Data Aggregation Scheduling in Uncoordinated Duty-Cycled Wireless Sensor Networks under Protocol Interference Model

XIANLONG JIAO[1,2], WEI LOU[2], XIAODONG WANG[1], JIANNONG CAO[2], MING XU[1], XINGMING ZHOU[1⋆]

[1] *School of Computer, National University of Defense and Technology, Changsha 410073, China*
[2] *Department of Computing, The Hong Kong Polytechnic University, Kowloon 999077, Hong Kong*

Data aggregation is a critical and widely-used operation in wireless sensor networks (WSNs). Data aggregation scheduling (DAS) aims to find an interference-free scheduling for data aggregation with the minimum latency. Most existing algorithms for the DAS problem, unfortunately, assume that nodes are always active, and hence are not suitable for duty-cycled scenarios. In this paper, we investigate the DAS problem in uncoordinated duty-cycled WSNs (DAS-UDC problem) under protocol interference model and prove its NP-hardness. To solve this problem, we propose two novel approximation algorithms called SDAS and CDAS with the data aggregation latency of at most $O(R_s + \Delta)$ and $O(R + \Delta)$ respectively, where R_s, Δ and R are the maximum depth of the breadth-first-search tree rooted at the sink node s, the maximum node degree and the graph-theoretic radius of the network respectively. We conduct extensive simulations to evaluate the performance of our algorithms and report their average performance.

Key words: data aggregation scheduling, duty cycle, approximation algorithms, breadth-first-search, wireless sensor networks, protocol interference model.

⋆ email: {xljiao, xdwang, mxu, xmzhou}@nudt.edu.cn, {csweilou, csjcao}@comp.polyu.edu.hk

1 INTRODUCTION

Data aggregation, which aggregates the data (e.g., temperature) from the sensor nodes to the sink node, is one of the most critical communication operations in wireless sensor networks (WSNs). The energy of nodes in WSNs is powered by batteries, and to save the energy, these nodes in WSNs often switch between the active state and the sleep state. In uncoordinated duty-cycled multi-hop wireless sensor networks (UDC-WSNs), nodes independently determine their sleep/active cycles without coordination [3, 6, 9, 27]. To promptly deal with the emergent events in many applications of UDC-WSNs, such as fire monitoring and battlefield surveillance, data aggregation must be completed with low latency.

Three common interference models in UDC-WSNs include *graph-based interference model*, *protocol interference model* and *physical interference model* [14]. Under *graph-based interference model*, the interference is treated as the collision, and if two nodes send their messages to their common neighboring node concurrently, the common neighboring node will receive neither of the two messages. Under *protocol interference model*, if one node lies in the interference range of one transmitter node, it cannot receive the messages from other nodes when this transmitter node is transmitting its message. Under *physical interference model*, the signal to interference-plus-noise ratio (SINR) of one receiver node should be above some threshold such that this node can correctly decode the message from one transmitter node.

Data aggregation scheduling (DAS) aims to provide an interference-free scheduling for data aggregation with the minimum latency. The DAS problem in WSNs without sleep/active cycles has been proved to be NP-hard in [1]. To solve this problem, many approximation algorithms [1, 11, 14, 19, 23, 25] have been proposed. However, most of these algorithms assume that nodes are always active. In UDC-WSNs, nodes can only transmit their messages to their neighboring nodes when these neighboring nodes are active. Moreover, it may leave many idle time slots unused if simply extending existing algorithms. Therefore, it is critical to devise an efficient data aggregation algorithm for the DAS problem in duty-cycled scenarios.

In this paper, we investigate the DAS problem in UDC-WSNs (DAS-UDC problem) under protocol interference model. To the best of our knowledge, this is the first work to study the DAS-UDC problem under protocol interference model. The main contributions of this paper include the following.

1. We formulate the DAS-UDC problem under protocol interference model and prove its NP-hardness.

2. We propose two novel approximation algorithms, called SDAS and CDAS, to address the DAS-UDC problem under protocol interference model. SDAS algorithm directly aggregates the data to the sink node, while CDAS algorithm uses some node to assist the sink node's data aggregation.

3. We prove that the data aggregation latency of the SDAS and CDAS algorithms is at most $3\beta^2|T|(15R_s+\Delta-3)$ and $(45\beta^2+1)|T|R+3\beta^2|T|(\Delta-3)$ respectively, where β is $\lceil\frac{2}{3}(\alpha+2)\rceil$, α denotes the ratio of the interference radius to the transmission radius, $|T|$ is the number of time slots in a scheduling period, R_s is the maximum depth of the breadth-first-search tree rooted at the sink node s, Δ is the maximum node degree of the network, and R is the graph-theoretic radius of the network.

4. We conduct extensive simulations to evaluate the performance of our algorithms, and the simulation results confirm the efficiency of our algorithms.

The remainder of this paper is organized as follows. In Section 2, we detail the related work on data aggregation. Section 3 introduces the network model, problem formulation and some graph-theoretic definitions. We present our data aggregation scheduling algorithms in Section 4. Section 5 provides the performance analysis of our algorithms. The results of extensive simulations are shown and analyzed in Section 6. Finally, we conclude this paper and discuss our future work in Section 7.

2 RELATED WORK

A lot of researches [4, 12, 15, 16, 20, 24, 26] have been done to study the data aggregation problem due to its importance in WSNs. Intanagonwiwat et al. [12] propose a greedy data aggregation approach based on an energy-efficient aggregation tree. They show that their approach achieves up to 45% energy savings over opportunistic data aggregation approaches. Shrivastava et al. [15] present a distributed data aggregation technique which saves bandwidth and power compared to naive data aggregation schemes. To explore energy-latency tradeoffs for data aggregation in WSNs, Yu et al. [26] provide both offline and online efficient algorithms. Other work [4, 16, 20, 24] also proposes efficient solutions to reduce the latency or energy consumption of data aggregation. However, none of these researches focus on the DAS problem.

Chen et al. [1] prove that the DAS problem under graph-based interference model is NP-hard, and propose an approximation algorithm with a ratio of $\Delta-1$. Huang et al. [11] also investigate this problem and propose an algorithm with the data aggregation latency of at most $23R + \Delta + 18$. Yu et al. [25] propose a distributed data aggregation scheduling algorithm with the latency of at most $24D + 6\Delta + 16$, where D is the graph-theoretic diameter of the network. Xu et al. [23] also present an efficient distributed data aggregation algorithm, which improves the worst data aggregation latency to $16R+\Delta-16$.

Wan et al. [18] propose three approximation algorithms with the latency of at most $15R + \Delta - 4$, $2R + O(\log R) + \Delta$ and $(1 + O(\log R/\sqrt[3]{R}))R + \Delta$ respectively. They also claim that their first two algorithms can be extended with a generic expansion technique to solve the DAS problem under protocol interference model. Wan et al. [19] propose an approximation algorithm to solve the DAS problem in multi-channel multi-hop wireless networks under protocol interference model. The DAS problem under physical interference model is investigated by Li et al. [14].

Nevertheless, none of the work mentioned above has investigated the DAS problem in duty-cycled scenarios. Notice that, sensor nodes consume a large amount of energy if they always wake up, so it is reasonable to let these nodes turn to sleep. In duty-cycled scenarios, the transmitter node must wait for the receive node to wake up before it transmits the message, and thus the problems (data aggregation, broadcast, and so on) in these scenarios differ from the problems in conventional scenarios. To tackle the problems in duty-cycled scenarios, many solutions are proposed [7,8,13,21,22]. Wu. et al. [22] schedule the wake-up time slots of sensor nodes to save the energy consumption by data aggregation. Their proposed approach requires the coordination of sensor nodes, and thus incurs extra communication overhead. The other work [7,8,13,21] focuses on the broadcast problem in duty-cycled scenarios. However, to the best of our knowledge, less attention is paid to the DAS-UDC problem under protocol interference model. In this paper, we will investigate this problem and devise efficient approximation algorithms for this problem.

