

Article

# Energy-Efficient Algorithm for Multicasting in Duty-Cycled Sensor Networks

Quan Chen <sup>1</sup>, Siyao Cheng <sup>1</sup>, Hong Gao <sup>1,\*</sup>, Jianzhong Li <sup>1</sup> and Zhipeng Cai <sup>2</sup>

Received: 19 November 2015; Accepted: 7 December 2015; Published: 11 December 2015

Academic Editors: Yunchuan Sun, Antonio Jara and Shengling Wang

<sup>1</sup> School of Computer Science and Technology, Harbin Institute of Technology, 92 West Dazhi Street, Harbin 150001, China; chenquan@hit.edu.cn (Q.C.); csy@hit.edu.cn (S.C.); lijzh@hit.edu.cn (J.L.)

<sup>2</sup> Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA; zcai@gsu.edu

\* Correspondence: honggao@hit.edu.cn; Tel.: +86-451-8641-5827

**Abstract:** Multicasting is a fundamental network service for one-to-many communications in wireless sensor networks. However, when the sensor nodes work in an asynchronous duty-cycled way, the sender may need to transmit the same message several times to one group of its neighboring nodes, which complicates the minimum energy multicasting problem. Thus, in this paper, we study the problem of minimum energy multicasting with adjusted power (the MEMAP problem) in the duty-cycled sensor networks, and we prove it to be NP-hard. To solve such a problem, the concept of an auxiliary graph is proposed to integrate the scheduling problem of the transmitting power and transmitting time slot and the constructing problem of the minimum multicast tree in MEMAP, and a greedy algorithm is proposed to construct such a graph. Based on the proposed auxiliary graph, an approximate scheduling and constructing algorithm with an approximation ratio of  $4\ln K$  is proposed, where  $K$  is the number of destination nodes. Finally, the theoretical analysis and experimental results verify the efficiency of the proposed algorithm in terms of the energy cost and transmission redundancy.

**Keywords:** multicasting; energy optimization; power aware; Steiner tree; duty cycle; wireless sensor networks

## 1. Introduction

In recent years, wireless sensor networks (WSNs) have been used in monitoring and retrieving sensory data from the physical world [1–11], which are usually expected to last over several months or years. Therefore, it is very necessary to design an energy conservation mechanism for WSNs to extend the network lifetime [12–14]. Thus, a schema of the duty-cycle is proposed in WSNs. According to the duty-cycle schema, each node switches between the active and the dormant state periodically; the period of the dormant state is much longer than that of the active state in order to save energy. According to the works in [15–19], the duty-cycled schema has high performance in terms of energy savings.

Multicasting is a fundamental component service for one-to-many communications in wireless sensor networks, such as to support data dissemination [20–22] for distributed data management and remote network configuration (e.g., [23–25]). Therefore, developing an energy-efficient multicast protocol is very meaningful in WSNs. Due to this consideration, the minimum energy multicasting (MEM) problem is proposed, which seeks to disseminate the messages from the source node to all of the destination nodes with minimum energy cost. Nowadays, the MEM problem has attracted extensive attentions from the research community, and it is studied in both nodes always-awake sensor networks [26–32] and duty-cycled sensor networks [33–36]. In the nodes always-awake sensor networks, where each node can deliver the messages to one group of its neighboring nodes by only

one transmission, the MEM problem is proven to be NP-hard, and some approximation algorithms have been proposed [26–32].

However, in the duty-cycled sensor networks, the MEM problem becomes more complicated. Since the nodes can only receive the messages in the active state, the sender may need to transmit the same message several times to one group of its neighboring nodes. Therefore, the methods for the MEM problem in the nodes always-awake sensor networks are not suitable for duty-cycled sensor networks. In such networks, designing the minimum energy multicasting algorithm requires us not only to select appropriate forwarding nodes, but also to schedule the transmitting time slot optimally. Considering this, several methods are proposed by [33–36], which seek to minimize the transmission redundancy and the number of transmissions during multicasting. However, they all assumed the energy cost for all of the transmissions is equivalent and did not consider the case that the transmitting power of the sensor nodes can be adjusted. According to [37], the sensor nodes can transmit at six different power levels, which range from 1 MW to 100 MW. In this case, the senders need not only to choose their transmitting time slot intelligently, but also schedule their transmitting power optimally to construct the multicast tree.

Therefore, in this work, we study the problem of minimum energy multicasting with adjusted power (MEMAP) in duty-cycled sensor networks, and it is proven to be NP-hard. To solve such a problem, an approximate scheduling and constructing algorithm is proposed.

In summary, the contributions of the paper are as follows.

- (1) The MEMAP problem in the duty-cycled sensor networks is proposed to minimize the energy cost during multicasting, and its NP-hardness is proven.
- (2) In order to solve the MEMAP problem, an auxiliary graph is defined, and a greedy algorithm is given to construct such a graph.
- (3) Based on the proposed auxiliary graph, an approximation algorithm is proposed for the MEMAP problem, and its approximation ratio is proven to be  $4\ln K$ , where  $K$  is the number of destination nodes.
- (4) Extensive simulations are carried out, which verify that the proposed algorithm has high performance in terms of energy cost.

The rest of this work is organized as follows. Section 2 surveys the related work. In Section 3, we present the preliminaries, including the network model and the problem definition. Section 4 proposes the algorithm design in detail. Simulation results are discussed in Sections 5. Section 6 concludes the paper.

## 2. Related Works

In recent years, there has been a tremendous amount of studies for the MEM problem in both nodes always-awake networks and duty-cycled sensor networks. In the nodes always-awake networks, the main works that studied the MEM problem are [26–32]. In [26], the author firstly studied the problem of constructing the minimum power broadcast/multicast tree in the wireless sensor network where each node can adjust its transmission power continuously, and three greedy heuristic algorithms were proposed. On the basis of [26], Wan *et al.* proved that the method in [26] has a linear approximation ratio, and then, they proposed several approximation algorithms with a constant approximation ratio for the min-power multicast routing problem [27]. In [28], the minimum energy broadcasting problem was proven to be NP-complete, and there is no polynomial algorithm with an approximation ratio better than  $\Omega(\log n)$  until  $NP = P$ . In [29], the author proposed a centralized approximation algorithm with at most  $8\ln K$ -times the optimum when the wireless nodes can adjust their transmitting power discretely. The minimum energy all-to-all multicasting problem was studied in [30], which tries to build a shared multicast tree to reduce the energy consumption. Recently, Qiu *et al.* [31] studied the minimum energy cooperative broadcasting problem where receivers can combat transmission errors by combing the received packets from different senders.

Baghaie *et al.* [32] try to formulate the optimal tradeoff between the energy cost and broadcast latency in the cooperative communication scheme. However, these methods were not suitable for the duty-cycled sensor networks.

In duty-cycled sensor networks, the main works that studied the energy-efficient multicasting problem are [18,19,33–36]. In [18,19], Feng and Guo *et al.* proposed an opportunistic forwarding scheme for reliable flooding and broadcasting by considering unreliable links in duty-cycled sensor networks. Lai *et al.* [33] proposed a broadcasting protocol to achieve a better tradeoff between the broadcast latency and transmission redundancy in the duty-cycled sensor networks. In [34], the authors studied the minimum transmission problem for broadcasting, and they proposed a centralized algorithm with an approximation ratio of  $3\ln(\Delta + 1)$ , where  $\Delta$  denotes the maximum node degree in the network. However, they adopted a restricted duty-cycling model where there is only one active time slot existing in the working schedule of each node. In [35], Su *et al.* proposed two optimal algorithms for the minimum energy multicasting problem and the delay-bounded minimum energy multicasting problem when the number of destinations (e.g.,  $K$ ) is small in the multicast session. Han *et al.* [36] removed the limitation of the size of destinations and proposed a polynomial time complexity approximation algorithm with an approximation ratio of  $6\rho H(\Delta + 1) + 2\rho$ , where  $H(\cdot)$  denotes the harmonic number and  $\rho$  is the approximation ratio of the minimum Steiner tree problem [38,39]. However, they assumed the energy cost to be equivalent for all of the transmissions. Therefore, their problem is actually to minimize the number of transmissions in the multicast session, and their method is not suitable for the case when the transmitting power can be adjusted.

Due to the above limitations, we consider the problem of minimum energy multicasting with adjusted power in duty-cycled sensor networks, in which we need to not only schedule the transmitting power optimally to construct the multicast tree, but also choose the transmitting time slot for each node intelligently.

### 3. Models and Problem Definition

Before presenting our algorithms in detail, we depict the models used in this work and the formal definition of the MEMAP problem.

#### 3.1. Network Model

We assumed a multihop duty-cycled sensor network  $G = (V, E)$ , where  $V$  is the set of sensor nodes and  $E$  denotes the set of edges. There exists an edge between two sensor nodes if they are within each other's transmission range. So as to conserve energy, each sensor node works between two states, e.g., the dormant state and the active state. In the dormant state, the sensor node turns off all of its functional models (*i.e.*, sensing the environment, sending and receiving packets) and just waits to be scheduled. The node switches between the active state and the dormant state periodically.

Let  $\mathcal{T}$  denote one working cycle for each node, which is divided into multiple time slots with equal lengths. The length of each time slot (e.g.,  $\tau$ ) can be determined according to [40] to guarantee that the data packet can be delivered to the neighbor successfully in one time slot. Since each sensor node has two states, all time slots in one working cycle, *i.e.*,  $\{1, 2, \dots, \mathcal{T}/\tau\}$ , can be separated into two disjointed subsets for any  $u \in V$ . We let  $\mathcal{W}(u) = \{t_u^1, t_u^2, \dots, t_u^k\}$  denote the working schedule of node  $u$ , which contains all active time slots of  $u$ . Then,  $\{1, 2, \dots, \mathcal{T}/\tau\} - \mathcal{W}(u)$  contains all dormant time slots of  $u$ . As the same setting in [35], we assume  $t_u^1, t_u^2, \dots, t_u^k$  are consecutive in this paper, and they can be set up according to the requirement of coverage or connectivity [41]. Nodes can switch to the active state according to their working schedule or when they have packets to be sent, but they can only receive the packets when they are in the active state. To deliver the data, the sender can switch to the active state when the receiver wakes up. Additionally, node  $u$ 's duty cycle can be calculated as  $|\mathcal{W}(u)| \times \tau/\mathcal{T}$ .

