Energy-Adaptive and Bottleneck-Aware Many-to-Many Communication Scheduling for Battery-Free WSNs

Bingkun Yao, Hong Gao, Quan Chen, and Jianzhong Li

***Abstract*—Battery-free wireless sensor networks (BF-WSNs) have captured the interest of research community in recent years. Compared with traditional battery-powered WSNs (BP-WSNs), BF-WSNs can prolong the lifetime of the network by exploiting ambient energy. Many-to-many communication is widely used in many applications of WSNs and the problem of minimum-latency many-to-many communication scheduling in BF-WSNs is of great significance. However, this problem has not been studied yet. The existing algorithms for BP-WSNs and BF-WSNs are not suitable for minimum-latency many-to-many scheduling problem in BF-WSNs. Also, different from BP-WSNs, energy-bottleneck nodes with low recharge rate and high workload in BF-WSNs make the problem more challenging. To address these issues, in this article, we first study the problem of many-to-many scheduling in BF-WSNs with the purpose of minimizing communication latency. The problem is formally defined and proved to be NP-hard. The energy-adaptive and bottleneck-aware scheduling algorithm for many to many in BF-WSNs is proposed. The correctness and average latency of the proposed algorithm are carefully analyzed. Extensive simulations show that our algorithm has high performance, in terms of communication latency and energy usage ratio. Furthermore, we also extend the proposed algorithm to other network models.**

***Index Terms*—Battery-free wireless sensor networks (BF-WSNs), bottleneck aware, energy adaptive, many-to-many communication scheduling, minimum latency.**

I. Introduction

WIRELESS sensor networks (WSNs) can be used to monitor the physical world, acting as significant components of the Internet of Things (IoT). Traditional WSNs powered by batteries (BP-WSNs) have limited lifetime since its infeasible to replace their batteries in many scenarios (e.g., in forests and volcanos). To address this challenge, battery-free WSNs (BF-WSNs) are proposed and widely studied [1]. Compared with BP-WSNs, nodes in BF-WSNs are equipped with energy collection devices to harvest energy from ambient sources (e.g., solar energy [2] and RF energy [3]). Therefore, the lifetime of BF-WSN is much longer than that of BP-WSN from the energy aspect.

Many-to-many communication where multiple sources transmit packets to multiple targets is an important data delivery pattern in both BP-WSNs and BF-WSNs, which can serve many other traffic patterns. For example, in-network data processing involves substantial data exchanges among many nodes (e.g., in-network machine learning [16]). To guarantee the time constraint for real-time applications, a fast, collision-free many-to-many communication schedule is highly required. For BP-WSNs, this problem has been studied extensively [4]–[18]. However, they are not suitable for BF-WSNs for the following three reasons.

1) Unlike BP-WSNs, nodes in BF-WSNs have to gain enough energy from the ambient environment to support their operations. However, the energy harvesting rate is usually lower than the energy consumption rate. Therefore, the time spent for nodes to get recharged is a significant cause of transmission latency in BF-WSNs. Protocols for BP-WSNs do not take this into consideration. When computing routes, not only nodes with high energy harvesting rates but also nodes with low energy harvesting rates will be involved in the transmission path. Once nodes with low energy harvesting rates are selected as relay nodes, the transmission will suffer from huge latency.
2) Nodes are generally assumed to work in always-awake or duty-cycled manner in BP-WSNs, which means a node can be scheduled to work at any time (for always-awake WSNs) or in any fixed timeslots (for duty-cycled WSNs) if no transmission collision occurs. However, in BF-WSNs, the residual energy of a node may not be enough to support its operation and some time must be spent to harvest enough energy. Thus, nodes in BF-WSNs cannot assign their working slots arbitrarily.
3) For a node in BF-WSN, its residual energy fluctuates with time due to alternate energy consumption and harvesting. Thus, there may be *energy collision*s between its operations [19]. For example, assume that a node receives a packet at timeslot 7. After that, if it needs four slots to harvest enough energy to receive another packet, it cannot be scheduled to receive before timeslot 11. Under this circumstance, "reception at timeslot 7"

Manuscript received August 13, 2020; revised October 21, 2020; accepted December 15, 2020. Date of publication December 21, 2020; date of current version May 7, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61632010 and Grant 61802071; and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2018A030310541 and Grant 2020A1515011132. *(Corresponding author: Hong Gao.)*

Bingkun Yao, Hong Gao, and Jianzhong Li are with the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: bingkunyao@stu.hit.edu.cn; honggao@hit.edu.cn; lijzh@hit.edu.cn).

Quan Chen is with the School of Computers, Guangdong University of Technology, Guangzhou 510006, China (e-mail: quan.c@gdut.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.3045979

has energy collision with "reception at timeslot 10." This problem is not considered in protocols for BP-WSNs.

As mentioned above, scheduling algorithms for many-to-many transmission in BF-WSNs must be *energy adaptive*, which means nodes' energy status (i.e., recharge rate and initial energy) must be considered in constructing transmission structures and energy collision is avoided.

For BF-WSNs, many-to-many scheduling problem has not been investigated yet. In recent years, some related scheduling problems have been studied. [21]–[23] and [24]–[27] considered broadcast (one-to-all) and data collection (all-to-one), respectively, which are two special cases of many-to-many transmission. However, they are not applicable to many-to-many transmission scheduling in BF-WSNs.

In addition to the above issues, the scheduling problem in BF-WSNs itself is also challenging due to *energy bottlenecks* [21]. In BF-WSNs, each node has two states: 1) working and 2) recharging. We define the *recharge time* of a node as the time required for charging to satisfy its assigned workloads, as explained in Fig. 1. Fig. 1 illustrates two examples of transmission scheduling, in which source nodes s_1, s_2, and s_3 need to transmit a packet to target t, respectively. The energy of each node before scheduled transmission is 0 and each node can only send or receive a packet at a time. Energy consumption for sending and receiving a packet is 100 and 80, respectively. The recharge rate is labeled at each node. As shown in Fig. 1(a), all source nodes share the same route toward t (i.e., $v_5 \rightarrow v_2 \rightarrow t$). Take node v_5 as an example. v_5 needs to receive three packets from source nodes then relay them. Hence, its recharge time is the quotient of its required harvested energy divided by its recharge rate, namely, $3 \times (80+100)/25 = 21.6$. Among all nodes involved (i.e., $\{t, v_2, v_5, s_1, s_2, s_3, v_2\}$) has the maximum recharge time [i.e., $3 \times (80 + 100)/20 = 27$]. Obviously, this is a lower bound of overall latency (i.e., the time when the last transmission occurs). As v_2 is one hop away from t, the overall latency is 28. In Fig. 1(b), three different routes toward t are exploited. Among all involved nodes (i.e., $\{t, v_1, v_2, v_3, v_4, v_5, v_6, s_1, s_2, s_3\}$), v_3 and v_4 have the maximum recharge time [i.e., $(80 + 100)/15 = 12$] and v_4 is two hops away from t. Hence, the overall latency is 14.

As can be seen from above, the overall latency is dominated by nodes with the maximum recharge time. These nodes are referred to as energy bottlenecks [e.g., v_2 marked in red in Fig. 1(a), and v_3 and v_4 marked in red in Fig. 1(b)]. If energy bottlenecks are assigned with heavy workloads while their recharge rates are low, their recharge time is likely to be very huge, resulting in a huge overall latency.[1] Compared with Fig. 1(a), the schedule in Fig. 1(b) exploits harvested energy of more nodes (i.e., v_1, v_3, v_4, and v_6), which contributes to lower maximum recharge time. Therefore, we argue that the scheduling algorithm must be *bottleneck aware*, which means the harvested energy of all nodes in the network should be used to the best advantage and the recharge time of bottlenecks should be minimized. This is quite different from the bottleneck problem in BP-WSNs and rarely considered in existing works. As energy status varies dramatically among nodes, this is not a trivial problem.

[1]In the rest of this article, "bottleneck" refers to the energy bottleneck.

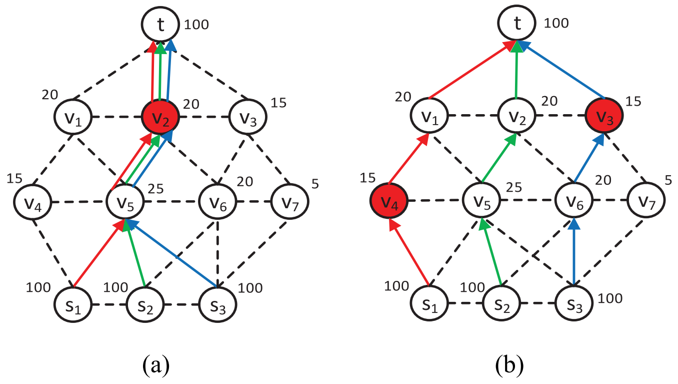


Fig. 1. Energy bottlenecks on scheduling. a) Overall latency: 28. b) Overall latency: 14.

To address the above issues, in this article, we first investigate minimum-latency many-to-many transmission scheduling problem in BF-WSNs, which considers two requirements mentioned above at the same time: 1) energy adaptive and 2) bottleneck aware. The main contributions of this article are as follows.

1) The problem of minimum-latency many-to-many transmission scheduling problem in BF-WSNs is formally defined and first studied.
2) An energy-adaptive and bottleneck-aware scheduling algorithm for many-to-many transmission in BF-WSNs (EBSA for short) is proposed to generate collision-free many-to-many schedule with low latency. EBSA consists of two phases. First, multiple evenly distributed clusters are formed to improve the energy efficiency of the schedule and a heuristic dynamic programming-based algorithm is proposed to compute the transmission structure for each cluster to minimize the recharge time of bottlenecks, in which nodes' energy status and load balancing are jointly considered. Second, an energy-adaptive scheduling algorithm is proposed to derive low-latency and collision-free many-to-many schedule.
3) The correctness, time complexity, and average latency of the EBSA algorithm are carefully analyzed.
4) Extensive simulations are carried out and the results verify that EBSA has high performance in terms of transmission latency and energy utilization compared with baselines.

The remainder of this article is as follows. Section II surveys the related work. Section III presents the definition of the problem of minimum-latency many-to-many transmission scheduling in BF-WSNs and proves its NP-hardness. Section IV describes the EBSA algorithm in detail. Section V analyzes EBSA theoretically. Section VI shows simulation results. EBSA for other network models is discussed in Section VII. Finally, Section VIII concludes this article.

II. Related Works

In terms of many-to-many transmission in BP-WSNs, lots of works have been done [4]–[18], which can be divided into three categories. The first category [4], [13]–[18] studied routing and in-network processing protocols for BP-WSNs. The second category [5], [6], [9], [10], [12] investigated many-to-many scheduling algorithms. The third category focused on data collection with multiple mobile sinks [7], [8], [11]. The above works focused on networks with nonrenewable batteries, they did not consider energy harvesting when constructing transmission structures and energy collision is not addressed. Hence, they cannot be used to solve the minimum-latency many-to-many transmission scheduling problem in BF-WSNs.

So far, the minimum-latency many-to-many transmission scheduling problem in BF-WSNs has not been considered yet. In recent years, some transmission scheduling algorithms related to many-to-many transmission for BF-WSNs have been studied [21]–[27].

Zhu *et al.* [21], Gu and He [22], and Baknina and Ulukus [23] studied data broadcast problems in BF-WSNs. Among them, Zhu *et al.* [21] investigated minimum-latency broadcast scheduling algorithm in multihop BF-WSNs. Given a preassigned duty-cycle for each node, Gu and He [22] tried to minimize the energy consumption satisfying a delay threshold. However, Zhu *et al.* [21] and Gu and He [22] only considered a special case of many-to-many transmission (i.e., one-to-all). In many-to-many scenarios, only a subset of nodes in the network needs to be involved in scheduling. How to select these nodes is very important. Besides, Baknina and Ulukus [23] studied the problem of optimal transmission scheduling on a multiuser AWGN broadcast channel. Nevertheless, it only discussed the single-hop scenario.

Zhu *et al.* [24], Liu and Chen [25], Dong *et al.* [26], and Mehrabi and Kim [27] focused on data collection in BF-WSNs. Zhu *et al.* [24] investigated the problem of minimum-latency data collection scheduling. However, like [21], it only studied a special case of many-to-many transmission (i.e., all-to-one transmission). Liu and Chen [25] discussed how to maximize throughput subject to energy neutrality constraints, but it did not account for transmission latency. Dong *et al.* [26] proposed NERF, a packet prioritization and transmission protocol for gathering information of all nodes with a delay bound. Nevertheless, Dong *et al.* [26] only considered a single-hop star network. Mehrabi and Kim [27] studied the problem of maximizing data collection utility with a mobile sink. However, it assumed that each sensor can reach the sink with no more than two hops.

