Energy-efficient Broadcast Scheduling with Minimum Latency for Low-Duty-Cycle Wireless Sensor Networks

Lijie Xu*, Jiannong Cao†, Shan Lin‡, Haipeng Dai*, Xiaobing Wu* and Guihai Chen*
*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
†Internet and Mobile Computing Lab, The Hong Kong Polytechnic University, Hong Kong
‡Department of Computer and Information Sciences, Temple University, Philadelphia, USA
Email: {ljxu83, dhpphd2003}@gmail.com, csjcao@comp.polyu.edu.hk, shan.lin@temple.edu, {wuxb, gchen}@nju.edu.cn

***Abstract*—For low-duty-cycle wireless sensor networks, multi-hop broadcasting is a very challenging problem, since every node has its own working schedules. Existing solutions usually use unicast instead of broadcast to forward packets from a node to its neighbors according to their working schedules. However, using unicast to implement broadcasting is not efficient. The broadcast nature of wireless communication offers opportunity for further energy saving even for low-duty-cycle networks. Thus, it is essential to design more efficient and robust broadcasting algorithms for low-duty-cycle networks. In this paper, we design a novel broadcasting algorithm to address this limitation for such networks. The key idea is to let some early wake-up nodes postpone their wake-up slots to overhear broadcasting message from its neighbors. This design utilizes the spatiotemporal locality of broadcasting to reduce the number of transmissions. We prove that to find the schedule for minimal network broadcasting latency and optimized total energy consumption is NP-hard, and then design an approximation algorithm that can achieve an approximation ratio of $O(\log N \cdot \log d_{max})$ where N and d_{max} denote the number of sensor nodes and the maximum node degree respectively. Compared with the traditional solution, extensive experimental results show that our algorithm achieves the minimal broadcasting latency while reducing energy consumption significantly.**

***Index Terms*—low-duty-cycle WSNs; broadcast scheduling; energy efficient; minimal latency**

I. Introduction

Wireless sensor networks (WSNs) have been widely used for various applications, such as environmental monitoring, scientific exploration, and medical care systems. Many of these applications require broadcasting to disseminate system configurations and code updates to the whole network after the system is deployed. Total energy consumption, as well as the broadcasting latency which denotes the time that the broadcasting message is received by the last uncovered node, is normally the main metric to evaluate the performance of any broadcasting schedule.

It is important and very challenging to minimize the latency and the energy consumption of the broadcasting, especially for low-duty-cycle WSNs, in which every sensor node has its own working schedule to wake-up periodically to perform sensing and communication tasks. Compared with the always-awake networks, low-duty-cycle sensor networks usually yield a notable increase on communication latency due to the periodic sleeping, and thus latency is always taken as the first consideration in such networks. In this work, we focus on the following problem: how to design a broadcasting schedule that can achieve the minimal latency while reducing total energy consumption for low-duty-cycle WSNs. Existing solutions [1]–[8] for broadcasting in low-duty-cycle WSNs usually implement broadcast with multiple unicasts, which is energy-inefficient especially for applications of large message broadcasting, such as *code updates*. Actually, the broadcast nature of wireless communication offers opportunities to reduce the total number of transmissions in broadcasting, even for duty-cycled networks where every node has its own schedule. To improve the energy efficiency of broadcasting, nodes should adjust their working schedules to maximize the number of receivers for each forwarding packet.

To achieve the minimal broadcasting latency and reduce the total energy consumption, we propose a novel broadcasting algorithm based on broadcasting spatiotemporal locality. The algorithm allows nodes to adjust their wake-up schedules to overhear forwarding message sent by their neighbors, thus improving the energy efficiency. Some nodes may postpone their wake-up slots to receive broadcasting message, increasing their latency. But these nodes can be carefully selected so that they are not on latency-critical paths. Therefore, their schedule changes do not affect the minimal broadcasting latency.

The contributions of this work are as follows:

- By capturing the spatiotemporal characteristic of multi-hop broadcasting, we model our scheduling problem as the Latency-optimal Group Steiner Tree Problem, which is proved to be NP-hard.
- This work proposes a novel wake-up schedule to realize broadcasting for low-duty-cycle networks by utilizing spatiotemporal locality of the broadcasting, which significantly improves energy efficiency without sacrificing broadcasting latency. We also prove that this design can achieve sub-linear approximation ratio for optimal broadcasting latency.
- Extensive simulations results show that our solution makes a significant improvement compared with the

traditional solution.

The rest of the paper is organized as follows: Section II summarizes the related work. Section III illustrates the network model and formally states the problem. Detailed description of our proposed scheme and performance analysis are presented in Section IV. Followed by the simulation results in Section V, we conclude the paper in Section VI.

II. Related Work

The broadcasting/multicasting problem in low-duty-cycle WSNs has received lots of attentions by the research community in the past few years [1]–[9].

Guo *et al.* [1] propose Opportunistic Flooding to make probabilistic forwarding decisions at the sender based on the delay distribution of next-hop nodes. In [5], authors consider link correlation and devise a novel flooding scheme to reduce energy consumption of broadcasting by making nodes with high correlation be assigned to a common sender. ADB [8], which is designed to be integrated with the receiver-initiated MAC protocol, reduces both redundant transmissions and delivery latency of broadcasting by avoiding collisions and transmissions over poor links. Lai *et al.* [9] propose a Hybridcast protocol which adopts opportunistic forwarding with delivery deferring to shorten broadcast latency and transmission number. However, all of these existing works either do not utilize the spatiotemporal locality of broadcasting to improve energy efficiency, or fail to provide provable approximation ratio for their solutions.

To the best of our knowledge, this is the first work that both considering the spatiotemporal locality of broadcasting and proposing the solution with provable approximation ratio, for the broadcast scheduling problem in low-duty-cycle WSNs.

III. Network Model and Problem Formulation

A. Network Model and Assumptions

In this paper, we assume that N sensor nodes are uniformly deployed in a circular sensory field with a radius of R and the sink node is located at the center of the sensory field. Also, it is assumed that time is divided into a number of equal time slots and each time slot is set long enough so that it can accommodate the transmission of the potential large broadcasting message. Each time slot is either in *sleep state* where each node will turn its radio off, or in *active state*, where each node will keep awake for a short duration of *listening interval* to make the event sensing and channel listening at the beginning. In our model, we assume all the sensor nodes are operated at low-duty-cycle mode, where each sensor node independently determines its own working schedule. For simplicity, we assume the working schedule of each node is periodic and alternates between one *active state* and $L-1$ *sleep states*. Fig. 1 explicitly illustrates an example of the periodic working schedule, where $L = 5$ and the node only stays in *active state* at time slot 3 for each period of working schedule.

As the same with most of literature for low-duty-cycle WSNs [1]–[6], we assume *time synchronization is achieved, and each node can transmit its packets at any time according its neighbors working schedules while can only receive the packets from its neighbors in active states*. Specifically, each node i will wake up at the beginning of the *active state* and keep listening for a period of *listening interval*, if any broadcasting packet in which the target receiver ID is i is received, it will keep receiving until all packets of the broadcasting message are received and then go to sleep immediately; otherwise, it will go to sleep immediately. If any sender wants to send the broadcasting message to its receiver, it will set a timer to wake up itself at the beginning of the receiver's next *active state* to finish the transmission, and then go to sleep immediately.