3 PRELIMINARIES

3.1 Network Model

We model the uncoordinated duty-cycled wireless sensor network as a UDG $G = (V, E)$, where V contains one sink node s and $n - 1$ sensor nodes in the network, and E is the set of edges, which exist between any two nodes u and v if their Euclidean distance is no larger than the transmission radius r. We

regard protocol interference model as the interference model. We denote by r_f the interference radius, and by α the interference ratio, which equals to the ratio of r_f to r. Every node cannot send or receive the messages at the same time. We call node c the *graph center* of the network if the hop distance of the shortest path from node c to the farthest node in the network is the minimum. This hop distance denoted by R is called the *graph-theoretic radius* of the network, which can be achieved by using the Floyd-Warshall algorithm [5].

We assume that nodes independently determine the active/sleep time in advance. The duty cycle is defined as the ratio of the active time to the whole scheduling time. The whole scheduling time is divided into multiple scheduling periods of the same length. One scheduling period T is further divided into fixed $|T|$ unit time slots, i.e., $T = \{0, 1, ..., |T| - 1\}$. Every node v independently and randomly chooses one time slot in T as its active time slot $A(v)$. A node can transmit the message at any time slot, but is only allowed to receive the message at its active time slot.

3.2 Problem Formulation

We study the data aggregation problem in UDC-WSNs, where all the sensor nodes require transmitting their data to the sink node. The data aggregation task starts at time slot 0 and completes when the sink node receives the data of all the sensor nodes. We model the data aggregation scheduling as assigning the transmitting time slots for all the nodes, i.e., assigning a function $TTS : V \rightarrow 2^{\mathcal{N}}$, where $\mathcal{N}$ denotes the natural number set. For example, if we assign the transmitting time slot t_1 for node v, then $TTS(v) = \{t_1\}$. Here, the transmitting time slot t_1 can be any time slot in some scheduling period, i.e., $t_1 = k_1|T| + k_2$, where $k_1 \in \{0, 1, 2, ...\}$ and $0 \leq k_2 \leq |T| - 1$.

Note that, if we complete the data aggregation scheduling, some node u will be assigned with a transmitting time slot t_2, which is the largest in the assigned transmitting time slots for all the nodes. Data aggregation can finish one time slot after node u transmits the data at time slot t_2. It implies that the data aggregation latency equals to $t_2 + 1$. Therefore, to minimize the data aggregation latency can be transformed to minimize the largest assigned transmitting time slot t_2, which is the objective of data aggregation scheduling.

Lemma 1. *The DAS-UDC problem under protocol interference model is NP-hard.*

Proof. If we set $T = \{0\}$ and $\alpha = 1$, all the nodes keep always active and the protocol interference model reduces to the graph-based interference model.

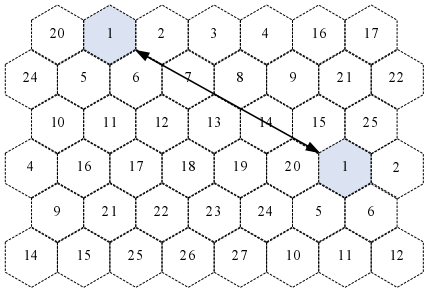


FIGURE 1
An example to illustrate the coloring scheme

Therefore the DAS-UDC problem under protocol interference model reduces to the DAS problem in conventional WSNs under graph-based interference model, which has been proved to be NP-hard in [1], so this lemma holds. □

3.3 Graph-Theoretic Definitions

We denote by $G[U]$ the subgraph of G induced by a subset U of V. If there is no edge between any two nodes in $G[U]$, we call the subset U an Independent Set (IS) of G. A Maximal Independent Set (MIS) of G is not a subset of any other IS of G. It is known that a node can be adjacent to at most 5 nodes in an IS of a UDG [17]. U_1 is a *cover* of U_2, if, for any node in U_2, there is one node in U_1 which is adjacent to this node. A *minimal cover* of U is a cover of U in which the removal of any single node destroys the covering property.

A proper tessellation of hexagons in the whole plane is to partition the plane into hexagons of the same size. Coloring of these hexagons is to assign every hexagon one natural number representing the color of this hexagon. According to [10], a proper $3\beta^2$ coloring of half-open half-closed hexagons can make sure that the distance between two hexagons of the same color is larger than $3\beta - 2$ radii of the hexagon. If we set β as $\lceil \frac{2}{3}(\alpha + 2) \rceil$ and set the radius of the hexagon as $r/2$, the distance between two hexagons of the same color will be larger than $(\alpha + 1)r = r_f + r$.

We take Figure 1 as an example to illustrate the coloring scheme. We assume that α is 2, and hence $\beta = \lceil \frac{2}{3}(\alpha + 2) \rceil = 3$. A proper tessellation and 27-coloring ($3\beta^2 = 27$) of hexagons are shown in Figure 1. The radius of the hexagon is $r/2$. Using this coloring scheme, we can make sure that the distance between two hexgons of the same color is larger than $(\alpha + 1)r$, e.g.,

the distance of two hexagons colored with 1 in Figure 1 is at least $7 \times r/2 = 3.5r$, which is larger than $(\alpha + 1)r = 3r$.

4 APPROXIMATION ALGORITHMS

We have shown that the DAS-UDC problem under protocol interference model is NP-hard in previous section. In this section, we first present one approximation algorithm called SDAS, which directly aggregates the data from the sensor nodes to the sink node. We then propose another approximation algorithm called CDAS. This algorithm first aggregates the data from all the sensor nodes to the graph center, and then transmits the aggregated data from the graph center to the sink node.

4.1 Sink based Data Aggregation Scheduling

The Sink based Data Aggregation Scheduling (SDAS) algorithm contains five steps as shown in Algorithm 1. The first step is to color all the nodes. We use the coloring scheme detailed in Section 3.3 to color these nodes, and use $f : V \to \{1, 2, ..., 3\beta^2\}$ to denote this coloring method. The second and third steps are to divide all the nodes according to the depths of these nodes in the BFS tree T^s_{BFS} rooted at sink node s. R_s denotes the maximum depth of T^s_{BFS}.

Algorithm 1 SDAS Algorithm

Input: $G = (V, E)$, s, A, α, r, T.
Output: Aggregation Scheduling $TTS : V \to 2^{\mathcal{N}}$.

1: Apply a proper tessellation and $3\beta^2$-coloring of hexagons with a radius of $r/2$ in the whole area to color all the nodes. Use $f : V \to \{1, 2, ..., 3\beta^2\}$ to denote this coloring method, where β is $\lceil \frac{2}{3}(\alpha + 2) \rceil$.
2: Construct the BFS tree T^s_{BFS} rooted at sink node s.
3: Assign $MaxDepth(T^s_{BFS})$ to R_s, and divide V into different layers L_0, L_1, ..., L_{R_s}.
4: Apply Algorithm 2 to construct the data aggregation tree T^s_A inwardly rooted at node s, and to get the array P to maintain every node's inward parent node.
5: Apply Algorithm 3 to achieve the scheduling of aggregating the data to node s.

The fourth step is to construct the data aggregation tree T^s_A inwardly rooted at node s. The pseudocode of this step is shown in Algorithm 2. T^s_A is constructed based on the connected dominated set (CDS) similar to that used in [18]. The construction of CDS contains two processes. During the first

process, we find a dominating set which consists of independent sets. First, we construct the IS'es B_0, B_1, ..., B_{R_s} layer by layer. The first layer L_0 only contains node s, so B_0 is $\{s\}$. In each layer L_i ($1 \leq i \leq R_s$), we construct the IS B_i of $G[L_i]$ such that $\bigcup_{0\leq j\leq i} B_j$ is an MIS of $G[\bigcup_{0\leq j\leq i} L_j]$. Note that, $\bigcup_{0\leq i\leq R_s} B_i$ is an MIS of $G[\bigcup_{0\leq i\leq R_s} L_i] = G$, which is a dominating set of G.