III. Problem Statement

A. Network and Energy Model

Consider a multihop BF-WSN $G = (V, E)$ deployed in a monitoring area in which V is the set of battery-free nodes. Let $\text{dist}(v_1, v_2)$ denote the Euclidean distance between any two nodes $v_1, v_2 \in V$ and r_t denote the transmission radius of each node. If $\text{dist}(v_1, v_2) \le r_t$, v_1 and v_2 can communicate with each other directly and there is an edge $(v_1, v_2) \in E$, also which denotes the neighborhood relationship between node v_1 and v_2. The working period of each node is divided into timeslots with the same length that is enough for sending or receiving one packet successfully. All nodes share a common channel. We assume the network is synchronized as in [19]–[21] and [24].

Each battery-free node is equipped with a supercapacitor whose capacity is $b_{\max}$ to store its harvested energy, which is the same for each node. A node must have enough residual energy if it is scheduled to work (i.e., send or receiving packets) in any timeslot. Let e_s and e_r denote the energy consumption for sending and receiving a packet, respectively. We assume that $b_{\max} \ge \text{Max}\{e_s, e_r\}$. The recharge rate of each node v denoted by $eh(v)$ can be predicted accurately in a period of time (e.g., 20 min). Therefore, the residual energy of node v at the start of the ith slot denoted by $b_v[i]$ is updated as follows:

$$b_v[i] = \begin{cases} \text{Min}\{b_v[i-1] + \text{eh}(v) - e_r,\ b_{\max}\} \\ \quad \text{If } b_v[i-1] \ge e_r \text{ and } v \text{ receives a packet at slot } i-1 \\ \text{Min}\{b_v[i-1] + \text{eh}(v) - e_s,\ b_{\max}\} \\ \quad \text{If } b_v[i-1] \ge e_s \text{ and } v \text{ sends a packet at slot } i-1 \\ \text{Min}\{b_v[i-1] + \text{eh}(v),\ b_{\max}\} \\ \quad v \text{ stays idle at timeslot } i-1. \end{cases}$$

B. Problem Definition

Now, we formalize our problem in detail. In a BF-WSN, the set of all sources is denoted as S and $S \subseteq V$. We use mapping $\Gamma : S \to 2^V/\emptyset$ to represent the relationship between each source s and its corresponding targets $\Gamma(s)$. That is, each source $s \in S$ is expected to send a packet to all nodes in $\Gamma(s)$.

In the process of scheduling, we use a quintuple referred as *schedule item* to denote a one-hop transmission, namely, sd = (sender, source, time, receiver, and type) where sender, source $\in V$, time $\in \mathbb{Z}^+$, receiver $\subseteq V$, and type $\in \{1, 2, 3\}$. It means that the node sender is scheduled to send the packet originated from the node source to nodes in the set receiver at timeslot time and sender is a one-hop neighbor of each node in receiver (The meaning of field type is explained in Section IV). A many-to-many schedule is a set of schedule items denoted by I. Any pair of schedule items in I may have two types of collision, namely, *time collision* and energy collision [19], which are explained in detail as follows.

1) *Time Collision:* In the protocol interference model, a node cannot receive more than one data packet from different senders at the same timeslot successfully. Therefore, given a schedule-item set I and its two different schedule items, namely, $sd_1 = (\text{sender}_1, \text{source}_1, \text{time}_1, \text{receiver}_1, \text{type}_1)$ and $sd_2 = (\text{sender}_2, \text{source}_2, \text{time}_2, \text{receiver}_2, \text{type}_2)$ are free of time collision, if and only if one of the two following conditions satisfies: a) $\text{time}_1 \ne \text{time}_2$ and b) $\text{time}_1 = \text{time}_2, \text{sender}_1 \notin \cup_{v \in \text{receiver}_2} NB(v)$ and $\text{sender}_2 \notin \cup_{v \in \text{receiver}_1} NB(v)$, where $NB(v)$ is the set of all one-hop neighboring nodes of v.
2) *Energy Collision [19]:* Two one-hop transmissions may conflict with each other for energy constraints. Fig. 2 shows two examples of scheduling. Assume that energy consumed for sending and receiving a packet is $e_s = 100$ and $e_r = 80$, respectively. In Fig. 2(a), there are two schedule items sd_1 and sd_2 and $b_{v_1}[10] = 120$. If sd_1 is executed (i.e., v_1 sends the packet originated from s

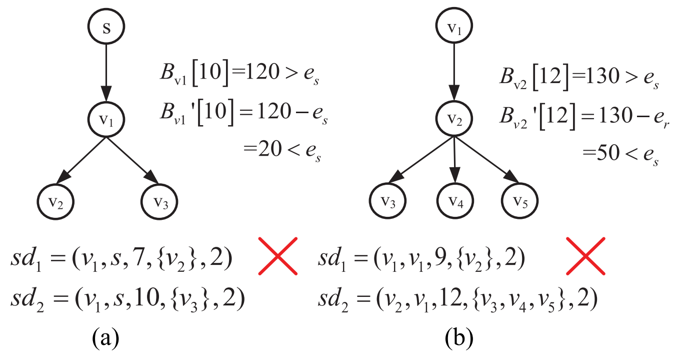


Fig. 2. Examples of energy collision where $e_s = 100$ and $e_r = 80$. a) Node v_1 can not be scheduled to send the packet at timeslot 7. b) Node v_2 can not be scheduled to receive the packet at timeslot 9.

at timeslot 7), b_{v_1} [10] would reduce from 120 to 20 < e_s, which is not enough to send a packet at timeslot 10, making sd_2 (i.e., v_1 is scheduled to send the packet originated from s at timeslot 10) nonexecutable. In this condition, we say that sd_1 has energy collision with sd_2. Similarly, in Fig. 2(b), sd_2 will become nonexecutable if sd_1 is executed (b_{v_2} [12] will decrease by e_r to 50 < e_s). Therefore, energy collision is defined as follows.

Definition 1 (Energy Collision): Given a set of schedule item I and two schedule items in I, namely, $sd_1 = (sender_1, source_1, time_1, receiver_1, type_1)$ and $sd_2 = (sender_2, source_2, time_2, receiver_2, type_2)$, there is an energy collision between them if and only if the following conditions are satisfied: a) $time_1 < time_2$ and b) if sd_1 is executed, sd_2 becomes nonexecutable for energy constraints.

The *latency* of I is defined as the maximum time of all schedule items in I. I is a *proper* many-to-many schedule, which means the schedule is free of both types of collision and all data packets should be received by their targets. We expect a proper many-to-many schedule with the minimum latency. To be specified, the minimum-latency many-to-many transmission scheduling problem in BF-WSNs is defined as follows.

Input:

1) A BF-WSN $G = (V, E)$.
2) The set of sources $S \subseteq V$, and a mapping $\Gamma : S \to 2^V/\emptyset$ indicating each source and its corresponding target nodes.
3) $b_v[1]$ and $eh(v)$ for each $v \in V$, e_s and e_r.

Output: Output a set $I = \{sd_1, sd_2, \ldots, sd_n\}$ composed of schedule items that satisfy the following.

1) For each $sd_i (1 \le i \le n)$, $b_{\text{sender}_i}[\text{time}_i] \ge e_s$ and $b_v[\text{time}_i] \ge e_r$ for each $v \in \text{receiver}_i$.
2) For each $s \in S$ and for each $v \in \Gamma(s)$, there exists $sd_i \in I$ such that $\text{source}_i = s$ and $v \in \text{receiver}_i$.
3) For each sd_i, one of the following conditions must be satistied: a) $\text{sender}_i = \text{source}_i$ and b) there exists $sd_j (j \neq i)$ such that $\text{source}_j = \text{source}_i$, $\text{sender}_i \in \text{receiver}_j$ and $\text{time}_j < \text{time}_i$.
4) For each $sd_i \in I$ and for each $v \in \text{receiver}_i$, $\text{sender}_i \in NB(v)$.
5) Schedule-items in I are pairwise free of time collision and energy collision.
6) The latency of I is minimized.

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$G = (V, E)$	a BF-WSN
$dist(v_1, v_2)$	the Euclidean distance between v_1 and v_2
r_t	the transmission radius of node
S	the set of sources
$eh(v)$	energy harvested per timeslot of node v
$b_v[i]$	residual energy of node v at the beginning of i-th timeslot
$\Gamma(s)$	target nodes of the data packet generated by node s
$NB(v)$	the set of one-hop neighbors of node v
b_{max}	the battery capacity
e_s	energy consumption for sending a packet
e_r	energy consumption for receiving a packet
sd	a schedule-item sd = ($sender$,$source$,$time$,$receiver$,$type$)
I	a many-to-many schedule consisting of schedule-items
CH	the set of all cluster heads
$CM(c)$	the set of sources that select c as its cluster head where $c \in CH$
$rct(v)$	the recharge time of node v
$LC(s, c)$	the data collection route from s to c where $s \in CM$(c)
$DT(s)$	multicast tree for packet generated by s
Q	the set of all schedule-items
$SEQ(H)$	A directed graph indicating a sequence for determining *time*s of schedule-items in H
$SD(c)$	the set of all schedule-items of data collection and data distribution rooted at c
$CS_c[i]$	the i-th group of schedule-items of data collection rooted at c
$DM_s[i]$	the i-th group of schedule-items whose *sender*s are dominators for distributing the packet generated by s
$CN_s[i]$	the i-th group of schedule-items whose *sender*s are connectors for distributing the packet generated by s
$F2S_v$	the forbidden-to-send vector of node v
$F2R_v$	the forbidden-to-receive vector of node v
$R(c)$	the network radius of G centered at c
$h(v_1, v_2)$	the minimum hop count between v_1 and v_2
R	the radius of network G = (V, E)

Condition 1 ensures that the energy constraints are being satisfied. Condition 2 means that the packet from each source will be received by all its corresponding targets. As for condition 3, for a schedule item sd_i, sender_i must have received the packet generated by source_i before slot time_i. Conditions 1–5 ensure that I is a proper many-to-many schedule.

Theorem 1: The minimum-latency many-to-many transmission scheduling problem in BF-WSNs is NP-hard.

Proof: Consider a special case of our problem. When $|\Gamma(s)| = 1$ for each $s \in S$ and $eh(v) \ge \text{Max}(e_s, e_r)$ for each $v \in V$, each node is always awake. In this condition, the minimum-latency many-to-many transmission scheduling problem in BF-WSNs is equal to the minimum latency flow scheduling problem, where each source has only one target, which has been proved to be NP-hard in [9]. Since this special case is NP-hard, our problem is also NP-hard. ■

Table I lists key symbols and notations used in this article.

IV. Energy-Adaptive and Bottleneck-Aware Scheduling

In this section, we describe EBSA in detail. First, based on a clustering method, we use a dynamic programming-based method to construct the transmission structure that consisted of routes and trees for each cluster in a bottleneck-aware manner.

Then, schedule items for each cluster are extracted from the transmission structure. Second, an energy-adaptive scheduling algorithm is proposed to generate the many-to-many schedule.

A. Compute Many-to-Many Transmission Structures and Corresponding Schedule Items

In this section, we discuss the computation of transmission structures and corresponding schedule items based on the clustering method. In our algorithm, the network is divided into several clusters and each source belongs to a cluster. In each cluster, the transmission procedure includes two parts: 1) data collection, where the cluster head gathers all packets generated by sources in the cluster and 2) data distribution, which means the cluster head distributes its gathered packets to corresponding targets. Thus, in this section, the transmission structures and corresponding schedule items are computed in the following three stages: 1) clustering stage, in which clusters are formed by dividing the monitoring region into uniform areas; 2) collection stage, which computes transmission structures and corresponding schedule items of data collection; and 3) distribution stage, which computes transmission structures and corresponding schedule items for data distribution. Next, we discuss three stages in detail.

1) Clustering Stage: In the clustering stage, the network is divided into clusters. To fully exploit the harvested energy of all nodes in the network, cluster heads should be distributed uniformly. Also, the average distance between sources and corresponding cluster heads needs to be as small as possible. Thus, we exploit the *field division algorithm* proposed in [28] to partition the monitoring field into similar-size and isotropic regions. Since cluster heads are likely to be assigned with heavy workloads, in each region, a node with the maximum energy harvesting rate is designated as the cluster head. The set of all cluster heads is denoted as CH. For each source $s \in S$, s selects the node in CH with the minimum hop count from s as its cluster head. The set of all sources that select c as cluster head is denoted as $CM(c)$.