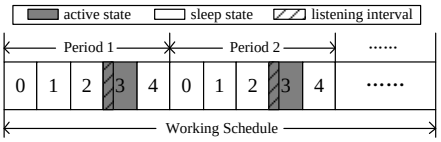


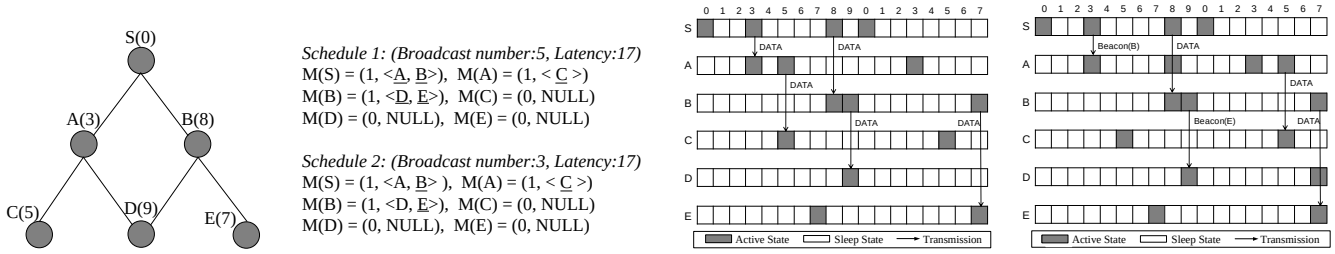
Fig. 1: An example of working schedule with $L = 5$

Besides, we also have the following basic assumptions: (1) Each node is aware of the working schedules of all its neighboring nodes within 2 hops, this can be realized via local information exchange between neighboring nodes initially after the network is deployed; (2) We do not consider the packet collision problem here due to the fact that the low-duty-cycle operation inherently reduces the probability of collision to a great extent, which has been experimentally verified in [2]; (3) We regard the broadcasting message transmission as the main energy consumption source.

B. Problem Statement

Here, we take a simple example to illustrate our problem. As shown in Fig. 2(a), the sink S wants to broadcast a message to the whole network, the number labeled within the pair of brackets denotes the scheduled wake-up time slot (i.e. *active state*) and L is set as 10. For simplicity, we first assume that for any node, the working schedules of its neighbors are different for each other. This assumption will then be relaxed in the next section.

Obviously, all nodes will receive the broadcasting message at their scheduled wake-up time slots which could bring the shortest broadcasting latency, however, draw much more energy consumption since any one-hop broadcasting is actually realized by a number of unicasts. Nevertheless, we find that the transmission number of the broadcasting message could be further reduced by considering broadcasting spatiotemporal locality, i.e. deferring the receiving time of some nodes. For any sender, here, we define two kinds of receivers: DelayedReceiver and InstantReceiver. In our model, the sender will send the broadcasting message to each InstantReceiver, and also it will send a short Beacon packet that only contains the ID of some InstantReceiver j, saying $Beacon(j)$, to each DelayedReceiver. Upon receiving the $Beacon(j)$ from the sender, any DelayedReceiver will go to sleep immediately and defer its receiving time by setting a timer to wake up at the *active state* of the InstantReceiver j. Due to assumption (1), actually, the DelayedReceiver can be aware of the working schedule of the InstantReceiver j.



(a) The original topology graph with two defined broadcasting schedules (b) Illustration of Schedule 1 (c) Illustration of Schedule 2

Fig. 2: An example with different broadcasting schedules.

Definition 1 (**Forwarding Sequence**). *For any forwarder i of the broadcasting message, its Forwarding Sequence $S_f(i)$ is defined as a sequence of its receivers sorted based on the scheduled wake-up time, namely* $S_f(i) =< r_1^1, \ldots, r_1^{k_1}, \underline{r_1}, r_2^1, \ldots, r_2^{k_2}, \underline{r_2}, \ldots, r_j^1, \ldots, r_j^{k_j}, \underline{r_j} >$ *where $r_j^k (k = 1, \ldots, k_j)$ and the underlined r_j respectively denote the DelayedReceivers and InstantReceivers of node i. Specifically, the forwarder i will send the short control packet $Beacon(r_j)$ to each DelayedReceiver r_j^k and send the broadcasting message to each InstantReceiver r_j.*

Based on broadcasting spatiotemporal locality, here, we present the definition of the broadcasting schedule in low-duty-cycle WSNs as follows.

Definition 2 (**Broadcasting Schedule**). *Given a communication graph $G = (V, E)$ in which V is the set of N nodes including the sink v_0 and all sensing nodes $I = \{v_i | i = 1, ..., N-1\}$, and E is the set of all communication edges, the schedule strategy of any sensing node $v_i (i = 1, \ldots, N-1)$ in G, saying $M(v_i)$, can be defined as follows:*

$$M(v_i) = (\alpha, \beta) \tag{1}$$

where

$$\alpha \in \{0, 1\}, \qquad \beta = \begin{cases} S_f(v_i) & \alpha = 1 \\ NULL & \alpha = 0 \end{cases}$$

In Equ. (1), the binary variable α denotes whether node v_i be a forwarder after receiving the broadcasting message, and if v_i is the forwarder (i.e. $M(v_i).\alpha = 1$), β will denote the Forwarding Sequence $S_f(v_i)$, which represents that once receiving the broadcasting message, node v_i will send the short Beacon packet or the broadcasting message to each node in $S_f(v_i)$ in sequence. Here, $NULL$ denotes the omitted item and it is obvious that $M(v_i).\beta = NULL$ for any node v_i with $M(v_i).\alpha = 0$.

Here, a broadcasting schedule $\mathbb{M}$ in the network can be defined as the set of all nodes' schedule strategies:

$$\mathbb{M} = \{M(v_i) | i = 0, \ldots, N-1\} \tag{2}$$

so that 1) I_α is a connected vertices subset in G; 2) $\bigcup_{i \in I_\alpha} M(v_i).\beta = I$; and 3) $\bigcap_{i \in I_\alpha} M(v_i).\beta = \emptyset$, where $I_\alpha = \{v_i | i = 0, \ldots, N-1$ and $M(v_i).\alpha = 1\}$.

In the above definition, assumption (3) still holds since in practice, the energy consumption for transmitting and receiving the short Beacon packet in our model is so small that it can be neglected compared with that for transmitting the broadcasting message, especially for the applications of large message broadcasting such as *code updates*. Further, we take two broadcasting schedules shown in Fig. 2 as an example. There is no waiting for each node (i.e. no DelayedReceiver but only InstantReceiver exists in the network) when adopting Schedule 1, which achieves the minimum broadcasting latency 17 but the maximum broadcasting number 5. For Schedule 2, the broadcasting number can be reduced to 3 without increasing the broadcasting latency as nodes A and D defer their receiving time to the scheduled wake-up time slots of B and E respectively. From the above example, we can find that there could exist multiple broadcasting schedules which have the same minimum broadcasting latency but different transmission numbers of the broadcasting message. Accordingly, our objective is to address the following *Latency-optimal Minimum Energy Broadcast (L-MEB) Problem*.