In addition, we assumed that each sensor node is equipped with an omnidirectional antenna, and the transmission power of each sensor node can be adjustable. There are  $\mathcal{L}$  power levels at

each node, *i.e.*,  $P = \{p_1, p_2, \dots, p_{\mathcal{L}}\}$ , where  $p_i (1 \leq i \leq \mathcal{L})$  denotes the transmitting power of the  $i$ -th power level. Without loss of generality,  $P$  is sorted ascendingly, that is  $p_i \leq p_j$  if  $1 \leq i \leq j \leq \mathcal{L}$ . For each neighboring node  $v$  of  $u$ , there exists a minimum transmitting power required to guarantee  $v$  is under the transmission range of  $u$ , and we assume that the transmission power is symmetric. Let the Euclidean distance between  $u$  and  $v$  be  $d(u, v)$ . According to the path loss model,  $u$  can communicate with  $v$  with power level  $l$  only if:

$$\frac{p_l / d(u, v)^\alpha}{N(1 + \phi)} \geq \beta \quad (1)$$

where  $\alpha$  is the pass loss exponent,  $\beta$  is the minimum SINR value (signal to interference plus noise ratio) to guarantee successful reception and  $N(1 + \phi)$  is the background noise. In these parameters,  $\beta > 1$  and  $\alpha$  usually belong to  $[2, 4]$ .

### 3.2. The MEMAP Problem in Duty-Cycled Sensor Networks

Given a multicast request, which includes a source node  $s$  and a set of destination nodes  $D (D \subseteq V - \{s\})$ , the MEM problem in the nodes always-awake sensor networks is to construct a multicast tree that satisfies that: (1) it is rooted at the source node  $s$  and spanning all of the nodes in  $D$ ; (2) the sum of the transmission power at the non-leaf nodes is minimized. This problem involves selecting the nodes to transmit the message and scheduling the transmitting power, as well.

However, multicasting in the duty-cycled sensor networks is quite different. According to the discussion in the above section, the working schedule of receivers decides the time slot for which the sender can transmit the packet. Then, in such a network, the broadcasting character can only be used when the receivers wake up simultaneously. In other words, as for a node  $u$  with multiple children in the multicast tree, it would need to transmit the same packet several times if the working schedules of all of the children are not overlapped. Therefore, the MEM problem in such a network not only involves selecting the transmitting nodes, but also the transmitting time slot and the transmitting power for each non-leaf node. We call the transmitting power and the transmitting time slot the transmitting schedule, which is defined as follows:

**Definition 1.** (Transmitting schedule) Given a node  $u \in V$ , a transmitting power  $p (p \in \{p_1, p_2, \dots, p_{\mathcal{L}}\})$  and a time slot  $t (t > 0)$ , then the transmitting schedule  $(u, p, t)$  means node  $u$  can transmit the packet at time slot  $t$  with transmitting power  $p$ .

By exploiting the definition of the transmitting schedule, the MEMAP problem in duty-cycled sensor networks is then to construct a multicast tree  $T$  and determine the transmitting schedules for each non-leaf node, while the total energy cost is minimized. Before we present the formal definition of the MEMAP problem, we need to give some notations here.

We use  $NB(u)$  to denote the set of neighboring nodes of  $u$  in  $G$ , which means the set of nodes  $u$  can communicate with the maximal transmitting power. As for an arbitrary multicast tree  $T$ , we use  $nl(T)$  to denote the set of non-leaf nodes in  $T$  and use  $fa(u)$  to denote  $u$ 's father and  $ch(u)$  to denote the set of  $u$ 's children for any node  $u \in T$ . Let  $(u, v)$  be a tree edge in  $T$ ; then,  $u$  denotes the father node of  $v$ , and  $v$  is a child node of  $u$ .

In the following, we firstly give the definition of a feasible solution.

**Definition 2.** (Feasible solution) Given a source node  $s$  and a set of destination nodes  $D = \{d_1, d_2, \dots, d_K\}$ , a multicast tree  $T$  and the transmitting schedules on multicast tree  $T$ , denoted by  $\mathcal{M}(T)$ , where  $\mathcal{M}(T) = \{(u, p, t) | u \in nl(T)\}$ .  $T$  and  $\mathcal{M}(T)$  are called a feasible solution for the MEMAP problem if they satisfy the following conditions:

1.  $T$  is rooted at  $s$  and spans all of the nodes in  $D$ ;
2. For any two nodes  $u \in nl(T)$ ,  $v \in ch(u)$ , there exists a transmitting schedule  $(u, p, t) \in \mathcal{M}(T)$ , where  $v$  is under the transmission range of  $u$  with transmitting power  $p$  and  $t \in \mathcal{W}(v)$ .

According to the above definition, the definition of the MEMAP problem is presented as follows:

Input:

1. A duty-cycled sensor network  $G = (V, E)$ ;
2. A source node  $s$  and a set of destination nodes  $D = \{d_1, d_2, \dots, d_K\}$ .

Output: A multicast tree  $T$  and the transmitting schedules on the multicast tree  $T$ , denoted by  $\mathcal{M}(T)$ . The multicast tree  $T$  and  $\mathcal{M}(T)$  satisfy the following conditions:

1.  $T$  and  $\mathcal{M}(T)$  are a feasible solution for the MEMAP problem;
2. For any feasible solution for the MEMAP problem, denoted by  $T'$  and  $\mathcal{M}(T')$ , we have

$$\sum_{(u', p', t') \in \mathcal{M}(T')} p' \geq \sum_{(u, p, t) \in \mathcal{M}(T)} p.$$

In the following, we will prove that the MEMAP problem in the duty-cycled sensor networks is NP-hard in Theorem 1.

**Theorem 1.** *The MEMAP problem in the duty-cycled sensor network is NP-hard.*

**Proof.** Let the working schedule of all of the nodes be the same, then the MEMAP problem in the duty-cycled sensor networks is converted to the MEM problem in the nodes always-awake sensor networks. Since the MEM problem in the nodes always-awake sensor networks is proven to be NP-hard by reducing the set cover problem to it, there is no polynomial-time algorithm with an approximation ratio better than  $\Omega(\log K)$  for it unless  $NP = P$  [28]. Therefore, the MEMAP problem is also NP-hard, as it is a general case of MEM problem.  $\square$

Based on Theorem 1, there is no polynomial time algorithm for the MEMAP problem unless  $NP = P$ , so that we study the approximate method to solve it, which is discussed in the next section.

## 4. Approximation Algorithms for MEMAP Problem

### 4.1. An Overview of the Proposed Algorithms

Since the above problem is NP-hard, we propose an approximate algorithm with an approximation ratio of  $4\ln K$ . The proposed algorithm mainly includes four steps. Firstly, we construct a weighted auxiliary graph based on the original graph  $G$ , which is used to integrate the transmitting power and transmitting time slot scheduling problem and the minimum multicast tree constructing problem. Secondly, We derive the MEMAP problem from the minimum node-weighted Steiner tree problem on the auxiliary graph and exploited approximation algorithms for the minimum node-weighted Steiner tree problem. Thirdly, we transform the obtained approximate minimum node-weighted tree  $T$  in the auxiliary graph into a valid multicast tree  $T'$ , which can be mapped into a feasible solution for the MEMAP problem through a series of transformations. Finally, according to the valid multicast tree  $T'$  in the auxiliary graph, we obtain the multicast tree  $T''$  on the original graph and the transmitting schedules for each non-leaf node.

### 4.2. The Auxiliary Graph

To clarify,  $G = (V, E)$  is referred to as the original graph, and  $u \in V$  is referred to as the original node. The first step of our approach is to transform the original graph into a weighted auxiliary graph  $G'$  to assist in building the node-weighted Steiner tree. Before we introduce the definition of the auxiliary graph, we need to give two concepts used in the auxiliary graph.

Generation node: For any node  $u \in V$ , we use  $u_g$  to denote  $u$ 's generation node in the auxiliary graph  $G'$ .

Schedule node: For any node  $u \in V$ , we use  $u_s$  to denote  $u$ 's schedule node in the auxiliary graph  $G'$ . Each schedule node  $u_s$  owns three properties, *i.e.*,  $(u_s.g, u_s.p, u_s.t)$ , which denote its corresponding generation node, transmitting power and transmitting time slot, respectively. For any

node  $u \in V$ , we call  $u_g$  the corresponding generation node of  $u_s$ , and  $u_s$  is derived from  $u_g$  in this paper. Obviously, we can get  $u_s.g = u_g$ .

Definition 3. (Auxiliary graph) Given a duty-cycled sensor network  $G = (V, E)$ , its auxiliary graph  $G' = (V', E')$  denotes the graph containing the scheduling information, where  $V'$  and  $E'$  denote the set of nodes and edges.  $V'$  and  $E'$  are constructed as follows:

- (i) Initially,  $V' = \emptyset, E' = \emptyset$ ;
- (ii) For each node  $u \in V$ , we add a generation node  $u_g$  in the auxiliary graph  $G'$ , i.e.,  $V' = \{u_g | u \in V\}$ ;
- (iii) For each node  $u \in V$ , we then add  $\mathcal{L}$  schedule nodes in  $G'$ , i.e.,  $u_{s1}, u_{s2}, \dots, u_{s\mathcal{L}}$ , where the three properties of  $u_{sl} (1 \leq l \leq \mathcal{L})$  are set as  $u_{sl}.g = u_g, u_{sl}.p = p_l$  and  $u_{sl}.t = 0$ , respectively. It is to be noticed that the transmitting time slot of  $u_s$  is initialized as zero, and we will introduce the method to determine  $u_{sl}.t$  later. Let the set of all of the schedule nodes of  $u$  be denoted by  $Y(u)$ . Then, we can have  $V' = \bigcup_{u \in V} Y(u)$ .
- (iv) For each node  $u \in V$ , we create an edge between  $u_g$  and each schedule node  $u_s \in Y(u)$ , which means node  $u$  can transmit with power  $u_s.p$  on time slot  $u_s.t$ . Let  $E'_u = \{(u_g, u_s) | u_s \in Y(u)\}$ , then we can have  $E' = \bigcup_{u \in V} E'_u$ ;
- (v) Let  $v$  be a neighboring node of  $u$  in the original graph  $G$ , and  $v_g$  is the generation node of  $v$ . For any schedule node  $u_s \in Y(u)$ , we add an edge  $(u_s, v_g)$  in  $G'$  if only if  $v$  is under the transmission range of  $u$  with transmitting power  $u_s.p$ , and we use  $R(u_s)$  to denote the set of such nodes  $v$  of  $u_s$ . After then,  $E'$  can be updated as  $E' = E' \cup \{ \bigcup_{u_s \in Y(u)} E'_{u_s} \}$ , where  $E'_{u_s} = \{(u_s, v_g) | v \in R(u_s)\}$ .