2) Collection Stage: This stage consists of two phases. In the first phase, the transmission structure of data collection is constructed while in the second phase, corresponding schedule items are extracted from the transmission structures. Two phases are summarized in Algorithm 1.

a) Computation of transmission structure: For each cluster head $c \in CH$, the transmission structure in its cluster consists of data collection routes from each source $s \in CM(c)$ to c, which are calculated as follows. First, all nodes are partitioned into layers by breadth-first search starting from c. For each $v \in V$, let $h(c, v)$ denote the minimum hop count from c to v and v is in the $h(c, v)$th layer. Second, for each source $s \in CM(c)$, the data collection route from s to c denoted by $LC(s, c)$ is computed, in which a node v can only choose a node in its upper layer as the next hop. Third, the recharge time of each node on the route is updated. Let $snum(v)$ and $rnum(v)$ denote the number of packets to be sent and received by node v, respectively. Then, as mentioned in Section I, the *recharge time* of node v denoted by $rct(v)$ can be computed

Algorithm 1 Collection Stage

```
Require: $G = (V, E)$, the set of sources $S$, the set of cluster heads $CH$ and
    $CM(c)$ for each $c \in CH$.
Ensure: Schedule-items of data collection phase.
1: for $\forall c \in CH$ do                ▷ Computing transmission structure.
2:     Partition $V$ into layers by BFS starting from $c$;
3:     for $\forall s \in CM(c)$ do
4:         Compute $LC(s, c)$;
5:         Update $rct(v)$ for each $v$ on $LC(s, c)$;
6:     end for
7: end for
8: Generate schedule-items of data collection;          ▷
   Extraction of schedule − items.
```

as: $rct(v) = (snum(v) \times e_s + rnum(v) \times e_r - b_v[1])/eh(v)$. For each $s \in CM(c)$, $LC(s, c)$ is computed in the following two steps.

Step 1) Before the computation, we mention that new route $LC(s, c)$ brings extra workload to the network. For each $v \in V, v$ is scheduled to send/receive the packet originated from s if v is on the route from s to c. In this condition, its recharge time will become $rct'(v)$, which is calculated as follows.

a) If $v = s$, on $LC(s, c)$, v only needs to send the packet originated from itself. Thus, $rct'(v) = rct(v) + e_s/eh(v)$.

b) If $v = c$, on $LC(s, c)$, v only needs to receive the packet originated from s. Hence, $rct'(v) = rct(v) + e_r/eh(v)$.

c) Otherwise, v will relay the packet originated from s. Therefore, $rct'(v) = rct(v) + (e_r + e_s)/eh(v)$.

Step 2) The data collection route from s to c should be computed in a bottleneck-aware manner. Thus, according to the definition of bottlenecks, $LC(s, c)$ is computed with the aim to minimize the maximum $rct'(v)$, where $v \in LC(s, c)$. For an arbitrary route L from a node to another node, the maximum $rct'(v)$ where $v \in L$ is denoted by $\max\_\mathrm{rt}(L)$. Let $M_r(s, v) = \mathrm{Min}\{\max\_\mathrm{rt}(L) | L \textit{ is a route from } s \textit{ to } v\}$. Then, we have following recursive relation, where $L_{\mathrm{can}}(v)$ denotes the last hop candidate of v, namely, $\{u | u \in V \& h(c, u) = h(c, v) + 1 \& u \in NB(v)\}$

$$M_r(s, v) = \mathrm{Min}\big\{\mathrm{Max}[M_r(su), rct'(v)] | u \in L_{\mathrm{can}}(v)\big\}. \tag{1}$$

According to (1), $LC(s, c)$ can be computed similarly to a dynamic-programming shortest path algorithm. Since the last hop candidates of a node are all in its lower layer, the time consumption is $O(|V| + |E|)$. Notice that in this procedure, for each node $v \in V$, when selecting v's last hop, sometimes there are multiple nodes $\{u_1, u_2, \ldots, u_n\} \subseteq L_{\mathrm{can}}(v)$ with the same $\mathrm{Max}[M_r(s, u_i), rct'(v)] (1 \le i \le n)$. In this case, u with the minimum summation of $rct'(v)$ of all nodes on $LC(s, u)$ is selected.

Theorem 2: $LC(s, c)$ derived by the dynamic-programming algorithm above is the route from s to c with the minimum $\max\_\mathrm{rt}(L)$.

Algorithm 2 Distribution Stage

```
Require: $G = (V, E)$, the set of sources $S$, target set $\Gamma(s)$ for each $s \in S$, the
    set of cluster heads $CH$ and $CM(c)$ for each $c \in CH$.
Ensure: Schedule-items of data distribution phase.
 1: for $\forall c \in CH$ do                    ▷ Computing transmission structure.
 2:     Construct a CDS starting from $c$;
 3:     for $\forall s \in CM(c)$ do
 4:         for $\forall t \in \Gamma(s)$ do
 5:             Compute $LD(c, t)$;
 6:             Update $rct(v)$ and $DT(s)$;
 7:         end for
 8:     end for
 9: end for
10: Generate schedule-items of data distribution;     ▷
    Extraction of schedule − items.
```

Proof: We only need to prove that (1) is correct by contradiction. Assume that there exists an optimal route that satisfies $M_r'(s, c) < M_r(s, c)$. There are two cases: 1) the maximum recharge rate of all nodes along this route is $rct'(c)$. Let h denote the last hop node of c. Obviously, $M_r(s, h) \le M_r'(s, h) < rct'(c)$ so that $\text{Max}[M_r(s, h), rct'(c)] = rct'(c)$. Therefore, $\text{Min}\{\text{Max}[M_r(s, u), rct'(c)] | u \in L_{\text{can}}(c)\} = rct'(c)$, which makes a contradiction and 2) in this case, we have $rct'(c) \le M_r^{prime}(s, h) = M_r^{prime}(s, c)$. Nevertheless, $M_r(s, h) \le M_r(s, h)$. Thus, the maximum recharge rate along the route through node h to c is $\text{Max}[M_r(s, h), rct'(c)]$, which also makes a contradiction. ∎

b) Extraction of schedule items: For each cluster head $c \in CH$, after $LC(s, c)$ is constructed for each $s \in CM(c)$, corresponding schedule items are generated as follows. For each source $s \in S$ and each edge (v_1, v_2) on $LC(s, c)$, a new schedule item $sd = (v_1, s, -1, \{v_2\}, 1)$ is generated, in which type $= 1$ indicates that sd is a schedule item of data collection. These generated schedule items are referred to as *schedule items of data collection rooted at c*.

3) Distribution Stage: Similar to the collection stage, in this stage, transmission structure and corresponding schedule-items for data distribution are also computed in two phases, respectively, which are summarized in Algorithm 2.

a) Computation of transmission structure: For each cluster head $c \in CH$, the transmission structure for data distribution rooted at c is constructed by computing the multicast tree for the packet originated from each source s in $CM(c)$, which is denoted by $DT(s)$. Notice that $DT(s)$ spans all targets in $\Gamma(s)$ and all leaves of $DT(s)$ are targets of s. Next, we discuss how to compute $DT(s)$ for each source $s \in CM(c)$ in the following two steps. In the first step, a connected dominating set (CDS) is built starting from c in a top-to-bottom manner to divide all nodes into two disjoint sets: *dominators* and *connectors*. In the second step, for each $s \in CM(c)$, $DT(s)$ is constructed by computing the route from c to each target in $\Gamma(s)$, in which the route connects dominators and connectors alternately. The details for the above two steps are described as follows.

In the first step, to divide all nodes into dominators and connectors, a CDS is built from the top layer to the bottom layer starting from c. In the CDS, the set of dominators is actually a maximal independent set (MIS) while nodes not in this MIS are called connectors. Obviously, a connector is adjacent to at least one dominator. Thus, this step actually computes an MIS of the network by the method of computing a CDS. Mention that the minimization of the maximum recharge rate is considered in the computation of CDS. The details for this step are as follows.

1) Mark c as a dominator and all other nodes in V as connectors. For any node v, if v is adjacent to at least one node marked as a dominator, we say that v is *dominated*.
2) Let $R(c)$ denote the network radius centered at c. For $2 \le i \le R(c)$, select the dominators of ith layer as follows. Node set X is initialized as the set of connectors in ith layer that has not been dominated. Obviously, X is the set of dominator candidates. Similar to [10], in a CDS, a dominator in ith layer selects a connector in the $(i-1)$th layer as its parent. Thus, for each $v \in X$, the set of v's last-hop candidate nodes $L_{\text{can}}(v)$ is set as $NB(v) \cap \{u | h(c, u) = h(c, v) - 1$ & u is connector$\}$. Then, dominators in this layer are selected by repeating the following steps until X becomes empty. First, similar to Phase (a) of the collection stage , the recharge time of v if v is involved in CDS, denoted by $rct'(v)$ is computed as
$$rct'(v) = \text{Min}\big\{\text{Max}[rct'(u)rct(u) + (e_s + e_r)/eh(u), rct(v) + eh(v)] | u \in L_{\text{can}}(v)\}. \quad (2)$$
Then, let v' denote the node with the minimum $rct'(v')$ in X. v' is marked as dominator. v' and all nodes adjacent to v' are removed from X. After X becomes empty, nodes not marked as dominators in ith layer are connectors. Each connector v in the ith layer selects a dominator in $(i-1)$th or ith layer as its last hop in CDS. Thus, v sets $L_{\text{can}}(v)$ as $NB(v) \cap \{u | u \in \text{level}(h(c, v) - 1) \cup \text{level}(h(c, v))$ & *u is dominator*$\}$. Then, update $rct'(v)$ by (2).

In the second step, for each source $s \in CM(c)$, compute the data distribution route $LD(c, t)$ from c to each target node t in $\Gamma(s)$. After being computed, the route is integrated into $DT(s)$ and the recharge time of each node on the route is updated. $L(c, t)$ is computed as follows.

1) Like $LC(s, c)$, $LD(c, t)$ may bring new workloads to the network. For each node $v \in V$, let $rct'(v)$ denote its recharge time if v is on $LD(c, t)$. In the multicast tree for s, a node sends and receives the packet originated from s at most once. Thus, $rct'(v)$ is updated as follows.
   a) If v is a leaf node of $DT(s)$, v has been scheduled to receive the packet originated from s. Thus, $rct'(v) = rct(v) + e_s/eh(v)$.
   b) Else, if v is a internal node of $DT(s)$, v has been scheduled to relay the packet. Therefore, $rct'(v) = rct(v)$.
   c) Else, if $v = t$, as t is the destination of $LD(c, t)$, on this route, v only needs to receive the packet. $rct'(v) = rct(v) + e_r/eh(v)$.
   d) Otherwise, $rct'(v) = rct(v) + (e_s + e_r)/eh(v)$.
2) $LD(c, t)$ is computed with the aim of minimizing the maximum $rct'(v)$ that v is on $LD(c, t)$. The method is the same as computing for routes of data collection, which also ends in $O(|V| + |E|)$-time. In this procedure, a dominator in the ith layer can select a connector in the

$(i-1)$th layer as its last-hop node while a connector in the ith layer can select a dominator in the $(i-1)$th or ith layer as its last hop. But there is a restriction: if node v is already in $DT(s)$, its last hop is fixed as its parent in $DT(s)$. This ensures that after $LD(c, t)$ is integrated into $DT(s)$, $DT(s)$ is still a tree structure.

b) Extraction of schedule items: For each cluster head $c \in CH$, after $DT(s)$ is constructed for each source $s \in CM(c)$, corresponding schedule-items are derived accordingly as collection stage. For each source $s \in CM(c)$ and each edge (v_1, v_2) in $DT(s)$, if v_1 is a dominator, generate new schedule item $sd = (v_1, s, -1, \{v_2\}; 2)$ otherwise, v_1 is a connector and a new schedule item $sd = (v_1, s, -1, \{v_2\};$ and3) is generated. These schedule items are called *schedule items of data distribution rooted at c*.

B. Energy-Adaptive Scheduling

In the last section, schedule items for data collection and data distribution are derived. In this section, we discuss how to generate the many-to-many schedule. That is, for each schedule item sd = (sender, source, time, receiver, type), we determine time of sd. Given a set of schedule items H, a graph $SEQ(H) = (V_h, E_h)$ is defined as follows.

