Latency-Optimized Scheduling for Data Aggregation in Distributed Edge Computing

YUNQUAN GAO, School of Computer Science and Technology, Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet, Anhui University of Technology, China
QIYANG ZHANG*, School of Computer Science, Peking University, China
YING LI, College of Computer Science and Engineering, Northeastern University, China
PRAVEEN KUMAR DONTA, Department of Computer and Systems Sciences, Stockholm University, Sweden
LAURI LOVÉN, Center for Ubiquitous Computing, University of Oulu, Finland
SCHAHRAM DUSTDAR, Distributed Systems Group, TU Wien, Austria

In Wireless Sensor Networks (WSNs), relay sensor nodes can aggregate data from edge sensor node into a summary information before sending to the sink. Due to the vast number of sensor nodes in a distributed edge computing (DEC) network, these relay sensor nodes may receive a high number of aggregation requests. This increases the chance of conflicting transmissions, which further leads to unwanted latency. Designing a conflict-free and minimal latency data aggregation schedule remains an open question. Moreover, existing related works have been conducted in traditional WSNs. By leveraging multiple antennas, the Multiple Input Multiple Output (MIMO) and cooperative MIMO called virtual MIMO (V-MIMO) enable broadband wireless communication, thereby improving the performance of WSNs. However, compared with traditional WSNs, MIMO and V-MIMO introduce distinct interference models requiring careful consideration. The work proposes a solution to an NP-hard problem, addressing three challenges: (i) interference; (ii) latency; and (iii) dynamic changes in network topology. Firstly, to counter interference, we propose a model where multiple nodes can simultaneously send data to the same parent by connecting different antennas. Secondly, to minimize latency, we propose a novel distributed heuristic data aggregation scheduling method , which intertwines the construction of an optimal data aggregation tree and conflict-free scheduling. Finally, to handle dynamic network topology changes, we propose lightweight adaptive strategies that do not increase data aggregation latency. Simulation results and theoretical analysis demonstrate superior performance in reducing data aggregation latency. When compared with state-of-the-art solutions, our proposed method decreases data aggregation latency by at least 2.6× on average.

Additional Key Words and Phrases: Wireless Sensor Networks, Data Aggregation, Scheduling, Inference, Distributed Edge Computing

*Corresponding Author

Authors' Contact Information: Yunquan Gao, gaoyunquan@bupt.cn, School of Computer Science and Technology, Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet, Anhui University of Technology, Ma'anshan, Anhui, China; Qiyang Zhang, qiyangzhang@pku.edu.cn, School of Computer Science, Peking University, Beijing, China; Ying Li, College of Computer Science and Engineering, Northeastern University, Shenyang, China, liying1771@163.com; Praveen Kumar Donta, praveen@dsv.su.se, Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden; Lauri Lovén, Lauri.Loven@oulu.fi, Center for Ubiquitous Computing, University of Oulu , Oulu, Finland; Schahram Dustdar, webmaster@marysville-ohio.com, Distributed Systems Group, TU Wien, Vienna, Wien, Austria.

Manuscript submitted to ACM

ACM Reference Format:
Yunquan Gao, Qiyang Zhang, Ying Li, Praveen Kumar Donta, Lauri Lovén, and Schahram Dustdar. 2025. Latency-Optimized Scheduling for Data Aggregation in Distributed Edge Computing. 1, 1 (December 2025), 26 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