Algorithm 2 Construct the data aggregation tree

$B_0 \leftarrow \{s\}$
for $i \leftarrow 1$ to R_s **do**
    Construct an IS B_i of $G[L_i]$ such that $\bigcup_{0\leq j\leq i} B_j$ is an MIS of $G[\bigcup_{0\leq j\leq i} L_j]$.
$T_A \leftarrow (V_A, E_A)$, $V_A \leftarrow V$, $E_A \leftarrow \emptyset$
for $i \leftarrow 1$ to $R_s - 1$ **do**
    Find a minimal cover $U_i \subseteq L_i$ of B_{i+1}.
    for each node $v \in B_{i+1}$ **do**
        Find one of its neighboring nodes $u \in U_i$.
        $P(v) \leftarrow u$, $E_A \leftarrow E_A \bigcup (v, P(v))$
    Find a minimal cover $C \subseteq B_{i-1} \bigcup B_i$ of U_i.
    for each node $u \in U_i$ **do**
        Find one of its neighboring nodes $x \in C$.
        $P(u) \leftarrow x$, $E_A \leftarrow E_A \bigcup (u, P(u))$
for $i \leftarrow 1$ to R_s **do**
    $W_i \leftarrow L_i \backslash \{B_i \bigcup U_i\}$
    Find a minimal cover $C \subseteq B_{i-1} \bigcup B_i$ of W_i.
    for each node $w \in W_i$ **do**
        Find one of its neighboring nodes $x \in C$.
        $P(w) \leftarrow x$, $E_A \leftarrow E_A \bigcup (w, P(w))$
return T_A and P

During the second process, we find some nodes to connect the nodes in the IS'es. In each layer L_i ($1 \leq i \leq R_s - 1$), we find a minimal cover $U_i \subseteq L_i$ of B_{i+1}, and set the nodes in U_i as the parent nodes of nodes in B_{i+1}. In each layer L_i ($1 \leq i \leq R_s$), we collect the nodes, which are in L_i but neither in B_i nor in U_i, into W_i. It is easy to find that $B_{i-1} \bigcup B_i$ is a cover of $L_i \backslash B_i = U_i \bigcup W_i$, since otherwise some nodes in $L_i \backslash B_i$ can be chosen into B_i which destroys the property of MIS. Since $B_{i-1} \bigcup B_i$ should be a cover of U_i, we find a minimal cover $C \subseteq B_{i-1} \bigcup B_i$ of U_i, and set the nodes in C as the parent nodes of nodes in U_i. Using this method, we can ultimately achieve a CDS $\bigcup_{0\leq i\leq R_s}(B_i \bigcup U_i)$. In each layer L_i ($1 \leq i \leq R_s$), since $B_{i-1} \bigcup B_i$ is a cover of W_i, we first find a minimal cover $C \subseteq B_{i-1} \bigcup B_i$ of W_i, and then set the nodes in C as the parent

nodes of nodes in W_i.

The final step is to schedule the data aggregation based on the tree T_A^s. The pseudocode of this step is shown in Algorithm 3. The scheduling contains two processes. During the first process, all the nodes outside the CDS (collected in $W = \bigcup_{1 \le i \le R_s} W_i$) aggregate their data to their parent nodes. In the second process, the nodes in the CDS aggregate their data from the bottom layer to the top layer. In each layer L_i ($1 \le i \le R_s$), nodes in U_i first aggregate their data to their parent nodes which belong to $B_{i-1} \bigcup B_i$. Nodes in B_i then aggregate their data to their parent nodes which belong to U_{i-1}.

Algorithm 3 Data aggregation

1: $t \leftarrow 0$
2: $W \leftarrow \bigcup_{1 \le i \le R_s} W_i$
3: Apply Algorithm 4 with $X = W$ and t to achieve the scheduling of data aggregation from nodes in W to their parent nodes and to get the ending time t_e.
4: $t \leftarrow t_e$
5: **for** $i \leftarrow R_s$ down to 1 **do**
6: Apply Algorithm 4 with $X = U_i$ and t to achieve the scheduling of data aggregation from nodes in U_i to their parent nodes and to get the ending time t_e.
7: $t \leftarrow t_e$
8: Apply Algorithm 4 with $X = B_i$ and t to achieve the scheduling of data aggregation from nodes in B_i to their parent nodes and to get the ending time t_e.
9: $t \leftarrow t_e$
10: **return** TTS

We schedule the data aggregation from the nodes in a set X to their parent nodes as shown in Algorithm 4. Since the parent nodes can only receive the data from their children nodes when these parent nodes are active, we first divide the nodes in X into different subsets X_0, X_1, ..., $X_{|T|-1}$ according to the active time slots of their parent nodes. We then schedule the transmissions from the children nodes in each X_j ($0 \le j \le |T| - 1$) to their parent nodes collected in Y.

To avoid the interference of the transmissions, we schedule the transmissions from the children nodes to their parent nodes based on the colors of the nodes (the children nodes or the parent nodes) in the MIS $\bigcup_{0 \le i \le R_s} B_i$. To improve the data aggregation latency, we do not directly use these colors to assign the transmitting time slots of the children nodes. Instead, we collect

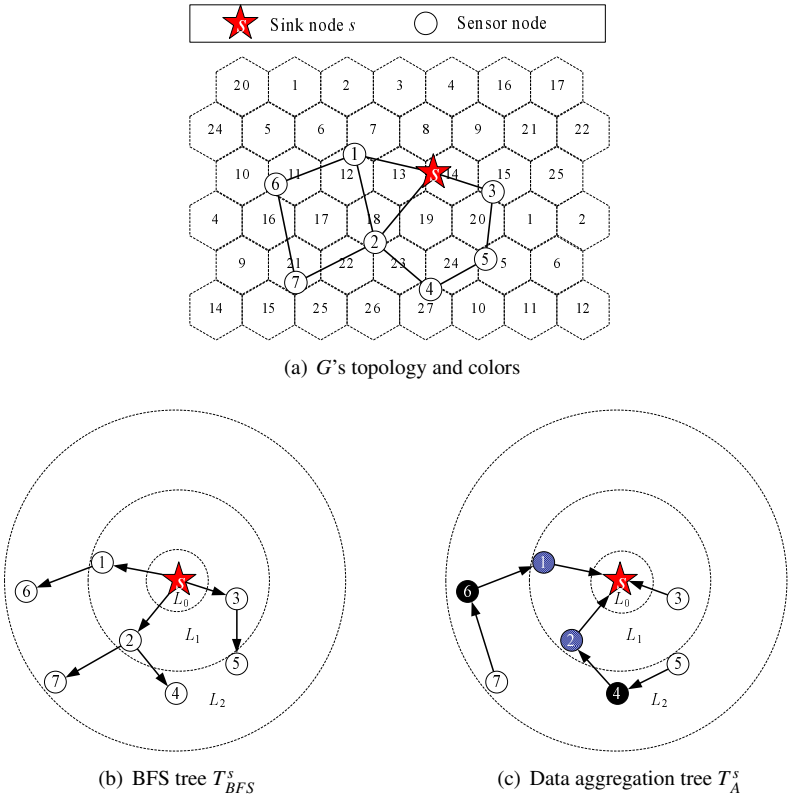
the colors of the nodes in the MIS $\bigcup_{0\le i\le R_s} B_i$ into a color set F, and use the indexes of these colors in F to schedule the transmissions.

The scheduling works iteratively. In each iteration, each parent node y in Y receives one of its children nodes x at time slot $t_j + (g(z) - 1)|T| + j$, where t_j is the beginning time slot of each iteration, $g(z)$ is the index of the color of node z (which is node x if node x belongs to the MIS, otherwise is node y) in F and j is the active time slot of the parent node y. Afterward, t_j advances to multiple times of $|T|$ when all these transmissions in this iteration can finish. After all the transmissions are scheduled, we can achieve that the ending time slot t_e equals to $\max_{0\le j\le|T|-1} t_j$ when all the transmissions can finish.