1) $SEQ(H)$ is directed and acyclic.
2) For each vertex $A \in V_h, A$ is a set of schedule items and $A \subseteq H$. $\cup_{A \in V_h} A = H$. For any two vertexes A, $B \in V_h, A \cap B = \emptyset$.
3) For any two vertexes A and B, if there exists a directed edge from A to B, for each $sd_1 \in A$ and each $sd_2 \in B$, time$_2$ will be determined after time$_1$ and time$_1$ < time$_2$.

Obviously, $SEQ(H)$ indicates a sequence for determining times of all schedule items in H.

In this section, for the set Q of all schedule items derived by the collection stage and distribution stage, we first construct $SEQ(Q)$, which indicates a sequence for determining times of all schedule items in Q. Then, the many-to-many schedule is derived according to $SEQ(Q)$. Therefore, this section contains the following two stages: 1) sequence stage, which constructs $SEQ(Q)$ and 2) schedule stage, in which times of all schedule items are determined according to $SEQ(Q)$. Next, we discuss two stages in detail.

1) Sequence Stage: For a cluster head $c \in CH$, let $SD(c)$ denote the set of all schedule items of data collection and data distribution rooted at c. $SEQ(Q)$ is constructed by constructing $SEQ(SD(c))$ for each $c \in CH$. $SEQ(SD(c))$ is constructed in the following two phases. In the first phase, we get the vertex set of $SEQ(SD(c))$; In the second phase, edges are added between vertexes. Two phases are summarized in Algorithm 3 and details of them are as follows.

a) Getting vertexes of SEQ(SD(c)): Schedule items in $SD(c)$ are divided into groups and each group is a vertex. $SD(c)$ consists of two types of schedule-items, one of data collection and the other of data distribution. First, schedule items of data collection rooted at c are divided into groups satisfying that schedule items in a group are pairwise free of time collision in the example shown in Fig. 3. The set of all

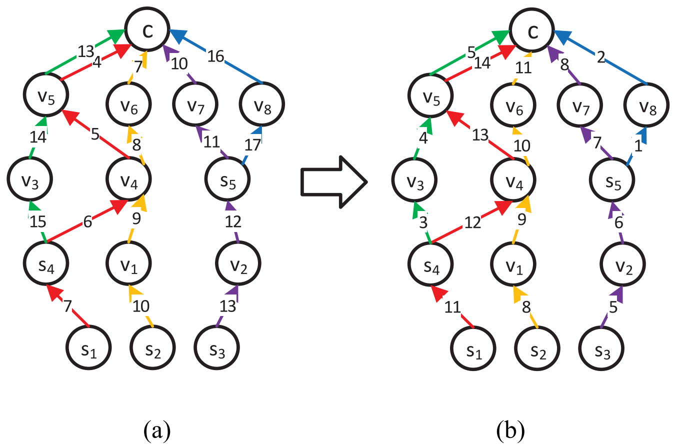


Fig. 3. Schedule items of data collection in the cluster headed at c are divided into 14 groups, namely, CS_c[1–14]. a) Compute seq number of all schedule items. b) Schedule item sd is put into CS_c [M+1-seq (sd)] where M = 17.

Algorithm 3 Sequence Stage

```
Require: The set of all schedule-items $Q = \cup_{c \in CH} SD(c)$;
Ensure: $SEQ(Q)$.
1: for $\forall c \in CH$ do
2:     Generate $CS_c$;                    ▷ Getting vertexes of SEQ(SD(c)).
3:     for $\forall s \in CM(c)$ do
4:         Generate $DM_s$ and $CN_s$;
5:     end for
6:     Adding edges between vertexes in $CS_c, DM_s$ and $CN_s$ for each $s \in$
       $CM(c)$.                            ▷ Adding edges to SEQ(SD(c)).
7: end for
8: $SEQ(Q) = \cup_{c \in CH} SD(c)$.
```

groups is denoted as CS_c and each group has a positive group number. The group whose number is i is denoted by $CS_c[i]$.

Each schedule item sd belongs to a group. The number of the group that sd belongs to is denoted as $seq(sd)$. As shown in Fig. 3, all source nodes in $CM(c)$ are ranked in ascending order of hop counts to c (i.e., $\{s_1, s_2, s_3, s_4, s_5\}$). Obviously, for any two schedule items sd_1 and sd_2, if the difference between the layer number of sender$_1$ and sender$_2$ is at least 3, sd_1 must be free of time collision with sd_2 [29]. Thus, for each schedule item $sd = (v_1, s_i, -1, \{v_2\}, 1)$, $seq(sd)$ is computed as

$$\text{seq(sd)} = 3 \times i + h(c, v_1). \tag{3}$$

Assume that the maximum seq of all schedule items of data collection rooted at c is M. Then, sd is put into $CS_c[M + 1 - seq(sd)]$. For example, in Fig. 3(a), the maximum seq of all schedule items is 17. In terms of schedule items extracted from the route starts from s_3 (marked in purple), schedule items corresponding to edges (s_3, v_2), (v_2, s_5), (s_5, v_7), and (v_7, c) are put into $CS_c[5]$, $CS_c[6]$, $CS_c[7]$, and $CS_c[8]$, respectively.

As we can see above, for a schedule item sd on the route starting from the source, compared with other schedule items extracted from this route, the closer its sender to the source, the smaller $seq(sd)$ is. Thus, times of those schedule items with smaller seqs should be determined before those with larger seqs. Namely, times of schedule items in $CS_c[i]$ should be determined after those in $CS_c[i-1]$.

Second, all schedule items of data distribution rooted at c are divided into groups. For each source $s \in CM(c)$, as mentioned in the last section, all nodes in $DT(s)$ can be divided into

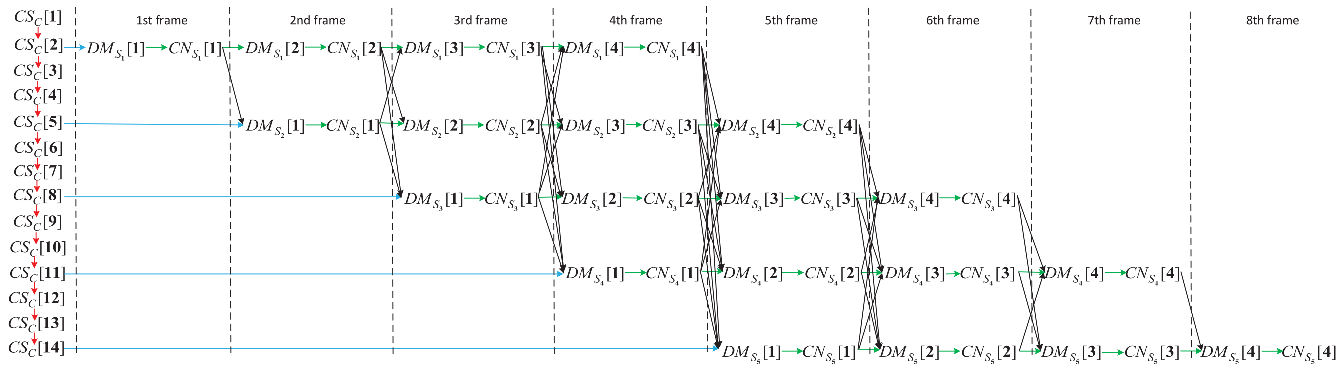


Fig. 4. Adding edges to $SEQ(SD(c))$ where $CM(c) = \{s_1, s_2, s_3, s_4, s_5\}$, $R(c) = 4$ and round$(s_{1-5}) = [2, 5, 8, 11, 14]$.

two types: 1) dominators and 2) connectors. Schedule items whose senders are dominators are divided into groups and the set of all groups is denoted as DM_s, in which schedule items with ith layer senders (i.e., the minimum hop count between sender and c is i) are divided into the same group denoted by $DM_s[i]$. Notice that for a schedule item sd, if sender $= c$, sd belongs to $DM_s[1]$. Likewise, schedule items whose senders are connectors are also divided into groups and the set of all groups is denoted as CN_s, in which the group denoted by $CN_s[i]$ stores schedule items whose senders are in ith layer. Obviously, we have $|DM_s|, |CN_s| \le R(c)$.

b) Adding edges to SEQ(SD(c)): In the last phase, we have gotten vertexes of $SEQ(SD(c))$: $CS_c[i]$ is a vertex where $1 \le i \le |CS_c|$; For each source $s \in CM(c)$ and $1 \le i \le R(c)$, $DM_s[i](CN_s[i])$ is a vertex. In this phase, edges are added between these vertexes. Before we introduce the details, for each source s, the *arrival round* of s is defined as follows.

Definition 2 (Arrival Round): For each $c \in CH$ and each source $s \in CM(c)$, there must exist a schedule item sd = (*sender, source, time, receiver, type*) where source $= s$ and receiver $= \{c\}$. If sd $\in$ CS$_c$[i], the arrival round of s is i, denoted as round$(s) = i$.

For example, as shown in Fig. 3(b), $CM(c) = \{s_1, s_2, s_3, s_4, s_5\}$ and their arrival rounds are 14, 11, 8, 5, and 2, respectively.

The procedure of adding edges to $SEQ(SD(c))$ consists of following three steps, which is illustrated by an example given in Fig. 4. Notice that $CM(c) = \{s_1, s_2, s_3, s_4, s_5\}$, $R(c) = 4$, and round$(s_{1-5}) = [2, 5, 8, 11, 14]$.

Step 1) Adding edges between vertexes in CS_c. times of schedule items in $CS_c[i-1]$ should be determined before those in $CS_c[i]$. Thus, for $2 \le i \le |CS_c|$, edge $(CS_c[i-1], CS_c[i])$ is added, which is marked in red in Fig. 4.

Step 2) Adding edges between vertexes in $\cup_{s \in CM(c)}(DM_s \cup CN_s)$. Assume that sources in $CM(c)$ have been ranked in ascending order of arrival round, namely, $CM(c) = \{s_1, s_2, \ldots, s_n\}$. Obviously, the packet of s_i is the ith packet to be received by c among all packets generated by sources in $CM(c)$. In data distribution, this packet is transmitted layer by layer in a top-to-bottom manner starting from c. In each layer, dominators send the packet of s_i earlier than connectors. Therefore, times of schedule items in $DMs_i[j]$ are determined before those in $CNs_i[j]$. Hence, edges $(DMs_i[j], CNs_i[j])$ and $(CNs_i[j], DMs_i[j+1])$ are added for $1 \le i \le n$ and $1 \le j \le R(c) - 1$ (marked in green in Fig. 4). Next, we notice that when the packet of s_i is sent by jth-layer nodes, the packet of s_{i+1} can be sent by nodes in $(j-1)$th layer, which means that for any two schedule items $sd_1 \in DMs_i[j] \cup CNs_i[j]$ and $sd_2 \in DMs_{i+1}[j-1] \cup CNs_{i+1}[j-1]$, time$_1$ is determined with the same priority with time$_2$. Similarly, for each schedule item $sd_1 \in DMs_i[j] \cup CNs_i[j]$, $sd_2 \in DMs_{i-p}[j+p] \cup CNs_{i-p}[j+p] (p \le i-1$ and $p \le R(c) - j)$, and $sd_3 \in DMs_{i+q}[j-q] \cup CNs_{i+q}[j-q] (q \le n-i$ and $q \le j-1)$, time$_1$, time$_2$, and time$_3$ are determined with the same priority. As mentioned before, there is an edge from $CNs_i[j]$ to $DMs_i[j+1]$. Since times of schedule items in $DMs_i[j+1], DMs_{i-p}[j+p+1] (p \le i-1$ and $p \le R(c)-j-1)$ and $DMs_{i+q}[j-q+1] (q \le n-i$ and $q \le j)$ are determined with the same priority, edges are added from $CNs_i[j]$ to $DMs_{i-p}[j+p+1] (p \le i-1$ and $p \le R(c)-j-1)$ and $DMs_{i+q}[j-q+1] (q \le n-i$ and $q \le j)$ (marked in black in Fig. 4).

Step 3) Adding edges between vertexes of CS_c and $\cup_{s \in CM(c)}(DM_s \cup CN_s)$. For each $s \in CM(c)$, the packet originated from s should be scheduled to be received by c before to be sent by c. Thus, edge $(CS_c[\text{round}(s)], DM_s[1])$ is added for each $s \in CM(c)$, which is marked in blue in Fig. 4.