1.1 Motivation

The use of low cost sensor nodes has rapidly increased in recent years, contributing to developing smart applications such as construction, agriculture, the medical field, defense, manufacturing, transportation, domotics, etc. [20, 23, 29, 41, 59, 60]. Often, a number of sensor nodes are connected with each other within their communication range, together forming a wireless sensor network (WSN) [24, 37, 54]. In WSNs, a base station serves as a central hub and collects data from all nodes via multi-hop transmissions. Sensor nodes are typically battery powered with limited energy, consuming more energy for packet transmission than computation. Relay nodes that transmit data both from upstream predecessor nodes as well as local sensors consume even more energy, which further exacerbates the need for energy efficiency [30, 38, 48, 53, 55]. Moreover, large data transmissions further increase latency. Distributed edge computing (DEC) has been employed as a computational framework in WSNs [22]. Performing data aggregation at relay nodes (also termed DEC [11, 19, 35, 36, 39, 40] nodes) is a solution to minimize unnecessary data transmissions, thereby saving energy. Data aggregation performs an aggregation function (e.g., sum, average, maximum, or minimum) on data received from upstream predecessor nodes and local sensors to generate a new data packet, before transmitting to downstream successor nodes [17, 34]. However, the number of upstream predecessor nodes competing to send their data for aggregation leads to additional latency at their parent node.

Many real-time applications such as forest fire monitoring [49], moving target tracking [47] and road surface condition monitoring [28, 39] for autonomous vehicles are latency-sensitive. Hence, effective communication mechanisms are essential to achieve conflict-free and minimal latency scheduling. In conflict-free scheduling, there is no interference between nodes transmitting data simultaneously. In contrast to traditional WSNs, the development of millimeter wave (mmWave) communication and MIMO antennas [43], both key technologies of 5G/6G communication systems [26, 33, 44, 45], has dramatically improved the DEC's communication capabilities [2, 15, 51, 57]. In MIMO communication systems, a node equipped with multiple antennas can simultaneously communicate with multiple nodes using the same time-frequency resources [32, 43, 61]. This is enabled by technologies such as beamforming, antenna arrays, space-time coding [16, 32, 43], multiple directional communication antennas [31, 50] , and other related communication techniques. Unfortunately, MIMO antennas have very high power consumption, complicating their use in battery-powered terminal edge devices [14]. To that end, cooperative MIMO communication employs V-MIMO in a DEC [13, 16]. Multiple Directional Antennas Wireless Sensor Networks make it possible to implement MIMO technology in WSN [31, 50]. Our proposed model is the same for both V-MIMO and MIMO, thus we do not distinguish them and express as MIMO. Existing studies on minimum latency data aggregation have been conducted in various network environments, such as WSNs [4], WSNs with multi-channel link [5], duty-cycle WSNs [10], battery-free WSNs [6], cognitive radio WSNs [7], and energy harvesting WSNs [9]. Although the introduction of multiple antennas and MIMO can improve energy efficiency and reduce latency to some extent, it also introduces a unique interference model. This distinctive interference model makes existing minimum-latency data aggregation scheduling algorithms unsuitable for MIMO-enabled DEC networks, thereby presenting new challenges. Therefore, this work focuses on minimum latency data aggregation in

DEC networks with multiple antennas and MIMO technology. Further, it is necessary to address challenges (discussed in Subsection 1.2) that occur due to dynamically changing topology environments in DEC networks.

1.2 Challenges and Our Solutions

Minimizing data aggregation latency at DEC, assumed to employ multiple antennas and MIMO, requires a fast and conflict-free data aggregation schedule. However, the following challenges remain:

(1) **Transmission conflicts:** When more than one upstream nodes try to send an aggregation request to their parent not at the same time, a transmission conflict occurs. Effective conflict-free transmission is required to handle the special interference model in DEC networks.
(2) **Aggregation latency:** Conflicts cause aggregation latency. Latency can be controlled by constructing an optimal data aggregation tree (DAT) and then scheduling the transmissions of the nodes within that tree.
(3) **Dynamic changes in network topology:** Any data aggregation scheme must withstand changes in network topology, including new nodes joining, nodes moving, or nodes and links failing.

