Kai Han, Yang Liu, and Jun Luo, Member, IEEE, ACM

Abstract—In duty-cycled wireless sensor networks, the nodes switch between active and dormant states, and each node may determine its active/dormant schedule independently. This complicates the Minimum-Energy Multicasting (MEM) problem, which was primarily studied in always-active wireless ad hoc networks. In this paper, we study the duty-cycle-aware MEM problem in wireless sensor networks both for one-to-many multicasting and for all-to-all multicasting. In the case of one-to-many multicasting, we present a formalization of the Minimum-Energy Multicast Tree Construction and Scheduling (MEMTCS) problem. We prove that the MEMTCS problem is NP-hard, and it is unlikely to have an approximation algorithm with a performance ratio of \((1 - o(1)) \ln \Delta\), where \(\Delta\) is the maximum node degree in a network. We propose a polynomial-time approximation algorithm for the MEMTCS problem with a performance ratio of \(O(H(\Delta + 1))\), where \(H(\cdot)\) is the harmonic number. In the case of all-to-all multicasting, we prove that the Minimum-Energy Multicast Backbone Construction and Scheduling (MEMBCS) problem is also NP-hard and present an approximation algorithm for it, which has the same approximation ratio as that of the proposed algorithm for the MEMTCS problem. We also provide a distributed implementation of our algorithms, as well as a simple but efficient collision-free scheduling scheme to avoid packet loss. Finally, we perform extensive simulations, and the results demonstrate that our algorithms significantly outperform other known algorithms in terms of the total transmission energy cost, without sacrificing much of the delay performance.

Index Terms—Approximation algorithm, duty-cycle-aware, minimum-energy, multicasting, wireless sensor networks (WSNs).

I. INTRODUCTION

WIRELESS sensor networks (WSNs) are decentralized systems without any preexisting infrastructures, and the sensor nodes are usually powered by batteries. As the limited battery lifetime imposes a severe constraint on the network performance, it is imperative to develop energy conservation mechanisms for WSNs. One common approach for energy conservation in WSNs is duty-cycling, in which each node switches between active and dormant states, and the active/dormant schedule can vary from node to node [1]–[5]. Duty-cycling is easily implementable and is proven to be an effective way for energy conservation [1]. As a result, duty-cycled wireless sensor networks (DC-WSNs) have been adopted by various applications [6]–[8].

As a crucial component of wireless networking, multicasting has been applied to WSNs in supporting data dissemination for distributed data management (e.g., [9]). Therefore, designing an energy-efficient multicast protocol is of great importance. In an always-active wireless ad hoc network (AA-WANET), the network topology is static, and each forwarding node can cover all its neighboring nodes by only one transmission. Therefore, the main task of the Minimum-Energy Multicasting (MEM) problem in AA-WANETs is to select appropriate forwarding nodes such that a multicast tree with minimum energy cost can be constructed. This problem was proved to be NP-hard, and some approximation algorithms have been proposed [10]–[13].

In DC-WSNs, however, new challenges to the MEM problem arise. More specifically, the network topology is now only intermittently connected, and a forwarding node may need to transmit the same data packet many times to reach its neighboring nodes. Therefore, designing energy-efficient multicasting algorithms in DC-WSNs requires not only that the forwarding nodes should be selected appropriately to construct a multicast tree, but also that the transmissions of each forwarding node need to be scheduled intelligently to cover the receiving nodes with a minimum number of transmissions. More importantly, these two aspects must be handled jointly so that the total energy cost can be reduced to the largest extent. Consequently, the existing solutions for the MEM problem in AA-WANETs are not suitable for DC-WSNs, and we need to design new energy-efficient multicasting algorithms to meet the challenges in DC-WSNs.

A. Background and Motivations

The MEM problem in AA-WANETs has been studied in [10]–[13]. Wan et al. [13] studied the minimum-power multicast routing problem in a scenario where each node can adjust its transmission power continuously, and the communication links can be symmetric or asymmetric. They proposed several centralized approximation algorithms with constant approximation ratios. Liang [11] considered a scenario in which each wireless node can adjust its transmission power in a discrete
fashion and the communication links are symmetric. He proposed a centralized approximate algorithm with performance ratio $4 \ln |M|$ for building a minimum-energy multicasting tree, where $M$ is the set of terminal nodes in a multicast request. Li et al. [10] considered a case in which all nodes have fixed transmission power and the communication links are asymmetric. They converted the minimum-energy multicasting problem to an instance of the Directed Steiner Tree (DST) problem [14] and presented several heuristics. Liang et al. [12] further considered a scenario where the transmission power is either fixed or adjustable. They studied the minimum-energy all-to-all multicasting problem in such a network and tried to build a shared multicast tree such that the total energy consumption of realizing an all-to-all multicast session by the tree is minimized. They proved that finding such a multicast tree is an NP-complete problem and proposed several approximation algorithms for it. All the aforementioned algorithms assume that the network nodes are always-active; they cannot directly apply to DC-WSNs.

Recently, the routing problem in DC-WSNs has started to attract attention from the research community. Su et al. [15] studied the minimum-latency unicasting problem in DC-WSNs and provided some optimal algorithms. Guo et al. [4] considered the effect of unreliable links on broadcasting and proposed an opportunistic forwarding scheme (a heuristic) to reduce the broadcast delay and total broadcast transmissions in DC-WSNs. Hong et al. [3] adopted a restricted duty-cycling model where only one active time-slot exists in the working period of any node, and they proposed several approximation algorithms for the Minimum-Transmission Broadcasting problem in DC-WSNs. However, extending [3] to the multicasting problem in DC-WSNs can be highly nontrivial.

To the best of our knowledge, the only work studying the MEM problem in DC-WSNs is [5]. In [5], the authors adopted a duty-cycling model in which the active time-slots of any node must be consecutive in a round and proposed two optimal algorithms (“oCast” and “DB-oCast”) for the minimum-energy one-to-many multicasting problem in DC-WSNs. Although oCast and DB-oCast were both claimed to be optimal in [5], their time complexity grows exponentially with respect to the number of terminal nodes. Consequently, they both require the number of terminal nodes in a multicast session to be very small, which may not hold in many applications.

B. Our Contributions

In this paper, we study the MEM problem in DC-WSNs using a generic duty-cycling model, where each wireless node determines its active/dormant schedule without any constraints. We formulate the MEM problem for DC-WSNs in the case of both one-to-many multicasting and all-to-all multicasting. We prove the NP-hardness of these two problem instances, and we propose approximation algorithms with guaranteed performance ratios. We also present a distributed implementation of our algorithms. Moreover, we propose a simple but efficient collision-free scheduling scheme on top of a multicast tree to avoid packet loss. Our main contributions are summarized as follows.

1) In one-to-many multicasting, we formulate the Minimum-Energy Multicast Tree Construction and Scheduling (MEMTCS) problem and prove its NP-hardness. We also prove that, unless $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)})$, the MEMTCS problem cannot be approximated within a performance ratio of $(1 - o(1))\Delta$, where $\Delta$ is the maximum node degree of the input network.

2) We propose a polynomial-time approximation algorithm for the MEMTCS problem with an approximation ratio of $6pH(\Delta + 1) + 2\rho$, where $H(\cdot)$ is the harmonic number and $\rho$ is the approximation ratio of a given algorithm for the Minimum Steiner Tree (MST) problem.

3) In all-to-all multicasting, we formulate the Minimum-Energy Multicast Backbone Construction and Scheduling (MEMBCS) problem and prove its NP-hardness. We present an approximation algorithm for the MEMBCS problem, which has the same performance ratio as the proposed algorithm for the MEMTCS problem.

4) We present a distributed implementation of the proposed algorithms, and we conduct extensive simulations to evaluate the performance of our algorithms. The simulation results demonstrate that our algorithms significantly outperform other known algorithms in terms of the total transmission energy cost.

5) We propose a collision-free scheduling scheme on top of a multicast tree (constructed by our algorithm for either MEMTCS or MEMBCS) in DC-WSNs. The simulation results based on this scheme show that the delay performance of our multicast trees is comparable to other proposals in the literature.

To the best of our knowledge, we are the first to present polynomial-time approximation algorithms with provable approximation ratios for the MEM problem in DC-WSNs. Moreover, as the Minimum-Transmission Broadcasting/Gossiping problems can be seen as special cases of our problem, we also provide the first approximation algorithms with provable approximation ratios for these problems in DC-WSNs under a generic duty-cycling model.

The rest of our paper is organized as follows. In Section II, we introduce the wireless network model and formulate both the MEMTCS and the MEMBCS problems. In Section III, we analyze the complexity of MEMTCS, and we propose an approximation algorithm for it. In Section IV, we extend the algorithm for MEMTCS to approximate MEMBCS. A distributed implementation of the proposed algorithms is presented in Section V, and a collision-free scheduling scheme is described in Section VI. In Section VII, we evaluate the performance of the proposed algorithms by simulations. Section VIII concludes the paper. In order to maintain fluency and clarity, we postpone most of the proofs to the Appendixes and list important notations in Table I.

II. ASSUMPTIONS AND DEFINITIONS

In this section, we first describe our network model and related parameters, then we present the formulations of the problems that we tackle in this paper.