Problem 1 (**L-MEB**). *Given a communication graph $G = (V, E)$, how to find an efficient broadcasting schedule $\mathbb{M}$ to optimize the transmission number of the broadcasting message, i.e. to minimize $\sum_{i=0}^{N-1} M(v_i).\alpha$, while guaranteeing that the broadcasting latency is minimized.*

C. Problem Formulation

Definition 3 (**Coverage Set**). *The Coverage Set of any Sender-InstantReceiver pair (v_s, v_r) at time slot t $(0 \leq t \leq L-1)$, saying $CS(v_s, v_r, t)$, is defined as follows:*
if $t < T_s(v_r)$,

$$CS(v_s, v_r, t) = \{x \in N(v_s) - \{v_0\} | t < T_s(x) \leqslant T_s(v_r)\} \tag{3}$$

otherwise,

$$CS(v_s, v_r, t) = \{x \in N(v_s) - \{v_0\} | t < T_s(x) \leqslant L-1 \text{ or } 0 \leqslant T_s(x) \leqslant T_s(v_r)\} \tag{4}$$

in which v_0 denotes the sink node, $N(v_i)$ and $T_s(v_i)$ denote the neighboring nodes set and the scheduled wake-up time slot of node v_i respectively.

Observation 1. *Given a communication graph $G = (V, E)$, if any node v_s decides to send the broadcasting message to its neighbor v_r, then an efficient broadcasting schedule must make sure that when being received by v_r, the broadcasting message also has been received by all the nodes in the coverage set $CS(v_s, v_r, T_c(v_s))$ where $T_c(v_s)$ denotes the time slot that the uncovered node v_s receives the broadcasting message.*

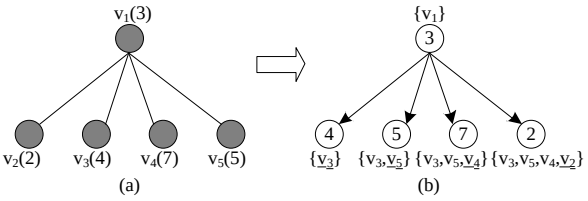


Fig. 3: (a) The original topology graph with one-hop case; (b) The corresponding SRG of the topology in (a) (underlined letters denote the InstantReceivers).

As an example, in Fig. 3(a), the sender v_1 is assumed to receive the broadcasting message at its scheduled wake-up time slot, namely $T_c(v_1) = T_s(v_1) = 3$. If we make the sender v_1 transmit the broadcasting message to node v_4, according to Observation 1, all the nodes in $CS(v_1, v_4, T_s(v_1)) = \{v_3, v_5, v_4\}$ must be ensured to have been covered at the coverage time of v_4 (i.e. the time when the uncovered node v_4 receives the broadcasting message), this is because any schedule which makes the coverage time of v_3 or v_5 be preceded by that of v_4 will never benefit from both the transmission number and the broadcasting latency. In order to better exhibit the spatiotemporal characteristic of broadcasting, the example in Fig. 3(a) can be transformed into the Spatiotemporal Relationship Graph (SRG) as shown in Fig. 3(b), where each edge represents one broadcasting message transmission and its ending vertex represents the resulting coverage set after this transmission, the number labeled within the vertex denotes the time slot of the InstantReceiver in the coverage set that represents this vertex. For convenience, each vertex in SRG is represented by its coverage set.

The following Spatiotemporal Relationship Graph Construction Algorithm (SRGC-A) will introduce how to efficiently construct a SRG in detail: Initially, SRG only contains a vertex $\{v_0\}$. Starting with considering the sink v_0 as the sender, we respectively regard each neighbor i of the sink as the InstantReceiver and insert a directed edge from the vertex $\{v_0\}$ to the newly added vertex $CS(v_0, i, T_s(v_0))$ $(i \in N(v_0))$. For each newly added vertex, saying CS_{new}, we in turn select each node $j \in CS_{new}$ as the sender, and then search all the vertices in SRG to check whether the vertex $CS(j, i, T_s(k))$ $(i \in N(j) - \{v_0\})$, where node k denotes the InstantReceiver in CS_{new}, has existed[1]. If so, we just insert a directed edge between the vertex CS_{new} and this existing vertex; otherwise, we add the vertex $CS(j, i, T_s(k))$, as well as an edge connecting $CS(j, i, T_s(k))$ with CS_{new}, into SRG. The above process repeats until no further vertex insertion to SRG is possible. Finally, we attach each vertex in SRG with the time slot of the InstantReceiver in its corresponding coverage set, and for each edge representing one message transmission, we also mark it with its corresponding Sender-InstantReceiver pair.

Theorem 1. *The worst-case time complexity of SRGC-A is $O(N^2 d_{max}^6)$, where d_{max} denotes the maximum node degree in the network.*

[1]We call a vertex A has existed in SRG iif the vertex which has the same coverage set and the same InstantReceiver with A can be found in SRG.

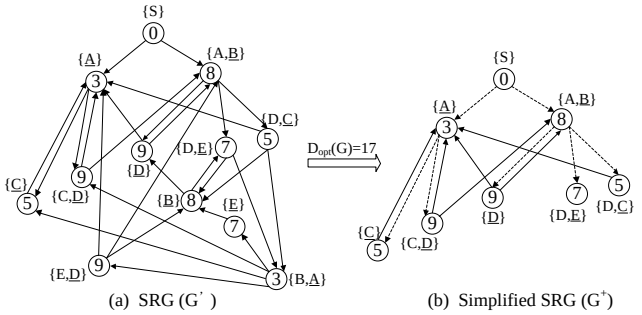


Fig. 4: (a) The corresponding SRG of the original topology graph in Fig. 2(a) (underlined letters denote the InstantReceivers); (b) The simplified SRG (dashed edges constitute LGT).

Proof: Seeing from SRGC-A, we can find that the vertex search operation actually dominates the whole construction procedure of SRG. In SRGC-A, it can result in at most d_{max}^2 SRG vertices for any node in G where d_{max} denotes the maximum node degree in G, thus, there are totally at most $N \cdot d_{max}^2$ vertices in SRG. We assume there are finally x vertices in SRG after executing SRGC-A ($x \leq N \cdot d_{max}^2$). As we know, the size of any vertex's coverage set is at most d_{max}, which means any vertex will result in at most d_{max}^2 edges and the total number of edges in SRG is thus at most $x \cdot d_{max}^2$. Actually, we can divide all the edges in SRG into two categories: (1) $x-1$ edges which are connected to the new added vertices after the search operation; (2) $x \cdot d_{max}^2 - x + 1$ edges which are connected to the existing vertices after the search operation. The total search time for the first category is at most $\sum_{m=0}^{x-2} m = \frac{(x-2)(x-1)}{2}$, and that for the second category is at most $(x-1)(x \cdot d_{max}^2 - x + 1)$.