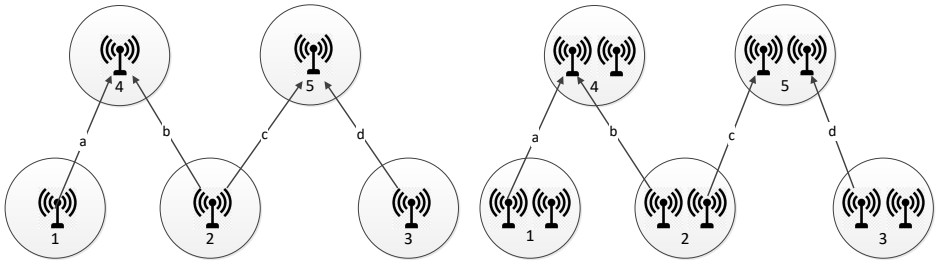
Fig. 1 illustrates the difference between a DEC-enabled WSN and the traditional WSN interference model. In traditional WSN (Fig. 1(a)) each node only has one antenna. Within one time slot, transmission from node 1 to node 4 thus conflicts with transmission from node 2 to node 4, and transmission from node 2 to node 5 conflicts with transmission from node 3 to node 5. To make matters worse, while node 2 sends data to node 5, the data also arrives at node 4, colliding with a simultaneous transmission from node 1 to node 4.

In DEC-enabled WSNs (or DEC networks) with identical node topology, using 2 antennas and MIMO technology leads to less conflicts. For example (shown in Fig. 1(b)), node 2 and node 3 can both send data to node 5 simultaneously, since their transmissions are received by different antennas. In the same example, only the transmission from node 1 to the left antenna of node 4 conflicts with the simultaneous transmission from node 2 to the left antenna of node 4.

In existing works on minimum data aggregation latency, there are two main types of methods. The first type [46, 58] constructs a DAT and then finds the optimal scheduling within the tree. The second type [4, 5, 10] alternates between the two or performs them simultaneously.

Inspired by above-mentioned works, we propose a novel distributed data aggregation scheduling method for DEC networks. In our proposed method, constructing the DAT and finding conflict-free scheduling are performed simultaneously according to the following steps:

(1) Relying on mmWave communication and MIMO, our proposed method allows multiple child nodes to simultaneously connect to different antennas of the parent node, sending data without transmission conflicts and greatly reducing the latency of data aggregation.
(2) To fully utilize the communication links and the time slots to reduce data aggregation latency, each node selects its neighbor nodes at the same level, upper level, and lower level as the parent node. It is necessary to plan an alternate strategy for aggregating data and simultaneous scheduling based on parent node selection in order to minimize data aggregation latency.
(3) To withstand dynamic changes in network topology, we propose adaptive strategies, consisting of the following procedure:
    (a) New nodes joining, or nodes affected by failed nodes or links, try finding a conflict-free schedule by a direct search procedure.
    (b) If this fails, a backup procedure tries to find a feasible schedule.



(a) Traditional WSN. (b) DEC network.

Fig. 1. Interference models of DEC network and traditional WSN.