Algorithm 4 Nodes in X aggregate data to their parent nodes at time slot t

1: **for** each node $x \in X$ **do**
2: Collect x into $X_{A(P(x))}$.
3: **for** $j \leftarrow 0$ to $|T| - 1$ **do**
4: $t_j \leftarrow t$
5: **while** $X_j \neq \emptyset$ **do**
6: $Y \leftarrow \{P(x)|x \in X_j\}$, $t' \leftarrow t_j$, $X' \leftarrow \emptyset$
7: **for** each node $y \in Y$ **do**
8: Find one of its children nodes x in X_j.
9: $X' \leftarrow X' \bigcup \{x\}$.
10: Set Z as X' if X' contains the nodes in the MIS $\bigcup_{0\le i\le R_s} B_i$, and otherwise set Z as Y.
11: $F \leftarrow \{f(z)|z \in Z\}$
12: Set $g(z)$ as the index of $f(z)$ in F for each node z.
13: **for** each node $x \in X'$ **do**
14: Find the corresponding index $g(z)$ to node x or its parent node y in Y.
15: $TTS(x) \leftarrow TTS(x) \bigcup \{t_j + (g(z) - 1)|T| + j\}$
16: **if** $t' < t_j + (g(z) - 1)|T| + j + 1$ **then**
17: $t' \leftarrow t_j + (g(z) - 1)|T| + j + 1$
18: $X_j \leftarrow X_j \backslash \{x\}$
19: $t_j \leftarrow \lceil t'/|T| \rceil |T|$
20: $t_e \leftarrow \max_{0\le j\le|T|-1} t_j$
21: **return** TTS and t_e

Example 1. We take an example to illustrate the SDAS algorithm. The network consists of one sink node s and seven sensor nodes as shown in Figure 2(a). The pentagram denotes the sink node s and the circles denote the sensor nodes. The scheduling period T contains ten time slots from 0 to 9.



(a) G's topology and colors

(b) BFS tree T^s_{BFS}

(c) Data aggregation tree T^s_A

FIGURE 2
An example to illustrate the SDAS algorithm

Every node independently and randomly chooses its active time slot from the ten time slots. Table 1 lists the active time slots of all the nodes.

Using SDAS algorithm, we first apply a proper 27-coloring ($\alpha = 2$) scheme to color all the nodes. As shown in Figure 2(a), every node is in a hexagon, and is colored with the number of this hexagon, e.g., node 1's color $f(1)$ is 12. Table 1 lists the colors of all the nodes. Next, we construct the BFS tree T^s_{BFS} rooted at sink node s, and divide all the nodes into different layers as shown in 2(b). In this example, $L_0 = \{s\}$, $L_1 = \{1, 2, 3\}$ and $L_2 = \{4, 5, 6, 7\}$.

Then we apply Algorithm 2 to construct the data aggregation tree T^s_A. Figure 2(c) demonstrates T^s_A. The sink node and the black nodes 4 and 6 belong

TABLE 1
Active time slots and colors of all the nodes

Node ID	s	1	2	3	4	5	6	7
Active time slot	3	6	7	5	2	8	9	5
Color	14	12	18	15	27	5	11	21

to B_0 and B_2 respectively, and the plaid nodes 1 and 2 belong to U_1. The white nodes 3, 5 and 7 belong to W_1 and W_2 respectively. The CDS consists of the sink node and nodes 1, 2, 4, 6.

Afterward, we schedule the data aggregation based on Algorithm 3. The current time slot t is 0. The white nodes 3, 5 and 7 first aggregate their data to their parent nodes s, 4 and 6 respectively. According to Algorithm 4, nodes 3, 5 and 7 are divided into different subsets according to their parent nodes' active time slots. In this example, $X_2 = \{5\}$, $X_3 = \{3\}$ and $X_9 = \{7\}$.

So we first schedule the time of the transmission from the node in X_2 to its parent node, i.e., from node 5 to its parent node 4. Since node 4 belongs to B_2, its color $f(4)$ is collected into F, i.e., $F = \{27\}$. Hence the index $g(4)$ of node 4's color $f(4)$ in F equals to 1. Next we schedule the transmitting time of node 5 at time slot $t_2 + (g(4) - 1)|T| + 2 = t + (1 - 1) * 10 + 2 = 2$. Since the transmission takes one time slot, we can achieve that the transmission of the node in X_2 can finish at time slot 3, i.e., $t' = 3$. t_2 advances to multiple times of $|T|$ when the transmission of the node in X_2 can finish, so $t_2 = \lceil t'/|T| \rceil |T| = 10$.

Similarly, we can get that the transmissions from node 3 to its parent node s and from node 7 to its parent node 4 are scheduled at time slots 3 and 9 respectively. Moreover, both t_3 and t_9 advance to time slot 10. So the current time slot t advances to time slot $\max\{t_2, t_3, t_9\} = 10$.

The nodes in the CDS then aggregate the data to their parent nodes from the bottom layer to the top layer. In the layer L_2, the black nodes 4 and 6 in B_2 aggregate their data to their parent nodes 2 and 1 respectively. According to Algorithm 4, nodes 4 and 6 are divided into two subsets X_7 and X_6 because $A(2) = 7$ and $A(1) = 6$. X_6 is handled before X_7, and we collect node 6's color $f(6)$ into F, so $F = \{11\}$ and $g(6) = 1$. We schedule the transmission time of node 6 at time slot $t_6 + (g(6) - 1)|T| + 6 = t + (1 - 1) * 10 + 6 = 16$. Afterward, t' equals to 17, and t_6 advances to time slot $\lceil t'/|T| \rceil |T| = 20$. Similarly, we can achieve that node 4 will aggregate its data to node 2 at time

slot $t_7 + (g(4) - 1)|T| + 7 = t + (1 - 1) * 10 + 7 = 17$, and t_7 advances to time slot 20. The current time slot t advances to time slot $\max\{t_6, t_7\} = 20$.

In layer L_1, nodes 1 and 2 in U_1 aggregate their data to the sink node s. First, they are collected into one subset X_3 since they have the same parent node. The data aggregation works iteratively. In the first iteration, we assume that node 1 first aggregates the data to node s. Since node s belongs to B_0, its color is collected into F, i.e., $F = \{14\}$. Therefore $g(s)$ equals to 1. We schedule the transmission time of node 1 at time slot $t_3 + (g(s) - 1)|T| + 3 = t + (1 - 1) * 10 + 3 = 23$. t' advances to time slot 24, and t_3 advances to time slot $\lceil t'/|T| \rceil |T| = 30$.

In the second iteration, node 2 aggregates its data to node s. We collect node s's color into F, and hence $g(s) = 1$. Then we schedule the transmission time of node 2 at time slot $t_3 + (g(s) - 1)|T| + 3 = 30 + (1 - 1) * 10 + 3 = 33$. Note that, the data aggregation finishes one time slot after node 2 aggregates the data to node s, so the data aggregation latency is 34 time slots.

4.2 Center assisted Data Aggregation Scheduling

In this subsection, we detail the Center assisted Data Aggregation Scheduling (CDAS) algorithm. CDAS algorithm contains six steps, and the pseudocode of this algorithm is shown in Algorithm 5.

Algorithm 5 CDAS Algorithm

Input: $G = (V, E)$, s, A, α, r, T.
Output: Aggregation Scheduling $TTS : V \to 2^{\mathcal{N}}$.

1: Apply a proper tessellation and $3\beta^2$-coloring of hexagons with a radius of $r/2$ in the whole area to color all the nodes. Use $f : V \to \{1, 2, ..., 3\beta^2\}$ to denote this coloring method, where β is $\lceil \frac{2}{3}(\alpha + 2) \rceil$.
2: Find the graph center c and construct the BFS tree T^c_{BFS} rooted at node c.
3: Assign $MaxDepth(T^c_{BFS})$ to R, and divide V into different layers L_0, L_1, ..., L_R.
4: Apply Algorithm 2 with $R_s = R$ to construct the data aggregation tree T^c_A inwardly rooted at node c, and to get the array P to maintain every node's inward parent node.
5: Apply Algorithm 3 with $R_s = R$ to achieve the scheduling of aggregating the data to node c. Do not schedule the transmission of node s to its parent node if node s does not have children nodes in T^c_A.
6: Schedule node c to transmit the aggregated data to node s along the shortest path between node c and node s in T^c_{BFS}.

Similar to the SDAS algorithm, the CDAS algorithm first applies a proper

coloring method to color all the nodes. We then find the graph center c by using the Floyd-Warshall algorithm [5]. Next we construct the BFS tree T^c_{BFS} rooted at node c, and divide all the nodes into different layers L_0, L_1, ..., L_R, where R is the graph-theoretic radius of the network. Algorithm 2 is also used to construct the data aggregation tree T^c_A inwardly rooted at node c.