A. Network Model and Parameters

A WSN is modeled by an undirected graph $G = (V, E)$, where $V$ is the set of wireless nodes, and $E$ is the set of links.
TABLE I
SYMBOLS AND NOTATIONS

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>The graph representing a WSN</td>
</tr>
<tr>
<td>$V$</td>
<td>Node set of $G$</td>
</tr>
<tr>
<td>$E$</td>
<td>Edge set of $G$</td>
</tr>
<tr>
<td>$K$</td>
<td>Length of the working period of any node in $V$</td>
</tr>
<tr>
<td>$M$</td>
<td>The terminal set</td>
</tr>
<tr>
<td>$s_a$</td>
<td>The source node in one-to-many multicasting</td>
</tr>
<tr>
<td>$e_a$</td>
<td>The energy cost for sending a data packet</td>
</tr>
<tr>
<td>$\Gamma(u)$</td>
<td>Set of active time slots in the working period of $u$</td>
</tr>
<tr>
<td>$n_{hG}(u)$</td>
<td>Set of neighboring nodes of $u$ in $G$</td>
</tr>
<tr>
<td>$nI(T)$</td>
<td>Set of non-leaf nodes in rooted tree $T$</td>
</tr>
<tr>
<td>$N(T)$</td>
<td>Set of nodes in tree $T$</td>
</tr>
<tr>
<td>$E(T)$</td>
<td>Set of edges in tree $T$</td>
</tr>
<tr>
<td>child$(u, T)$</td>
<td>Set of child nodes of $u$ in rooted tree $T$</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Approximation ratio of a given algorithm for the minimum Steiner tree problem</td>
</tr>
<tr>
<td>$H(k)$</td>
<td>The $k$-th Harmonic number</td>
</tr>
<tr>
<td>$T_{opt}$</td>
<td>An optimal multicast tree for a MEMTCS problem</td>
</tr>
<tr>
<td>$B_{opt}$</td>
<td>An optimal feasible schedule for $T_{opt}$</td>
</tr>
<tr>
<td>$\Pi(T, B)$</td>
<td>The total energy cost of an one-to-many multicast session using the multicast tree $T$ and the feasible schedule $B$</td>
</tr>
<tr>
<td>$G$</td>
<td>The extended graph of $G$</td>
</tr>
<tr>
<td>$\tilde{G}$</td>
<td>Node set of $G$</td>
</tr>
<tr>
<td>$\tilde{E}$</td>
<td>Edge set of $G$</td>
</tr>
<tr>
<td>$X(u, i)$</td>
<td>The set of nodes in tree $T$</td>
</tr>
<tr>
<td>$\Psi(u)$</td>
<td>Set of all satellite nodes of $u$</td>
</tr>
<tr>
<td>$V_G$</td>
<td>Set of all satellite nodes in $V$</td>
</tr>
<tr>
<td>$G_i$</td>
<td>The sub graph of $G$ induced by $V_G$</td>
</tr>
<tr>
<td>$\gamma(u, T)$</td>
<td>A minimum hitting set of the collection ${\Gamma(v)\mid v \in n_{hG}(u)}$</td>
</tr>
<tr>
<td>$SB^*$</td>
<td>A minimum satellite bridge</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>The maximum node degree in $G$</td>
</tr>
<tr>
<td>$d^i(T)$</td>
<td>Set of nodes in $T$ with degree $i$</td>
</tr>
<tr>
<td>$d^i(T)$</td>
<td>Set of nodes in $T$ with degree greater than one</td>
</tr>
<tr>
<td>$S(T)$</td>
<td>Summation of $</td>
</tr>
<tr>
<td>$T_i$</td>
<td>A minimum isotropic scattering tree</td>
</tr>
<tr>
<td>$P^{(m)}$</td>
<td>The rooted tree by designating $m$ as the root of tree $P$</td>
</tr>
<tr>
<td>$S^{(m)}$</td>
<td>A feasible schedule for $P^{(m)}$</td>
</tr>
<tr>
<td>$P_A$</td>
<td>An optimal multicast backbone for a MEMBCS problem</td>
</tr>
<tr>
<td>$S_A$</td>
<td>The set ${S^{(m)}\mid m \in M}$</td>
</tr>
<tr>
<td>$\Pi(P_A, S_A)$</td>
<td>The total energy cost of an all-to-all multicast session by using $P_A$ and $S_A$</td>
</tr>
</tbody>
</table>

The nodes in $V$ are distributed in a two-dimensional plane, and each node is equipped with an omnidirectional antenna. All nodes have the same fixed transmission power, and there exists a link between two nodes if they are within the transmission range of each other. We also assume that each node has a unique ID and knows the IDs of its one-hop neighbors.

We assume that time is divided into equal-length slots, and each time-slot is long enough for sending or receiving a data packet. Without loss of generality, we assume that the working schedule of each node is periodic, and the working period of any node has $K$ time-slots. To save energy, every node switches between active and dormant states, and we denote by $\Gamma(u)$ the set of active time slots in the working period of node $u$, i.e., $\Gamma(u) \subseteq \{1, 2, \ldots, K\}$ and $\Gamma(u) \neq \emptyset$. The active/dormant schedule is independently and arbitrarily determined by each node. Our duty-cycling model is similar with the model used in [2] and [4], and the duty-cycling models used in [3] and [5] can be considered as special cases of our model. Following a very common setting in DC-WSNs, we assume that a node can wake up its transceiver to transmit a packet at any time-slot, but can only receive a packet when it is active. We also assume that time synchronization is achieved in network, and each node knows the active/dormant schedule of its neighboring nodes. These are common assumptions in the literature [2]–[5]. Finally, we assume that a packet transmission is always successful unless it collides with other transmission(s).

We denote by $e_a$, the energy cost for sending a data packet by any node, and denote by $e_r$, the energy cost for receiving a data packet. It is well known that a wireless sensor node has different power consumption levels at different working states such as transmitting, receiving, and idle-listening. Let $e_{tx}$, $e_{rx}$, and $e_{il}$ be the energy consumptions of radio for transmitting, receiving, and idle-listening, respectively. Usually, $e_{tx}$ is larger than $e_{rx}$, whereas $e_{il}$ is almost the same as $e_{rx}$ [18]. Since a node has to consume at least $e_{il}$ when it is active, we set $e_a = r_{tx} - e_{il}$, and set $e_r = e_{rx} - e_{il} \approx 0$. Consequently, we neglect $e_{il}$ in our later problem formulation.

For the convenience of description, we clarify some other notations here. For any node $u$, we denote by $n_{hG}(u)$ the set of neighboring nodes of $u$ in $G$. Suppose that $T$ is an arbitrary tree. We denote by $N(T)$ the set of nodes in $T$. Denote by $E(T)$ the set of edges in $T$. Denote by $d^0(T)$ the set of nodes in $T$ with degree one. Denote by $d^*(T)$ the set of nodes in $T$ with degree greater than one. If $T$ is a rooted tree, then we denote by $nI(T)$ the set of nonleaf nodes in $T$, and denote by $child(u, T)$ the set of child nodes of node $u$ in $T$. Suppose that $\tau$ is the root node of $T$. Clearly, if $\tau \in d^0(T)$, then $nI(T) = d^*(T) \cup \{\tau\}$; otherwise $nI(T) = d^*(T)$.

B. Basic Concepts and Problem Formulations

We first make a few crucial definitions for DC-WSNs, then we formally define our problems.

1) Concepts for Duty-Cycle-Aware Multicasting: In a one-to-many multicast session, there exist a terminal set $M \subseteq V$ and a source node $s \in M$, such that the data sent by $s$ should be received by all the nodes in $M - \{s\}$. A multicast tree $T$ is a subtree of $G$ rooted at $s$, and each terminal node in $M$ is a tree node of $T$.

Since all nodes have the same transmission power, the energy cost for sending a data packet using a multicast tree $T$ can be determined by the total number of transmissions of the nodes in $nI(T)$. In DC-WSNs, the time-slot at which a node transmits the packet decides which neighbors can receive it. Therefore, the transmission schedule of each forwarding node plays a key role on the energy cost for multicasting. Furthermore, incorrect schedules can even prevent certain terminal nodes from receiving a data packet.

Consider the WSN shown in Fig. 1(a) as an example. Suppose that the terminal set is $M = \{n_1, n_6, n_7, n_8\}$, and node $n_1$ is the source node. The set of numbers associated with each node indicates the active time slots in the working period of that node. Fig. 1(b) and (c) are two possible multicast trees $T_1$ and $T_2$, respectively. If node $n_2$ in $T_1$ transmits at time-slot 3 or 4, then $\ldots$
only one child node \( (n_6 \text{ or } n_4) \) can receive the data. Therefore, a correct transmission schedule of node \( n_2 \) for \( T_1 \) must be a set of time-slots that contains \( \{3, 4\} \).

From this observation, we introduce the concept of “feasible schedule.” Actually, finding a feasible schedule of any nonleaf node in a multicast tree equals finding a “hitting set” \([19]\). We clarify this by Definitions 1 and 2.

**Definition 1 (Hitting Set \([19]\)):** Given a collection \( \mathcal{C} \) of subsets of a finite set \( \mathcal{F} \), a hitting set is a subset \( \mathcal{F}' \subseteq \mathcal{F} \) such that \( \mathcal{F}' \) contains at least one element from each subset in the collection \( \mathcal{C} \).

**Definition 2 (Feasible Schedule):** Given a multicast tree \( T \) in \( G \), a function \( B : \text{nl}(T) \rightarrow 2^{\{1, 2, \ldots, K\}} \) is called a feasible schedule for \( T \) if and only if for any node \( u \in \text{nl}(T) \), \( B(u) \) is a hitting set of the collection \( \{T[v] : v \in \text{child}(u, T)\} \).

In other words, a feasible schedule for \( T \) is such that each nonleaf node \( u \in \text{nl}(T) \) has to transmit at a set of time-slots to reach all its children, and this time-slot set is denoted by \( B(u) \).

According to Definition 2, the energy cost for sending a data packet using a feasible schedule \( B \) on multicast tree \( T \) can be written as \( \sum_{u \in \text{nl}(T)} |B(u)| \cdot \epsilon_s \).