Due to $x \leq N \cdot d_{max}^2$, the total search time of SRGC-A is therefore at most $\frac{(x-2)(x-1)}{2} + (x-1)(x \cdot d_{max}^2 - x + 1) = O(x^2 d_{max}^2) \leq O(N^2 d_{max}^6)$. ■

As an example, Fig. 4(a) shows the resulting SRG by performing SRGC-A on the original topology graph in Fig. 2(a). Specifically, we can find that SRG well captures the spatiotemporal characteristic of broadcasting and one broadcasting schedule can be implicitly represented by a subtree of SRG which is rooted from the vertex $\{v_0\}$ and consists of vertices that collectively cover all the nodes in the original topology graph.

Next, we first define the *Latency-optimal Group Steiner Tree (L-GST) Problem* and then show that our target problem can be transformed into L-GST problem.

Definition 4 (Latency of Tree). *The latency of any tree T, saying $D(T)$, can be defined as follows:*

$$D(T) = \max_{t \in leaf(T)} \{D_T(r, t)\}$$

where r denotes the root of the tree, $D_T(i, j)$ and $leaf(T)$ denote the E2E latency from vertex i to vertex j on T and the set of the leaves of T, respectively.

Problem 2 (L-GST). *Given a directed graph $G' = (V', E')$ with weight $w_{e'} = 1$ for each edge $e' \in E'$ and a family of subsets (groups) of vertices $f = \{g_1, g_2, \ldots, g_k\}(g_i \subseteq V')$, how to find a minimum weight subtree $T_{opt} = (V_T \subseteq V', E_T \subseteq E')$ rooted from a specified vertex such that:*
(1) $V_T \cap g_i \neq \phi$ for all $i \in \{1, \ldots, k\}$;
(2) the latency of T_{opt} is minimal over all the subtrees that satisfy constraint (1).

Theorem 2. *L-MEB problem on the original topology graph is equivalent to L-GST problem on its corresponding SRG.*

Proof: In SRG, we can partition all the vertices into N groups according to the common members in their coverage sets. For any node i in the original topology graph, we classify all the SRG vertices whose coverage sets containing node i into a group. Consequently, one broadcasting schedule can be represented by a subtree of SRG which is rooted from the vertex $\{v_0\}$ and connects at least one vertex in each group of SRG. Besides, for any subtree of SRG containing multiple edges that represent the same transmission (since they are marked with the same Sender-InstantReceiver pair and the same sending time), we can always find an equivalent subtree in which all edges respectively represent different transmissions. Here, two subtrees of SRG are called equivalent if and only if their representative broadcasting schedules have the same transmission number and broadcasting latency. Our target problem, i.e. Problem 1, is therefore equivalent to Problem 2 in which G' is the corresponding SRG of the original topology graph G. ∎

Here, we define the Minimum Latency Path Tree (MLPT) in any graph G as the spanning subset of G which consists of the minimum latency paths from the root to all the vertices in G. Further, we can derive the following conclusion.

Theorem 3. *Under our model, the latency of MLPT in the original topology graph is the optimal broadcasting latency.*

Proof: Actually, MLPT in the original topology graph can be regarded as a broadcasting schedule without waiting. Assume that node v_b is the leaf on MLPT of which sink-to-node latency is the maximum overall, namely the E2E latency from the sink v_0 to node v_b on MLPT, saying $D_T(v_0, v_b)$, just equals to the latency of MLPT. Obviously, we cannot find a broadcasting schedule whose latency is less than $D_T(v_0, v_b)$, given that the schedule guarantees v_b is covered. This is because the E2E latency will not benefit from waiting in duty-cycled WSNs which has been shown in [10]. Thus, the optimal broadcasting latency must be $D_T(v_0, v_b)$. ∎

Theorem 4. *L-MEB problem on the original topology graph is NP-hard.*

Proof: By Theorem 2 and Theorem 3, L-MEB problem on the original topology graph is virtually equivalent to Latency Constrained Group Steiner Tree (LC-GST) Problem on the corresponding SRG where the latency-constraint is the latency of MLPT in the original topology graph. Clearly, LC-GST problem is a generalization of Latency Constrained Steiner Tree Problem which has been proved to be NP-hard in [11]. This completes the proof. ∎

IV. Approximation Algorithm

In order to solve the aforementioned problem, in this section, we propose an efficient approximation solution which consists of the following steps: (1) Latency-optimality Guaranteed Tree (LGT) construction; (2) Edge selection on LGT; (3) Broadcasting schedule construction.

A. LGT Construction

Given any original topology graph G and its corresponding SRG G', according to Theorem 3, we can derive the optimal broadcasting latency $D_{opt}(G)$ by figuring out the latency of MLPT in G. In addition, we can further simplify SRG by merely retaining the vertices in G' whose minimum root-to-vertex latencies in G' are not more than $D_{opt}(G)$. This is because our expected subtree of G' which represents the latency-optimal broadcasting schedule, will absolutely not include any vertex whose minimum root-to-vertex latency in G' is more than the optimal broadcasting latency. Here, we take the topology graph in Fig. 2(a), of which latency of MLPT equals to 17, as an example. By removing the vertices whose the minimum root-to-vertex latency is more than 17 and the associated edges in Fig. 4(a), we can obtain the simplified SRG as shown in Fig. 4(b). Thus, our target problem can be further reduced to L-GST problem on the simplified SRG.

We use $OPT_{GST}(T)$ and $OPT_{L-GST}(G)$ to denote the cost of the optimal solution for Group Steiner Tree (GST) Problem on any tree T and that for L-GST problem on any graph G, respectively, and the following conclusion holds.

Theorem 5. *For any latency-optimal spanning subtree T^+ of the simplified SRG G^+, we have $OPT_{GST}(T^+) \leq h(T^+) \cdot OPT_{L-GST}(G^+)$, where $h(T^+)$ denotes the height of tree T^+ and is bounded by a constant ξ.*

Proof: We denote by $T_{opt} = (V_T, E_T)$ the optimal solution of our target problem. Given any latency-optimal spanning subtree T^+, it is easy to see that the subtree of T^+ whose leaves are all the vertices in $V_T - \{r\}$, saying $\widetilde{T} = (\widetilde{V}, \widetilde{E})$, must be a broadcasting schedule solution. For the worst case in which all the vertices in $V_T - \{r\}$ are just the leaves of T^+, we have $OPT_{GST}(T^+) \leq |\widetilde{E}| \leq \sum_{i \in V_T - \{r\}} |P(r,i)| \leq h(T^+) \cdot |V_T - \{r\}| = h(T^+) \cdot |E_T| = h(T^+) \cdot OPT_{L-GST}(G^+)$, where $|P(r,i)|$ denotes the hop count of the path from root r to vertex i and $h(T^+)$ denotes the height of tree T^+. Furthermore, we can find that under our uniform deployment, the latency of MLPT in the original topology graph, i.e. the optimal broadcasting latency according to Theorem 3, is at most about $\frac{R \cdot L}{r_c}$ time slots where r_c denotes the communication range of each node, and obviously each hop in T^+ will cost at least one time slot, which implies $h(T^+)$ must be at most $\xi = \frac{R \cdot L}{r_c}$, a constant independent of N. As shown in our simulations, indeed, $h(T^+)$ is always a small value that is far less than ξ in practice. ∎