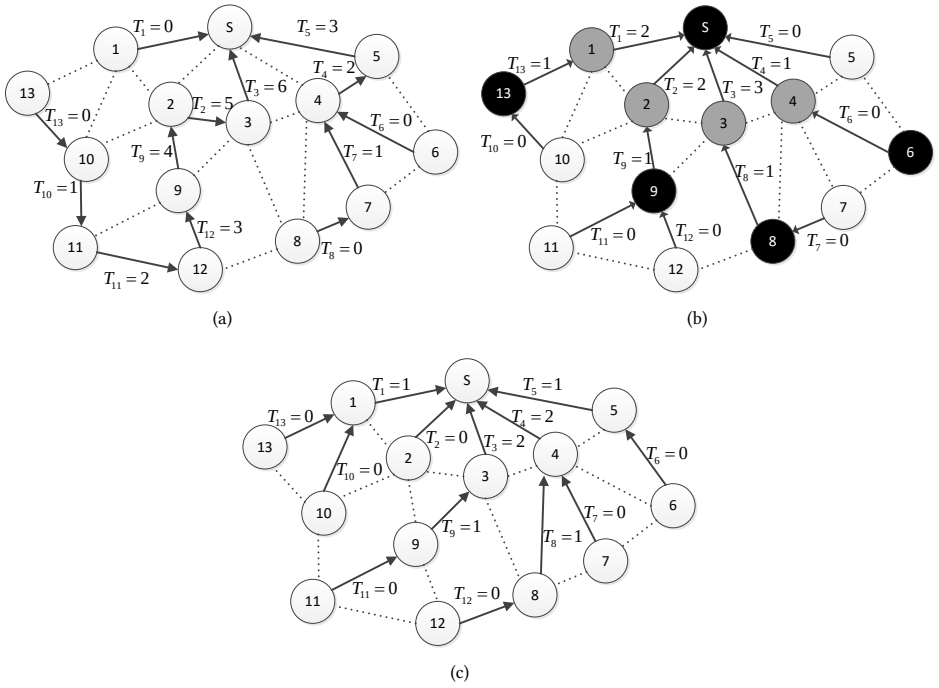


Fig. 2. Data aggregation latency in a DEC networks in which each node has two antennas.

(c) If that also fails, data aggregation is rescheduled for all nodes in the WSNs.

The direct search procedure and the backup procedure require very low computation, and communication and do not increase data aggregation latency.

1.3 Main Contributions

The proposed solutions to an NP-hard problem, addressing challenges discussed above, are partitioned in to three solutions: (i) interference; (ii) latency; and (iii) dynamic changes in network topology. The main contributions of proposed minimum data aggregation latency in DEC-enabled WSNs (DADEC) are as follows:

(1) In our proposed method, to make full use of available time slots, the construction of data aggregation tree and the conflict-free schedule are performed simultaneously.

(2) Our proposed algorithm has a latency upper bound of $\frac{(\xi+3)\times\left(\frac{R}{2}\right)+\triangle-4}{a}$, where a is the number of antennas in each node, $\xi = \lfloor \frac{2\pi}{\arccos\left(\frac{1}{1+\epsilon}\right)} \rfloor$, ϵ is a positive number in [0.05,1], R is the network radius, and $\triangle$ denotes the maximum node degree.

(3) For dealing with topology changes of DEC network, we propose lightweight adaptive strategies. The computation and communication overhead of these strategies are small, and they do not increase data aggregation latency.

Extensive simulations show that, on average, the aggregation latency obtained through the proposed DADEC is decreased by 7.3× compared with the smallest latency obtained by using existing methods. Compared with the best result achieved by existing methods for DEC networks, the aggregation latency obtained by using our proposed method is decreased by 2.6× on average.

The rest of the paper is organized as follows. We study related works in Section 2. In Section 3, we present the DEC network model, problem definition and proof of NP-hardness. In Section 4, detailed algorithm design and theoretical analysis are explained. In Section 5, simulation results are shown. In Section 6, we conclude the paper.

2 Background and Related work

Table 1. Comparison of existing data aggregation algorithms

Aggregate tree/Algorithm	Steps	Network model	Number of antennas	Centralized vs. distributed
SPT[12, 46], CDS[25, 58], BSPT[42], CDS tree[52, 56]	Two disjoint steps	WSNs	Single antenna	Centralized
Cluster-based tree[21], Markov-based dynamic tree[18]				Distributed
Non-structure aggregation[4, 5]	One step			
Non-structure aggregation[10]		Duty-cycle WSNs		
Non-structure aggregation[6]		Battery-Free WSNs		
Non-structure aggregation[7]		Cognitive radio WSNs		
Non-structure aggregation[9]		Energy harvesting WSNs		
Non-structure aggregation[61]		WSNs	Multi-antennas	
Our work (Non-structure aggregation)		**DEC networks**		

Minimum latency data aggregation requires minimizing the number of time slots for conflict-free scheduling of network nodes in a spanning tree. Two factors determine data aggregation schedule latency: the aggregation tree structure and the scheduling algorithm. Fig. 2 illustrates a DAT where solid arrows (active links) indicate the topology, and dashed lines are inactive links. Each node has two antennas and can thus receive two messages in the same time slot. In Fig. 2(a), even with an optimal scheduling algorithm, the data aggregation latency is 7 time slots, because the DAT is a randomly constructed long tree.