2) **Two Minimum-Energy Multicasting Problems:** Based on the above discussions, we first introduce the MEMTCS problem in Definition 3:

**Definition 3 (MEMTCS):** Given a WSN \( G \), \( M \subseteq V \), and a source node \( s \in M \), the MEMTCS problem seeks a 2-tuple \( \{T_{\text{opt}}, B_{\text{opt}}\} \) in which \( T_{\text{opt}} \) is a multicast tree rooted at \( s \) and \( B_{\text{opt}} \) is a feasible schedule for \( T_{\text{opt}} \), such that the total energy cost

\[
\Pi(T_{\text{opt}}, B_{\text{opt}}) = \sum_{u \in \text{nl}(T_{\text{opt}})} |B_{\text{opt}}(u)| \cdot \epsilon_s
\]

is minimized.

For example, we can find two feasible schedules \( B_1 \) and \( B_2 \) for \( T_1 \) and \( T_2 \) in Fig. 1, respectively, where \( B_1(n_1) = \{1\} \), \( B_1(n_2) = \{3, 4\} \), \( B_1(n_4) = \{2, 3\} \), and \( B_2(n_1) = B_2(n_3) = \{3\} \), \( B_2(n_5) = \{2\} \). Assuming \( \epsilon_s = 1 \), we have \( \Pi(T_1, B_1) = 5 \) and \( \Pi(T_2, B_2) = 4 \). Actually, \( \{T_2, B_2\} \) is an optimal solution for \( n_1 \) to send data to \( \{n_6, n_7, n_8\} \) in Fig. 1.

Besides one-to-many multicasting, all-to-all multicasting is also a fundamental operation in WSNs. In an all-to-all multicast session, each node in \( M \) serves as both a source node and as a receiver node and must send data to all other nodes in \( M \). A naive approach is to build a one-to-many multicast tree for each source node in \( M \) and find a feasible schedule for each multicast tree. As a result, there will be \( |M| \) multicast trees. However, the cost of maintaining so many multicast trees will be high, and shifting from one multicast tree to the other will cause delay and energy overhead \([12]\). Therefore, an alternative way is to construct a shared multicast tree serving as a multicast backbone and find \( |M| \) feasible schedules based on the multicast backbone. Similar to the one-to-many multicasting case, the energy cost of an all-to-all multicast session is determined by the transmission schedules of the nodes in the multicast backbone. Based on these considerations, we introduce the MEMBCS problem, as described in Definition 4.

**Definition 4 (MEMBCS):** Given a WSN \( G \) and a terminal set \( M \), a multicast backbone \( P \) is an unrooted subtree of \( G \) spanning all the nodes in \( M \). If we designate a node \( v \in M \) as the source node and all other nodes in \( M \) as receiver nodes, then \( P \) becomes a rooted multicast tree \( P(v) \), in which the edges have a natural orientation away from the root \( v \). The MEMBCS problem seeks a multicast backbone \( P_A \) and a set of transmission schedules \( S_A = \{S_A^{(m)} : m \in M\} \), such that we have the following.

1) Each transmission schedule \( S_A^{(m)} \in S_A \) is a feasible schedule for \( P_A^{(m)} \).
2) The total energy cost

\[
\Pi(P_A, S_A) = \sum_{m \in M} \sum_{u \in \text{nl}(P_A^{(m)})} |S_A^{(m)}(u)| \cdot \epsilon_s
\]

is minimized.

**III. MEMTCS PROBLEM**

We first briefly evaluate the hardness of MEMTCS. We prove it is NP-hard using a reduction from the Minimum Hitting Set (MHS) problem \([19]\), and we claim this in Theorem 1.

**Theorem 1:** The MEMTCS problem is NP-hard.

The MHS problem was proved to be equivalent to the Minimum Set Cover (MSC) problem \([19], [20]\). Moreover, Fiege \([21]\) has proved that, unless NP has quasi-polynomial-time algorithms, there exists no polynomial-time algorithm for the MSC problem with performance ratio of \((1 - o(1))\ln n\), where \( n \) is the size of the MSC problem. Therefore, with the proof of Theorem 1, we can easily get the following corollary.

**Corollary 1:** Unless \( \text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)}) \), there exists no polynomial-time approximation algorithm with performance ratio of \((1 - o(1))\ln \Delta\) for the MEMTCS problem, where \( \Delta \) is the maximum node degree of a WSN.

Next, we propose an approximation algorithm for the MEMTCS problem. We first provide a brief overview of our algorithm in Section III-A, then describe our methods in details from Sections III-B–III-E.

**A. Overview of the Proposed Algorithm**

Our approximation algorithm consists of several steps. First, we use a graph transformation method to extend the original network graph \( G \) into \( \hat{G} \), where the possible transmitting time-slots of the nodes in \( G \) are represented as satellite nodes, and the nodes in \( \hat{G} \) are connected in a particular way to facilitate the design of our approximation algorithm (Section III-B). Second, we propose the concept of Minimum Satellite Bridge (MSB) in \( \hat{G} \) as well as an algorithm for finding an approximation MSB. The MSB is actually a special tree in \( \hat{G} \) whose nodes can cover...
all the nodes in $M$ (Section III-C). Finally, we map the approximation for MSB to a multicast tree in $G$ and a feasible schedule for the multicast tree (Section III-E). This resulting multicast tree along with its schedule serves as an approximate solution to the MEMTCS problem.

To find the approximation ratio of our algorithm, we propose another concept, Minimum Isotropic Scattering Tree (MIST); it is a special multicast tree $T_f$ in $G$ spanning the nodes in $M$ (Section III-D). We prove that $T_f$ serves as a quantitative “bridge” between the number of nodes in an MSB and $(T_{opt}, B_{opt})$. As a result, we obtain the approximation ratio of our algorithm.

B. Graph Transformation

The first step of our approach is to transform the original network graph into an extended graph, whose definition is shown in Definition 5. Note that the work in [5] has also provided a graph transformation method, and one can use it to convert the MEM problem into an instance of the DST problem. Unfortunately, the best-known approximation ratio of any polynomial-time algorithm for the DST problem is only linear [14]. In contrast, employing our new graph transformation method proposed in Definition 5 enables us to apply an MST algorithm and hence leads to a much better logarithmic approximation ratio for the MEM problem.

Definition 5 (Extended Graph): The extended graph of $G$ is an undirected graph $\tilde{G} = (\tilde{V}, \tilde{E})$, where $\tilde{V}$ is the set of nodes and $\tilde{E}$ is the set of edges. The nodes and edges in $\tilde{G}$ are created by the following steps.

i) Initially, $V = V$ and $\tilde{E} = \emptyset$.

ii) For each node $u \in V$ and each time-slot $i \in \bigcup_{v \in nb_G(u)} \Gamma(v)$, create a new node $\lambda(u, i)$ in $\tilde{V}$. The node $\lambda(u, i)$ is called a satellite node of $u$ on slot $i$, and $u$ is called a nuclear node of $\lambda(u, i)$. The set of all satellite nodes of $u$ is denoted by $\Psi(u)$.

iii) For each node $u \in V$, create an undirected edge between each pair of nodes in $\Psi(u) \cup \{u\}$. In other words, the subgraph induced by $\Psi(u) \cup \{u\}$ is a complete graph.

iv) For each edge $(u, v) \in E$, each time-slot $i \in \Gamma(v)$, and each time-slot $j \in \Gamma(u)$, create three undirected edges $(\lambda(u, i), \lambda(v, j)), (\lambda(u, i), v)$, and $(u, \lambda(v, j))$.

From Definition 5, we can see that $\tilde{V}$ can be partitioned into two disjoint subsets: $V$ and $V_S$, where $V$ is the set of all nuclear nodes (or the original nodes of $G$), and $V_S$ is the set of all satellite nodes. A nuclear node may have multiple satellite nodes, but any satellite node only has one nuclear node. The satellite nodes of any nuclear node $u$ actually correspond to the possible transmitting time-slots of $u$ that can cover $u$’s neighbors. The linking structure of the extended graph is designed purposely to facilitate our algorithm design, and we will further elaborate this in the following sections. An example of the extended graph is shown in Fig. 2.

According to the construction rules in Definition 5, we have some properties of $\tilde{G}$ useful to later complexity and approximation analysis, as described in Lemmas 1 and 2.

Lemma 1: There are at most $(K + 1)|V|$ nodes and $\binom{K + 1}{2}|V| + 3K^2|E|$ edges in $\tilde{G}$.

Algorithm 1: Finding an approximate MSB

<table>
<thead>
<tr>
<th>Input</th>
<th>The extended graph $\tilde{G}$ and the terminal set $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>An approximate MSB $SB$</td>
</tr>
</tbody>
</table>

1. $C \leftarrow \emptyset$, $UC \leftarrow M$

2. while $UC \neq \emptyset$ do

3. $v \leftarrow \arg \max_{v \in V_S \setminus C} |nb_{\tilde{G}}(v) \cap UC|$

4. $C \leftarrow C \cup \{v\}$; $UC \leftarrow UC - (nb_{\tilde{G}}(v) \cap UC)$

5. Let $G_x$ be the sub-graph of $\tilde{G}$ induced by $V_S$. Assign each edge in $G_x$ a weight of 1. Compute an approximate minimum Steiner tree $ST$ in $G_x$ which connects the nodes in $C$; $SB \leftarrow ST$

We can see that Algorithm 1 consists of two stages. The first stage is lines 1–4, and the second stage is line 5. In the first stage, we use a greedy set cover algorithm to find a small node set $C$ that can cover all the nodes in $M$. In each loop, we first find a node $v$ that has the maximum number of adjacent nodes in the uncovered node set $UC$ (line 3). Then, we add $v$ into $C$ and update $UC$ (line 4). In the second stage, an approximate Steiner tree algorithm is applied upon $C$.

