantic University, USA in 2004. Dr. Lou's current research interests are in the areas of mobile ad hoc and sensor networks, peer-to-peer networks, mobile computing, and computer networks. 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.



Dr. Xiang-Yang Li is a professor at the Illinois Institute of Technology. He is recipient of China NSF Outstanding Overseas Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include the mobile computing, cyber physical systems, wireless networks, security and privacy, and algorithms.