$SEQ(Q)$ is the union of all $SEQ(SD(c))$ in which $c \in CH$.

Before we introduce the schedule stage, an important definition is given as follows.

Definition 3 (Candidate Time): Given a schedule item sd = (sender, source, time, receiver, type) that sender, source, receiver, and type have been determined, a set of schedule items H that for each sd' $\in H$, all fields of sd' have been determined. The candidate time of sd regarding H is defined as the minimum slot i such that if sd executes in ith slot, sd is free of collision with all schedule items in H.

2) Schedule Stage: In this stage, times of all schedule items are determined according to $SEQ(Q)$ iteratively. A schedule item set denoted by RE records schedule items whose times can be determined and another set denoted by I records schedule items whose times have been determined. In each

Algorithm 4 Schedule Stage

```
Require: $SEQ(Q) = (V_q, E_q)$;
Ensure: The schedule-item set $I$ indicating the many-to-many schedule;
1: $RE \leftarrow \cup_{A \in V_q}\{A|In\_degree(A) = 0\}$;
2: while $|RE| \neq 0$ do
3:     Compute the candidate time for each $sd \in RE$;
4:     $sd' \leftarrow argmin_{sd \in RE}(the\ candidate\ time\ of\ sd)$;
5:     All fields of $sd'$ is determined; $I \leftarrow I \cup \{sd'\}$; $RE \leftarrow RE - \{sd'\}$;
       Remove $sd'$ from $A'$ that $sd' \in A'$; Remove $A'$ from $SEQ(Q)$ if $|A'| = 0$;
       Update $F2R$s and $F2S$es;
6:     $RE \leftarrow RE \cup (\cup_{A \in V_q}\{A|In\_degree(A) = 0\})$;
7: end while
8: return $I$;
```

iteration, if the in-degree of a vertex A becomes zero, schedule items in A are added into RE. For each schedule item sd in RE, compute its *candidate time* regarding I. Then, the schedule item with the minimum *candidate time* in RE is removed from RE and its time is determined as its *candidate time*. If times of all schedule items in A are determined, A is deleted from $SEQ(Q)$. This stage is shown in Algorithm 4.

Next, we discuss how to compute the *candidate time* for a schedule item sd = (sender, source, time, receiver, type) regarding I. Assume that $sd \in A$, where A is a vertex of $SEQ(Q)$. As in Definition 3, sd must be free of time collision and energy collision with all schedule items in I. To avoid two kinds of collisions, two vectors, namely, $F2S$ and $F2R$ are maintained for each node $v \in V$.

1) *Forbidden-to-Send (F2S):* $F2S_v[i] = 1$ means that v can not be scheduled to send data at ith timeslot. Otherwise, it equals to 0.
2) *Forbidden-to-Receive (F2R):* $F2R_v$ is a mapping between timeslots and binaries $[v_1, v_2]$. $F2R_v[i] = [v_1, v_2]$ indicates that v can only be scheduled to receive the packet originated from v_2 sent by v_1 at timeslot i and $v_1 \in NB(v)$. If $F2R_v[i] = [-1, -1]$, v cannot be scheduled to receive at ith slot.

By exploiting $F2R$ and $F2S$ above, it is very easy to compute the *candidate time* for sd. Let basetime(A) denote the maximum time of all schedule items in A's precursors in $SEQ(Q)$. According to the definition of $SEQ(H)$, if $sd \in A$, time must be greater than basetime(A). Besides, nodes must have harvested enough energy before their operations. Thus, the *candidate time* of sd is computed as the minimum integer i satisfying the following conditions.

1) $i >$ basetime(A).
2) $F2S_{\text{sender}}[i] = 0$ and $b_{\text{sender}}[i] \geq e_s$.
3) ($F2R_u[i] = \emptyset$ *or* $F2R_u[i]$ = [sender, source]) and $b_u[i] \geq e_r$ both satisfy for each $u \in$ receiver.

In order to avoid two types of collisions, after time of a schedule item sd = (sender, source, time, receiver, type) is determined, $F2R$ and $F2S$ are both updated. First, time collision is avoided as follows.

1) For each node $v \in$ receiver, one-hop neighbors of v cannot send packet at timeslot time because v is scheduled to receive. Thus, set $F2S_u$[time] = 1 for every $u \in NB(v)$/sender. Besides, v cannot be scheduled to receive other packets. Therefore, $F2R_v$[time] is set to be $[-1, -1]$.
2) For each node $v \in NB$(sender)/receiver, if $F2R_v$[time] $= \emptyset$, in timeslot time, v can only be scheduled to receive the packet originated from source sent by set sender. Hence, $F2R_v$[time] = [sender, source]. Else, v is in the transmission radius of a node $u \neq$ sender that u has been scheduled to send in timeslot time. In this condition, just set $F2R_v$[time] $= [-1, -1]$.

Second, we discuss how to cope with energy collision. Consider a node v and we use $SDS(v)$ and $RDS(v)$ to denote its already-determined slots for sending and receiving, respectively. Assuming we try to schedule v to send a packet at timeslot i. If $i > Max\{SDS(v) \cup RDS(v)\}$, we only need to consider whether $b_v[i] \geq e_s$. Otherwise, v's residual energy at the beginning of already-determined working timeslots bigger than i is likely to be reduced, which may cause energy collision, making some schedule items whose times have been determined not executable. The reduced energy can be calculated using the following theorem proposed in [19].

Theorem 3: Let $r^i_m(v, e)$ denote the reduction of $b_v[m]$ if v is scheduled to consume e units of energy at timeslot i ($m > i$ and $i \notin SDS(v) \cup RDS(v)$). $r^i_m(v, e)$ is computed as follows.

1) $b_v[j] < b_{\max}$ for all $i < j < m$. In this case, $r^i_m(v, e) = e$.
2) There exists j such that $b_v[j] = b_{\max}$ and $i < j < m$. In this case, some harvested energy is wasted due to the capacity of battery. In this case, Let $S^i_m = \{t|b_v[t] \geq b_{\max} - eh(v) \& t \notin SDS(v) \cup RDS(v) \& i \leq t < m\}$, then $r^i_m(v, e) = \text{Max}\{e - \sum_{t \in S^i_m}(eh(v) + b_v[t] - b_{\max}), 0\}$.

According to Theorem 3, after time of a schedule item sd is determined, for nodes in {sender} $\cup$ receiver, some slots smaller than time should be forbidden to send or receive packets.

1) In terms of sender, if there exists $i <$ time such that $b_{\text{sender}}[\text{time}] - e_s - r^i_{\text{time}}(\text{sender}, e_r) < 0$, set $F2R_{\text{sender}}[i] = [-1, -1]$. If there exists $i <$ time such that $b_{\text{sender}}[\text{time}] - e_s - r^i_{\text{time}}(\text{sender}, e_s) < 0$, set $F2S_{\text{sender}}[i] = 1$.
2) In terms of nodes in receiver, for each $u \in$ receiver, if there exists i<time such that $b_u[\text{time}] - e_r - r^i_{\text{time}}(u, e_r) < 0$, set $F2R_u[i] = [-1, -1]$. Set $F2S_u[i] = 1$ if there exists $i <$ time such that $b_u[\text{time}] - e_r - r^i_{\text{time}}(v, e_s) < 0$.

So far, we have discussed how to compute *candidate time* for a schedule item sd. Here, we introduce a trick to reduce the energy consumption of transmission indicated by many-to-many schedule. For a schedule item sd = (sender, source, time, receiver, type) of data distribution phase (i.e., type = 2 or 3) and $sd \in A$, there might be node $v \notin$ receiver such that there exists another schedule item sd' = (sender', source', time', receiver', type') $\in A$, in which source' = source and $v \in$ receiver'. In this condition, v should be scheduled to receive the packet originated from source. If $v \in NB$(sender), v can also be scheduled to receive the packet of source from sender. Therefore, for a schedule item $sd \in RE$, if type = 2 or 3, before computing its *candidate time*, its receiver is set as $((\cup_{sd' \in A}\text{receiver}_{sd'}) \cap NB(\text{sender})) \cup \text{receiver}$, where $sd' \neq sd$.

V. Theoretical Analysis

In this section, we analyze the time complexity, correctness and the bound of the average many-to-many transmission latency for EBSA.

Lemma 1: The time complexity of the clustering stage, collection stage, and distribution stage is $O(|CH||V|^2+|S|\overline{\Gamma}(|V|+|E|))$, in which $\overline{d}$ is the average degree of the network and $\overline{\Gamma} = Avg\{|\Gamma(s)||s \in S\}$.

Proof: First, according to [28], the clustering stage takes $O(|V|)$-time. Second, we discuss collection stage (Algorithm 1). In line 2, BFS is used to compute $h(c, v)$ for each $v \in V$ and each $c \in CH$, which takes $O(|CH|(|V|+|E|))$-time. The time consumption of line 4 is $O(|V| + |E|)$. The update process in line 5 finishes in $O(|V|)$-time. There are totally $|S|$ routes to be computed. Therefore, the time consumption of lines 4 and 5 is $O(|S|(|V| + |E|))$. The average hop count between each source and cluster head is $R/2$ and each schedule item is visited once in line 8, so that the time consumed by line 8 is $O(|S|R/2) = O(|S||V|)$. In summary, the time complexity of the collection stage is $O((k + |S|)(|V| + |E|)) = O(|S|(|V| + |E|))$. Third, we focus on the distribution stage (Algorithm 2). The time complexity of line 2 is $O(|V|^2)$ and line 2 is repeated $|CH|$ times, resulting in a time complexity of $O(|CH||V|^2)$. Thus, similar to collection stage, distribution stage consumes $O(|CH||V|^2 + |S|\overline{\Gamma}(|V| + |E|))$-time. Summarizing the time complexities of three stages ends the proof. ■

Lemma 2: The time complexity of sequence stage and schedule stage is $O((|S|R\overline{\Gamma})(|S|R\overline{\Gamma} + \overline{d}^2) + |S|R^3)$.

Proof: As for sequence stage (Algorithm 3), each schedule item is visited once in computation for vertexes of $SEQ(Q)$ and there are $O(|S|R\overline{\Gamma})$ schedule items. Besides, the number of edges in $SEQ(Q)$ is $O(|S|R^3)$. Thus, the time complexity of Algorithm 3 is $O(|S|R^3 + |S|R\overline{\Gamma})$. As for schedule stage (Algorithm 4), the time consumption of line 1 is $O(|S|R)$. The number of all schedule items is within $O(|S|R\overline{\Gamma})$ such that lines 3–6 are repeated $O(|S|R\overline{\Gamma})$ times. Lines 3 and 4 both consume $O(|S|R\overline{\Gamma})$-time. In line 5, the time complexity of updating $F2S$es and $F2R$s is $O(\overline{d}^2)$ and $O(\overline{d})$ respectively. Therefore, lines 2–7 consume $O((|S|R\overline{\Gamma})(|S|R\overline{\Gamma} + \overline{d}^2))$-time. ■

Theorem 4: The time complexity of EBSA is $O(|CH||V|^2 + |S|^2R^2\overline{\Gamma}^2 + |S|(\overline{\Gamma}|V|\overline{d}^2 + R^3))$, where $\overline{d}$ is the average degree of the network.

Proof: By Lemmas 1 and 2, the overall time complexity of EBSA is $O(|CH||V|^2 + |S|^2R^2\overline{\Gamma}^2 + |S|(\overline{\Gamma}(|V| + |E| + R\overline{d}^2) + R^3))$. Also, $|E| = O(|V|\overline{d})$ and $R = O(|V|)$, $O(|V| + |E| + R\overline{d}^2) = O(|V|\overline{d}^2)$. This finishes the proof. ■

Lemma 3: If times of all schedule items are determined according to $SEQ(Q)$, the scheduling result satisfies conditions 2 and 3 defined in the minimum-latency many-to-many transmission scheduling problem in BF-WSNs.

Proof: For each $s \in S$, since a multicast tree $DT(s)$ is constructed spanning all nodes in $\Gamma(s)$ and DM_s and CN_s have been derived according to $DT(s)$, for each $t \in \Gamma(s)$, if $t \neq s$, there exists $sd \in DM_s \cup CN_s$ such that $t \in$ receiver and source $= s$. Therefore, condition 2 holds.

Now, we prove that condition 3 also holds. Consider a schedule item $sd =$ (*sender, source, time, receiver, type*) and source $= s \in CM(c)$. There are two cases.