Since each satellite node can cover at most $\Delta + 1$ nodes in $M$ (according to Lemma 2), the first stage of Algorithm 1 has an approximation ratio of $H(\Delta + 1)$ [22]. The approximation ratio of the second stage is a constant $\rho$. By taking advantage of the special structure of the extended graph, we can get Theorem 2.
The approximation ratio of Algorithm 1 is $3pH(\Delta + 1) + \rho$, where $H(\cdot)$ is the harmonic number and $\rho$ is the approximation ratio of the algorithm used in Algorithm 1 for finding a minimum Steiner tree.

D. Minimum Isotropic Scattering Tree

Now we link an MSB to a special tree in $G$ called the MIST and a feasible schedule for the internal nodes in MIST. Although an approximate MSB will be involved to construct an approximation to MEMTCS in Section III-E, the quantitative relation between an MSB and an optimal solution to MEMTCS is not straightforward. Therefore, we use MIST as a medium to derive the approximation ratio. We first introduce the concept of MIST in Definition 7.

Definition 7 (Minimum Isotropic Scattering Tree): For any tree $T$ in $G$ and any node $u \in d^+(T)$, we denote by $\mathcal{Y}(u, T)$ a minimum hitting set of the collection $\{\Gamma(u)\mid u \in \text{nb}_T(u)\}$, and define $\Xi(T) = \sum_{u \in d^+(T)} |\mathcal{Y}(u, T)|$. The MIST $T_T$ is a tree in $G$ such that $T_T$ spans $M$ and that $\Xi(T_T)$ is minimized.

We claim that $\Xi(T_T)$ actually equals the number of nodes in a minimum satellite bridge, which is proved by Lemmas 3 and 4 and Theorem 3. Lemma 3 claims that the number of nodes in an MSB is no more than $\Xi(T_T)$, whereas Lemma 4 implies that $\Xi(T_T)$ is no more than the number of nodes in an MSB. With these two lemmas, Theorem 3 follows.

Lemma 3: $|N(SB^*)| \leq \Xi(T_T)$.

Lemma 4: Any satellite bridge $SB$ can be mapped to a 2-tuple $\{R, F\}$, where $R$ is a tree in $G$ spanning the nodes in $M$, and $F$ is a function that satisfies the following.

1) For any $u \in d^+(R)$, $F(u)$ is a hitting set of the collection $\{\Gamma(u)\mid u \in \text{nb}_R(u)\}$.

2) $\sum_{u \in d^+(R)} |F(u)| \leq |N(SB^*)|$.

Theorem 3: $\Xi(T_T) = |N(SB^*)|$.

The proof of Lemma 4 actually provides a method of mapping any satellite bridge to a tree that spans $M$ and a feasible schedule for the internal nodes in this tree. This mapping procedure and its outcome can be roughly described as follows. According to the construction rules of the extended graph, the satellite nodes in a satellite bridge can be mapped to the nuclear nodes that they belong to, as well as the transmitting time-slots on these nuclear nodes. Furthermore, we can find a tree spanning these mapped nuclear nodes and the terminal nodes in $M$, and most importantly, the internal nodes in this tree are all the mapped nuclear nodes. According to the special node-connecting method of the extended graph, the mapped transmitting time-slots of any internal node in this tree can cover all its neighboring nodes in the tree.

For example, we can find a satellite bridge containing two satellite nodes $\lambda(b, 1)$ and $\lambda(b, 3)$ in the extended graph shown in Fig. 2. These two satellite nodes both belong to the same nuclear node $b$, and they dominate two other nuclear nodes $a$ and $c$. According to the mapping method of Lemma 4, this satellite bridge can be mapped to the tree shown in the left of Fig. 2, and the nodes $\lambda(b, 1)$ and $\lambda(b, 3)$ are mapped to the transmitting time-slots $\{1, 3\}$ of node $b$. Clearly, these two time-slots can cover the neighboring nodes of $b$: $a$ and $c$.

E. Approximation Algorithm for MEMTCS

Based on the methods introduced by the previous sections, we propose our algorithm for the MEMTCS problem, as shown in Algorithm 2.

Algorithm 2: Approximation for MEMTCS

Input: A DC-WSN $G$, a terminal set $M$, and a source node $s \in M$

Input: A multicast tree $\overline{T}$ and a feasible schedule $\overline{B}$

1. Construct the extended graph $G = (\overline{V}, \overline{E})$ of $G$

2. Use Algorithm 1 to compute an approximate minimum satellite bridge $SB$

3. Use the method in the proof of Lemma 4 to map $SB$ to a 2-tuple $\{\overline{T}, \overline{F}\}$. Let $\overline{T}$ be the rooted tree resulting from designating $s$ as the root of $\overline{T}$

4. For each node $u \in \text{nl}(\overline{T})$

5. If $u \in d^+(\overline{T})$

6. Prune the time slots in $\overline{F}(u)$ that do not cover any child nodes of $u$ in $\overline{T}$; $\overline{B}(u) \leftarrow \overline{F}(u)$

7. Else

8. Let $v$ be $u$’s child node in $\overline{T}$. Find an arbitrary $i \in \Gamma(v)$; $\overline{B}(u) \leftarrow \{i\}$

The output of Algorithm 2 is a 2-tuple $\{\overline{T}, \overline{B}\}$. From Lemma 4, it is easy to know that $\overline{T}$ is a multicast tree spanning the nodes in $M$, and $\overline{B}$ is a feasible schedule for $\overline{T}$. Next, we prove the approximation ratio of Algorithm 2 by Lemmas 5 and 6 and Theorem 4. Lemma 5 is actually based on a special property of the MHS problem, i.e., if we add a subset in an instance of the MHS problem, then the cardinality of the result minimum hitting set will increase at most 1. Using Lemma 5, Lemma 6 finds out a quantitative relationship between $\Xi(T_T)$ and $\{T_{opt}, B_{opt}\}$, which is used in the proof of Theorem 4.

Lemma 5: For any node $u \in d^+(T_{opt})$, we have $|\mathcal{Y}(u, T_{opt})| \leq |B_{opt}(u) + 1$.

Lemma 6: $\{T_{opt}, B_{opt}\}$ is related to $T_T$ by $\sum_{u \in \text{nl}(T_{opt})} |B_{opt}(u)| \geq |\Xi(T_T) - |d^+(T_{opt})|| + 1$.

Theorem 4: The approximation ratio of Algorithm 2 is $6pH(\Delta + 1) + 2p$.

For example, if node $a$ is the source node and node $c$ is the receiver node in Fig. 2, then we can find a multicast tree shown in the left of Fig. 2 and a feasible schedule $\overline{B}$ such that $B(a) = \{4\}$ and $B(b) = \{3\}$ using Algorithm 2. In fact, Algorithm 2 first finds the multicast tree and the transmitting time-slots $\{1, 3\}$ of node $b$ (lines 1–3), then adds another transmitting time-slot $\{4\}$ for node $a$ since $a$ only has degree 1 (line 8). Finally, the transmitting time-slot $\{1\}$ of node $b$ is eliminated by a trivial pruning process since it does not cover any child nodes in the rooted multicast tree (line 6).

The dominating running time of Algorithm 2 is the time on constructing $SB$ in line 2, using Algorithm 1. Lines 1–4 in Algorithm 1 can be implemented in $O(\sqrt{V})$ time. If we use the 2-approximation algorithm proposed in [23] to compute an approximate minimum Steiner tree in line 5, the resulting time complexity is $O(\sqrt{V} \log |\overline{V}| + |\overline{F}|)$. Given $K$ as a predefined
constant, the time complexity of Algorithm 2 is $O(|V|^2)$, and
the approximation ratio is $12H(D + 1) + 4$.

IV. MEMBCS PROBLEM

Just as the MEMTCS problem, the MEMBCS problem is also
NP-hard. We claim this in Theorem 5. The NP-hardness of the
MEMBCS problem can be proved by using a reduction from the
Maximum Leaf Spanning Tree (MLST) problem [19].

**Theorem 5:** The MEMBCS problem is NP-hard.

Next, we provide an approximation algorithm for the
MEMBCS problem, as shown in Algorithm 3. We can see that
Algorithm 3 is actually adapted from Algorithm 2 and has the
same time complexity as Algorithm 2. Note that the transmission
schedule of any internal node in $P$ is actually obtained by
using the mapping method provided in the proof of Lemma 4.
Therefore, according to Lemma 4, it can be easily known that
the output of Algorithm 3 is a correct solution to the MEMBCS
problem. Surprisingly, the approximation ratio of Algorithm 3
is the same as that of Algorithm 2, as we state in Theorem 6.

**Algorithm 3:** Approximation for the MEMBCS problem

**Input:** A DC-WSN $G$ and a terminal set $M$.

**Output:** A multicast backbone $P$ and a set of feasible
schedules for the rooted trees in $\{P^m; m \in M\}$.

1. Find an arbitrary node $v$ in $M$.
2. Call Algorithm 2 to get a multicast tree $T$ rooted at $v$
   and a feasible schedule $\mathcal{B}$ for $T$.
3. Let $P$ be the unrooted tree that has the same edges and
   nodes as $T$.
4. For any node $m \in M$ and any node $u \in d^+(P)$, let
   $S^m(u) = \mathcal{B}(u)$.
5. For any node $w \in M \cap d^+(P)$, find an arbitrary active
time-slot $i$ of the neighboring node of $w$ in $P$, and let
   $S^w(i) = \{i\}$.