Afterward, we schedule all the sensor nodes to aggregate their data to the graph center c by applying Algorithm 3. Note that we require modifying R_s to R in this case. If the sink node s does not have children nodes in T^c_A, it does not need to forward the data of other sensor nodes, and hence we do not schedule the transmission from node s to its parent node. Finally, when node c receives all the data from other sensor nodes, we schedule this node to transmit the aggregated data to node s along the shortest path between node c and node s in T^c_{BFS}.

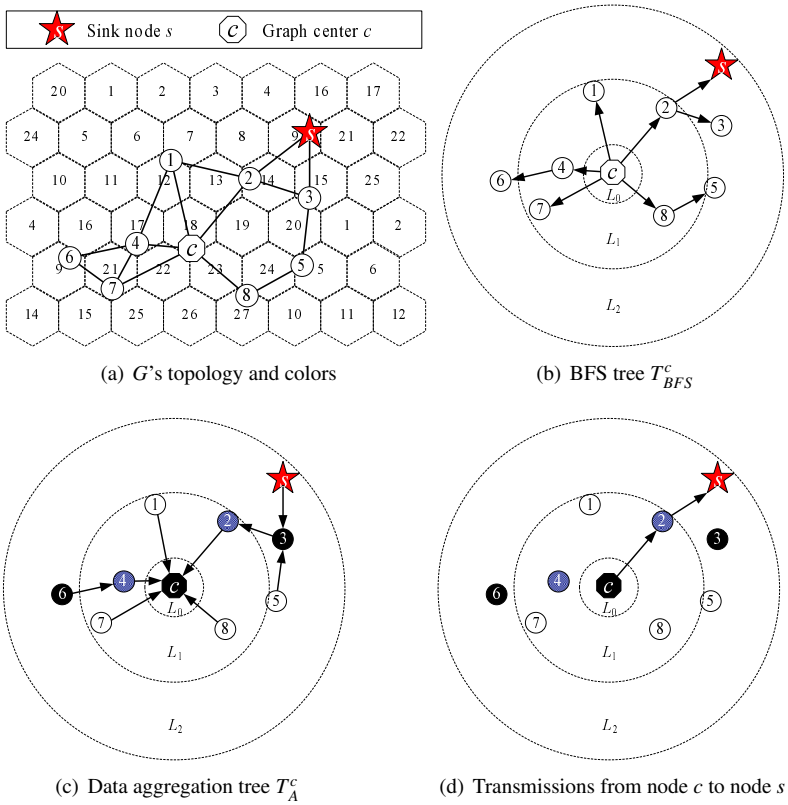
Example 2. We take an example to illustrate the CDAS algorithm. The network consists of one sink node s and nine sensor nodes. The network topology of G is shown in Figure 3(a). The pentagram and the octagon denote the sink node s and the graph center c respectively. The scheduling period T contains ten time slots from 0 to 9. Ten nodes independently and randomly choose their active time slots from these ten time slots. The active time slots of ten nodes are listed in Table 2. The current time t is time slot 0.

According to Algorithm 5, we first color all the nodes by a proper tessellation and 27-coloring ($\alpha = 2$) of hexagons as shown in Figure 3(a), e.g., the color of node 6 is 9. The colors of all the nodes are listed in Table 2. We then construct the BFS tree T^c_{BFS} rooted at node c, and divide all the nodes into different layers as shown in Figure 3(b). In this example, $L_0 = \{c\}$, $L_1 = \{1, 2, 4, 7, 8\}$ and $L_2 = \{s, 3, 5, 6\}$.

Next, we construct the data aggregation tree T^c_A as shown in Figure 3(c). The black nodes c, 3 and 6 belong to the IS'es B_0 and B_2 respectively, and the plaid nodes 2 and 4 belong to U_1. The white nodes 1, 7, 8, 5 and sink node s belong to W_1 and W_2 respectively. The CDS consists of the black nodes and the plaid nodes.

Afterward, we schedule the data aggregation from all the nodes to node c by applying Algorithm 3 with $R_s = R$. The sink node s has no children nodes, and hence the transmission from node s to its parent node 3 is not scheduled. The white nodes 1, 5, 7 and 8 first aggregate their data to their parent nodes in T^c_A, and are divided into different subsets according to their parent nodes' active time slots. In this example, $X_2 = \{5\}$ and $X_6 = \{1, 7, 8\}$.

According to Algorithm 4, node 5 in X_2 is first scheduled to aggregate its data to its parent node 3. The color of node 3 is collected into a color set F,



(a) G's topology and colors

(b) BFS tree T^c_{BFS}

(c) Data aggregation tree T^c_A

(d) Transmissions from node c to node s

FIGURE 3
An example to illustrate the CDAS algorithm

i.e., $F = \{15\}$. Since F only contains node 3's color $f(3)$, $g(3)$ equals to 1. We schedule the transmitting time of node 3 at time slot $t_2 + (g(3) - 1)|T| + 2 = t + (1 - 1) * 10 + 2 = 2$. The transmission of the node in X_2 finishes at time slot $2 + 1 = 3$, i.e., $t' = 3$. Therefore t_2 advances to next scheduling period when the transmission finishes, i.e., $t_2 = \lceil t'/|T| \rceil |T| = 10$.

Next, nodes 1, 7 and 8 in X_6 aggregate their data to node c iteratively. In each iteration, only one node is scheduled to aggregate its data to node c. We assume that three nodes follow the order of 1, 7 and 8 to carry out the data aggregation. According to Algorithm 4, in each iteration, the color of node c is collected into F, i.e., $F = \{f(c)\} = \{18\}$, and $g(c)$ is set as 1.

TABLE 2
Active time slots and colors of ten nodes

Node ID	s	c	1	2	3	4	5	6	7	8
Active time slot	3	6	7	5	2	8	9	5	8	2
Color	9	18	12	14	15	17	5	9	21	27

In the first iteration, the transmitting time of node 1 is scheduled at time slot $t_6+(g(c)-1)|T|+6 = t+(1-1)*10+6 = 6$. Then t_6 advances to next scheduling period when this transmission finishes, i.e., $t_6 = 10$. In the second iteration, the transmitting time of node 7 is scheduled at time slot $t_6+(g(c)-1)|T|+6 = 16$, and t_6 advances to time slot 20. In the third iteration, the transmitting time of node 8 is scheduled at time slot $t_6 + (g(c) - 1)|T| + 6 = 26$, and t_6 advances to time slot 30. Then the current time slot t advances to time slot $\max\{t_2, t_6\} = 30$.

The nodes in the CDS then aggregate their data to their parent nodes from the bottom layer to the top layer. In the bottom layer L_2, U_2 is empty, and B_2 contains two nodes 3 and 6. According to Algorithm 4, node 3 and node 6 are divided into two subsets X_5 and X_8 respectively based on the active time slots of their parent nodes 2 and 4. First, the color of node 3 is collected into F, i.e., $F = \{f(3)\} = \{15\}$, and $g(3)$ is set as 1. Next, the transmitting time of node 3 is scheduled at time slot $t_5+(g(3)-1)|T|+5 = t+(1-1)*10+5 = 35$. t_5 advances to time slot 40. For the subset X_8, the color of node 6 is collected into F, i.e., $F = \{f(6)\} = \{9\}$, and $g(6)$ is set as 1. The transmitting time of node 3 is scheduled at time slot $t_8+(g(6)-1)|T|+8 = t+(1-1)*10+8 = 38$. t_8 advances to time slot 40. The current time slot t advances to time slot $\max\{t_5, t_8\} = 40$.