As the example in Figure 1, there is an original graph in Figure 1a, where the number in the braces denotes the working schedules. There are two power levels  $\{p_1, p_2\}$  for each node, and  $p_1 < p_2$ . As for the forwarder  $f$ , it can reach  $a$  and  $c$  with transmission power  $p_1$  and  $a, b, c$  with transmission power  $p_2$ . As for the above original graph, we do as follows according to Definition 3, where the result is shown in Figure 1b. Firstly, the generation nodes are created for each original node, i.e., the blue nodes in Figure 1b, which are  $f_g, a_g, b_g$  and  $c_g$ , respectively. Then, we create two schedule nodes  $f_{s1}$  and  $f_{s2}$  with transmitting power  $p_1$  and  $p_2$ , respectively, and their three properties are  $(f_g, p_1, 0)$  and  $(f_g, p_2, 0)$ . According to the above discussion, we connect  $a_g$  and  $c_g$  to  $f_{s1}$  and connect  $a_g, b_g$  and  $c_g$  to  $f_{s2}$ .

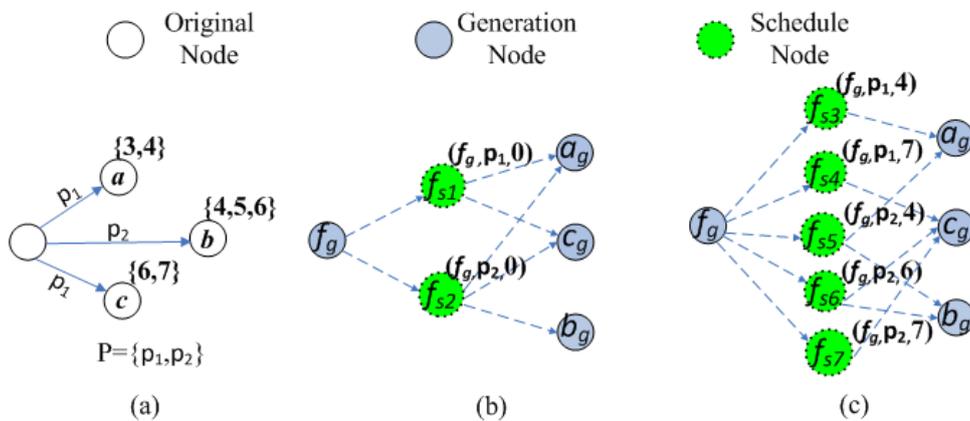


Figure 1. An example of constructing the auxiliary graph. (a) The original graph; (b) The intermediate auxiliary graph; (c) The final auxiliary graph.

In the following, we will introduce the transmitting time slot-determining algorithm to determine the transmitting time slot for each schedule node  $u_s$  in the constructed auxiliary graph  $G'$ . Before that, we need to give some notations.

Let  $start(u) = \min\{t_u \mid t_u \in \mathcal{W}(u)\}$  and  $end(u) = \max\{t_u \mid t_u \in \mathcal{W}(u)\}$  denote the first active time slot and the last active time slot of node  $u$ , respectively. Since the sensor nodes can only receive the data message when it is in the active state, the transmitting time slot must be assigned as the slot that all of its reaching nodes are active. For a schedule node  $u_s$  and all of the reaching nodes in  $R(u_s)$ , it may require several slots to deliver the message to all of them. For this case, the schedule node  $u_s$  may be split into multiple schedule nodes with the same generation node and transmitting power (e.g.,  $u_s.g$  and  $u_s.p$ ), but different transmitting time slots (e.g.,  $u_s.t$ ). In addition, for any transmission schedule  $(u, p, t)$  in the optimal schedule, we should find a corresponding schedule node in the auxiliary graph.

For any schedule node  $u_s$  in Definition 3, the transmitting time slot-determining algorithm exploits a greedy strategy, which mainly works as follows:

Firstly, all of the nodes in  $R(u_s)$  are sorted by their ending time slots in ascending order.

Secondly, we greedily choose the last active time slot of the first node in  $R(u_s)$  to create a new schedule node. Let  $v_1$  be such a node, and then, we create a new schedule node  $u_{s\mathcal{L}+1}$  and set  $u_{s\mathcal{L}+1}.t = end(v_1)$ . For any node  $v \in R(u_s)$ , we connect it to the new schedule node  $u_{s\mathcal{L}+1}$  if  $start(v) \leq u_{s\mathcal{L}+1}.t$ . After that, we remove  $v_1$  from  $R(u_s)$ . For the nodes left in  $R(u_s)$ , we repeat the same procedure, until all of the nodes in  $R(u_s)$  are handled.

Finally, since the original schedule node  $u_s$  will not be used afterwards, we delete it from  $G'$  directly. The detailed procedure of the transmitting time slot-determining algorithm is shown in Algorithm 1.

---

**Algorithm 1** Transmitting time slot-determining algorithm.

---

**Input:** A schedule node  $u_s$ , the set of its reaching nodes  $R(u_s)$ ;

**Output:** The set of splitting schedule nodes  $u_{sj}$  and its transmitting time slot  $u_{sj}.t$ ;

- 1: Sorting the nodes in  $R(u_s)$  according to their last active time slot;
  - 2:  $j \leftarrow \mathcal{L} + 1$ ;
  - 3: **while**  $R(u_s)$  is not empty **do**
  - 4:    $v_1 \leftarrow$  the first node in  $R(u_s)$ ;
  - 5:   Create a new schedule node  $u_{sj}$  identical to  $u_s$ ;
  - 6:    $u_{sj}.t \leftarrow end(v_1)$ ;
  - 7:   Add edge  $(u_s, u_{sj})$  and  $(u_{sj}, v_1)$  in  $G'$ ;
  - 8:   **for**  $i = 2$  to  $|R(u_s)|$  **do**
  - 9:      $v_i \leftarrow$  the  $i$ -th node in  $R(u_s)$
  - 10:    **if**  $start(v_i) \leq u_{sj}.t$  **then**
  - 11:     add edge  $(u_{sj}, v_i)$ ;
  - 12:    **else**
  - 13:     break;
  - 14:    **end if**
  - 15:   **end for**
  - 16:    $j \leftarrow j + 1$ ;
  - 17:   Remove  $v_1$  from  $R(u_s)$ ;
  - 18: **end while**
  - 19: Delete  $u_s$  from  $G'$ ;
  - 20: **return** the set of splitting schedule nodes of  $u_s$ ;
- 

As shown in the above example, since the working schedules of nodes  $a$  and  $c$  are not overlapped, we split the schedule node  $f_{s1}$  into two schedule nodes  $f_{s3}$  and  $f_{s4}$  to connect to  $a_g$  and  $c_g$ , respectively. The transmitting time slot for  $f_{s3}$  and  $f_{s4}$  are set as 4 and 7 respectively, since  $end(a) = 4$  and  $end(c) = 7$ . As for the schedule node  $f_{s2}$ , its reaching node  $R(f_{s2}) = \{a, b, c\}$ . We first choose

the minimal last active time slot from  $\{end(v)|v \in R(f_{s2})\}$ , e.g.,  $end(a) = 4$ , to create a new schedule node  $f_{s5}$  with  $f_{s5}.t = 4$ . Then, we connect  $f_{s5}$  to node  $a_g$  and  $b_g$ . After that, we remove  $a$  from  $R(f_{s2})$ . For the rest of nodes in  $R(f_{s2})$ , we do similarly and choose Time Slot 6 to create a new schedule node to connect node  $b_g$  and node  $c_g$ . The complete auxiliary graph is shown in Figure 1c, and the three properties for each schedule node are shown in the brackets above the node.

Through the above procedure, we can see that the original schedule node is “split” into several schedule nodes with different transmitting time slots. The above greedy strategy can guarantee that for any transmission schedule  $(u, p, t)$  in the optimal solution, we can find a corresponding schedule node in the auxiliary graph, which is shown in Theorem 2.

**Theorem 2.** Assuming  $T_{opt}$  and  $M_{opt}(T_{opt})$  are the optimal multicast tree and its transmitting schedules. Then, for any  $(u, p, t) \in M_{opt}(T_{opt})$ , we can find a schedule node in the auxiliary graph  $G'$ .

**Proof.** Let  $Ch(u, p, t)$  denote the set of the children of  $u$  in the multicast tree  $T_{opt}$  where  $u$  can communicate with at time  $t$  by transmitting power  $p$ , and  $v$  is the node of the minimum last active time slot in  $Ch(u, p, t)$ . In the following, we will prove this from two aspects.

(1) We will firstly prove that all of the working schedules of nodes in  $Ch(u, p, t)$  contain the time slot  $end(v)$ . Assuming there is at least one node whose working schedule does not contain time slot  $end(v)$ , since the working schedules are consecutive, then there must exist a node  $v'$  ( $v' \neq v \wedge v' \in Ch(u, p, t)$ ) that  $end(v') < end(v)$  or  $start(v') > end(v)$ . As  $v$  denotes the node with the minimum last active time slot in  $Ch(u, p, t)$ , so we have  $end(v') \geq end(v)$ . In addition, according to the definition of the MEMAP problem, the working schedules of all of the nodes in  $Ch(u, p, t)$  are overlapped, and all contain time slot  $t$ . Then, we can have  $t \leq end(v)$  and  $start(v') \leq t$ , which result in  $start(v') \leq end(v)$ . Combining the two reasons, we can get  $end(v') \geq end(v)$  and  $start(v') \leq end(v)$ , which contradicts that  $end(v') < end(v)$  or  $start(v') > end(v)$ . Therefore, all of the working schedules of nodes in  $Ch(u, p, t)$  contain the time slot  $end(v)$ .