1) If type $= 1$, assume $sd \in CS_c[i]$. If sender $= s$, Condition 3 holds; otherwise, there exists a schedule item $sd' =$ (sender', source', time', receiver', type') such that $sd \in$ receiver', source $=$ source', type' $= 1$ and sender' is the last hop node of sender on $LC(s, c)$. Since $h(c,$ sender'$) = h(c,$ sender$) + 1$, $sd' \in CS_c[i-1]$; In $SEQ(Q)$, there exists an edge between $CS_c[i-1]$ and $CS_c[i]$. Thus, we have time' $<$ time.
2) Otherwise, assume that type $= 2$. Then sender is a dominator. If $sd \in DM_s[1]$, sender $= c$, there exists $sd' =$ (sender', source', time', receiver', type') $\in CS_c[\text{round}(s)]$ such that $c \in$ receiver' and source' $= s$. Since edge $(CS_c(\text{round}(s)), DM_s[1])$ exists in $SEQ(Q)$, time' $<$ time. Otherwise, let $sd \in DM_s[i]$ and $i > 1$. There exists $sd' =$ (sender', source', time', receiver', type') such that $sd \in$ receiver', source $=$ source' and sender' is the parent node of sender in $DT(s)$. According to the method of constructing CDS starting from c, sender' must be a connector and $h(c,$ sender'$) = h(c,$ sender$) - 1$. Thus, $sd' \in CN_s[i-1]$ and there exists an edge from $CN_s[i-1]$ to $DM_s[i]$ in $SEQ(Q)$, so that condition 3 holds when type $= 2$. The case when type $= 3$ can be verified similarly. ■

Lemma 4: The scheduling result satisfies conditions 1 and 5 defined in the minimum-latency many-to-many transmission scheduling problem in BF-WSNs.

Proof: First, according to the method of computing *candidate time* for a schedule item sd, *candidate time* must satisfy condition 1. Thus, condition 1 holds.

Second, we prove that condition 5 holds by contradiction. Assume that two schedule items in I, namely $sd_1 = (\text{sender}_1, \text{source}_1, \text{time}_1, \text{receiver}_1, \text{type}_1)$ and $sd_2 = (\text{sender}_2, \text{source}_2, \text{time}_2, \text{receiver}_2, \text{type}_2)$. First, we consider time collision. Without loss of generality, assume that time collision occurs at $v \in \text{receiver}_2$. Thus, we have $v \in NB(\text{sender}_1)$ and $\text{time}_1 = \text{time}_2$. If time_1 is determined before time_2, $F2R_v[\text{time}_1] = (\text{sender}_1, \text{source}_1)$ or $[-1, -1]$. According to the method of computation for *candidate time*, we have $\text{sender}_1 = \text{sender}_2$, $\text{source}_1 = \text{source}_2$ and $\text{time}_1 = \text{time}_2$, which is impossible. Otherwise, if time_2 is determined before time_1, since $\text{time}_1 = \text{time}_2$, $F2S_{sender_1}[\text{time}_1]$ is set to be 1 just after time_2 is determined. sender_1 is forbidden to send at slot time_1, which makes a contradiction. Thus, time collision will not happen between sd_1 and sd_2. Then, we consider energy collision. Without loss of generality, assume that there exists $v \in V$ such that v is scheduled to receive and send at slot t_1 and t_2, respectively, in which $b_v[t_2] \geq e_s$ and sending at t_2 causes $b_v[t_1] < e_r (t_1 > t_2)$. For node v, after being scheduled to receive a packet at t_1, $F2S_v[t_2]$ is set to be 1 and v cannot be scheduled to send at timeslot t_2, which leads to a contradiction. Cases of other kinds of energy collisions can be verified similarly. Hence, energy collision will not occur. ■

Theorem 5: Schedule item set I generated by EBSA is a proper many-to-many schedule.

Proof: According to the definition of the schedule item and Lemmas 3 and 4, conditions 1–5 in the definition of the problem hold. Thus, I is a correct many-to-many schedule. ■

Lemma 5: For each cluster head $c \in CH$, to ensure that schedule items of data distribution rooted at c are pairwise free of time collision, at most $26(|CM(c)|+R(c)-1)$ different timeslots needs to be assigned to them.

Proof: Consider a cluster head $c \in CH$. In the phase of adding edges to $SEQ(SD(c))$, step 2 indicates that for $CNs_i[j](1 \le i \le |CM(c)|$ and $1 \le j \le R(c))$, times of schedule items in $DMs_i[j+1] \cup DMs_{i-p}[j+p+1] \cup DMs_{i+q}[j-q+1] (p \le i-1$ and $p \le R(c)-j-1, q \le n-i$ and $q \le j)$ must be determined after those in $CNs_i[j]$. Also, times of schedule items in $CNs_i[j]$ are determined after those in $DMs_i[j]$. We define the schedule item set $DMs_i[j+1] \cup DMs_{i-p}[j+p+1] \cup DMs_{i+q}[j-q+1] \cup CNs_i[j+1] \cup CNs_{i-p}[j+p+1] \cup CNs_{i+q}[j-q+1] (p \le i-1$ and $p \le R(c)-j-1$, $q \le n-i$ and $q \le j)$ as a *frame*. For example, eight *frame*s are marked in Fig. 4. It is not difficult to see, there are at most $(|CM(c)|+R(c)-1)$ *frame*s and times of schedule items in the ith *frame* are determined after those in $(i-1)$th frame. According to [10], to ensure that schedule items in one *frame* are pairwise free of time collision, at most 26 different slots are needed to be assigned to them. Therefore, at most $26(|CM(c)|+R(c)-1)$ slots are needed for all *frame*s. ■

Theorem 6: The expected latency of I generated by EBSA is at most $26|S|R+3|S|+2\sqrt{|S|}R+|S|(2e_s+2e_r)/\overline{eh}$, in which $\overline{eh} = Avg\{eh(v)|v \in V\}$.

Proof: The latency of I is caused by time-collision and energy-collision avoidance in the scheduling algorithm. First of all, we discuss latency caused by time collision. Consider latency for data collection in a cluster headed at $c \in CH$. The average size of $CM(c)$ is $|S|/|CH|$ and its average radius is $R/\sqrt{|CH|}$. Therefore, the maximum hop count from a source in $CM(c)$ to c is $2R/\sqrt{|CH|}$ on average. According to (3), $|CS_c| \le 3|CM(c)| + R(c)$. Since schedule items in $CS_c[i]$ are pairwise free of time collision, at most $3|S|/|CH|+2R/\sqrt{|CH|}$ different slots are needed on average to avoid time collision between schedule items in CS_c. There is time collision between data collection of different clusters. Thus, the average latency of data collection is at most $3|S| + 2R\sqrt{|CH|} = y_1$. For data distribution of the cluster, according to Lemma 5, the number of different slots needed for time collision avoidance is at most $26(|S|/|CH| + R(c) - 1) \le 26(|S|/|CH| + R - 1)$ on average. Since target nodes may not in the cluster, there is time collision between data distribution phases of different clusters. Thus, the average latency caused by time collision avoidance for data distribution of the whole network is at most $26|CH|(R-1) + 26|S| = y_2$. To sum up, the average delay caused by time collision avoidance is at most $x = y_1 + y_2$. Since $|CH| \le |S|, x \le 26|S|R + 3|S| + 2\sqrt{|S|}R$, which is the upper bound of the average delay by time collision avoidance.

Next, we consider the latency caused by energy collision avoidance, which is caused by recharge time. First, we discuss the latency of data collection. Consider a node v and v belongs to the cluster centered at $c \in CH$. $b_v[1] = 0$. As the average size of $CM(c)$ is $|S|/|CH|, v$ joins at most $|S|/|CH|$ data collection routes on average. In this case, v needs to relay $|S|/|CH|$-packets, resulting in an average recharge time of $y_3 = |S|(e_s+e_r)/(|CH|\overline{eh})$, where $\overline{eh}$ is the average recharge rate of nodes in the network. Then, in terms of data distribution, in the worst case, v relays all data packets originated from sources in S. In this condition, the average recharge time of v in the data distribution phase is at most $y_4 = |S|(e_s + e_r)/\overline{eh}$. In summary, the average latency caused by energy-collision-avoidance is at most $y_3 + y_4 \le |S|(2e_s + 2e_r)/\overline{eh}$. Adding y_{1-4} together finishes the proof. ■

VI. Simulation Results

In this section, the performance of EBSA is evaluated through extensive simulations under different energy conditions and distributions of nodes.

A. Simulation Setups

So far, the problem researched in [21] is the most similar to our problem. To compare with the state of the art, experimental parameters are set similar to [21] as follows. First, we randomly deploy 200 nodes in a 200 m $\times$ 200 m area, the transmission range of each node is 27m. The average number of one-hop neighbors is 10. Second, we set $e_s = 100$ and $e_r = 80$. The average recharge rate and initial energy of each node are set to be 10 and 0 in general cases, respectively. The number of timeslots spent on harvesting the energy for sending and receiving a packet varies from 1 to 250.[2] Third, $|CM(c)|$ is 4 on average for each $c \in CH$. 10% of nodes in V are picked as sources and each source selects 6% of other nodes randomly as its targets. In terms of baseline algorithms, since no existing works focus on the minimum-latency many-to-many transmission scheduling problem in BF-WSNs, the following algorithms related to many-to-many transmission scheduling are modified to be suitable for our problem and compared with EBSA.

1) MUSTER [4] focused on many-to-many transmission routing for BP-WSNs. To increase network lifetime, MUSTER minimizes the number of nodes involved in routing. Since transmission scheduling has not been discussed in MUSTER, after routing trees are formed, a proper many-to-many schedule is generated similar to the method in Section IV-B.
2) Huang *et al.* [10] proposed scheduling algorithm for gossiping. First, a central node is selected to gather all data packets, then broadcast them to other nodes. We modify the algorithm to make it aware of energy collision.
3) Recently, EWTSBS [21] has been proposed to deal with minimum-latency broadcast scheduling problem in BF-WSNs. This algorithm can avoid both kinds of collisions. Thus, we extend it to the scenario of multisources by constructing the broadcast tree rooted at s using EWTSBS for each source $s \in S$.

The latency of many-to-many schedule (many-to-many latency for short) and energy usage ratio is evaluated. The energy usage ratio is defined as the ratio between energy consumed and harvested of all nodes in the network. Data for each experiment are the average of 200 executions.

[2]In [21], this number varies from 1 to 1000 and there is only one packet to be transmitted. Since in our scenario, nodes may transmit multiple packets, this setting is also fair and reasonable.

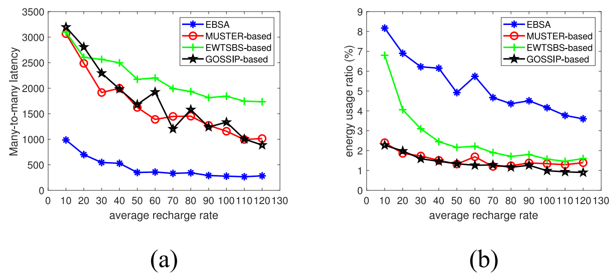


Fig. 5. Average recharge rate with uneven energy distribution. (a) Many-to-many latency. (b) Energy usage ratio.

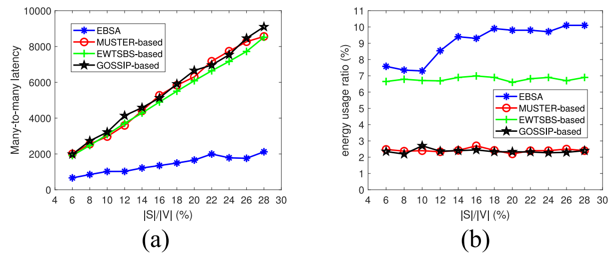


Fig. 6. Source ratio with uneven distributed ambient energy. (a) Many-to-many latency. (b) Energy usage ratio.