**Theorem 6:** The approximation ratio of Algorithm 3 is also $6\rho H(D + 1) + 2\rho$.

**Proof:** Note that $S^m_A(m)$ is a minimum hitting set of the
collection $\{\Gamma(v) | v \in \text{child}(u, P^m)\}$, for all $m \in M$ and $\forall u \in \text{n}(P^m)$. Therefore, similar to the proof of Lemma 6, we have:

\[ \Pi(P_A, S_A) = \sum_{m \in M} \sum_{u \in \text{n}(P^m)} |S^m_A(u)| \cdot e_n \]
\[ \geq |M| \cdot (\Xi(T_I) - d^+(P_A) + 1) \cdot e_n. \]

Since each node in $d^+(P_A)$ should transmit at least $M$ times
(once for each source node in $M$), we have

\[ \Pi(P_A, S_A) \geq |M| \cdot (\Xi(T_I) + 1) \cdot e_n. \]

Hence, we have

\[ 2 \cdot \Pi(P_A, S_A) \geq |M| \cdot (\Xi(T_I) + 1) \cdot e_n. \]

On the other hand, we have

\[ \sum_{u \in n(P^m)} |S^m(u)| \leq 1 + \sum_{u \in d^+(P)} |S^m(u)| \]
\[ = 1 + \sum_{u \in d^+(P)} \mathcal{B}(u) \]
\[ \leq \alpha \Xi(T_I) + 1 \]

where $\alpha = 3\rho H(D + 1) + \rho$. Therefore

\[ \Pi(P, \{S^m | m \in M\}) = \sum_{m \in M} \sum_{u \in \text{n}(P^m)} |S^m(u)| \cdot e_n \]
\[ \leq \sum_{m \in M} (\alpha \Xi(T_I) + 1) \cdot e_n \]
\[ \leq \alpha \cdot |M| \cdot (\Xi(T_I) + 1) \cdot e_n \]

hence the approximation ratio $6\rho H(D + 1) + 2\rho$. □

V. DISTRIBUTED IMPLEMENTATION

In this section, we provide a distributed implementation of our algorithms. First, we propose the distributed implementation of Algorithm 2 for the MEMTCS problem. As the main operation of Algorithm 2 is line 2, where Algorithm 1 is called to compute an approximate minimum satellite bridge, we start by a distributed implementation of Algorithm 1.

As we have described in Section III, Algorithm 1 consists of two stages: The first stage is lines 1–4, in which a greedy strategy is used to find a small satellite node set $C$ covering the nodes in $M$, and the second stage is line 5, in which an approximate minimum Steiner tree is computed. The first stage of Algorithm 1 can be decentralized using Algorithm 4.

**Algorithm 4:** Distributed implementation of the first stage of Algorithm 1

1. Each white node $u$ with nonempty $\text{route}(u)$ broadcasts a
   message (“election”, |$\text{route}(u)$|, u.ID).
2. Each red node checks all the “election” messages it
   received, and finds a node $v$ whose value of $\text{route}(v)$ is
   the maximum (break tie by choosing the node with largest
   ID). Then sends $v$ a message “you win.”
3. If a white node $u$ receives “you win” messages from all
   nodes in $\text{route}(u)$, then it colors itself blue, and broadcasts
   a message “I am dominator.”
4. If a red node receives an “I am dominator” message, then
   it colors itself green, and broadcasts a message “I am
   dominated.”
5. If a white node $u$ receives an “I am dominated” message
   from a neighboring node $v$, then it deletes $v$ from $\text{route}(u)$.

In Algorithm 4, each node in $\tilde{G}$ is colored red, green, white, or
blue. The red nodes are the nodes in $M$ that are not covered yet.
The green nodes are the nodes in $M$ that are already covered by
some blue nodes. The blue nodes are the nodes that are selected
into the resulting node set $C$, and the white nodes are the nodes
that are not selected. Initially, each node in $M$ is colored red, and
all other nodes are colored white. Moreover, each white node $u$
owns a set \( \text{nb}(u) \), which is initialized to be the set of IDs of all \( u \)’s red neighbors.

It can be seen that Algorithm 4 is a faithful implementation of the greedy strategy in lines 1–4 of Algorithm 1, so the approximation ratio of Algorithm 4 is \( H(\Delta + 1) \). There are at most \( |M| \) rounds in Algorithm 4 because it repeats until no red nodes exist, and at least one red node turns green in each round. Therefore, the message complexity of Algorithm 4 is \( O(|M| \cdot V) \) and the time complexity is \( O(|M| \cdot D) \), where \( D \) is the diameter of \( G \).

The second stage of Algorithm 1 can be decentralized by using a distributed Steiner tree algorithm in the literature [24], [25]. If we adopt the 2-approximation distributed algorithm proposed in [24], then the message complexity is \( O(|M| \cdot |V|) \) and the time complexity is \( O(|M| \cdot D) \), where \( D \) is the diameter of \( G \).

The distributed implementation of Algorithm 2 (except line 2) is trivial: An arbitrary spanning tree of the subgraph induced by \( N(SB) \) needs be found in line 3. To accomplish this, the distributed Depth-First Search (DFS) algorithm proposed by Makki et al. [26] can be applied. The time complexity and message complexity of the distributed DFS algorithm are both \( O(|V|) \). Based on these discussions, we can get Corollary 2.

**Corollary 2:** There exists a distributed algorithm for MEMTCS. It has an approximation ratio of \( 24H(\Delta + 1) + 8 \).

The time complexity and message complexity of the distributed algorithm are \( O(D \cdot |V|) \) and \( O(|M| \cdot |V|) \), respectively, where \( D \) is the diameter of \( G \).

Though our distributed implementation for Algorithm 2 is based on the extended graph \( \mathcal{G} \), it can be easily adapted to run on \( G \). According to the construction method of \( G \), any satellite node in \( V_S \) can be seen as a local “pseudo node” governed by its nuclear node in \( V \). Therefore, each nuclear node can send messages for its satellite nodes and do the computation that its satellite nodes need to do. Let \( \mathcal{A} \) be the distributed algorithm running on \( G \), and let \( \mathcal{A} \) be the adapted version of \( \mathcal{A} \) running on \( G \). If several satellite nodes of the same nuclear node \( u \) send their messages simultaneously to other nuclear nodes in \( \mathcal{A} \), then \( u \) in \( A \) can send these messages at different time-slots. Note that any nuclear node has at most \( K \) satellite nodes and \( K \) is a pre-defined constant. Therefore, if we define a constant \( \mathcal{E} \) \((\mathcal{E} > K)\), then any nuclear node \( u \) in \( \mathcal{A} \) can receive messages for all its satellite nodes after waiting \( \mathcal{E} \) time, and then \( u \) can do the computation for its satellite nodes based on the messages it received. Note that the message passing processes between the satellite nodes governed by the same nuclear node in \( \mathcal{A} \) become local computations of that nuclear node in \( \mathcal{A} \), and no new messages are generated in \( \mathcal{A} \). Therefore, \( \mathcal{A} \) has the same time and message complexity with \( \mathcal{A} \).

The major operation of Algorithm 3 for solving the MEMBPCS problem involves in invoking Algorithm 2. Therefore, Algorithm 3 can be decentralized in a way similar as that of Algorithm 2. The approximation ratio, time complexity, and message complexity of the resulting distributed implementation are the same as those claimed in Corollary 2.

### VI. Collision-Free Scheduling

In this section, we propose a collision-free scheduling scheme based on the multicast trees we built in Sections III and IV; it guarantees collision-free data transmissions in one-to-all and all-to-all multicasting without compromising the energy efficiency. This scheme aims only to avoid collision loss during data transmissions, rather than to minimize the multicast delay. We use this scheme to compare our algorithm to others in terms of the delay performance. The overall idea of the scheduling scheme is based on graph-coloring, i.e., if we assign different colors to the conflicting transmissions and then schedule them accordingly, no collisions would occur. We hereby describe our scheduling scheme in detail.

Let \( (\mathcal{T}, \mathcal{B}) \) be the output of Algorithm 2. For any node \( u \in \mathcal{T} \) and any time-slot \( i \in \mathcal{B}(u) \), we use \( \text{tr}(u, i) \) to denote a transmission of node \( u \) on time-slot \( i \), and use \( C_u(i) \) to denote the possible receiving nodes of \( \text{tr}(u, i) \) on \( \mathcal{T} \), i.e., \( C_u(i) = \{ v | v \in \text{nb}_G \{u\} \cap N(\mathcal{T}) \land i \in \mathcal{T}(i) \} \). Since transmitting on different time-slots would never cause a collision, two transmissions \( \text{tr}(u, i) \) and \( \text{tr}(v, j) \) collide iff \( i = j \) and they interfere with each other according to certain interference models. For brevity, we consider only a graph-based interference model in our paper, i.e., \( \text{tr}(u, i) \) and \( \text{tr}(v, j) \) satisfy any of the following two conditions: 1) \( u \in C_u(i) \) or \( v \in C_v(j) \); and 2) \( C_u(i) \cap C_v(j) \neq \emptyset \). The former condition is called the primary interference, and the latter condition is called the secondary interference.

Based on the interference conditions, we design a coloring process as follows. For each time-slot \( 1 \leq i \leq K \), we create a conflict graph where each node corresponds to one transmission \( \text{tr}(u, i) \) in \( \mathcal{T} \), and two nodes are adjacent iff their corresponding transmissions collide. Then, we employ a traditional node-coloring algorithm (e.g., [27]) to color the conflict graph, hence find the total color number \( \chi \), and the color \( x_u(i) \) (a positive integer no more than \( \chi \)) of any transmission \( \text{tr}(u, i) \). Upon completing the coloring process, we can schedule the multicast transmissions by forcing each tree node to transmit only at the time-slots corresponding to its assigned colors, hence the conflicting transmissions are scheduled in different working periods. More specifically, a multicast tree node \( u \in \mathcal{T} \) is allowed to transmit only at the time-slots in \( \{ i | x_u(i) - 1 \cdot K + i \in \mathcal{B}(u), i \in \{0, 1, 2, \ldots \} \} \), and it picks an earliest allowed time-slot to transmit when it has some packets to forward.