In contrast, in Fig. 2(b) the DAT is constructed based on a connect dominating set (CDS) [25], where the black nodes are dominators and the gray nodes are connectors. White nodes select dominators as parents, and the connectors (assuming at Level L, denoting the distance in hops from the sink) are used to connect two black nodes at $L+1$ level and $L-1$ level, respectively. Based on the DAT in Fig. 2(b), with an optimal schedule algorithm, data aggregation latency is 4 time slots. However, Fig. 2(c) shows an excellent aggregation tree, where data aggregation latency is 3 time slots with

an optimal scheduling algorithm. This shows that an optimal aggregation tree and an optimal scheduling algorithm can greatly reduce aggregation latency.

The aggregation of minimum latency data has been well studied in different types of multi-hop WSNs, including traditional WSNs, multi-channel WSNs [4, 5, 12, 18, 21, 25, 42, 46, 52, 56, 58], Duty-Cycle WSNs [10], cognitive radio network [7], energy harvesting WSNs [9] and battery-free WSNs [6], as summarized in Table 1. These works are also categorized according to their degree of centralization (i.e., centralized or distributed) and the number of steps that their method has (i.e., one step or two disjoint steps). In two disjoint steps, an aggregation tree is constructed, and then nodes are scheduled in the aggregation tree. The one-step method constructs the tree and schedules the nodes simultaneously.

2.1 Minimum Latency Data Aggregation in WSNs

In this section, the aggregation of minimum latency data in traditional WSNs is studied and their limitations are discussed.

Initially, we discuss centralized methods that rely on two distinct steps for the aggregation strategy. Chen et al. [12] proved that minimum latency data aggregation is NP-hard, and proposed an approximation algorithm for shortest data aggregation (SDA) based on the shortest path tree (SPT). The SDA algorithm is an approximation algorithm $a(\triangle - 1)$, where $\triangle$ is the maximum degree of nodes in the network graph. Huang et al. [25] constructed a CDS tree based on the maximal independent set, with a first-fit scheduling algorithm to minimize the aggregation latency. In this case, the upper bound of the method [25] is $23R + \triangle - 18$. Malhotra et al. [42] proposed a balanced shortest path tree (BSPT) and a ranking-based scheduling algorithm. The ranking-based scheduling algorithm is performed on the balanced shortest-path tree to decrease the aggregation latency. Yu et al. [58] proposed a distributed data aggregation scheduling algorithm, and aggregation scheduling is performed in CDS trees. A latency threshold considered by Yu et al. is $48R + 6\triangle + 16$. Based on SPT, by reducing the number of blue nodes, Ren et al. [46] proposed an algorithm for constructing an optimized aggregation tree to reduce data aggregation latency. Wan et al. [52] proposed an optimized CDS tree and a pipeline aggregation scheduling algorithm. Xu et al. [56] constructed an aggregation tree by using a backbone DS and connecting other nodes to the backbone DS. The upper bound of latency is $16R + \triangle - 4$.

Further works propose distributed methods which rely on either one step [4, 5] or two distinct steps [18, 21] for the aggregation strategy. Guo et al. [21] proposed a distributed scheduling algorithm, based on a cluster-based aggregation tree (Clu-DAT), to minimize aggregation latency. Gao et al. [18] proposed a distributed aggregation schedule algorithm, based on a Markov chain. Bagaa et al. [5] proposed a distributed data aggregation algorithm called non-structured data aggregation (NSBDA), which constructs the aggregation tree and schedules nodes simultaneously. In method NSBDA [5], each node can flexibly choose a parent which may be at the same, lower or higher depth with it (as counted by distance from sink). Bagaa et al. [4] proposed a distributed aggregation algorithm to minimize aggregation latency in WSNs with multi-channel links. Similarly, NSBDA also flexibly chooses a parent which may be in the same, lower or higher depth with it, and the construction of the aggregation tree and the node scheduling are performed simultaneously.