(2) Now, we will prove that the schedule node  $u_s$  is the target node in two cases:

- If  $t = end(v)$ ,  $u_s$  is the correspondent schedule node obviously.
- If  $t < end(v)$ , since all of the working schedule of the nodes in  $Ch(u, p, t)$  contains the time slot  $end(v)$ , so  $(u, p, end(v))$  is also a feasible schedule that  $u$  can communicate with all of the nodes in  $Ch(u, p, t)$  with transmitting power  $p$  at time slot  $end(v)$ . In this case, we can just map the transmitting schedule  $(u, p, t)$  to the schedule node  $u_s$ , as well.

Combing the above two reasons, the theorem is proven.  $\square$

So far, the auxiliary graph has been constructed. We can find that  $V'$  in the auxiliary graph can be partitioned into two subsets  $V_g$  and  $V_s$ , where  $V_g$  is the set of all generation nodes and  $V_s$  is the set of all of the schedule nodes. In order to exploit the node-weighted Steiner tree algorithm, we set the weight of each generation node  $u_g$  as  $w(u_g) = 0$  and set the weight of each schedule node  $u_s$  as  $w(u_s) = u_s.p$ .

The size of the auxiliary graph is analyzed in Theorem 3.

**Theorem 3.** The number of nodes and edges in the auxiliary graph  $G'$  are at most  $n + n \times \mathcal{L} \times \Delta$  and  $(1 + \Delta) \times (n \times \mathcal{L} \times \Delta)$ , respectively, where  $\Delta$  denotes the maximum degree and  $n = |V|$  denotes the number of nodes in the original graph.

**Proof.** Firstly, as in Definition 3, for each generation node  $u_g$  in the auxiliary graph  $G'$ , there is only one schedule node  $u_s$  with power level  $l$  in  $Y(u)$ , where  $Y(u)$  denotes all of the schedule nodes of  $u$ . Then, the schedule node  $u_s$  is split into several schedule nodes with power level  $l$  after executing the transmitting time slot-determining algorithm. Let  $Y(u_s, l)$  denote the set of schedule nodes derived

from  $u_g$ , and its power level is  $l$ . In the worst case, we can have  $|Y(u_s, l)| = |NB(u)| \leq \Delta$ . Then, for each node  $u \in V$ , we can have:

$$|Y(u)| = \sum_1^{\mathcal{L}} |Y(u_s, l)| \leq \sum_1^{\mathcal{L}} |NB(u)| \leq \mathcal{L} \times \Delta \quad (2)$$

Thus, according to Equation (2), the total number of nodes in the auxiliary graph  $G'$  can be calculated as:

$$|V'| = |V_g| + |V_s| = |V_g| + \sum_{u \in V} |Y(u)| \leq n + n \times \mathcal{L} \times \Delta \quad (3)$$

where  $|V_g| = |V| = n$  denotes the number of nodes in the original graph.

Secondly, according to Definition 3 and Algorithm 1, for each generation node  $u_g$ , there exists an edge between  $u_g$  and  $u_s$  ( $u_s \in Y(u)$ ). For each schedule node  $u_s \in Y(u)$ , there exists at most  $|NB(u)|$  edges from  $u_s$  to its neighboring generation nodes. Then, we can have:

$$|E'| = \sum_{u \in V} |E'_u| + \left| \bigcup_{u \in V} \left\{ \bigcup_{u_s \in Y(u)} E'_{u_s} \right\} \right| \leq \sum_{u \in V} |Y(u)| + \sum_{u \in V} (|Y(u)| \times \Delta) \leq (1 + \Delta) \times (n \times \mathcal{L} \times \Delta) \quad (4)$$

□

As we can see, compared to the original graph, the nodes in the auxiliary graph increased  $\mathcal{L} \times \Delta$  times and the edges increased  $\mathcal{L} \times \Delta^2$  times. Since  $\mathcal{L}$  and  $\Delta$  are usually constant, the size of the auxiliary graph is controlled. In addition, we can notice that the auxiliary graph  $G'$  has the following properties:

(1) Given two nodes in  $G'$ , there are no edges between them if they are both schedule nodes or generation nodes. In other words, the neighbors of a schedule node are all generation nodes, and the neighbors of a generation node are all schedule nodes. Two generation nodes are connected through a schedule node.

(2)  $R(u_{si}) \subseteq R(u_{sj})$ , if  $1 \leq u_{si}.p \leq u_{sj}.p \leq \mathcal{L}$  and  $u_{si}.t = u_{sj}.t$ , where  $u_{si}$  and  $u_{sj}$  are two schedule nodes derived from the same generation node  $u_g$ .

(3) For any two generation nodes  $u_g$  and  $v_g$ , which are connected to a same schedule node  $w_s$ , then the working schedules of node  $u$  are overlapped with  $v$ , which means they can receive the packet simultaneously.

#### 4.3. Minimum Node-Weighted Steiner Tree

Given a multicast request that includes a source node  $s$  and a set of destination nodes  $D$  in the duty-cycled sensor network  $G$ , let  $s_g$  be the generation node of  $s$  in the auxiliary graph  $G'$ , and  $D_g = \{d_{1g}, d_{2g}, \dots, d_{kg}\}$  are the set of the generation nodes of the destination nodes. Now, our objective is to find a minimum node-weighted Steiner tree in  $G'$ , which is rooted at  $s_g$  and spanning all of the nodes in  $D_g$ . The minimum node-weighted Steiner tree was used to help us obtain a feasible solution for the MEMAP problem, which includes a multicast tree  $T$  and the transmitting schedules  $M(T)$ .

However, it is unlikely to have a polynomial-time algorithm to find such a minimum node-weighted tree in the auxiliary graph  $G'$  unless  $P = NP$ . Thus, a approximation algorithm [38] is exploited to obtain the near optimal minimum node-weighted Steiner tree with approximation ratio of  $2 \ln K$ .

Let  $T_{ag}^{app}$  denote the obtained approximate minimum node-weighted Steiner tree in the auxiliary graph  $G'$ . In the following, we will introduce the method to map  $T_{ag}^{app}$  to a feasible solution for the MEMAP problem, which is guaranteed by the following theorem.

**Theorem 4.** Let  $T_{ag}$  be a node-weighted Steiner tree, which is rooted at  $s_g$  and spans all of the nodes in  $D_g$  in the auxiliary graph  $G'$ ;  $T_{ag}$  can be mapped to a feasible solution for the MEMAP problem if it satisfies the following conditions:

1. For any leaf node  $i \in T_{ag}$ ,  $i$  must be a generation node in  $D_g$ ;
2. For each schedule node  $u_s$  in the node-weighted Steiner tree  $T_{ag}$ , there exists a generation node  $u_g$  in  $T_{ag}$ , where  $u_g$  is the corresponding generation node of  $u_s$ ;
3. For any non-leaf generation node  $u_g$  in the tree  $T_{ag}$  and any node  $i \in ch(u_g)$ , node  $i$  must be a schedule node derived from  $u_g$ .

**Proof.** In the following, we will prove the theorem by constructing a feasible solution for the MEMAP problem with the node-weighted Steiner tree  $T_{ag}$ .

Firstly, the multicast tree  $T$  in the original graph is constructed by the following three steps:

Step 1. For any schedule node in  $T_{ag}$ , we create an edge between its father and all of its child nodes;

Step 2. Remove all of the schedule nodes from  $T_{ag}$ ;

Step 3. Replace all of the generation nodes with their original nodes.

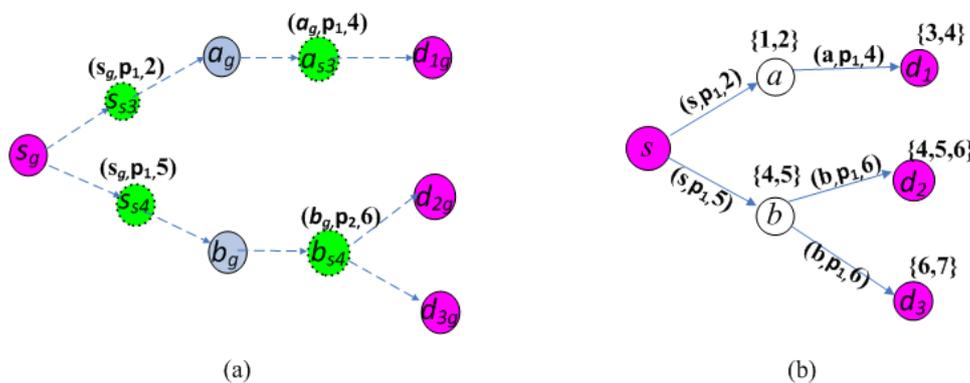
As we can see, for any tree edge  $(fa(u), u) \in T$ , there exists a schedule node between  $fa(u)$  and  $u$  in the node-weighted Steiner tree  $T_{ag}$ , which means that  $fa(u)$  can communicate with  $u$  with a certain power. Additionally,  $s_g$  and  $D_g$  are all in  $T_{ag}$ , then  $s$  and  $D$  are in  $T$  accordingly. Therefore, the source node  $s$  can deliver the messages to all of the destination nodes by the tree  $T$ .

Secondly, we will show how to determine the transmitting schedules for  $T$ , i.e.,  $M(T)$ . For each node  $u$  in  $T$ ,  $u_g$  is its corresponding generation node in  $T_{ag}$ . Then, for each schedule node  $u_s \in ch(u_g)$  in  $T_{ag}$ , we add a transmitting schedule  $(u, u_s, p, u_s, t)$  in  $M(T)$ . Since for each non-leaf node  $u$  and its child node  $v \in T$ , there exists a schedule node between them, then we can have a transmitting schedule  $(u, u_s, p, u_s, t)$  with which  $u$  can communicate with  $v$  with transmitting power  $u_s, p$  at time slot  $u_s, t$ .