It is straightforward to see that no collisions would occur by using the scheduling scheme described above. Moreover, the total energy cost \( \Pi(\mathcal{T}, \mathcal{B}) \) of the one-to-all multicasting using \( (\mathcal{T}, \mathcal{B}) \) is not affected by the collision-free scheduling. We omit the scheduling of all-to-all multicasting for brevity; it follows directly from the aforementioned scheme with only minor adaptations. Our collision-free scheduling scheme can be implemented in a distributed manner by adapting existing distributed Distance-2 coloring algorithms for WANETs (e.g., [28]). Moreover, the scheduling scheme can also be extended to accommodate more complex interference models by adapting the corresponding (distributed) coloring algorithms under those models [e.g., [29] for the signal-to-interference-plus-noise ratio (SINR) model].

### VII. Performance Evaluation

In this section, we evaluate the performance of our algorithms via simulations. Our simulations focus on the effect
of various network conditions on the performance of different one-to-many and all-to-all multicasting algorithms. In the simulations, we deploy wireless nodes randomly in a 1000 $\times$ 1000-m$^2$ square, and the transmission range of each node is set to 300 m. Each node randomly picks some time-slots in the working period as its active time-slots. Without loss of generality, the energy cost $e_a$ for sending a data packet by any node is set to 1.

We first compare our algorithms to a set of commonly used multicasting algorithms in Section VII-A, demonstrating that our algorithms indeed perform much better than them. Then, in Section VII-B, we compare our algorithm to oCast [5], which may obtain the optimal solution for a one-to-many multicast with a small terminal set, showing that our algorithm actually obtains nearly optimal solution in practice. Finally, we compare the delay perform of our algorithms to others in Section VII-C, confirming that the near-optimal energy efficiency achieved by our algorithms do not come at a cost of a significantly increased delay. Each data point reported in our figures results from an average of 10 simulation results.

A. Comparing to Conventional Multicasting Algorithms

To the best of our knowledge, there is no polynomial-time minimum-energy multicasting algorithms designed for DC-WSNs. Thus, we compare our algorithms with several conventional multicasting algorithms, including the Shortest Path Tree (SPT) algorithm, the Approximate Minimum Steiner Tree (AMST) algorithm, and the minimal data overhead tree (the MNT algorithm) proposed by [30] and [31]. The SPT algorithm computes shortest paths from the source node to the receiver nodes and aggregates these shortest paths to construct a multicast tree. The AMST algorithm computes an approximate minimum Steiner tree spanning all the nodes in the terminal set $M$. Here, we adopt the AMST algorithm provided by Kou et al. [32], which was also used by Liang et al. [12] to solve the minimum-energy all-to-all multicasting problem in always-active wireless ad hoc networks. The MNT algorithm was designed for reducing the total number of transmissions for a multicast session in AA-WANETs. The work in [30] and [31] has proved that MNT can reduce the number of transmissions in a one-to-many multicast session more effectively than other heuristics.

To use the multicast trees constructed by the SPT, AMST, and MNT algorithms in a DC-WSN environment, we need to find a transmission schedule for each forwarding node in the multicast trees. Obviously, the most energy-efficient transmission schedule for any forwarding node $u$ in a multicast tree $T$ is the minimum hitting set of the collection $\{\{r\}|r \in \text{child}(u, T)\}$. However, finding a minimum hitting set is an NP-hard problem. Therefore, we use a greedy hitting set algorithm [33] to find the transmission schedules of the forwarding nodes. Since the greedy algorithm is essentially the best-possible polynomial-time approximation algorithm for the minimum hitting set problem (unless $\text{NP} \subseteq \text{DTIME}(2^{O(\log \log n)})$), each forwarding node in the multicast trees generated by the SPT, AMST, and MNT algorithms has the best-possible energy-efficient transmission schedule. In Table II, we show that the all these algorithms have the same order of time complexity. Note that as the complexity of finding transmission schedules is one order lower than that of the tree constructions (for SPT, AMST, and MNT), we only see the latter under an asymptotic notation.

We set the length of working period to $K = 20$, then every node randomly picks up a number in $[1, 20]$ and randomly chooses this number of active time-slots within the whole working period. In Fig. 3, we compare Algorithm 2 (denoted by TCS) to SPT, AMST, and MNT in the case of one-to-many multicasting. The percentage of terminal nodes scales from 20% to 100% with an increment of 10%. The algorithm proposed in [32] is adopted in Algorithm 2 for computing an approximate minimum Steiner tree. Since $e_a$ is set to one, we only plot the total number of transmissions in multicasting, which is the same with the total energy cost of one-to-many multicasting.

Fig. 3(a) and (b) shows the total number of transmissions in one-to-many multicasting for networks of size 100 and 300, respectively. As promised by its inventors, MNT outperforms SPT and AMST because the multicast tree generated by MNT has less forwarding nodes (nonleaf nodes) than the other multicast trees [30]. We also see that TCS significantly outperforms all the other algorithms, and the total number of transmissions can be reduced by about 20% even compared to MNT. The reason is that since the traditional SPT, AMST, and MNT algorithms generate multicast trees regardless of the duty cycles of the wireless nodes, they cannot optimize the transmission schedules of the forwarding nodes in a global manner. Therefore, although the transmission schedules are optimized locally in the SPT, MST, and MNT algorithms using a best-possible optimization algorithm (the greedy algorithm for finding a minimum hitting set), their total number of transmissions can still be high. On the contrary, our TCS algorithm builds the multicast tree and finds the transmission schedules of the forwarding nodes in a holistic manner by taking advantage of the special structure of the extended graph, so the total number of transmissions is reduced more effectively than the others.
In Fig. 4, we evaluate the performance of Algorithm 3 (denoted by BCS) for all-to-all multicasting. A revised version of the SPT algorithm, namely, SPT\_A, is used for computing the energy cost of all-to-all multicasting based on a shortest path tree. Given the terminal set $V$, SPT\_A randomly selects a root node in $V$ and builds a shortest path tree $T$ just as SPT does. However, the output of SPT\_A is the sum of the energy cost of every rooted tree in $V$, where $T(m)$ is the rooted tree by designating $m$ as the root of tree $T$. The greedy hitting set algorithm is used again for computing the transmission schedule of each tree. The AMST and MNT algorithm are similarly revised for all-to-all multicasting, and the revised AMST and MNT algorithms are denoted by AMST\_A and MNT\_A, respectively.

The network parameters in Fig. 4(a) and (b) are the same as those in Fig. 3(a) and (b), respectively. It can be seen that the simulation results in Fig. 4 show similar patterns with Fig. 3, and the BCS algorithm again outperforms the other algorithms in terms of the total energy cost. This can be explained by the same reasons that we have described in the one-to-many multicasting case.

In Fig. 5, we study how the length of the working period impacts the performance of different algorithms. The number of network nodes is set to 150, and the number of terminal nodes is fixed to 100. The length of the working period scales from 5 to 40 with an increment of 5. Fig. 5(a) and (b) plots the total number of transmissions for one-to-many and all-to-all multicasting, respectively. It is evident that our algorithms still outperform the other algorithms when the length of the working period changes. Meanwhile, we also see that for all algorithms under comparison, the total number of transmissions does not vary much. An explanation is that since the active time-slots of any node are randomly selected from the working period, the number of common active time-slots of any two different nodes does not vary much when the length of the working period increases. Therefore, the performance of all these algorithms does not depend on the length of the working period under our generic duty-cycling model.

### B. Comparing to oCast

As we mentioned in Section I-A, oCast [5] is the only algorithm for min-energy one-to-many multicasting in DC-WSNs, though it adopts a restricted duty-cycling model and has a time complexity exponential in $|M|$. In this section, we compare our algorithm to oCast in terms of both total transmissions and CPU time for one-to-many multicasting. Due to the prohibitive CPU time of oCast, we have to compare our algorithm with oCast only in small network instances, i.e., the number of network nodes scales from 50 to 150, and the number of terminal nodes is fixed to 10. We again set the length of the working period as $K = 20$. However, as oCast requires that the active time-slots of any node must be consecutive, we randomly pick up a number in $[1, 10]$ and randomly choose this number of consecutive active time-slots during the whole working period. We implement oCast and our algorithm using MATLAB running on a Dell PC with 8 G memory and one 3.4-GHz Intel Core i7-2600.

Fig. 6 plots the total number of transmissions of oCast and our algorithm TCS, and the corresponding CPU times of the two algorithms are listed in Table III. As the size of the terminal set is fixed, the total number of transmissions incurred by both

| $|V|$ | 50  | 60  | 70  | 80  | 90  | 100         |
|------|-----|-----|-----|-----|-----|-------------|
| oCast| 234m| 322m| 436m| 514m| 627m| 723m        |
| TCS  | 4s  | 7s  | 11s | 14s | 18s | 23s         |

| $|V|$ | 110 | 120 | 130 | 140 | 150 | |
|------|-----|-----|-----|-----|-----|-----|
| oCast| 797m| 895m| 1022m| 1102m| 1224m| |
| TCS  | 31s | 37s | 47s | 52s | 64s | |

...
The delay performance is another important performance factor that we could consider for multicasting, where the multicast delay is defined as the time taken for the data packets to reach all the terminal nodes in a multicast session. However, the min-delay multicasting problem under wireless interference is itself an NP-hard problem even in always-active wireless networks [34], and most work on min-energy multicasting for AA-WANETs neglects the delay problem (e.g., [11]–[13]). This problem gets even harder in DC-WSNs since the impact of duty cycling and interference should be jointly considered for designing a delay-efficient multicast scheduling scheme. To the best of our knowledge, no work has ever provided a collision-free scheduling algorithm to minimize the multicast delay in DC-WSNs. Consequently, we have to compare the delay of various multicasting algorithms using our collision-free scheduling algorithm provided in Section VI.