B. Uneven Distributed Ambient Energy

In general cases, the recharge rate varies dramatically among different nodes. Under this condition, we discuss the latency and energy usage ratio as functions of the average recharge rate (Fig. 5) and the source ratio (i.e., $|S|/|V|$). (Fig. 6). As Fig. 5 shows, the latency and energy utilization both decrease as ambient energy grows for four algorithms and EBSA outperforms all baseline protocols. Fig. 5(a) indicates that EBSA reduces 70%–80% of many-to-many delay compared with baseline protocols. This is mainly because EBSA constructs transmission structures in a bottleneck-aware manner and many-to-many schedule is generated energy adaptively. Although designed for BF-WSNs, EWTSBS is not bottleneck aware, it does not consider recharge rate when constructing broadcast trees. Thus, the latency of the EWTSBS-based algorithm is also high as the MUSTER-based and GOSSIP-based algorithm. As Fig. 5(b) shows, the energy usage ratio of EBSA is also higher than all baseline algorithms. This is because our method is bottleneck-aware. Compared with baseline protocols, EBSA distributes energy consumption among all nodes in the network and more harvested energy can be exploited and the recharge time is reduced. This is also a direct cause of the trend in Fig. 5(a). Also, we can see that EWTSBS-based outperforms other baselines on energy usage ratio. This is because all nodes in the network are targets for each source in EWTSBS. In this condition, much harvested energy is wasted for useless transmissions, which causes a huge recharge time. MUSTER and GOSSIP only construct a single tree as the transmission structure. Thus, there are more hotspots for energy consumption, which also increases the recharge time. In this case, the harvested energy of those nodes not included in the tree is not exploited, which is also the cause of their low energy utilization.

Fig. 6 evaluates many-to-many latency and energy usage ratio as functions of the source ratio. As we can see, both many-to-many latency and energy usage ratio increase with the percentage of sources since the workload of the network grows. Under this circumstance, EBSA still outperforms baselines, in which many-to-many latency of EBSA is only 25%–30% of that for baselines and its energy usage ratio is the highest. This can be explained in the same reason for Fig. 5.

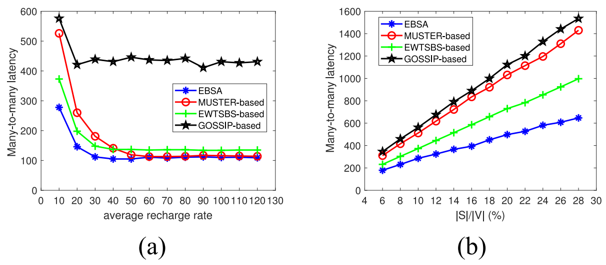


Fig. 7. Uniform distributed ambient energy. (a) Average recharge rate. (b) Source ratio.

C. Uniform Distributed Ambient Energy

In Section I, we mention that the overall latency is mainly affected by the recharge time of bottlenecks. Thus, in this part, we discuss the condition where each node has the same recharge rate so that there is less difference in recharge time between bottlenecks and other nodes in the network. Fig. 7 evaluates many-to-many latency as functions of average recharge rate and the percentage of sources. For all algorithms, the many-to-many latency shows in Fig. 7(a) is much less than that in Fig. 5(a). Compared with baselines, the latency of EBSA is 60% of that for MUSTER-based and 80% for EWTSBS-based when the average recharge rate is low (less than 50), which is not as good as in Fig. 5(a). This is because, in this condition, ambient energy is distributed uniformly. Although not taking energy harvesting rate and bottlenecks into account, the recharge times of bottlenecks are almost the same as other nodes in MUSTER based and EWTSBS based. However, EBSA can exploit the harvested energy of more nodes. Thus, it still outperforms two baselines. When the average recharge rate is high (over 60), the latency of EBSA is almost the same as two baselines. In this condition, the latency caused by time collision avoidance, rather than recharge time is the main cause for many-to-many latency. EBSA does not focus much on time collision avoidance so that it has almost no advantage. The latency of the gossip-based algorithm is the highest. This is because, in the GOSSIP-based algorithm, each packet has to be relayed by the central node, resulting in a huge delay near it.

As for Fig. 7(b), since there is not abundant ambient energy, the performance of EBSA is still better than baselines. As the percentage of sources increases, the advantage of EBSA over MUSTER based and EWTSBS based is more obvious (the percentage of EBSA's latency on two algorithms decreases from 55% to 43%). Compared with the other two baselines, EWTSBS based has lower many-to-many latency. This is because there are more nodes involved in transmission in EWTSBS based and more energy can be exploited.

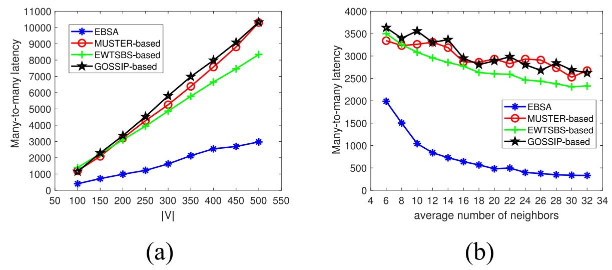


Fig. 8. (a) Network scale and (b) average degree.

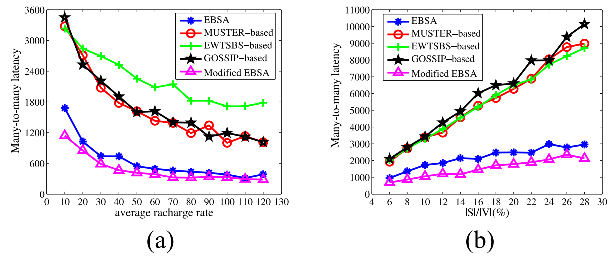


Fig. 9. Sources distribute in a particular area. (a) Average recharge rate. (b) Source ratio.

D. Network Size and Average Degree of Nodes

In this part, we evaluate many-to-many latency in terms of network size [Fig. 8(a)] and the average number of one-hop neighbors [Fig. 8(b)].

From Fig. 8(a), we can see that EBSA outperforms baselines in both small and large-scale BF-WSNs. Besides, the many-to-many latency for all algorithms grows linearly with network size. This is because hop counts from sources to their targets increases with the network scale. Also, in general cases, 10% of all nodes are selected as sources and each of them further picks 6% of all other nodes as corresponding targets. Thus, there are total $0.006|V|^2 = \theta(|V|^2)$ routes between all sources and their targets. Thus, each node is involved in $\theta(|V|)$ routes on average, such that the recharge time increases linearly with $|V|$.

As shown in Fig. 8(b), the latencies of all algorithms decrease as node distribution becomes denser. This is because when the network scale is fixed, the average hop count from sources to targets decreases, which reduces the possibility that a bottleneck node has a very long recharge time. Also, the advantage of our protocol over baselines is greater as the average degree increases (54% and 11% on many-to-many latency of EWTSBS based when $\overline{NB} = 6$ and 32). The reason is as follows. For any node v, the number of nodes with high energy harvesting rates and low workloads in $NB(v)$ increases with the growth of the average number of neighbors. Since our algorithm is bottleneck aware and energy adaptive, when constructing transmission routes, nodes with high recharge rates and few already-assigned sending and receiving slots are more likely to be selected as v's parents or children compared with three baselines.

E. Region Query

In many cases, multiple applications may require data from a particular area. To simulate this scenario, in this part, all sources are selected around a specific point randomly picked in the monitoring area. Fig. 9 shows many-to-many latency in terms of average recharge rate and source percentage. As in Fig. 9, EBSA still outperforms all baselines. However, compared with Figs. 5(a) and 6(a) in which sources are distributed uniformly, latencies of four algorithms almost all increases [Fig. 9(a) versus Fig. 5(a): 44% for EBSA, 0.7% for MUSTER based, 1.6% for EWTSBS based and −3.8% for GOSSIP based; Fig. 9(b) versus Fig. 6(a): 55% for EBSA, 2% for MUSTER-based, 5% for EWTSBS based, and 8% for GOSSIP-based]. Among them, EBSA sees the biggest increase. This is because EBSA distributes the workload in the network by constructing uniformly distributed clusters. When sources are distributed in a particular area, they are likely to choose several specific clusters whose cluster heads are close to them, whereas other cluster heads will have no cluster members (i.e., $|CM(c)| = 0$). In this case, all workloads are assigned to transmission structures of several specific clusters, which reduces the energy usage ratio and increases the recharge time of bottlenecks. Therefore, the performance of EBSA decreases. To solve this problem, when sources are distributed in a certain area, EBSA is modified as follows. After the election of cluster heads, each cluster head chooses $|S|/|CH|$ sources uniformly as its cluster members. Since cluster heads are evenly distributed by the modified EBSA, all workloads are distributed uniformly among all nodes in the network. This improves the energy usage ratio and, therefore, the many-to-many latency decreases. This is shown in Fig. 9. In Fig. 9(a), the modified EBSA reduces the latency by 25% on average compared with EBSA. Compared with EBSA in Fig. 5(a), the modified EBSA only increases the latency by 9%. In Fig. 9(b), the latency of the modified EBSA is 70% of that for EBSA on average, which only increases 8% of latency compared with EBSA in Fig. 6(a).

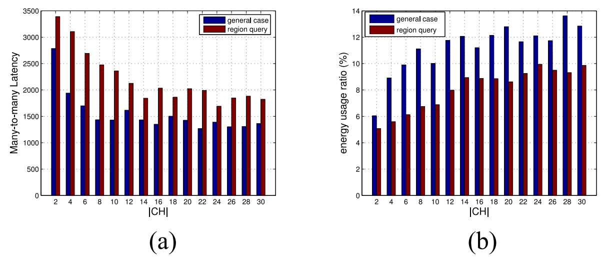


Fig. 10. Impact of $|CH|$ on EBSA's performance. (a) Many-to-many latency. (b) Energy usage ratio.

F. Impact of Cluster Number, $|CH|$

The cluster number $|CH|$ is an important parameter of EBSA. In this part, we briefly discuss its impact on the performance of EBSA in the general case and region query. Notice that the percentage of sources is set to be 20%. Fig. 10 shows many-to-many latency [Fig. 10(a)] and energy usage ratio [Fig. 10(b)] as functions of $|CH|$. As shown in Fig. 10(a), the many-to-many latency decreases as the number of cluster heads increases. This is because more clusters make more nodes involved in the transmission, which improves the energy usage ratio, making EBSA more bottleneck aware. This can also be shown in Fig. 10(b). As $|CH|$ increases, more nodes are participating in transmission and the amount of time collision also increases. However, like energy consumption, time collisions can also be distributed uniformly among all nodes by EBSA. Therefore, the many-to-many latency remains stable after $|CH|$ reaches a value [i.e., around 20 in Fig. 10(a)].

VII. Discussion

In previous sections, the EBSA under protocol interference model is discussed. To extend EBSA to more general cases, in this section, we discuss many-to-many scheduling under other assumptions with average many-to-many latency analyzed.

A. Source With Multiple Packets

In real-world applications, data of each source may not be the same sized but transmitted in packets of the same size. Thus, the number of packets of each source may be more than one. Assume that source s has n_s packets to be transmitted. A technique called *virtual node* is used in this case as follows: For each $s \in S$, if $n_s > 1, n_s - 1$ instances of s are added into S. Then running EBSA as in general. Let $W = \sum_{s\in S} n_s$, according to Theorem 6, the average latency is upper bounded by $26WR + 3W + 2\sqrt{WR} + W(2e_s + 2e_r)/\underline{eh}$.

B. Interference Radius Larger Than Transmission Radius

In general cases of protocol interference model, Let r_f denote the interference radius of each node, then we have $r_f = \mu r_t (\mu \geq 1)$. For each node v, let the set of nodes in the interference range of v be denoted as $INF(v) = \{u | u \in V \& \text{dist}(u, v) \leq r_f\}$. In Sections III and IV, we discuss the case when $\mu = 1$. When $\mu > 1$, time collision can be redefined as follows: Given a schedule-item set I and any two schedule items $sd_1 = (\text{sender}_1, \text{source}_1, \text{time}_1, \text{receiver}_1, \text{type}_1)$ and $sd_2 = (\text{sender}_2, \text{source}_2, \text{time}_2, \text{receiver}_2, \text{type}_2)$ that $sd_1, sd_2 \in I$, sd_1 is free of time collision with sd_2, if and only if one of the two following conditions satisfies: 1) $\text{time}_1 \neq \text{time}_2$ and 2) $v \notin INF(\text{sender}_1)$ for each $v \in \text{receiver}_2$ and $v \notin INF(\text{sender}_2)$ for each $v \in \text{receiver}_1$. In this case, to avoid time collision, after time of a schedule item sd = (sender, source, time, receiver, type) is determined, F2S and F2R are updated as follows.

1) For each $v \in$ receiver and for each $u \in INF(v)$/sender, set $F2S_u[\text{time}] = 1$, set $F2R_v[\text{time}] = [-1, -1]$.
2) For each node $v \in NB(\text{sender})$/receiver, if $F2R_v[\text{time}] = \emptyset$, set $F2R_v[\text{time}] = [\text{sender}, \text{source}]$. Else, set $F2R_v[\text{time}] = [-1, -1]$. For each node $v \in INF(\text{sender})$/receiver, if $v \notin NB(\text{sender})$, set $F2R_v[\text{time}] = [-1, -1]$.