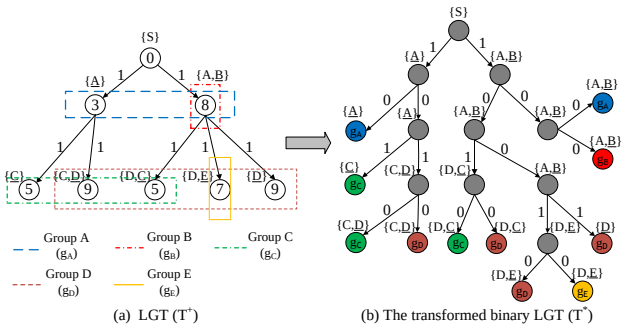


Fig. 5: The example of tree transformation.

According to Theorem 5, we are expected to find a latency-optimal spanning subtree of the simplified SRG G^+, which we call the Latency-optimality Guaranteed Tree (LGT), to provide the performance guarantee. Obviously, the MLPT in G^+ must be a latency-optimal spanning subtree of G^+ and therefore can be directly taken as the LGT here. In Fig. 4(b), the dashed edges show the LGT of the simplified SRG in Fig. 4(a). Here, we use $T^+ = (V^+, E^+)$ to denote the LGT.

B. Edge Selection on LGT

Seeing from the above subsection, actually, we can approximate our problem as the GST problem on LGT which has guaranteed the optimality of broadcasting latency. In [12], authors propose an efficient method to address the GST Problem on tree. However, [12] requires that the input tree should be a binary one where each group is a subset of its leaves and groups are pairwise disjoint, and it only gives a probabilistic solution. Based on the solution in [12], we devise a deterministic method, which consists of three steps:

(1) Tree Transformation

First, we convert the LGT into a binary tree in which each group is a subset of its leaves and groups are pairwise disjoint via the following operations:

- For any internal (i.e. non-root and non-leaf) vertex v_i in LGT, we insert a zero-weight edge from vertex v_i to a newly added vertex v_i' which shares the same coverage set as v_i.
- For any leaf vertex v_i in the tree, we insert m zero-weight edges from v_i to m newly added vertices sharing the same coverage set as v_i, in which m equals to the size of the coverage set of v_i.
- For any non-leaf vertex v_i with more than two children and its parent $p(v_i)$ (if any), we first add a new vertex v_i' with the same coverage set as v_i into the tree, then we replace v_i with v_i' to be the child of $p(v_i)$ and delete the edge from v_i to any child v_j of v_i, finally, we insert the zero-weight edges from v_i' to both v_i and v_j respectively. This process repeats until a binary tree is fully built.

By partitioning all the non-root vertices in LGT into $N-1$ groups (exclusive of the group that contains the sink), apparently, we can safely draw the conclusion that the GST Problem on LGT is equivalent to that on the transformed binary LGT in which each group is a subset of its leaves and groups are pairwise disjoint. Fig. 5 shows the example of tree transformation in which the members in one group are marked with the same color/dashed frame type.

(2) Randomized Rounding

Letting $T^* = (V^*, E^*)$ be the transformed binary LGT, as shown in [12], the GST Problem on T^* can be formulated as the following 0-1 Integer Programming:

$$\begin{aligned} (IP)\min & \sum_{e^* \in E^*} w_{e^*} x_{e^*} \\ s.t. & \sum_{e^* \in \partial S} e^* \geqslant 1,\ \forall S \subset V^* \ so\ that\ r \in S \\ & and\ S \cap g_i = \phi\ for\ some\ i \in \{1, \ldots, N-1\} \\ & w_{e^*} \in \{0,1\},\ x_{e^*} \in \{0,1\},\ \forall e^* \in E^* \end{aligned} \tag{5}$$

where r denotes the root vertex of T^*, and ∂S denotes the set of edges with only the starting endpoint in S.

In the above formulation, the binary variable x_{e^*} indicates whether to select the edge e^* or not. Given a group g_i, apparently, it requires that at least one edge with only the starting endpoint in S should be selected for any vertex set S which separates the root from g_i. Here, x_{e^*} can be relaxed to the range of [0,1] and regarded as the capacity of edge e^*, which implies that any cut that separates the root from all the vertices in a given group has capacity of at least one. According to the Max-flow Min-cut Theorem, the maximum flow from the root to any group must be at least one. In other words, there must exist a flow whose value is exactly one from the root to any group. Thus, we can relax the above Integer Programming to the following Linear Programming.

$$\begin{aligned} (LP)\min & \sum_{(u,v) \in E^*} w_{(u,v)} x_{(u,v)} \\ s.t. & \sum_{(u,v) \in E_g} f_g(u,v) = \sum_{(v,w) \in E_g} f_g(v,w),\ v \in V_g - g - \{r\} \\ & \sum_{u \in g} \sum_{v \in V_g} f_g(v,u) = 1 \\ & 0 \leqslant f_g(u,v) \leqslant x_{(u,v)} \leqslant 1, w_{(u,v)} \in \{0,1\},\ (u,v) \in E_g \\ & g \in \{g_1, g_2, \ldots, g_{N-1}\} \end{aligned} \tag{6}$$

where f_g denotes the flow from the root to group g and $T_g = (V_g, E_g)$ denotes the subtree of T^* which consists of the paths from the root r to each leaf vertex in group g.

Similar to [12], we adopt the following Randomized-rounding based Edge Selection Algorithm (RES-A) to make the edge selection: first, each edge e^* is marked with probability $\frac{x_{e^*}}{x_{p(e^*)}}$ in which x_{e^*} can be figured out from Equ. (6) and $p(e^*)$ denotes the parent edge of e^*. For any edge e^* whose starting endpoint is the root, specially, it is marked with x_{e^*}. An edge is added into the Selected Edge Set which is initially null only if the edges including itself and all its ancestors are marked. Then, we check whether the GST is generated by combining all the edges in the Selected Edge Set and the zero-weight edges, if yes, the edge selection process is terminated; otherwise, we repeat the above random experiment until the edge selection is terminated or the random experiment has been repeated for $\lceil \eta \cdot \log(N-1) \cdot \log \max_{1 \leqslant i \leqslant N-1} |g_i| \rceil$ times (rounds), where η is a constant.

The following Lemma, which has been proven in [12], explicitly shows the performance of the aforementioned randomized rounding based approach.

Lemma 1. *[12]. For a binary tree in which each group is a subset of its leaves and groups are pairwise disjoint, the probability that its root fails to reach any group g after one time random experiment is at most about* $1-\frac{1}{64\log \max\limits_{1\leqslant i\leqslant N-1}|g_i|}$.