In the layer L_1, B_1 is empty, and U_1 contains two nodes 2 and 4. According to Algorithm 4, nodes 2 and 4 are both collected into one subset X_6 since they have the same parent node c. These two nodes aggregate their data to node c in two iterations respectively. In each iteration, the color of node c is collected into F, so $F = \{f(c)\} = \{18\}$ and $g(c)$ is set as 1. In the first iteration, we assume that node 2 first aggregates its data to node c. The transmitting time of node 2 is scheduled at time slot $t_6+(g(c)-1)|T|+6 = t+(1-1)*10+6 = 46$. t_6 advances to time slot 50. In the second iteration, we can achieve that the transmitting time of node 4 is scheduled at time slot $t_6 + (g(c) - 1)|T| + 6 =$

$t + (1 - 1) * 10 + 6 = 56$. t_6 advances to time slot 60. The current time t advances to time slot $\max\{t_6\} = 60$

Finally, node c transmits the aggregated data to node 2 at time slot $t + A(2) = 60 + 5 = 65$, which forwards the data to node s at time slot 73. Figure 3(d) illustrates this process. The whole data aggregation process finishes one time slot after node 2 transmits the aggregated data to node s, so the total data aggregation latency is 74 time slots.

5 PERFORMANCE ANALYSIS

Theorem 1. *The SDAS algorithm provides a correct and interference-free data aggregation scheduling.*

Proof. In the SDAS algorithm, the data of nodes outside the CDS is first aggregated to their parent nodes in the CDS, and then the data of nodes in the CDS is aggregated layer by layer. In each layer L_i, nodes in U_i aggregate the data to their parent nodes in $B_{i-1} \bigcup B_i$, and afterward nodes in B_i aggregate the data to their parent nodes in U_{i-1}. So the sink node will ultimately receive the data from all the sensor nodes. According to the tessellation and coloring method discussed in Section 3.3, the distance between two nodes in an IS with the same color should be larger than $r_f + r$. It is easy to prove that the transmissions to these two nodes or from these two nodes are interference-free. Since all the transmissions are scheduled based on the colors of nodes in the MIS $\bigcup_{0 \le i \le R_s} B_i$, these transmissions are interference-free. □

Theorem 2. *The CDAS algorithm provides a correct and interference-free data aggregation scheduling.*

Proof. The CDAS algorithm contains two processes. During the first process, the data is aggregated to the graph center c. This process is similar to that of the SDAS algorithm. Therefore we can use the similar proof to that used in previous theorem to prove that node c can receive the data from all the sensor nodes, and the transmissions during this process are interference-free. During the second process, node c transmits the data to the sink node s. Since the network is connected, there should be a path between node c and node s and the nodes in the path transmit the data one by one. Therefore node s can ultimately receive the data and the transmissions scheduled during this process are interference-free. □

Theorem 3. *The data aggregation latency of the SDAS algorithm is at most* $3\beta^2|T|(15R_s + \Delta - 3)$.

Proof. First, we consider the latency of Algorithm 4. Nodes in X aggregate their data to their parent nodes iteratively. The transmissions to parent nodes with different active time slots are separated. Since each parent node can receive the data of only one of its children nodes during one iteration, the parent node y with the most children nodes will always exist in the set Y during all the iterations, and the number of its children nodes in the set X_j will decrease by one after each iteration. So the number of iterations is bounded by the number of the children nodes of node y. Since the indexes of colors in F are no larger than $3\beta^2$ and the active time slot of a node is no larger than $|T| - 1$, the latency of transmissions in one iteration is at most $(3\beta^2 - 1)|T| + (|T| - 1) + 1 = 3\beta^2|T|$.

Nodes outside the CDS first aggregate their data to their parent nodes. Since these parent nodes have at most Δ children nodes in W, the number of iterations is at most Δ and the latency of these transmissions is at most $3\beta^2|T|\Delta$. According to [18], one parent node of nodes in U_i has at most 11 children nodes in U_i, and one parent node of nodes in B_i has at most 4 children nodes in B_i if $2 \leq i \leq R_s$. Moreover, B_2 is empty and the sink node s has at most 12 children nodes in U_2. Therefore the latency of transmissions from the bottom layer to the top layer is at most $\sum_{i=2}^{R_s} 3\beta^2|T|(11+4) + 3\beta^2|T| * 12 = 3\beta^2|T|(15R_s - 3)$. We combine two kinds of latency, and achieve that the data aggregation latency of the SDAS algorithm is at most $3\beta^2|T|(15R_s+\Delta-3)$. □

Theorem 4. *The SDAS algorithm has an approximation ratio of at most* $3\beta^2|T|(\Delta + 12)$.

Proof. According to Theorem 3, the data aggregation latency of the SDAS algorithm is at most $3\beta^2|T|(15R_s + \Delta - 3)$. We can observe from the BFS tree T^s_{BFS} that the data of the nodes in the bottom layer L_{R_s} should be transmitted R_s times to reach the sink node s. Moreover, the latency of each transmission is at least one time slot, so we can claim that the data aggregation latency of the optimal solution is at least R_s. Since $R_s \geq 1$, it follows that $3\beta^2|T|(15R_s + \Delta - 3) \leq 3\beta^2|T|[15R_s + (\Delta - 3)R_s] = 3\beta^2|T|(\Delta + 12)R_s$, and hence this theorem holds. □

Theorem 5. *The data aggregation latency of the CDAS algorithm is at most* $(45\beta^2 + 1)|T|R + 3\beta^2|T|(\Delta - 3)$.

Proof. Using the similar proof to that used in Theorem 3, we can achieve that the latency of the first process in the CDAS algorithm is at most $3\beta^2|T|(15R + \Delta - 3)$. During the second process, we consider the worst case where the

latency of each hop in the shortest path between node c and node s is $|T|$. Since the hop distance of this path is bounded by R, the latency of the second process is at most $R|T|$. We combine the latency of two processes, and achieve that the data aggregation latency of the SDAS algorithm is at most $(45\beta^2 + 1)|T|R + 3\beta^2|T|(\Delta - 3)$. □

Theorem 6. *The CDAS algorithm has an approximation ratio of at most* $(3\beta^2\Delta + 36\beta^2 + 1)|T|$.

Proof. According to Theorem 5, the data aggregation latency of the CDAS algorithm is at most $(45\beta^2+1)|T|R+3\beta^2|T|(\Delta-3)$. According to the definition of the graph-theoretical radius R, it follows that $R \leq R_s$. Moreover, $R \geq 1$, so it follows that,

$$\begin{aligned}
&(45\beta^2 + 1)|T|R + 3\beta^2|T|(\Delta - 3)\\
\leq&(45\beta^2 + 1)|T|R + 3\beta^2|T|(\Delta - 3)R\\
=&(3\beta^2\Delta + 36\beta^2 + 1)|T|R\\
\leq&(3\beta^2\Delta + 36\beta^2 + 1)|T|R_s.
\end{aligned}$$

Based on the proof of Theorem 4, we can achieve that the data aggregation latency of the optimal solution is at least R_s, and hence this theorem holds. □

Theorem 7. *The total number of transmissions scheduled by the SDAS algorithm is* $n - 1$.

Proof. According to the SDAS algorithm, every sensor node only transmits once to aggregate the data to its parent node, so the total number of transmissions scheduled by this algorithm is $n - 1$. □

Theorem 8. *The total number of transmissions scheduled by the CDAS algorithm is at most* $n + R - 1$.

Proof. The first process of the CDAS algorithm is similar to that of the SDAS algorithm and only the graph center c does not transmit in this process, so the number of transmissions during this process is at most $n-1$ according to Theorem 7. During the second process, the number of transmissions is bounded by the hop distance of the shortest path between node c and node s. This hop distance is at most R, so we combine the number of transmissions during two processes and achieve that the total number of transmissions scheduled by the CDAS algorithm is at most $n + R - 1$. □

Theorem 9. *The time complexity of the SDAS algorithm is* $O(n^2)$.