Combining the above two reasons and Definition 2, a feasible solution for the MEMAP problem is obtained. The theorem is proven. □

In this paper, we call a node-weighted Steiner tree a valid multicast tree if it satisfies the three conditions in Theorem 4.

As the example shown in Figure 2, there is a calculated Steiner tree in Figure 2a, where the pink node denotes the source node and the destination nodes. We can find that the tree in Figure 2a is a valid multicast tree since it satisfies the conditions in Theorem 4. According to the above method, then we can obtain the corresponding multicast tree and transmitting schedules on the original graph. The results are shown in Figure 2b, where the three tuple along the links denotes a transmitting schedule.



**Figure 2.** An example of a valid multicast tree.(a) A valid multicast tree; (b) The corresponding multicast tree and transmitting schedules.

#### 4.4. Constructing a Valid Multicast Tree

However, the obtained approximate minimum node-weighted Steiner tree  $T_{ag}^{app}$  on the auxiliary graph  $G'$  may not satisfy the three conditions in Theorem 4. There exist three violations.

- Violation 1.  $T_{ag}^{app}$  contains a leaf node  $i$ , and  $i$  does not belong to  $D_g$ . This violates Condition 1 in Theorem 4.
- Violation 2.  $T_{ag}^{app}$  contains a schedule node, which is not derived from any of its neighboring generation nodes in the tree  $T_{ag}^{app}$ . This violates Condition 2 in Theorem 4.
- Violation 3.  $T_{ag}^{app}$  contains a generation node  $u_g$  ( $u_g \neq s_g$ ), but  $u_g$  cannot be reached by the source node  $s_g$ , which means that there exist the tree edges  $(v_g, u_s)$  and  $(u_s, u_g)$  in  $T_{ag}^{app}$ , where  $u_s$  is the father node of  $u_g$  and a child node of  $v_g$ . This violates Condition 3 in Theorem 4.

In order to eliminate the three violations in the approximate minimum node-weighted Steiner tree  $T_{ag}^{app}$ , we introduce three correcting operations as follows.

For Violation 1: For any leaf node  $i$  that does not belong to  $D_g$ , we just delete it from  $T_{ag}^{app}$ . This procedure continues until all of the leaf nodes satisfy Condition 1 in Theorem 4.

For Violation 2: Assume  $u_s$  is the schedule node, which is not derived from any of its neighboring generation nodes. Let  $u_g$  be the generation node of  $u_s$ , then we do as follows:

- if  $u_g \notin T_{ag}^{app}$ , then the generation node  $u_g$  and the edge  $(u_g, u_s)$  are added in the tree  $T_{ag}^{app}$ ;
- if  $u_g \in T_{ag}^{app}$ , but  $u_g \notin NB(u_s)$ , where  $NB(u_s)$  denotes the set of neighbors of  $u_s$  in the tree  $T_{ag}^{app}$ ; in this case, we delete the tree edge  $(fa(u_s), u_s)$  from  $T_{ag}^{app}$  firstly, and then, we would add the tree edge  $(u_g, u_s)$  in the current tree  $T_{ag}^{app}$ .

For Violation 3: This correcting operation is done by checking all of the nodes in the tree  $T_{ag}^{app}$  through a breadth-first search. All of the nodes in the tree have two states, e.g., “unchecked” and “checked” states. Let the queue  $Q$  store the set of current nodes needed to be checked. Initially, all of the nodes in  $T_{ag}^{app}$  are marked “unchecked”, and the root  $s_g$  is pushed into  $Q$ . The correcting process works as follows:

Let the first node in  $Q$  be  $i$ ; we first marked node  $i$  “checked” and pop it from  $Q$ . Then, we handle  $i$  according to the following two cases:

Case 1:  $i$  is a generation node. Then, for each schedule node  $j_s \in ch(i)$ , we do as follows:

- If  $j_s$  is not derived from  $i$ , let  $j_g$  denote the corresponding generation node of  $j_s$ . We then check for any schedule node  $c_s$  ( $c_s \in ch(i)$ ), whether there exists an edge between  $c_s$  and  $j_g$  in the auxiliary graph  $G'$ . If yes, add an edge  $(c_s, j_g)$  in  $T_{ag}^{app}$ . Otherwise, we choose the schedule node  $c_s = \operatorname{argmin}\{w(i_s) | i_s \in Y(i) \wedge (i_s, j_g) \in E'\}$  to add into the current tree  $T_{ag}^{app}$ , and then, the tree edges  $(i, c_s)$  and  $(c_s, j_g)$  are added. After that, we delete the tree edge  $(i, j_s)$  from  $T_{ag}^{app}$ . Finally, the new added schedule node  $c_s$  is pushed into  $Q$ .
- If  $j_s$  is derived from  $i$ , we just push it into  $Q$ ;

Case 2:  $i$  is a schedule node. Then, for any generation node  $u_g \in ch(i)$ , we just push it into  $Q$  for the following computing.

The correcting operation for Violation 3 ends when  $Q$  is empty, and all of the nodes in  $T_{ag}$  are marked “checked”.

In the following, we will prove that the tree after the above three correcting operations can satisfy the three conditions in Theorem 4, which means it is a valid multicast tree.

**Theorem 5.** Let the tree after the three correcting operations be  $T_{ag}^c$ ; then,  $T_{ag}^c$  is a valid multicast tree.

**Proof.** In order to guarantee the correctness of Theorem 5, we just need to prove that the above three operations can eliminate the violations successfully, that is the three conditions in Theorem 4 are satisfied.

As for Condition 1 (for any leaf node  $i \in T_{ag}^c$ ,  $i$  must be a generation node in  $D_g$ ), according to correcting Operation 1, any leaf node  $i$  that is not a generation node in  $D_g$  is pruned. Then, Condition 1 is satisfied.

As for Condition 2 (for each schedule node  $u_s$  in the node-weighted Steiner tree  $T_{ag}^c$ , there exists a generation node  $u_g$  in  $T_{ag}^c$ , where  $u_g$  is the corresponding generation node of  $u_s$ ), according to correcting Operation 2, for any schedule node  $u_s$  in  $T_{ag}^c$ , its generation node  $u_g$  and the tree edge  $(u_g, u_s)$  are added in the tree. Then, Condition 2 is also satisfied.

As for condition 3 (for any non-leaf generation node  $u_g$  in the tree  $T_{ag}^c$ , then for any node,  $i \in ch(u_g)$  must be a schedule node derived from  $u_g$ ), according to correcting Operation 3, for each generation node  $i$  in  $T_{ag}^c$  and any schedule node  $j_s$  that is not derived from  $i$ , we delete the tree edge  $(i, j_s)$  in the obtained approximate minimum node-weighted Steiner tree and add a schedule node  $i_s$ , which is chosen to connect to  $j_s$ 's corresponding generation node  $j_g$ . Obviously, Condition 3 is satisfied.

Combining the above three reasons, the theorem is proven.  $\square$

After the approximate minimum node-weighted Steiner tree  $T_{ag}^{app}$  has been corrected to a valid multicast tree in the auxiliary graph, then we can transform it into a feasible solution for the MEMAP problem according to Theorem 4. So far, the complete approximate scheduling and constructing algorithms have all been introduced, which is shown in Algorithm 2.

---

**Algorithm 2** Approximate scheduling and constructing algorithm.

---

**Input:** A duty-cycled network  $G$ , a source node  $s$  and a set of destination nodes  $D$ ;

**Output:** A multicast tree  $T$  and the set of transmitting schedules  $M(T)$  for the multicast tree  $T$ ;

- 1: Construct the auxiliary graph  $G'$  according to Definition 3;
  - 2: **for all** schedule node  $u_s$  in  $G'$  **do**
  - 3:   Call Algorithm 1 to determine the transmitting time slot for each schedule node;
  - 4: **end for**
  - 5: Call the Steiner tree algorithm to get a multicast tree  $T_{ag}^{app}$  on  $G'$ ;
  - 6: Correct  $T_{ag}^{app}$  to a valid multicast tree  $T_{ag}^c$  on  $G'$  by using the three correction operations in Section 4.4;
  - 7: Map the valid multicast tree  $T_{ag}^c$  into a feasible solution for MEMAP using the method in Theorem 4, including the multicast tree  $T$  and the set of transmitting schedule  $M(T)$ ;
  - 8: **return** the multicast tree  $T$  and the set of transmitting schedule  $M(T)$ ;
- 

#### 4.5. Approximation Ratio Analysis

In the following, we give the approximation ratio analysis of the proposed algorithm in Lemma 1 and Theorem 6 below.

**Lemma 1.** *Given a multicast request  $(s:D)$  in the duty-cycled network, the weighted sum of the minimum node-weighted Steiner tree is the lower bound for the MEMAP problem.*

**Proof.** Let  $(T_{opt}, M_{opt})$  denote the optimal result for the MEMAP problem in the duty-cycled network  $G$ . Following the construction of the auxiliary graph  $G'$ ,  $(T_{opt}, M_{opt})$  can be mapped into a node-weighted Steiner tree  $T'_{ag}$  in  $G'$ , which is rooted at  $s_g$  and spanning all of the nodes in  $D_g$ , and the transmitting schedule  $(u, p, t) \in M_{opt}$  is mapped to a schedule node of  $u_s$  in  $T'_{ag}$ . Then, the total transmission power in  $M_{opt}$  is equal to the weighted sum of the tree  $T'_{ag}$ .

Assume  $T_{ag}^{opt}$  is the minimum node-weighted Steiner tree in the auxiliary graph  $G'$ , which spans all of the nodes in  $D_g \cup \{s_g\}$ . Obviously, we can get  $W(T_{ag}^{opt}) \leq W(T'_{ag})$ , where  $W(T_{ag}^{opt})$  denotes the weighted sum of all of the nodes in  $T_{ag}^{opt}$ . The lemma is proven.  $\square$

**Theorem 6.** The approximation ratio of our method is  $4\ln K$ , where  $K$  is the number of the destination nodes.

**Proof.** Let  $T_{ag}^{app}$  be the obtained approximate minimum node-weighted Steiner tree through [38]. According to [38], we have  $W(T_{ag}^{app}) \leq 2\ln KW(T_{ag}^{opt})$ , where  $T_{ag}^{opt}$  is the minimum node-weighted Steiner tree.