(3) Edge Compensation and Reduction

Different from [12] which only gives a probabilistic solution, we will make sure our solution is deterministic by edge compensation operation. If the root is not connected to some group g after executing RES-A, specifically, we will establish the minimum weight path from the root to group g and then add the edges on this path which have not been selected by RES-A into the Selected Edge Set. Finally, we further reduce the transformed binary LGT to the original LGT by removing all the zero-weight edges from the Selected Edge Set.

C. Broadcasting Schedule Construction

By adopting the above-mentioned solution, we can approximately obtain the minimum weight Group Steiner Tree on LGT that consists of the edges in Selected Edge Set, saying $T^G = (V^G, E^G)$, which implicitly represents an energy efficient latency-optimal broadcasting schedule. Now, we introduce how to transform T^G into the corresponding broadcasting schedule as defined in Definition 2.

For any vertex $v_i^G \in V^G$, $T_s(v_i^G)$ is used to denote the scheduled wake-up time slot of the InstantReceiver in v_i^G (namely $T_s(v_i^G) = T_s(v_j)$ where v_j is the InstantReceiver of vertex v_i^G). For any edge $e_i^G = (v_s^G, v_r^G)$ in E^G which represents one transmission, we use a four-tuple $(S(e_i^G), t_{S(e_i^G)}, T_c(S(e_i^G)), R(e_i^G))$ to characterize it, in which $S(e_i^G)$ and $R(e_i^G)$ respectively denote the Sender and the InstantReceiver of the transmission; $t_{S(e_i^G)}$ denotes the time when the sender receives the broadcasting message before the transmission e_i^G, and $T_c(S(e_i^G))$, which equals to $T_s(v_s^G)$, denotes the corresponding time slot of $t_{S(e_i^G)}$. In Fig. 5(a), for example, the edge $(\{A,B\},\{D,C\})$ can thus be represented by the four-tuple $(A,8,8,C)$. For any sensor node v_i, we use $t_{v_i}^{min}$ to denote the time when node v_i is covered for the first time in the schedule T^G, specifically,

$$t_{v_i}^{\min} = \min_{k\in V^G(v_i)} D_{T^G}(r^G, k) \tag{7}$$

where r^G is the root of T^G, $V^G(v_i)$ is the subset of V^G consisting of the vertices of which coverage sets contain node v_i. Further, for any forwarder v_j, we define $T_c^{min}(v_j)$ as follows:

$$T_c^{\min}(v_j) = T_s(\arg\min_{k\in V_s^G(v_j)} D_{T^G}(r^G, k)) \tag{8}$$

where $V_s^G(v_j)$ is the subset of $V^G(v_j)$ consisting of the vertices which have at least one output edge with the sender v_j in T^G. Here, we use a ring as shown in Fig. 6 to represent the periodical wake-up schedule of each node.

Observation 2. *Given a broadcasting schedule $\mathbb{M}$ and for any forwarding node v_r, making v_r receive the broadcasting message ahead of its coverage time in $\mathbb{M}$ will NOT defer its neighbors' coverage time in $\mathbb{M}$ and will NOT increase the transmission number of v_r in $\mathbb{M}$ as also.*

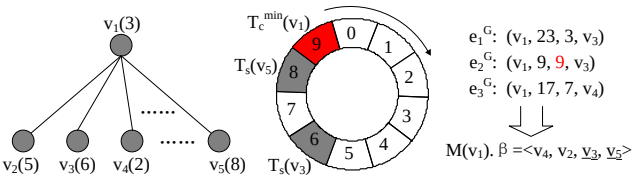


Fig. 6: An example of the initial build of $M(v_i).\beta$.

Our Broadcasting Schedule Construction Algorithm (BSC-A) includes two steps: Schedule Initialization and Schedule Adjustment.

(1) Schedule Initialization

For any node v_i, its schedule strategy $M(v_i)$ can be initially generated from T^G as follows: If there is no edge where the sender is v_i in E^G, we set $M(v_i).\alpha = 0$ and $M(v_i).\beta = NULL$. If there exists at least one edge indicating the sender is v_i in E^G, we set $M(v_i).\alpha = 1$ and $M(v_i).\beta$ which is initially null can be built by the following way: For any edge e_i^G in the set of edges where the sender is v_i, we check that whether $CS(v_i, R(e_i^G), T_c(S(e_i^G))) \subseteq CS(v_i, R(e_i^G), T_c^{min}(v_i))$, if yes, we add node $R(e_i^G)$ into $M(v_i).\beta$ if it is not in $M(v_i).\beta$ and mark it as the InstantReceiver; otherwise, node v_i' will be added into $M(v_i).\beta$ if it is not in $M(v_i).\beta$ and be marked as the InstantReceiver, where v_i' is the neighboring node of v_i whose scheduled wake-up time slot is the furthest away from the time slot $T_c^{min}(v_i)$ in the wake-up schedule ring along with the clockwise direction. Then, we sort $M(v_i).\beta$ as $< \underline{\beta_1}, \underline{\beta_2}, \ldots, \underline{\beta_{m(v_i)}} >$ according to the clockwise sequence of their scheduled time slots in the wake-up schedule ring with starting from the time slot $T_c^{min}(v_i)$. Afterwards, we add all the nodes in set $CS(v_i, \beta_{m(v_i)}, T_c^{min}(v_i)) - M(v_i).\beta$ into $M(v_i).\beta$ and mark them as the DelayedReceivers, and then we reorder $M(v_i).\beta$ according to the clockwise sequence of their scheduled time slots in the wake-up schedule ring with starting from the time slot $T_c^{min}(v_i)$. According to Observation 2, the above build process can ensure that the coverage time of each node will not exceed that scheduled in T^G, which implies the performance of the broadcasting schedule that represented by T^G will not be degraded.

(2) Schedule Adjustment

Here, suppose we have any three forwarders v_i, v_j and v_k, of which Forwarding Sequences can be represented as follows:

$$\begin{aligned} M(v_i).\beta &=< v_3, \underline{v_5}, v_6, v_7, \underline{v_8}, v_{10}, \underline{v_9} > \\ M(v_j).\beta &=< v_2, v_4, \underline{v_6}, v_7, v_{12}, \underline{v_8} > \\ M(v_k).\beta &=< \underline{v_5}, v_1, \underline{v_6}, v_{11}, v_8, \underline{v_9} > \end{aligned} \tag{9}$$

where the underlined nodes denote the InstantReceivers.

Definition 5 (Remove Back). *Given any Forwarding Sequence $M(v_i).\beta$ that contains v_j, the operation to Remove Back v_j in $M(v_i).\beta$ is defined as: (1) If v_j is not the InstantReceiver, to remove v_j from $M(v_i).\beta$; (2) Otherwise, to replace v_j with the previous node of v_j in $M(v_i).\beta$ to be*

the InstantReceiver and then remove v_j from it, particularly, if the previous node of v_j is also the InstantReceiver or v_j is the first node in $M(v_i).\beta$, just to remove v_j from $M(v_i).\beta$.