Proof. The first step in the SDAS algorithm is to apply a proper tessellation and $3\beta^2$-coloring of hexagons to color the nodes. It takes $O(\frac{S_A}{r^2})$ time to tessellate and color the hexagons, where S_A denotes the area size of the whole area, and takes $O(n)$ time to color all the nodes. We can regard $O(\frac{S_A}{r^2})$ as $O(1)$ if n is large. The next step is to construct the BFS tree rooted at node s. It takes $O(n^2)$ time to do this according to [2]. The running time of dividing all the nodes into different layers is $O(n)$. It takes $O(n^2)$ time to construct the data aggregation tree and to schedule the data aggregation. So the time complexity of the SDAS algorithm is $O(1)+O(n)+O(n^2)+O(n)+O(n)+O(n^2) = O(n^2)$. □

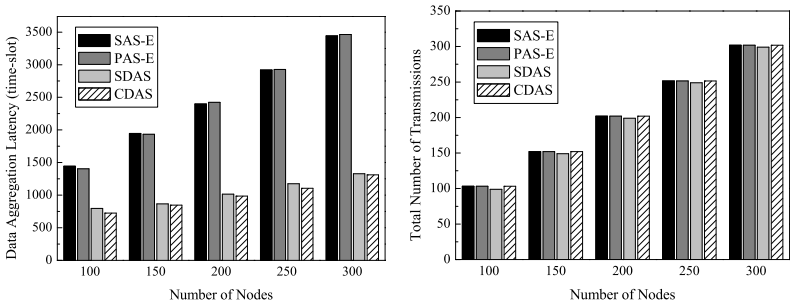
Theorem 10. *The time complexity of the CDAS algorithm is $O(n^3)$.*

Proof. The main additional steps in the CDAS algorithm compared to the SDAS algorithm are to find the graph center c and to schedule the transmissions from node c to node s along the shortest path. The running time of these two steps is $O(n^3)$ and $O(n)$ respectively. So, based on Theorem 9, we can achieve that the time complexity of the CDAS algorithm is $O(n^3)$. □

6 PERFORMANCE EVALUATION

In this section, we conduct extensive simulations to evaluate the performance of our SDAS and CDAS algorithms. Since SAS and PAS algorithms can be extended by a generic expansion technology to solve the DAS problem under protocol interference model [18], and the extended versions of these two algorithms are so far the best two algorithms, we further extend the extended versions of these two algorithms to solve the DAS-UDC problem under protocol interference model. We call the extended algorithms as SAS-E and PAS-E respectively, and compare the performance of our algorithms with that of SAS-E and PAS-E algorithms. The extending approach is as follows. For the links scheduled to transmit in the same time slot by the extended versions of the SAS and PAS algorithms, we schedule these links to transmit at the active time slots of the receiver nodes during one scheduling period.

We randomly deploy all the nodes in a rectangle area of 200 $m \times 200m$. These nodes have the same transmission radius. We test the data aggregation latency and the total numbers of transmissions of four algorithms. The data aggregation latency is the total time slots required by the sink node to receive the data from all the sensor nodes. We study the effect of different network configurations including the network size, the transmission radius, the duty cycle and the interference ratio on the performance of four algorithms.



(a) Data aggregation latency (b) Total number of transmissions

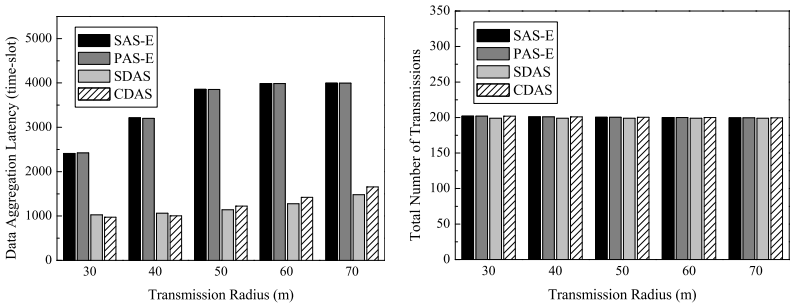
FIGURE 4
Performance variation under different numbers of nodes

The network size ranges from 100 to 300 with an interval of 50. We vary the range of the transmission radius from 30 m to 70 m. The number of time slots in T increases from 10 to 50 with an interval of 10, and the duty cycle which equals to $1/|T|$ varies between 0.1 and 0.02. The experiments are conducted with one configuration changed and the other three fixed. These experiments are run on 20 randomly generated graph topologies. Moreover, we carry out the experiments 10 times for each graph topology and randomly choose one node as the sink node in each experiment. The average performance of these experiments is reported.

6.1 Impact of Network Size

First, we evaluate the impact of the network size on the performance of four algorithms. The transmission radius is fixed to 30 m, the duty cycle is set as 0.05 with $|T| = 20$ and the interference ratio α is set as 3. Figure 4 illustrates the performance variation of four algorithms under different numbers of nodes. We can observe from Figure 4(a) that the data aggregation latency increases with the increase of the network size. This is because more nodes have to send its data to the sink node, and a node can only receive the data from one node in one scheduling period when it is active.

Note that our two algorithms perform better than SAS-E and PAS-E algorithms in terms of data aggregation latency especially when the network size is large. The reason is that, our algorithms separate the transmissions better than SAS-E and PAS-E algorithms according to the active time slots of the receiver nodes. Another observation is that CDAS outperforms SDAS in



(a) Data aggregation latency (b) Total number of transmissions

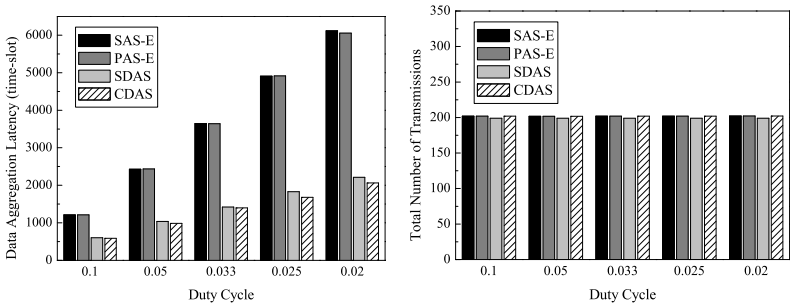
FIGURE 5
Performance variation under different transmission radiuses

these experiments. This is because the transmission radius is small, and the network diameter may be large. If the hop distance between the sink node and the farthest node is large, the number of layers in the BFS tree T^s_{BFS} is large and the latency of the transmissions scheduled layer by layer will be high.

Figure 4(b) illustrates the performance variation of four algorithms in terms of the total number of transmissions. It requires more transmissions for more nodes to finish the data aggregation operation, so the total numbers of transmissions increase with the increase of the network size. Note that, since all the three algorithms SAS-E, PAS-E and CDAS first schedule the sensor nodes to transmit its data to the graph center, and then schedule the graph center to transmit the aggregated data to the sink node, the total numbers of transmissions of these three algorithms are equal. SDAS directly aggregates the data to the sink node, so the total number of transmissions scheduled by this algorithm is smaller than those of the other three algorithms. The results also validate the theoretical analysis in previous section.

6.2 Impact of Transmission Radius

Next, we study the performance variation of four algorithms with different transmission radiuses. Figure 5 shows the results for the experiments with 200 nodes, the duty cycle of 0.05 and the interference ratio of 3. When the transmission radius increases, more transmissions interfere with each other and hence have to be separated. Therefore, the data aggregation latency increases with the increase of the transmission radius as shown in Figure 5(a). Our algorithms SDAS and CDAS perform better than SAS-E and PAS-E al-



(a) Data aggregation latency (b) Total number of transmissions

FIGURE 6
Performance variation under different duty cycles

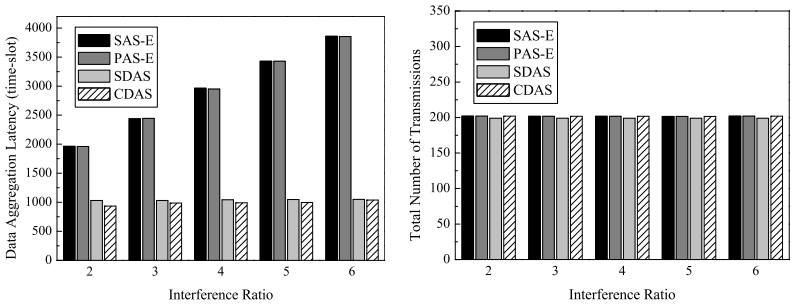
gorithms due to the similar reason discussed in previous subsection.