In Fig. 7(a), we plot the delay performance of one-to-many multicasting of various algorithms including oCast, TCS, and other conventional multicast algorithms. The network parameters of Fig. 7 are the same as those in Fig. 6. We can see that the delay performances of most algorithms are similar, except that the SPT algorithm has a relatively small delay. This can be roughly explained by the fact that the SPT has the shortest paths from the source node to other terminal nodes in multicasting.

In Fig. 7(b), we compare the all-to-all delay performance of various algorithms. Since the solution output by oCast is only for one-to-many multicasting and can be invalid for all-to-all multicasting, we only compare our algorithm BCS with the conventional multicast algorithms. This time, we can see that the AMST_A algorithm performs the best in general (though only slightly better than BCS). A possible reason for this is that AMST_A tends to involve the least number of links, as it directly approximates MST.

Appendix A

Proofs for the MEMTCS Problem

Proof of Theorem 1: Given an instance \((C, F)\) of the minimum hitting set problem, we create a wireless network graph \(G\) by the following method.

Let the elements in \(F\) be \(\{f_1, f_2, \ldots, f_p\}\), and let the subsets in \(C\) be \(C_1, C_2, \ldots, C_q\). For each \(C_j(1 \leq j \leq q)\), create a node \(v_j\) in \(G\), and let \(t(v_j) = \{i (1 < i < p) \land (f_i \in C_j)\}\). Create a node \(x\) in \(G\), and connect \(x\) to each \(v_j(1 \leq j \leq q)\).

Let \(x\) be the source node and \(\{x\} \cup \{v_j\}\) be the terminal set \(M\) in multicasting. Let \(e_s = 1\) and \(e_r = 0\). It is easy to prove that \(F\) has a hitting set of size at most \(k\) if and only if \(G\) has a multicast tree \(T\) and a feasible schedule \(B\) for \(T\) such that \(\Pi(T, B) \leq k\). Therefore, the MEMTCS problem is NP-hard.

Proof of Lemma 1: From Definition 5(ii), we know that for any node \(u \in V\), there are at most \(K\) satellite nodes in \(\Psi(u)\). Therefore, there are at most \((K + 1)|V|\) nodes in \(\tilde{G}\).

In Definition 5(iii), there are at most \((K + 1)^2|V| + 3K^2|E|\) edges created for each node in \(V\). In Definition 5(iv), there are at most \(K + 1\) edges created for each edge in \(E\). Therefore, there are at most \((K + 1)^2|V| + 3K^2|E|\) edges in \(\tilde{G}\).

Proof of Lemma 2: According to Definitions 5(iii) and (iv), we know that the neighboring nodes of any nuclear node in \(\tilde{G}\) are all satellite nodes. Thus, no nuclear nodes are adjacent in \(\tilde{G}\).

For any satellite node \(v \in V_S\), let \(u\) be the nuclear node of \(v\). According to Definitions 5(iii) and (iv), the set of neighboring nuclear nodes of \(v\) in \(\tilde{G}\) is a subset of \(\{u\} \cup \text{nb}_{\tilde{G}}(u)\), whose cardinality is bounded by \(\Delta + 1\).

Proof of Theorem 2: Each node \(u \in V_S\) may cover a set \(\text{nb}_{\tilde{G}}(u) \cap M\), whose cardinality is bounded by \(\Delta + 1\) (see Lemma 2). Therefore, lines 1–4 of Algorithm 1 are intrinsically...
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

12 IEEE/ACM TRANSACTIONS ON NETWORKING

Fig. 8. Connecting the node \( c \) in \( C \) to \( SB^* \).

a greedy algorithm for finding a minimum set cover [22]. Suppose that \( C^* \) is a smallest set of nodes covering the nodes in \( M \), we have

\[ |C^*| \leq H(\Delta + 1)|C^*|. \tag{1} \]

According to Definition 6, we know that the nodes in \( SB^* \) also cover the nodes in \( M \). Hence

\[ |C^*| \leq |N(SB^*)|. \tag{2} \]

Let \( ST^* \) be a minimum Steiner tree in \( \bar{G}_s \) that connects the nodes in \( C \). We have

\[ |E(SB)| \leq \rho |E(ST^*)|. \tag{3} \]

For any node \( c \in C - N(SB^*) \), there must exist a node \( m \in M \) adjacent to \( c \). Since \( SB^* \) is a satellite bridge, \( m \) must be adjacent to a certain node \( t \) in \( SB^* \). For convenience, we assume that \( c \not\in\Psi(m) \) and \( t \not\in\Psi(m) \) (otherwise the theorem can be proved in a similar way). According to Definition 5(iv), we know that there must exist two nodes \( c', t' \) such that \( c' \) is adjacent to \( c \) and \( t' \) is adjacent to \( t \). Since \( c' \) and \( t' \) are all satellite nodes of \( m \), if \( c' \neq t' \), then \( c' \) and \( t' \) must be adjacent, according to Definition 5(iii). Therefore, there exists a path from \( c \) to \( t \) in \( \bar{G}_s \) whose length is no more than 3, as shown in Fig. 8. In other words, there must exist a tree in \( \bar{G}_s \) whose node set contains \( N(SB^*) \cup C \), and it has at most \( |N(SB^*)| + 3|C| - 1 \) edges. Since \( ST^* \) is a minimum Steiner tree that connects the nodes in \( C \), it follows that

\[ E(ST^*) \leq |N(SB^*)| + 3|C| - 1. \tag{4} \]

Finally, with (1)–(4), we can get

\[ |E(SB)| = |E(SB^*)| + 1 \leq \rho |E(ST^*)| + 1 \leq \rho |N(SB^*)| + 3|C| - 1 + 1 \leq (3\rho H(\Delta + 1) + \rho)|N(SB^*)| - (\rho - 1) \leq (3\rho H(\Delta + 1) + \rho)|N(SB^*)| \]

hence the approximation ratio \( 3\rho H(\Delta + 1) + \rho \).

Proof of Lemma 3: Let \( S_1 \) be the set \( \{\lambda(u,i) | u \in d^+ (T_I) \wedge i \in \Upsilon(u,T_I)\} \). Clearly, \( S_1 = \Xi(T_I) \). Furthermore, we have the following statements.

1) According to Definition 3(iii), we know that for any node \( u \in d^+ (T_I) \) and any \( i_1, i_2 \in \Gamma(u) \), \( i_1 \neq i_2 \), the edge \( (\lambda(u,i_1), \lambda(u,i_2)) \) is in \( \bar{E} \).

2) For any two neighboring nodes \( v_1 \) and \( v_2 \) in \( d^+ (T_I) \), there must exist \( j_1 \in \Upsilon(v_1,T_I) \cap \Gamma(v_2) \) and \( j_2 \in \Upsilon(v_2,T_I) \cap \Gamma(v_1) \). According to Definition 5(iv), the edge \( (\lambda(v_1,j_1), \lambda(v_2,j_2)) \) is in \( \bar{E} \).

3) For any node \( m \in M \), if \( m \in d^+ (T_I) \), then for any \( i \in \Upsilon(m,T_I) \), \( \lambda(m,i) \) is adjacent to \( m \), according to Definition 3(iii). If \( m \not\in d^+ (T_I) \), then there must exist a node \( m_1 \in d^+ (T_I) \) adjacent to \( m \). Since \( \tau(m_1,T_I) \) is a minimum hitting set of the collection \( \{I(v) | v \in \text{nb}_{T_I} (m_1)\} \), there must exist \( k_i \in \Upsilon(m_1,T_I) \cap \Gamma(m) \). Therefore, \( \lambda(m_1,k_i) \in S_1 \). According to Definition 5(iv), we have \( \{\lambda(m_1,k_i), m_1, m\} \in \bar{E} \).

From 1) and 2), it is easy to know that the subgraph induced by \( S_1 \) in \( \bar{G} \) is connected. From 3), we know that every node in \( M \) is adjacent to a node in \( S_1 \). Let \( SB_1 \) be an arbitrary spanning tree of the subgraph induced by \( S_1 \). Then, \( SB_1 \) must be a satellite bridge. Since \( SB^* \) is a minimum satellite bridge, we have

\[ |N(SB^*)| \leq |N(SB_1)| = S_1 = \Xi(T_I). \]

Proof of Lemma 4: Suppose that the nodes in \( SB \) belong to \( q \) nuclear nodes. Therefore, \( N(SB) \) can be partitioned into \( q \) mutually disjoint subsets: \( A_1, A_2, \ldots, A_q \), such that all the nodes in \( A_i \) have the same nuclear node \( a_i \) and \( a_i \neq a_j \) for any \( i \neq j \). Let \( A \) be the set \( \{v_1, v_2, \ldots, a_q\} \). According to Definition 6, for any node \( u \in M - A \), we can find a node \( u' \in N(SB) \) adjacent to \( u \). Thus, we can connect \( u \) to \( SB \) by adding the edge \( (u, u') \). When all the nodes in \( M - A \) are connected to \( SB \), we get a new tree \( SB' \).

Then, we consider the graph \( G' = (V', E') \) whose node set \( V' = M \cup A \) and edge set \( E' \) is \( E'_1 \cup E'_2 \), where \( E'_1 = \{(a_i, a_j) | \exists (u, v) \in SB' \wedge u \in A_i \wedge v \in A_j \wedge \{1 \leq i \neq j \leq q\} \} \) and \( E'_2 = \{(a_i, v) | \exists (u, v) \in SB' \wedge u \in A_i \wedge v \in (M - A) \wedge \{1 \leq i \neq q\} \}. \) From Definition 5(iv), it is easy to know that \( G' \) is a connected subgraph of \( G \), and all nodes in \( M - A \) are degree-one nodes in \( G' \).