As the randomized approach is adopted in the building of T^G, the resulting broadcasting schedule could incur redundant transmissions. In the example as shown in Equ. (9), if node v_5, which is the InstantReceiver of both v_i and v_k, is not selected as a forwarder or receives the broadcasting message from v_i ahead of v_k, the transmission from v_k to v_5 will be redundant and thus can be removed from the schedule. In addition to the redundant transmission, unnecessary collision could also be inevitable for our derived schedule. For example, the collision probability would rise when the time v_6 takes to receive the broadcasting message from v_j is the same with that from v_k. If v_6 receives the broadcasting message from v_j no later than that from v_k, we can Remove Back v_6 in $M(v_k).\beta$ to get an equivalent Forwarding Sequence according to Observation 2. In order to avoid the redundant transmission and reduce the collision probability as greatly as possible, in this step, we propose the following approach to further adjust $M(v_i).\alpha$ and $M(v_i).\beta$ values for each node v_i.

For each non-sink node v_i, we first find the edge $e_i^G = (v_s^G, v_r^G)$ in E^G such that $v_i \in v_r^G$ and $D_{T^G}(r^G, v_r^G) = t_{v_i}^{min}$, and node $S(e_i^G)$ is thus selected to be the candidate sender for v_i. Then, we check the Forwarding Sequence of each forwarder v_j where $v_j \neq S(e_i^G)$, if $v_i \in M(v_j).\beta$, to Remove Back v_i in $M(v_j).\beta$. After the above process, if the new resulting Forwarding Sequence of any forwarder v_j is empty, we will set $M(v_j).\alpha = 0$ and $M(v_j).\beta = NULL$.

Obviously, the aforementioned approach can ensure each node appears in only one forwarder's Forwarding Sequence which essentially avoids the redundant transmission and unnecessary collision that exist in the schedule T^G. Accordingly, we can easily have the following proposition.

Proposition 1. *The broadcasting schedule resulted from BSC-A will keep the latency-optimality and must be no worse than the schedule that represented by T^G in terms of the transmission number.*

Theorem 6. *When $\eta \geq 64$, the approximation ratio of the broadcasting schedule resulted from BSC-A is $O(\log N \cdot \log d_{max})$.*

Proof: As Equ. (6), of which the optimal solution equals to the expected cost (i.e. the expected number of selected edges) $\mathbf{E}[cost/round]$ after each round of RES-A, is the LP-relaxation of Equ. (5) of which the optimal solution equals to $OPT_{GST}(T^*)$, we have $\mathbf{E}[cost/round] \leq OPT_{GST}(T^*)$. Also, we observe that for any group g, the number of the added edges in the Edge Compensation step after executing RES-A, saying N_g, will not exceed $OPT_{GST}(T^*)$ since the minimum weight path from the root to group g in the Edge Compensation step must be no longer than the path from the root to g that belongs to optimal GST on T^*. Further, we use A to denote the event that the root fails to reach any group g after executing RES-A. According to Lemma 1, we have

$$\begin{aligned}\mathbf{Pr}[A] &\leq (1 - \frac{1}{64 \log \max\limits_{1 \leqslant i \leqslant N-1} |g_i|})^{\lceil \eta \cdot \log(N-1) \cdot \log \max\limits_{1 \leqslant i \leqslant N-1} |g_i| \rceil} \\ &\approx (1 - \frac{1}{64 \log \max\limits_{1 \leqslant i \leqslant N-1} |g_i|})^{(64 \log \max\limits_{1 \leqslant i \leqslant N-1} |g_i|) \cdot \frac{\eta \cdot \log(N-1)}{64}}\end{aligned} \tag{10}$$

Due to $\lim\limits_{x \to \infty} (1 - \frac{1}{x})^x = e^{-1}$ and $\log \max\limits_{1 \leqslant i \leqslant N-1} |g_i| \geq 1$, we can find that if $\eta \geq 64$, then

$$\mathbf{Pr}[A] \leq e^{-\frac{\eta \cdot \log(N-1)}{64}} \leq e^{-\log(N-1)} \leq e^{\log \frac{1}{N-1}} \leq \frac{1}{N-1} \tag{11}$$

Let $\widetilde{N}$ denote the number of the groups that fail to reach the root after executing RES-A, thus, we have $\mathbf{E}[\widetilde{N}] = (N-1)\mathbf{Pr}[A] \leq 1$ since the $N-1$ groups in T^* are disjoint and independent. Letting $\Delta = \lceil \eta \cdot \log(N-1) \cdot \log \max\limits_{1 \leqslant i \leqslant N-1} |g_i| \rceil$, according to Theorem 5, the expected cost of the solution T^G, namely $\mathbf{E}[|E^G|]$, is thus at most

$$\begin{aligned}&\Delta \cdot \mathbf{E}[cost/round] + \mathbf{E}[\widetilde{N}] \cdot N_g \\ &\leq (\Delta + 1) \cdot OPT_{GST}(T^*) = (\Delta + 1) \cdot OPT_{GST}(T^+) \\ &\leq h(T^+) \cdot (\Delta + 1) \cdot OPT_{L-GST}(G^+) \\ &\leq \xi \cdot (\Delta + 1) \cdot OPT_{L-GST}(G^+)\end{aligned} \tag{12}$$

For the final solution $\mathbb{M}$ resulted from BSC-A, according to Proposition 1 and the conclusion $\log \max\limits_{1 \leqslant i \leqslant N-1} |g_i| \leq O(\log d_{max})$ which can be easily seen from SRGC-A, we have $\mathbf{E}[\sum\limits_{i=0}^{N-1} M(v_i).\alpha] \leq \mathbf{E}[|E^G|] \leq O(\log N \cdot \log d_{max}) \cdot OPT_{L-GST}(G^+)$ ∎

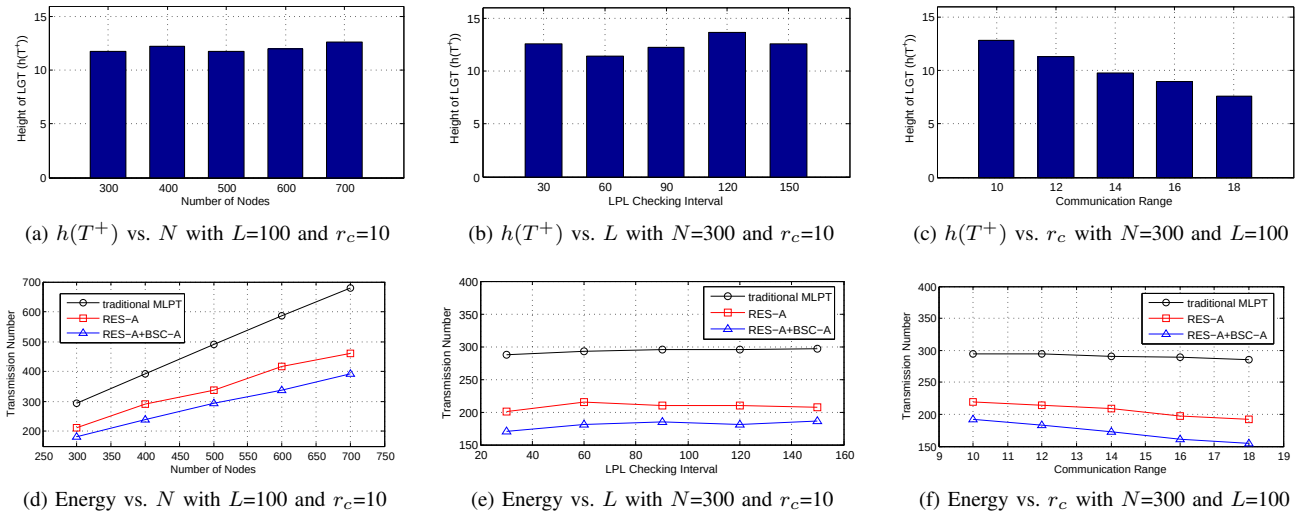
A straightforward observation from Theorem 6 is that we can set the parameter η as 64 in the RES-A so as to guarantee the approximation ratio of $O(\log N \cdot \log d_{max})$.