The following theorem indicates the expected many-to-many latency of EBSA in general cases of protocol interference model.

Theorem 7: The expected latency of EBSA when $r_f = \mu r_t (\mu > 1)$ is at most $(2\eta + 1)|S|R + |S|(2e_s + 2e_r)/\underline{eh}$, in which $\eta = \psi(\mu + 1)$ and [6]

$$\psi(x) = \left\lfloor \frac{\pi}{\sqrt{3}} x^2 + \left(\frac{\pi}{2} + 1\right) x \right\rfloor + 1. \tag{4}$$

C. Scheduling Under Physical Interference Model

In this part, we consider scheduling under physical interference model, which is more realistic but more complex than scheduling under a protocol interference model. In this case, a node v_2 can successfully receive the packet sent by v_1 if the SINR value between them exceeds a threshold

$$\text{SINR}(v_1, v_2) = \frac{P \times \text{dist}(v_1, v_2)^{-\alpha}}{N + \sum_{v' \in D/\{v_1\}} P \times \text{dist}(v', v_2)^{-\alpha}} \geq \beta \tag{5}$$

where P denotes the transmission energy, α is the path loss exponent and usually, $2 \leq \alpha \leq 6$, D is the set of nodes sending simultaneously with v_1, β is a threshold that is no less than 1, N is the background noise. Thus, time collision is redefined as follows: For any schedule item sd = (sender, source, time, receiver, type), sd is free of time collision with any other schedule items if and only if at slot time, $SINR(\text{sender}, v) \geq \beta$ for each $v \in$ receiver. We apply the strategy proposed in [6] to avoid time collision and EBSA is modified as follows.

1) Process the input BF-WSN $G = (V, E)$ before clustering stage. First, set $E = \emptyset$. Then for any pair of nodes (v_1, v_2), if $\text{dist}(v_1, v_2) \leq (P/cN\beta)^{1/\alpha} = r$, add an undirected edge (v_1, v_2) into E, where $c > 1$. The network radius in this case is denoted by R'.
2) Compute ρ as follows, where $\delta(x) = \sum_{j=1}^{\infty} 1/j^{x}$[3]:

$$\rho = 1 + \left(\frac{\beta(16\delta(\alpha - 1) + 8\delta(\alpha) - 6)}{1 - c^{-1}}\right)^{(1/\alpha)}. \tag{6}$$

3) Algorithm 4 is modified as follows. For each schedule item sd = (sender, source, time, receiver, type) and each $v \in$ receiver, variable $SINR$(sender, v, time) is maintained according to (5) to record $SINR$ value between sender and v at slot time. $F2R$ and $F2S$ are no longer set for time-collision avoidance. Assume that $sd \in A$ and A is a vertex of $SEQ(Q)$. The *candidate time* of sd regarding I is computed as the minimum integer i satisfying the following conditions.
   a) $i > \text{basetime}(A)$.
   b) $SINR(\text{sender}, v, i) \geq \beta$ for each $v \in$ receiver. For each sd' = (sender', source', time', receiver', type') $\in I$ that time' $= i$ and for each $v' \in$ receiver', $SINR$(sender', v', time') $\geq \beta$.
   c) $F2S_{\text{sender}}[i] = 0$ and $b_{\text{sender}}[i] \geq e_s$; $F2R_u[i] = \emptyset$ and $b_u[i] \geq e_r$ both satisfy for each $u \in$ receiver.
   d) If type $= 2$, for each schedule item sd' in A such that time' has been determined, if time' $= i$, dist(sender, sender') $\geq r$ holds.
   e) If type $= 3$, for each schedule item sd' in A such that time' has been determined, if time' $= i$, $\text{dist}(v, v') \geq r$ holds for each $v \in$ receiver and each $v' \in$ receiver'.

   After time of sd is determined, SINR values are updated as (5).

The following theorem indicates the expected many-to-many latency of EBSA under the physical interference model.

[3]δ values for some xs are given in [6].

Theorem 8: The expected latency of EBSA under the physical interference model is at most $(2\eta+1)|S|R' + |S|(2e_s + 2e_r)/\overline{eh}$, in which $\eta = \psi(\rho)$.

The proofs of Theorems 7 and 8 are omitted due to space limitations.

VIII. Conclusion

Many-to-many transmission is an important data dissemination pattern in WSNs. In this article, we first addressed the challenge for many-to-many transmission scheduling problem in BF-WSNs, then the minimum-latency many-to-many transmission scheduling problem in BF-WSNs is defined and an energy-adaptive and bottleneck-aware algorithm, namely, EBSA is proposed to generate the low-latency many-to-many transmission schedule. To deal with the problem of energy bottlenecks and fully exploit the harvested energy, EBSA constructs bottleneck-aware transmission structures to reduce the recharge time of bottlenecks. Then, EBSA computes many-to-many transmission schedule in an energy-adaptive manner. Both theoretical analysis and simulation verify the high performance of our algorithm, in which EBSA reduces 70%–80% many-to-many latency compared with baselines when ambient energy is distributed unevenly. From the simulation, we can see that being bottleneck aware can greatly reduce transmission latency. Thus, as for future work, we will further investigate the energy-bottleneck problem in BF-WSNs designed for other real-time applications.

References

[1] A. Kofi, A. Nadir, T. Cristiano, A. Hoda, and H. Wendi, "Energy-harvesting wireless sensor networks (EH-WSNs): A review," *ACM Trans. Sensor Netw.*, vol. 14, no. 2, pp. 1–50, 2018.

[2] Y. Yang, L. Su, Y. Gao, and T. F. Abdelzaher, "SolarCode: Utilizing erasure codes for reliable data delivery in solar-powered wireless sensor networks," in *Proc. INFOCOM* 2010, pp. 1–5.

[3] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 757–789, 2nd Quart., 2015.

[4] L. Mottola and G. P. Picco, "MUSTER: Adaptive energy-aware multi-sink routing in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 12, pp. 1694–1709, Dec. 2011.

[5] S. Chen, M. Coolbeth, H. Dinh, Y.-A. Kim, and B. Wang, "Data collection with multiple sinks in wireless sensor networks," in *Proc. WASA*, 2009, pp. 284–294.

[6] P.-J. Wan, L. Wang, and O. Frieder, "Fast group communications in multihop wireless networks subject to physical interference," in *Proc. IEEE MASS*, 2009, pp. 526–533.

[7] K. Hwang, J. S. In, and D. Eom, "Distributed dynamic shared tree for minimum energy data aggregation of multiple mobile sinks in wireless sensor networks," in *Proc. EWSN*, 2006, pp. 132–147.

[8] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Netw.*, vol. 11, nos. 1–2, pp. 161–175, 2005.

[9] B. Vincenzo, K. Peter, and M. Alberto, "Minimizing flow time in the wireless gathering problem," *ACM Trans. Algorithms*, vol. 7, no. 3, pp. 109–120, 2011.

[10] C.-H. Huang, P.-J. Wan, H. Du, and E. K. Park, "Minimum latency gossiping in radio networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 790–800, Jun. 2010.

[11] K. H. Seok, A. Tarek, and K. Wook-Hyun, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *Proc. SenSys*, 2003, pp. 193–204.

[12] W.-B. Pöttner, H. Seidel, J. Brown, U. Roedig, and L. C. Wolf, "Constructing schedules for time-critical data delivery in wireless sensor networks," *ACM Trans. Sensor Netw.*. vol. 10, no. 3, pp. 1–31, 2014.

[13] F. Mager, C. Herrmann, and M. Zimmerling, "One for all, all for one: Toward efficient many-to-many broadcast in dynamic wireless networks," in *Proc. HotWireless*, 2017, pp. 19–23.

[14] K. Yuen, B. Li, and B. Liang, "Distributed data gathering in multi-sink sensor networks with correlated sources," in *Proc. Networking*, 2006, pp. 868–879.

[15] C. Herrmann, F. Mager, and M. Zimmerling, "MIXER: Efficient many-to-all broadcast in dynamic wireless mesh networks," in *Proc. SenSys*, 2018, pp. 145–158.

[16] M. Mohammad and M. C. Chan, "Codecast: Supporting data driven in-network processing for low-power wireless sensor networks," in *Proc. IPSN*, 2018, pp. 72–83.

[17] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Lowpower wireless bus," in *Proc. SenSys*, 2012, pp. 1–14.

[18] O. Landsiedel, F. Ferrari, and M. Zimmerling, "CHAOS: Versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proc. SenSys*, 2013, pp. 1–14.

[19] Q. Chen, H. Gao, Z. Cai, L. Cheng, and J. Li, "Energy-collision aware data aggregation scheduling for energy harvesting sensor networks," in *Proc. INFOCOM*, 2018, pp. 117–125.

[20] K. Chen, H. Gao, Z. Cai, Q. Chen, and J. Li, "Distributed energy-adaptive aggregation scheduling with coverage guarantee for battery-free wireless sensor networks," in *Proc. INFOCOM*, 2019, pp. 1018–1026.

[21] T. Zhu, J. Li, H. Gao, and Y. Li, "Broadcast scheduling in battery-free wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 15, no. 4, p. 49, 2019.

[22] Y. Gu and T. He, "Bounding communication delay in energy harvesting sensor networks," in *Proc. IEEE ICDCS*, 2010, pp. 837–847.

[23] A. Baknina and S. Ulukus, "Optimal and near-optimal online strategies for energy harvesting broadcast channels," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3696–3708, Dec. 2016.

[24] T. Zhu, J. Li, H. Gao, and Y. Li, "Latency-efficient data collection scheduling in battery-free wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 16, no. 3, p. 25, 2020.

[25] R. Liu and Y. Chen, "Robust data collection for energy-harvesting wireless sensor networks," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 107025.

[26] Z. Dong, Y. Gu, J. Chen, S. Tang, T. He, and C. Liu, "Enabling predictable wireless data collection in severe energy harvesting environments," in *Proc. RTSS*, 2016, pp. 157–166.

[27] A. Mehrabi and K. Kim, "Maximizing data collection throughput on a path in energy harvesting sensor networks using a mobile sink," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 690–704, Mar. 2016.

[28] P. Zhou, C. Wang, and Y. Yang, "Self-sustainable sensor networks with multi-source energy harvesting and wireless charging," in *Proc. INFOCOM*, 2019, pp. 1828–1836.

[29] H. Choi, J. Wang, and E. A. Hughes, "Scheduling for information gathering on sensor network," *Wireless Netw.*, vol. 15, no. 1, pp. 127–140, 2009.

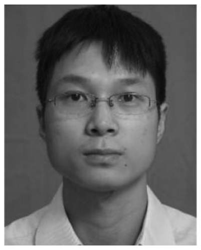
Bingkun Yao received the bachelor's degree in computer science from Harbin Institute of Technology, Harbin, China, in 2018, and the M.S. degree in computer science from Harbin Institute of Technology, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology.

His current research interests include energy harvesting wireless sensor networks and in-network data processing.

Hong Gao received the B.S. degree in computer science from Heilongjiang University, Harbin, China, in 1988, the M.S. degree from Harbin Engineering University, Harbin, in 1991, and the Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, in 2004.

She is a Professor with the School of Computer Science and Technology, Harbin Institute of Technology. Her research interests include graph data management, sensor network, and massive data management.

Prof. Gao is the winner of National Science and Technology Progress Award.



Quan Chen received the B.S., Master, and Ph.D. degrees from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2010, 2012, and 2017, respectively.

He is currently a Lecturer with the School of Computers, Guangdong University of Technology, Guangzhou, China. His research interests include routing and data transmission algorithms in wireless sensor networks and duty-cycle networks.

Jianzhong Li received the bachelor's degree from Heilongjiang University, Harbin, China, in 1975.

He was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA, as a Staff Scientist with the Information Research Group, Lawrence Berkeley National Laboratory, USA, and as a Visiting Professor with the University of Minnesota, USA. He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He has published over 150 papers in refereed journals and conferences. His research interests include data management systems, sensor networks, and data intensive computing.

Prof. Li has been involved in the program committees of major computer science and technology conferences, including SIGMOD, VLDB, ICDE, INFOCOM, ICDCS, and WWW. He has also served on the editorial boards for distinguished journals, including the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.