Let  $T_{ag}^1$ ,  $T_{ag}^2$  and  $T_{ag}^3$  denote the node-weighted Steiner tree after correcting Operations 1, 2 and 3, respectively. In the following, we will prove  $W(T_{ag}^3) \leq 2W(T_{ag}^{app})$ .

Firstly, in correcting Operation 1, we remove the leaf nodes that does not belong to  $D_g$ ; obviously, we can have:

$$W(T_{ag}^1) \leq W(T_{ag}^{app}) \quad (5)$$

Secondly, in correcting Operation 2, we add the generation node of the schedule node  $u_s$ , of which the generation node is not in the tree  $T$ . Since the weight of all of the generation nodes is zero, we can have:

$$W(T_{ag}^2) = W(T_{ag}^1) \leq W(T_{ag}^{app}) \quad (6)$$

Thirdly, in correcting Operation 3, let  $u_s$  be the schedule node that is not derived from its father  $i_g$ . We handle this case in the following two aspects:

1. If we can find a schedule node  $i_s \in ch(i_g)$  to reach the generation node  $u_g$  of  $u_s$ , then we add an tree edge  $(i_s, u_g)$ . No schedule node is added in this case. Thus, the weighted sum of the tree is not changed.
2. If we cannot find such a power, we need to add a schedule node  $i'_s$  with  $i'_s = \operatorname{argmin}\{w(i_s) | i_s \in Y(i) \wedge (i_s, u_g) \in E'\}$  in the correcting tree  $T_{ag}^2$ . Since we assume that the transmission is symmetric, so the weight of added schedule node  $i'_s$  is not larger than  $w(u_s)$ .

Therefore, for each schedule node  $u_s \in T_{ag}^2$ , at most a schedule node with weight  $w(u_s)$  is added in the correcting tree  $T_{ag}^3$ . Therefore, we can have:

$$W(T_{ag}^3) \leq 2W(T_{ag}^2) \quad (7)$$

According to Equations (5)–(7), we can have  $W(T_{ag}^3) \leq 2W(T_{ag}^{app})$ . In addition, according to Lemma 1, we can have  $W(T_{ag}^{app}) \leq 2\ln KW(T_{ag}^{opt})$  and  $W(T_{ag}^{opt}) \leq W(T'_{ag})$ , where  $T'_{ag}$  denotes the corresponding node-weighted Steiner tree for the optimal result for the MEMAP problem. Therefore, we can have  $W(T_{ag}^3) \leq 4\ln KW(T'_{ag})$ , and the approximation ratio is  $4\ln K$ . The theorem is proven.  $\square$

Additionally, the time complexity of the proposed algorithm can be proven to be polynomial by Lemma 2 and Theorem 7.

**Lemma 2.** The time complexity of Algorithm 1 is  $O(\Delta^2)$ .

**Proof.** In Algorithm 1, since the number of nodes of  $R(u_s)$  is less than  $\Delta$ , then Step 1 would take  $O(\Delta \times \log \Delta)$  time. In addition, we can see Steps 4–17 would take  $O(|R(u_s)|) = O(\Delta)$  time obviously. Therefore, the time complexity from Steps 3–18 is  $O(|R(u_s)| \times |R(u_s)|) = O(\Delta^2)$ . Therefore, combining the above analysis, the time complexity of Algorithm 1 is  $O(|R(u_s)| \times |R(u_s)|) = O(\Delta^2)$ .  $\square$

**Theorem 7.** The time complexity of the approximate scheduling and constructing algorithm is  $O(K^2 \mathcal{L} \Delta n \log(\mathcal{L} \Delta n))$ .

**Proof.** Firstly, according to Definition 3, Step 1 would take  $O(\mathcal{L}n)$  time to create the schedule nodes and  $O(\mathcal{L} \Delta n)$  time to create the edges. Then, the time complexity for Step 1 is  $O(\mathcal{L} \Delta n)$ . As for Step 2, there are  $O(\mathcal{L}n)$  schedule nodes, and the time complexity of Algorithm 1 is  $\Delta^2$  according to Lemma 1, so the time complexity for Step 2 is  $O(\mathcal{L} \Delta^2 n)$ . In Step 3, since the number of nodes and edges are  $n + n\mathcal{L} \Delta$  and  $(1 + \Delta) \times (n\mathcal{L} \Delta)$  according to Theorem 3, then the time complexity of Step 3 is

$O(K^2\mathcal{L}\Delta n\log(\mathcal{L}\Delta n))$  according to [38]. In Step 4, we need to correct the obtained Steiner tree with three correction operations, where correction Operations 1 and 2 take  $O(\mathcal{L}\Delta n)$  time by examining the tree once and correction Operation 3 would take  $O(\mathcal{L}\Delta n)$  time by a breadth-first search. Therefore, the time complexity for Step 4 is  $O(\mathcal{L}\Delta n)$ . In Step 5, the time complexity is also  $O(\mathcal{L}\Delta n)$  by visiting the corrected tree once. In summary, the time complexity of Algorithm 2 is  $O(K^2\mathcal{L}\Delta n\log(\mathcal{L}\Delta n))$ .  $\square$

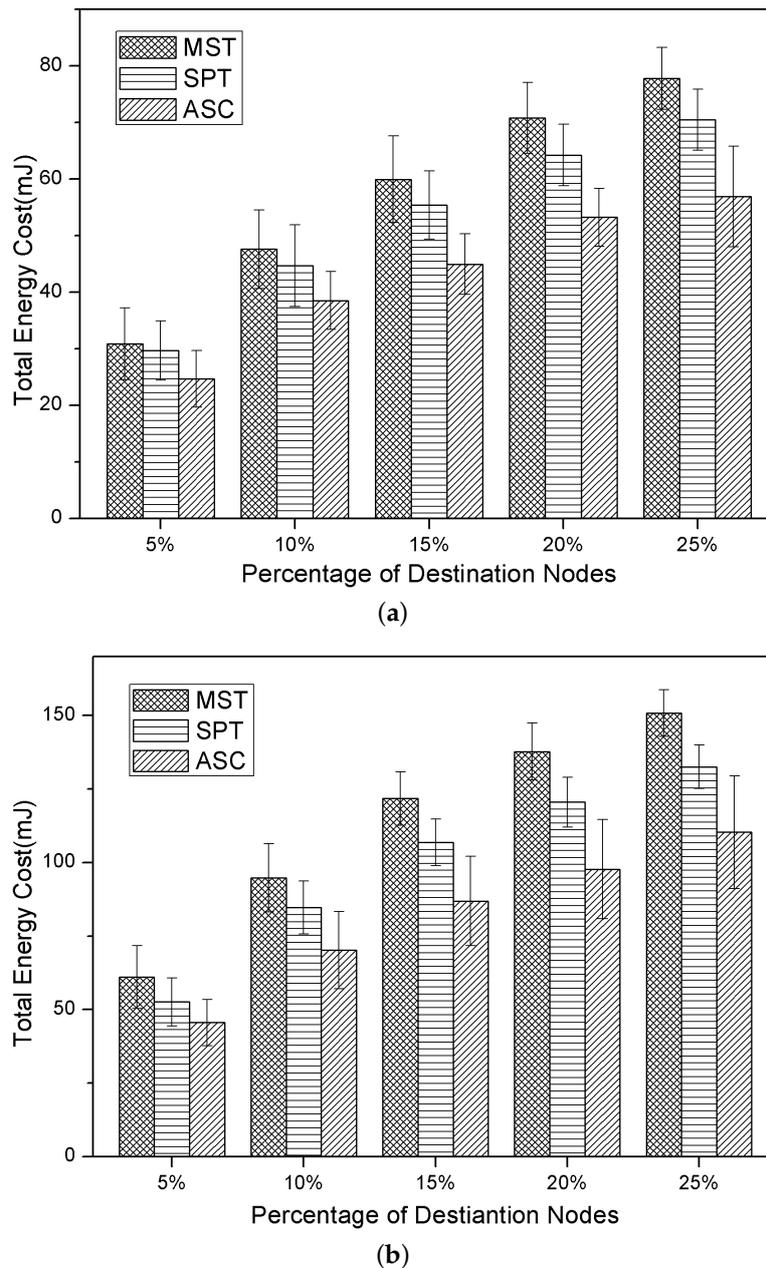
## 5. Experimental Results

In this section, we study the empirical performance of the proposed algorithm. In the experiments, we randomly deploy the wireless nodes in a 300 m  $\times$  300 m field, where the number of nodes ranges from 100–200. The duty cycle is set from 5%–25%, and one time slot is set 50 ms. The working schedule of each node is generated randomly to test a wide range of configurations. In addition, each node in the experiments can transmit at five power levels, which are 1 MW, 10 MW, 15 MW, 20 MW and 50 MW respectively. The bandwidth is set to 40 kbps, and the size of the data is 100 bytes. All experiments are repeated 100 times with different node deployments and working schedules.

Since most existing methods are not suitable for minimum-energy multicasting with adjusted power in the duty-cycled sensor networks, we compare our algorithm (denoted as ASC) with the following baseline methods: the minimum spanning tree (MST) method and the shortest path tree (SPT) method. The SPT method calculates the shortest paths from the source node to each destination node, and then, all of the shortest paths form a multicast tree. As for the MST method, the leaf nodes that are not destination nodes are removed. This method is most widely used to approximate the Steiner tree. Additionally, in order to use the multicast tree constructed by the MST and SPT methods in the duty-cycled sensor networks, the transmitting schedules, including the transmitting time slot and the transmitting power for each non-leaf node, should also be determined. In the experiments, we exploit the enumerating method to find the best transmitting schedules for the multicast tree constructed by the MST and SPT methods.