Note that we assumed the working schedules of neighboring nodes are different for each other previously. Nevertheless, our solution can also be applied to the more practical case where neighboring nodes could have the identical wake-up schedule by simply regarding the set of neighbors having identical wake-up time slot as one virtual node. In BSC-A, a virtual node is Removed Back in the Forwarding Sequence of any forwarder i only if node i is not the candidate sender for each node in this virtual node. otherwise, we only need to remove the nodes whose candidate senders are not i from the virtual node.

V. Performance Evaluation

In this section, we evaluate the performance of our solution via simulations. Here, we assume sensor nodes are uniformly distributed in a circular sensory field with a radius of $R = 50$ and the sink node is located at the center of the sensory field. Once the length of checking interval L is fixed, each node independently and randomly chooses a time slot to wake up in one period of L time slots and then repeat this working schedule. All the results are obtained by averaging results of 10 experiments.

First, we evaluate the value of $h(T^+)$ where T^+ is LGT, namely the MLPT on the simplified SRG. We respectively consider the following three cases of network configurations:



(a) $h(T^+)$ vs. N with L=100 and r_c=10 (b) $h(T^+)$ vs. L with N=300 and r_c=10 (c) $h(T^+)$ vs. r_c with N=300 and L=100

(d) Energy vs. N with L=100 and r_c=10 (e) Energy vs. L with N=300 and r_c=10 (f) Energy vs. r_c with N=300 and L=100

Fig. 7: The performance under different network configurations.

(L=100, r_c=10), (N=300, r_c=10) and (L=100, N=300). Fig. 7(a) and Fig. 7(b) exhibit the similar results, that is, the value of $h(T^+)$ almost keeps stable, namely around 12, as the number of nodes or the length of LPL checking interval increases. As shown in Fig. 7(c), however, $h(T^+)$ drops as the communication range of each node increases, which means it is only related to the communication range of each node on the condition that R is fixed. Intuitively, this is because the value of $h(T^+)$ can be approximately considered as the ratio of the latency of MLPT on the original topology graph, which is generally determined by the height of MLPT and average one-hop latency on MLPT, to the average one-hop latency on LGT. Actually, average one-hop latency on MLPT and that on LGT have the same order of magnitude once L is fixed, and the height of MLPT is generally determined by R and r_c. According to the simulation result depicted in Fig. 7(a)-(c), we can obviously find that the value of $h(T^+)$ is always a small constant without being related with N and is far less than its theoretical upper bound $\frac{R \cdot L}{r_c}$ under whatever the network configuration, thus it can be approximately neglected in the computation of the approximation ratio.

Next, we proceed to evaluate the performance of our proposed approximation solution by comparing it with the traditional MLPT-based latency-optimal strategy, in which no deferring policy is employed and the sink broadcasts the message directly along with the MLPT of the original topology graph. As shown in Fig. 7(d)-(f), our solution using both RES-A and BSC-A significantly reduces the total transmission number of the broadcasting message compared with the traditional MLPT-based approach under various network configurations, and achieves around 10%-25% improvement over the solution where only RES-A is adopted, i.e. $|E^G|$, which shows the high-efficiency of our proposed BSC-A on the reduction of redundant transmissions. Besides, seeing from Fig. 7(d)-(f), it is clear that the network density and the transmission power affect the performance of our solution to a greater extent compared with the duty cycle given that R is fixed, and specifically, our solution would perform better as the network density or the transmission power rises.

VI. Conclusion

In this paper, we consider how to utilize broadcasting spatiotemporal locality to address the broadcast scheduling problem in low-duty-cycle WSNs. We first transform our target problem into the Latency-optimal Group Steiner Tree Problem on the Spatiotemporal Relationship Graph, which is shown to be NP-hard, and then approximately solve this problem by using a deterministic randomized-rounding based method. Also, an efficient Broadcasting Schedule Construction Algorithm is proposed to further avoid the redundant transmission and reduce the collision probability as greatly as possible. Finally, the high-efficiency of our solution has been evaluated through theoretical analysis and simulations.

References

[1] S. Guo, Y. Gu, B. Jiang, and T. He. *Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links*. In MobiCom, 2009.

[2] F. Wang and J.C. Liu. *Duty-cycle-aware broadcast in wireless sensor networks*. In INFOCOM, pages 468-476, 2009.

[3] L. Su, B. Ding, Y. Yang, T.F. Abdelzaher, G. Cao, and J.C. Hou, *ocast: Optimal multicast routing protocol for wireless sensor networks*. In ICNP, pages 151-160, 2009.

[4] T. Zhu, Z.G. Zhong, T. He, Z.L. Zhang. *Exploring link correlation for efficient flooding in wireless sensor networks*. In NSDI, 2010.

[5] S. Guo, S.M. Kim, T. Zhu, Y. Gu, T. He. *Correlated flooding in low-duty-cycle wireless sensor networks*. In ICNP, 2011.

[6] X.L. Jiao, W. Lou, J.C. Ma, J.N. Cao, X.D. Wang, X.M. Zhou. *Duty-cycle-aware minimum latency broadcast scheduling in multi-hop wireless networks*. In ICDCS, 2010.

[7] Z.J. Li, M. Li, J.L. Liu, S.J. Tang. *Understanding the flooding in low-duty-cycle wireless sensor networks*. In ICPP, 2011.

[8] Y.J. Sun, O. Gurewitz, S. Du, L. Tang, and D.B. Johnson. *ADB: An efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks*. In SenSys, pages 43-56, 2009.

[9] S. Lai, B. Ravindran. *On multihop broadcast over adaptively duty-cycled wireless sensor networks*. In DCOSS, 2010.

[10] S. Lai, B. Ravindran. *On distributed time-dependent shortest paths over duty-cycled wireless sensor networks*. In INFOCOM, 2010.

[11] V.P. Kompella, J.C. Pasquale, G.C. Polyzos. *Multicasting for multimedia applications*. In INFOCOM, 1992.

[12] N. Garg, G. Konjevod, R. Ravi. *A polylogarithmic approximation algorithm for the group steiner tree problem*. In SODA, 2000.