Let \( R \) be an arbitrary spanning tree of \( G' \). Clearly, we have \( M \subseteq N(R) \) and \( d^+ (R) \subseteq A \). Let \( F(a_i) = \{I(\lambda(a_i, l) \in A_i) \} \) for \( 1 \leq i \leq q \). Clearly, \( |F(a_i)| = |A_i| \). For \( a_i \in A \) and \( u \in \text{nb}_{HT}(a_i) \), we have the following.

1) If \( (a_i, u) \in E'_1 \), then there must exist \( a_j \in A_i, u \in A_i \), and \( v \not\in A_j \) such that \( w = a_j, (u, v) \in SB' \) and \( i \neq j \). According to Definition 5(iv), there must exist a time-slot \( l_i \in \Gamma(a_i) \) such that \( u = \lambda(a_i, l_i) \). Note that \( l_i \) is also in \( F(a_i) \). Therefore, \( F(a_i) \cap \Gamma(w) = F(a_i) \cap \Gamma(a_i) \neq \emptyset \).

2) If \( (a_i, u) \in E'_2 \), then there must exist \( A_i \in A \) adjacent to \( u \) by reasoning similar to 1), we also have \( F(a_i) \cap \Gamma(w) \neq \emptyset \).

From 1) and 2), we know that any \( F(a_i), 1 \leq i \leq q \) is a hitting set of the collection \( \{I(v) | v \in \text{nb}_{HT}(a_i)\} \). Since \( d^+ (R) \subseteq A \), we have \( \sum_{u \in d^+ (R)} |F(u)| = \sum_{a_i \in A} |F(a_i)| = \sum_{1 \leq i \leq q} |A_i| = |N(SB)|. \)

Proof of Theorem 3: From Lemma 4, we know that we can find a tree \( R^* \) in \( G \) spanning the nodes in \( M \) and a function \( F^* \) such that we have the following.

1) For any \( u \in d^+ (R^*), F^*(u) \) is a hitting set of the collection \( \{I(v) | v \in \text{nb}_{HT}(u)\} \).
Clearing, we also have \( \sum_{u \in d^{+}(T_{\text{root}})} |F_{\text{out}}^{*}(u)| \leq N(SB^{*}) \).

Therefore, we have \( \sum_{u \in d^{+}(R^{*})} |B_{\text{opt}}(u)| \leq N(SB^{*}) \).

From Lemma 3, we know \( N(SB^{*}) \leq N(T_{\text{root}}) \).

Proof of Lemma 5: From Definition 3, we know that \( H_{\text{opt}}(u) \) must be a minimum hitting set of the collection \( \{T(v) | v \in \text{child}(u, T_{\text{root}})\} \). If \( u \) is the root of \( T_{\text{opt}} \), then \( \text{child}(u, T_{\text{opt}}) = \text{tree}(u) \).

Therefore, we have \( \sum_{u \in d^{+}(T_{\text{opt}})} |H_{\text{opt}}(u)| \leq |H_{\text{opt}}(u)| + 1 \).

Otherwise, let \( u' \) be the parent node of \( u \) in \( T_{\text{opt}} \). Find an arbitrary \( j \in T_{\text{opt}} \), then \( B_{\text{opt}}(u) \cup \{j\} \) must be a hitting set of the collection \( \{T(v) | v \in \text{nbd}(u)\} \).

Therefore, we still have \( \sum_{u \in d^{+}(T_{\text{opt}})} |B_{\text{opt}}(u)| \leq |H_{\text{opt}}(u)| + 1 \).

Proof of Lemma 6: If \( s \in d^{+}(T_{\text{opt}}) \), then \( B_{\text{opt}}(s) = |T(s, T_{\text{opt}}) \).

With Lemma 5, we have

\[
\sum_{u \in \text{nbd}(T_{\text{opt}})} |B_{\text{opt}}(u)| = |B_{\text{opt}}(s)| + \sum_{u \in d^{+}(T_{\text{opt}})} |B_{\text{opt}}(u)|
\]

\[
\geq 1 + \sum_{u \in d^{+}(T_{\text{opt}})} (T(u, T_{\text{opt}}) - 1)
\]

\[
- \Xi(T_{\text{opt}}) - d^{+}(T_{\text{opt}}) + 1.
\]

If \( s \in d^{+}(T_{\text{opt}}) \), then \( B_{\text{opt}}(s) = |T(s, T_{\text{opt}}) \).

Therefore, we still have

\[
\sum_{u \in \text{nbd}(T_{\text{opt}})} |B_{\text{opt}}(u)|
\]

\[
- |B_{\text{opt}}(s)| + \sum_{u \in d^{+}(T_{\text{opt}})} |B_{\text{opt}}(u)|
\]

\[
\geq |T(s, T_{\text{opt}})| + \sum_{u \in d^{+}(T_{\text{opt}})} (T(u, T_{\text{opt}}) - 1)
\]

\[
= \sum_{u \in d^{+}(T_{\text{opt}})} (|T(u, T_{\text{opt}})| - (d^{+}(T_{\text{opt}}) - 1)
\]

\[
= \Xi(T_{\text{opt}}) - d^{+}(T_{\text{opt}}) + 1.
\]

Now, \( \sum_{u \in \text{nbd}(T_{\text{opt}})} |B_{\text{opt}}(u)| \geq \Xi(T_{\text{opt}}) - d^{+}(T_{\text{opt}}) + 1 \)

follows from the fact that \( \Xi(T_{\text{opt}}) \geq \Xi(T_{\text{root}}) \).

Proof of Theorem 4: Using Lemma 6, we have

\[
\Pi(T_{\text{opt}}, B_{\text{opt}}) = \sum_{u \in \text{nbd}(T_{\text{opt}})} |B_{\text{opt}}(u)| \cdot e_{u}
\]

\[
\geq (\Xi(T_{\text{root}}) - d^{+}(T_{\text{opt}}) + 1) \cdot e_{u},
\]

Since each node in \( T_{\text{opt}} \) with degree greater than one must transmit at least once, we have \( d^{+}(T_{\text{opt}}) \cdot e_{u} \leq 1 \cdot (T(u, B_{\text{opt}}), \text{and hence:}) \).

\[
(\Xi(T_{\text{root}}) + 1) \cdot e_{u} \leq 2 \cdot \Pi(T_{\text{opt}}, B_{\text{opt}}).
\]

Let \( \alpha = 3pH(\Delta + 1) + r \). With Lemma 4 and Theorems 2 and 3, we have

\[
\sum_{u \in d^{+}(T_{\text{opt}})} |B(u)| - \sum_{u \in d^{+}(T_{\text{opt}})} |\bar{B}(u)|
\]

\[
\leq N(SB) \leq \alpha |N(SB^{*})| = \alpha \Xi(T_{\text{root}}).
\]

Using these inequalities, we can get

\[
\Pi(T_{\text{opt}}, B_{\text{opt}}) = \sum_{u \in \text{nbd}(T_{\text{opt}})} |B(u)| \cdot e_{u}
\]

\[
- \sum_{u \in d^{+}(T_{\text{opt}})(\{s\})} |B(u)| \cdot e_{u}
\]

\[
\leq \left( \sum_{u \in d^{+}(T_{\text{opt}})(\{s\})} |B(u)| + 1 \right) \cdot e_{u}
\]

\[
\leq (\alpha \cdot \Xi(T_{\text{root}}) + 1) \cdot e_{u}
\]

\[
\leq \alpha \cdot \Xi(T_{\text{root}}) \cdot e_{u}
\]

\[
\leq 2\alpha \cdot \Pi(T_{\text{opt}}, B_{\text{opt}})
\]

\[
= (6pH(\Delta + 1) + 2r)\Pi(T_{\text{opt}}, B_{\text{opt}})
\]

hence the approximation ratio \( 6pH(\Delta + 1) + 2r \).

\[
\Pi(P_{A}, S_{A}) = \sum_{m \in M} \sum_{u \in \text{nbd}(P_{A})(m)} S_{A}(m) \cdot e_{u}
\]

\[
= \sum_{m \in V} \sum_{u \in \text{nbd}(P_{A})(m)} 1
\]

\[
= \sum_{m \in V} (1 + d^{+}(P_{A})(m)) \sum_{m \in d^{+}(P_{A})(m)} |d^{+}(P_{A})(m)|
\]

\[
= V^{2} - (V - 1) \cdot |d^{+}(P_{A})|
\]

Clearly, \( \Pi(P_{A}, S_{A}) \) is minimized iff \( P_{A} \) is an MLST of \( G \). Since the MLST problem is NP-complete, the MEMBCS is NP-hard.

\section*{REFERENCES}


Kai Han received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China, Hefei, China, in 1997 and 2004, respectively.

He is now with the School of Computer Science, Zhongyuan University of Technology, Zhengzhou, China. He is currently a Visiting Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include wireless ad hoc and sensor networks, distributed computing, and combinatorial optimization.

Yang Liu received the M.Sc. and Ph.D. degrees in computer science from Northwestern Polytechnical University, Xian, China, in 2003 and 2007, respectively.

She is currently an Associate Professor with the School of Information Sciences and Engineering, Henan University of Technology, Zhengzhou, China. Her research interests include distributed algorithms, algorithmic game theory, grid computing, and cloud computing.

Jun Luo (M’09) received the Ph.D. degree in computer science from the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, in 2006.

From 2006 to 2008, he worked as a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University, Singapore, where he is currently an Assistant Professor. His research interests include wireless networking, distributed systems, multimedia protocols, network modeling and performance analysis, applied operations research, as well as network security.

Dr. Luo is a member of the Association for Computing Machinery (ACM).