### 5.1. Performance of Energy Cost

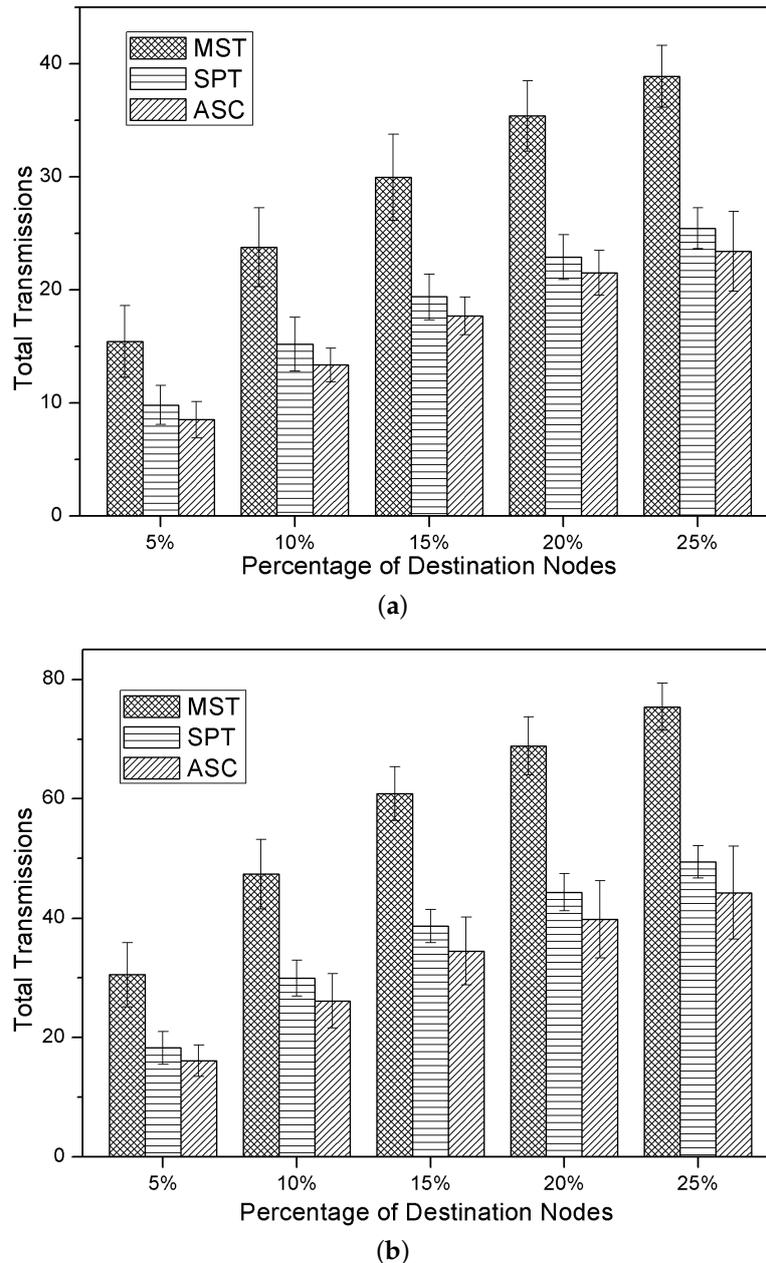
Firstly, we compare the total energy cost of three algorithms under different numbers of destination nodes. We conduct the experiments mainly in two network scenarios, while the number of nodes is set to 100 and 200, respectively, and the results are shown in Figure 3a,b. In Figure 3a, the x-coordinate denotes the percentage of the destination nodes of all nodes, which ranges from 5%–25%. As we can see, the total energy cost produced by our method is the lowest, which is about 20% lower than the MST method, and when the number of destination nodes is greater, the ratio can even reach 30%. This is because our method considers the working schedule and the transmitting power of each node in constructing the multicast tree, which can reduce the energy cost and the number of transmissions at the same time. Although the MST and SPT method determine the transmission schedule optimally in a localized way, they construct the multicast tree regardless of the node's working schedule and cannot optimize the multicast tree in a global manner. It's to be noticed that in Figure 3a, the total energy cost of the SPT method is a little lower than the MST method; this is because the SPT method reduces the number of transmissions through adjusting the transmitting power. In Figure 3b, the total energy cost of three methods is increased compared to Figure 3a; this is because the number of nodes and the destination nodes increased. However, our algorithm still generates the lowest energy cost compared to the other two methods. Additionally, we can see that the total energy cost of the three algorithms grows with the number of destination nodes increasing in both experiments.



**Figure 3.** Total energy cost. (a)  $|V| = 100$ ; (b)  $|V| = 200$ .

### 5.2. Performance of Total Transmissions

In this group of experiments, we analyzed the total number of transmissions of three methods in the multicasting process. In the duty-cycled sensor networks, besides the transmitting power of all of the nodes in the multicast tree, the total number of transmissions is also an important factor for the total energy cost. As shown in Figure 4a,b, the total number of transmissions of the the SPT method is far less than the MST method. This results in that the energy cost of the SPT method is less than the MST method (which has been demonstrated in Figure 3a,b), despite that the SPT method exploits higher transmitting power in multicasting. However, both the total number of transmissions of the MST method and the SPT method are larger than our method. Additionally, as shown in Figure 4a,b, the total number of transmissions of three methods increases with the number of destination nodes.



**Figure 4.** The total number of transmissions. (a)  $|V| = 100$ ; (b)  $|V| = 200$ .

### 5.3. Performance under Different Duty Cycles

Finally, we investigate the influences of different duty cycles on the performance of the three methods, and the results are shown in Figure 5a,b. In this group of experiments, the duty cycle of each node varies from 5%–20%, and the percentage of the destination node is set as 15%. As we can see in Figure 5a,b, the total energy cost and the number of transmissions of our method are both the lowest, which demonstrates the high performance of our method in terms of energy cost. Additionally, in Figure 5a, we can find that the total energy cost of our method decreases slightly with the duty cycle increasing. This may be because our method has exploited the working schedule of each node in constructing the multicasting tree, and the common active slots of the neighboring nodes do not vary much when the duty cycle increases 5%, which results in the total number of transmissions not reducing obviously. This can be illustrated in Figure 5b, where the number of transmissions of the three methods decrease slightly when the duty cycle of each node increases 5%.

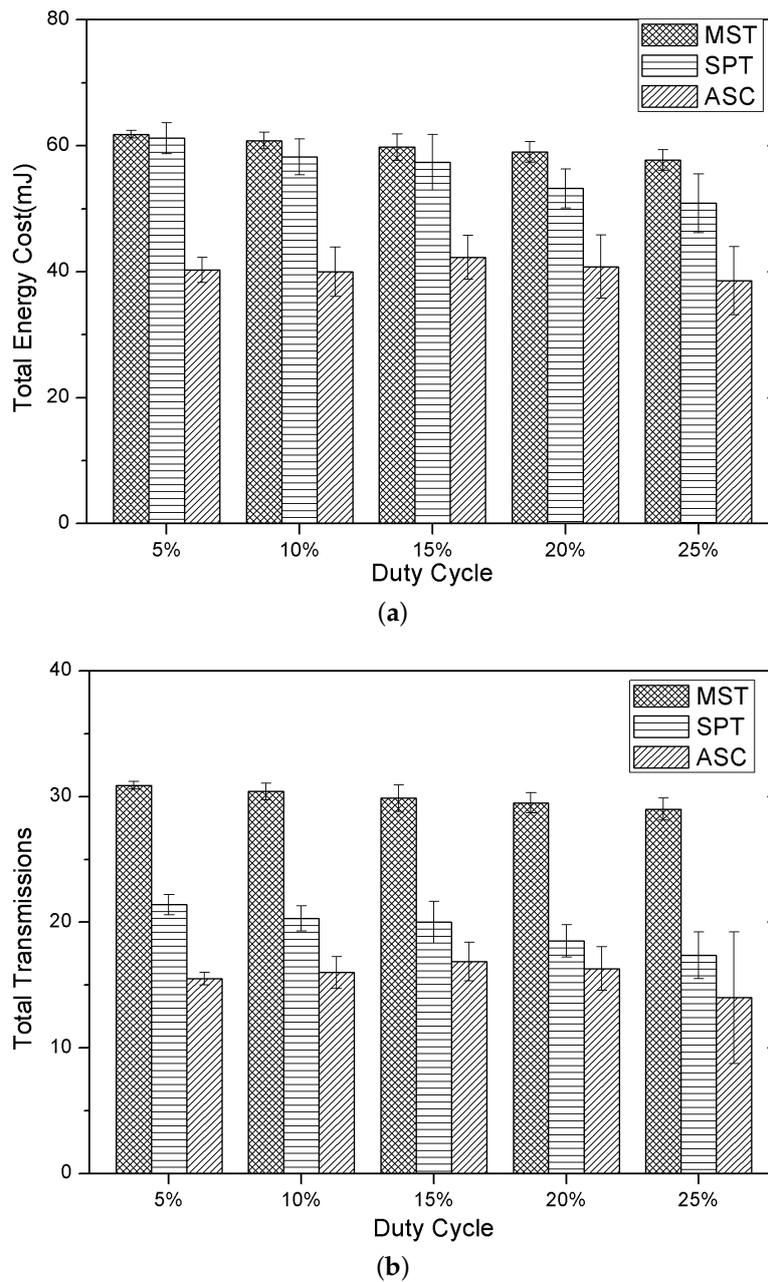


Figure 5. Performance under different duty cycles. (a) Total energy cost; (b) Total transmissions.

## 6. Conclusions

In this paper, the problem of minimum energy multicasting with adjusted power (MEMAP) in duty-cycled sensor networks was proposed, and it was proven to be NP-hard. To solve such a problem, an auxiliary graph was proposed to integrate the transmitting power and time slot scheduling problem and the minimum multicast tree constructing problem in MEMAP, and a greedy algorithm was exploited to construct such a graph. Based on the proposed auxiliary graph, an approximate scheduling and constructing algorithm with an approximation ratio of  $4lnK$  was proposed, where  $K$  is the number of destination nodes. Finally, we perform extensive simulations, and the results verify the high performance of the proposed algorithm in terms of the energy cost and transmission redundancy.