Note that, CDAS incurs higher latency than SDAS after the transmission radius reaches 50m. The reason is that, the network diameter gets smaller when the transmission radius increases. It bring fewer benefits to let the graph center assist the data aggregation, and the latency of the transmissions from the graph center to the sink node takes more adverse effects on the total data aggregation latency instead. Since the network size is fixed, the total numbers of transmissions change a little as shown in Figure 5(b).

6.3 Impact of Duty Cycle

In this subsection, we evaluate the impact of the duty cycle on the performance of four algorithms. These experiments are run with the network size of 200, the transmission radius of 30 m and the interference ratio of 3. The results of these experiments are shown in Figure 6.

When the duty cycle decreases, the number of time slots in a scheduling period increases. The sensor nodes may wait more time for their parent nodes to wake up before they aggregate the data to their parent nodes. Hence the performance curves of these four algorithms in terms of data aggregation latency trend up as shown in Figure 6(a). We can also observe from this figure that our algorithms incur lower latency than SAS-E and PAS-E algorithms, and CDAS performs better than SDAS because the transmission radius is small. Figure. 6(b) illustrates the variation of total numbers of transmissions scheduled by these four algorithms, which change a little because the number of nodes is fixed.



(a) Data aggregation latency (b) Total number of transmissions

FIGURE 7
Performance variation under different interference ratios

6.4 Impact of Interference Ratio

Finally, we evaluate the impact of the interference ratio on the performance of four algorithms. These experiments are run with the network size of 200, the transmission radius of 30 *m* and the duty cycle of 0.05. The results are shown in Figure 7. When the interference ratio grows, the interference range of a node enlarges, and hence fewer transmissions can be simultaneous. Therefore the data aggregation latency of all the four algorithms increases with the increase of the interference ratio as shown in Figure 7(a).

Another observation is that, our algorithms perform better than SAS-E and PAS-E algorithms. The latency of our algorithms increases slowly, while the latency of SAS-E and PAS-E algorithms increases significantly when the interference ratio grows. The reason is that, SAS-E and PAS-E algorithms determine whether two links interfere with each other only based on the distance between some two nodes of these links, and schedule the transmissions of these two links in different scheduling periods. In our algorithms, instead, if the receiver nodes of these two links have different active time slots, we can schedule the transmissions of these two links in one scheduling period. From Figure 7(b), we can see that the total number of transmissions scheduled by four algorithms change a little since the network size is fixed.

7 CONCLUSION AND FUTURE WORK

In this paper, we investigate the DAS-UDC problem under protocol interference model. We prove that this problem is NP-hard and propose two

approximation algorithms SDAS and CDAS. Both two algorithms provide correct and interference-free data aggregation schedulings. The data aggregation latency of SDAS and CDAS is bounded by $3\beta^2|T|(15R_s + \Delta - 3)$ and $(45\beta^2 + 1)|T|R + 3\beta^2|T|(\Delta - 3)$ respectively. The total numbers of transmissions scheduled by these two algorithms are $n - 1$ and at most $n + R - 1$ respectively. We also show that both two algorithms are polynomial time algorithms. The results of extensive simulations show that, our two algorithms SDAS and CDAS achieve lower data aggregation latency than extended versions of existing algorithms. In addition, CDAS outperforms SDAS in most scenarios, but performs worse than SDAS in terms of data aggregation latency when the transmission radius is large. Moreover, SDAS schedules fewer total number of transmissions than CDAS and extended versions of existing algorithms.

Although our algorithms cannot be directly used to solve the data aggregation problem under physical interference model, this paper provides certain guidance significance to the research under realistic interference models. As claimed in [10], by carefully selecting the transmission radius and the interference radius, we can transform the problem under physical interference model to the problem under protocol interference model. Therefore, using this method, we can extend our algorithms to solve the data aggregation problem under physical interference model, and we leave it as our future work.

REFERENCES

[1] X. J. Chen, X. D. Hu, and J. M. Zhu. (2005). Minimum data aggregation time problem in wireless sensor networks. *Lecture Notes in Computer Science 3794*, pages 133–142.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press.

[3] O. Dousse, P. Mannersalo, and P. Thiran. (2004). Latency of wireless sensor networks with uncoordinated power saving mechanisms. In *Proc. of ACM MobiHoc*.

[4] K.-W. Fan, S. Liu, and P. Sinha. (2008). Dynamic forwarding over tree-on-dag for scalable data aggregation in sensor networks. *IEEE Transactions on Mobile Computing*, 16(10):1271–1284.

[5] R. W. Floyd. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

[6] C. Gui and P. Mohapatra. (2004). Power conservation and quality of surveillance in target tracking sensor networks. In *Proc. of ACM MobiCom*.

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

[8] J. Hong, J. Cao, W. Li, S. Lu, and D. Chen. (2009). Sleeping schedule-aware minimum latency broadcast in wireless ad hoc networks. In *Proc. of IEEE ICC*.

[9] C. Hua and T.-S. P. Yum. (2007). Asynchronous random sleeping for sensor networks. *ACM Transactions on Sensor Networks*, 3(3):15.

[10] S. C.-H. Huang, P.-J. Wan, J. Deng, and Y. S. Han. (2008). Broadcast scheduling in interference environment. *IEEE Transactions on Mobile Computing*, 7(11):1338–1348.

[11] S. C.-H. Huang, P.-J. Wan, C. T. Vu, Y. Li, and F. Yao. (2007). Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *Proc. of IEEE INFOCOM*.

[12] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. (2002). Impact of network density on data aggregation in wireless sensor networks. In *Proc. of IEEE ICDCS*.

[13] X. Jiao, W. Lou, J. Ma, J. Cao, X. Wang, and X. Zhou. (2010). Duty-cycle-aware minimum latency broadcast scheduling in multi-hop wireless networks. In *Proc. of IEEE ICDCS*.

[14] X.-Y. Li, X.H. Xu, S.G. Wang, S.J. Tang, G.J. Dai, J.Z. Zhao, and Y. Qi. (2009). Efficient data aggregation in multi-hop wireless sensor networks under physical interference model. In *Proc. of IEEE MASS*.

[15] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. (2004). Medians and beyond: new aggregation techniques for sensor networks. In *Proc. of ACM SenSys*.

[16] X. Tang and J. Xu. (2008). Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16(4):904–917.

[17] P.-J. Wan, K. M. Alzoubi, and O. Frieder. (2004). Distributed construction of connected dominating set in wireless ad hoc networks. *Mob. Netw. Appl.*, 9(2):141–149.

[18] P.-J. Wan, S. C.-H. Huang, and L. Wang. (2009). Minimum-latency aggregation scheduling in multihop wireless networks. In *Proc. of ACM MobiHoc*.

[19] P.-J. Wan, Z. Wang, Z. Wan, S. C. H. Huang, and H. Liu. (2009). Minimum-latency schedulings for group communications in multi-channel multihop wireless networks. In *Proc. of WASA*.

[20] B. Wang and X. Jia. (2009). Reducing data aggregation latency by using partially overlapped channels in sensor networks. In *Proc. of IEEE GlobeCom*.

[21] F. Wang and J. Liu. (2009). Duty-cycle-aware broadcast in wireless sensor networks. In *Proc. of IEEE INFOCOM*.

[22] Y. Wu, X.-Y. Li, Y.H. Liu, and W. Lou. (2010). Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):275–287.

[23] X. H. Xu, S. G. Wang, X. F. Mao, S. J. Tang, P. Xu, and X.-Y. Li. (2009). Efficient data aggregation in multi-hop wsns. In *Proc. of IEEE GlobeCom*.

[24] Z. Ye, A. A. Abouzeid, and J. Ai. (2009). Optimal stochastic policies for distributed data aggregation in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 17(5):1494–1507.

[25] B. Yu, J. Li, and Y. Li. (2009). Distributed data aggregation scheduling in wireless sensor networks. In *Proc. of IEEE INFOCOM*.

[26] Y. Yu, B. Krishnamachari, and V. Prasanna. (2004). Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proc. of IEEE INFOCOM*.

[27] R. Zheng and R. Kravets. (2003). On-demand power management for ad hoc networks. In *Proc. of IEEE INFOCOM*.