**Acknowledgments:** This work is supported in part by the National Basic Research Program of China (973 Program) under grant No. 2012CB316200, the National Natural Science Foundation of China (NSFC) under Grant No. 61190115, 61370217, the Fundamental Research Funds for the Central Universities under grant No. HIT.KISTP201415, the National Science Foundation (NSF) under Grants No. CNS-1152001, CNS-1252292, the Research Fund for the Doctoral Program of Higher Education of China under grant No.20132302120045, and the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under grant No.HIT.NSRIF.2014070.

**Author Contributions:** Quan Chen developed the core algorithm of the intertwined scheduling and constructing algorithms for minimum-energy multicasting in duty-cycled sensor networks and described it. The contributions of Siyao Cheng focused mainly on the constructing algorithm of the auxiliary graph and the proof of the approximation ratio of the proposed algorithm; she also participated in the English correction of this paper. Hong Gao and Zhipeng Cai are the supervisors.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Chen, Q.; Cheng, S.Y.; Gao, H.; Cai, Z.P. Approximate Scheduling and Constructing Algorithms for Minimum-Energy Multicasting in Duty-Cycled Sensor Networks. In Proceedings of the International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), Beijing, China, 22–23 October 2015.
2. He, T.; Krishnamurthy, S.; Luo, L.; Ting, Y. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM TOSN* **2006**, *2*, 1–38.
3. Song, Y.; Liu, L.; Ma, H. A Biology-Based Algorithm to Minimal Exposure Problem of Wireless Sensor Networks. *IEEE Trans. Netw. Service Manag.* **2014**, *11*, 417–430.
4. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless Sensor Network Survey. *Comput. Netw.* **2008**, *52*, 2292–2330.
5. Gao, J.; Li, J.; Cai, Z.; Gao, H. Composite Event Coverage in Wireless Sensor Networks with Heterogeneous Sensors. In Proceedings of the 34th IEEE International Conference on Computer Communications (IEEE INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 217–225.
6. Cheng, S.; Cai, Z.; Li, J.; Fang, X. Drawing Dominant Dataset From Big Sensory Data in Wireless Sensor Networks. In Proceedings of the 34th IEEE International Conference on Computer Communications (IEEE INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 531–539.
7. Cai, Z.; Chen, Z.; Lin, G. A 3.4713-approximation algorithm for the capacitated multicast tree routing problem. *Theor. Comput. Sci.* **2009**, *410*, 5415–5424.
8. Cai, Z.; Randy, G.; Lin, G. Size-constrained tree partitioning: Approximating the multicast k-tree routing problem. *Theor. Comput. Sci.* **2011**, *412*, 240–245.
9. Cai, Z.; Xue, G.; Lin, G. Improved Approximation Algorithms for the Capacitated Multicast Routing Problem. *COCOON* **2005**, *8881*, 136–145.
10. He, Z.; Cai, Z.; Cheng, S.; Wang, X. Approximate Aggregation for Tracking Quantiles in Wireless Sensor Networks. *Theor. Comput. Sci.* **2015**, *607*, 381–390.
11. Zheng, X.; Li, J.; Gao, H.; Cai, Z. Capacity of Wireless Networks with Multiple Types of Multicast Sessions. In Proceedings of the 25th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Philadelphia, PA, USA, 11–14 August 2014; pp. 135–144.
12. Yao, Y.J.; Cao, Q. EDAL: An Energy-Efficient, Delay-Aware, and Lifetime-Balancing Data Collection Protocol for Wireless Sensor Networks. In Proceedings of the 10th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (IEEE MASS), Hangzhou, China, 14–16 October 2013; pp. 182–190.
13. Liu, X.; Luo, J.; Vasilakos, A. Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks. In Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Salt Lake City, UT, USA, 27–30 June 2011; pp. 46–54.
14. Liu, X.Y.; Zhu, Y.; Kong, L.; Liu, C. CDC: Compressive Data Collection for Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 2188–2197.
15. Han, K.; Luo, J.; Liu, Y.; Vasilakos, A. Algorithm Design for Data Communications in Duty-Cycled Wireless Sensor Networks: A survey. *IEEE Commun. Mag.* **2013**, *51*, 107–113.

16. Gu, Y.; He, T.; Lin, M.; Xu, J. Spatiotemporal Delay Control for Low-Duty-Cycle Sensor Networks. In Proceedings of the 30th Real-Time Systems Symposium (RTSS), Washington, DC, USA, 1–4 December 2009; pp. 127–137.
17. Gu, Y.; He, T. Dynamic Switching-Based Data Forwarding for Low-Duty-Cycle Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2011**, *10*, 1741–1754.
18. Wang, F.; Liu, J. On Reliable Broadcast in Low Duty-Cycle Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2012**, *11*, 767–779.
19. Cheng, L.; Gu, Y.; He, T.; Niu, J. Dynamic Switching-Based Reliable Flooding in Low-Duty-Cycle Wireless Sensor Networks. In Proceedings of the 32nd IEEE International Conference on Computer Communications (IEEE INFOCOM), Turin, Italy, 14–19 April 2013; pp. 1393–1401.
20. Zeng, Y.; Xiang, K.; Li, D.; Vasilakos, A. Directional routing and scheduling for green vehicular delay tolerant networks. *Wirel. Netw.* **2013**, *19*, 161–173.
21. Meng, T.; Wu, F.; Yang, Z. Spatial Reusability-Aware Routing in Multi-Hop Wireless Networks. *IEEE Trans. Comput.* **2015**, doi:10.1109/TC.2015.2417543.
22. Busch, C.; Kannan, R.; Vasilakos, A. Approximating Congestion + Dilation in Networks via “Quality of Routing” Games. *IEEE Trans. Comput.* **2012**, *61*, 1270–1283.
23. Li, P.; Guo, S.; Yu, S.; Vasilakos, A. Reliable Multicast with Pipelined Network Coding Using Opportunistic Feeding and Routing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 3264–3273.
24. Yen, Y.S.; Chao, H.C.; Chang, R.S. Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs. *Math. Comput. Model.* **2011**, *53*, 2238–2250.
25. Xing, G.; Li, M.; Luo, H.; Jia, X. Dynamic Multiresolution Data Dissemination in Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2009**, *8*, 1205–1220.
26. Wieselthier, J.; Nguyen, G.; Ephremides, A. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In Proceedings of the 19th IEEE International Conference on Computer Communications (IEEE INFOCOM), Tel Aviv, Israel, 26–30 March 2000; pp. 585–594.
27. Wan, P.J.; Calinescu, G.; Yi, C.W. Minimum-power multicast routing in static ad hoc wireless networks. *IEEE/ACM Trans. Netw.* **2004**, *12*, 507–514.
28. Cagalj, M.; Phubaux, J.; Enz, C. Minimum-Energy Broadcast in All-Wireless Networks: NP-Completeness and Distribution Issues. In Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MOBICOM), Atlanta, GA, USA, 23–28 September 2002.
29. Liang, W. Approximate minimum-energy multicasting in wireless ad hoc networks. *IEEE Trans. Mob. Comput.* **2006**, *5*, 377–387.
30. Liang, W.; Brent, R.; Xu, Y.; Wang, Q. Minimum-energy all-to-all multicasting in wireless ad hoc networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 5490–5499.
31. Qiu, C.; Shen, H.; Yu, L. Energy-efficient cooperative broadcast in fading wireless networks. In Proceedings of the 33rd IEEE International Conference on Computer Communications (IEEE INFOCOM), Toronto, ON, Canada, 27 April–2 May 2014; pp. 1114–1122.
32. Baghaie, M.; Krishnamachari, B. Delay constrained minimum energy broadcast in cooperative wireless networks. In Proceedings of the 30th IEEE International Conference on Computer Communications (IEEE INFOCOM), Shanghai, China, 10–15 April 2011.
33. Lai, S.; Ravindran, B. On multihop broadcast over adaptively duty-cycled wireless sensor network. In Proceedings of the 6th IEEE International on Distributed Computing in Sensor Systems (DCOSS), Santa Barbara, CA, USA, 21–23 June 2010; pp. 158–171.
34. Hong, J.; Cao, J.; Li, W.; Lu, S.; Chen, D. Minimum-transmission broadcast in uncoordinated duty-cycled wireless ad hoc networks. *IEEE Trans. Veh. Technol.* **2010**, *59*, 307–318.
35. Su, L.; Ding, B.; Yang, Y.; Abdelzaher, T.F.; Cao, G.; Hou, J.C. Ocast: Optimal multicast routing protocol for wireless sensor networks. In Proceedings of the 17th annual IEEE International Conference on Network Protocols (ICNP), Princeton, NJ, USA, 13–16 October 2009; pp. 151–160.
36. Han, K.; Liu, Y.; Luo, J. Duty-Cycle-Aware Minimum-Energy Multicasting in Wireless Sensor Networks. *IEEE Trans. Netw.* **2013**, *21*, 910–923.
37. Gerharz, M.; de Waal, C.; Martini, P.; James, P. A Cooperative Nearest Neighbours Topology Control Algorithm for Wireless Ad Hoc Networks. In Proceedings of the 12th International Conference on Computer Communications and Networks, Dallas, TX, USA, 20–22 October 2003; pp. 412–417.

38. Klein, P.N.; Ravi, R. A nearly best-possible approximation algorithm for node-weighted Steiner tree. *J. Algorithms* **1995**, *19*, 104–114.
39. Liu, L.; Song, Y.; Zhang, H.; Ma, H. Physarum Optimization: A Biology-Inspired Algorithm for the Steiner Tree Problem in Networks. *IEEE Trans. Comput.* **2013**, *64*, 818–831.
40. Chipcon, CC2420 Data Sheet. Available online: <http://www.ti.com/> (accessed on 6 September 2015).
41. Hsin, C.F.; Liu, M. Randomly duty-cycled wireless sensor networks: Dynamics of coverage. *IEEE Trans. Wirel. Commun.* **2006**, *5*, 3182–3192.



© 2015 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons by Attribution (CC-BY) license (<http://creativecommons.org/licenses/by/4.0/>).