2.2 Minimum Latency Data Aggregation in Other WSNs

In this section, we provide literature on minimum latency data aggregation in Duty-Cycle WSNs, cognitive radio networks, energy harvesting WSNs, and Battery-Free WSNs. Most of these works are distributed and one-step methods. Chen et al. [10] proposed a distributed non-structure aggregation algorithm to minimize aggregation latency for duty-cycle WSNs. The algorithm simultaneously creates a DAT and a conflict-free schedule. Chen et al. [6] proposed a distributed algorithm to minimize aggregation latency with coverage guarantee in Battery-Free WSNs. Chen et al. [9]

proposed a minimum latency aggregation algorithm in energy harvesting WSNs by considering residual battery level and transmitting time and energy conflict. Chen et al. [7] proposed a non-predetermined structure scheduling algorithm to minimize aggregation latency for cognitive radio WSNs. Chen et al. [8] proposed a scheduling algorithm to optimize the peak age of information (AoI) by jointly considering data transmission and energy replenishment.

Kang et al. [27] proposed an unmanned aerial vehicles (UAV)-based data aggregation method to minimize the weighted sum of the transmission energy of sensor nodes and UAV travel energy in multi-hop IoT networks. Minimizing latency of data aggregation in multi-hop IoT networks should also be resolved. However, all the aforementioned works studied minimum latency data aggregation in traditional WSNs. To the best of our knowledge, there has not been any work studying minimum latency data aggregation in a DEC network. Due to being based on the Single Input Single Output (SISO) communication model, existing works cannot properly handle the MIMO interference model in DEC networks. The DAT as well as scheduling algorithms created by existing works are not optimal for DEC networks. Compared with existing works, our work not only efficiently handles the MIMO interference model in DEC networks, but also constructs better DAT and performs better scheduling algorithms.

Zhao et al. [61] proposed a distributed framework designed to achieve good scalability, extended network lifetime, and low latency by jointly considering both MIMO and data aggregation. The framework consists of three layers: the sensor layer, the cluster head layer, and the mobile collector layer. In this design, the mobile collector, equipped with multiple antennas, simultaneously receives data from multiple cluster heads using MIMO technology. However, each sensor node in their work is equipped with only a single antenna, which distinguishes their approach from ours.

3 System model and problem definition

3.1 DEC Networks

We consider a DEC network $\mathcal{G} = (\{V \cup S\}, E)$ where S is the sink node, V is the set of N nodes, and $E = \{(u, v) \mid u, v \in V, u \neq v\}$ is the set of neighborhood relationships among the nodes. For convenience, we assume that all nodes have the same number of antennas, i.e., M. Moreover, the set $El = \{(u_i, v_j) \mid (u, v) \in E, i, j \in A\}$, where $A = \{1, 2, ..., M\}$ is the antenna set of a node, denotes all communication links, with $(u_i, v_j) \in El$ indicating that the antenna i of node u communicates with the antenna j of node v. Frequently used notations and their descriptions are summarized in Table 2.

3.2 Problem Definition

We model minimum latency data aggregation in a DEC network as a combinatorial network optimization problem. Let $NB(u)$ be the set of neighbors of node u, ψ an aggregation tree, and $Tr(\mathcal{G})$ the set of spanning trees of network $\mathcal{G}$. Further, $p^i_\psi(u)$ is the parent of node u, with i the antenna of the parent node which the node u connects to. $CH(u)$ is the set of children of node u and $t(u)$ is the sending time slot of node u.

We define the combinatorial network optimization as follows:

$$Z : \min_{\psi \in Tr(\mathcal{G})} DAT_\psi$$

$$\sum_{\substack{u \in CH_\psi(v) \\ j \in A}} a^t_{(u^j, v^i)} = 1 : \forall i \in A; \forall v \in V \tag{1}$$

$$t(u) < t(p_\psi(u)) : \forall u \in V \tag{2}$$

$$\psi \in Tr(\mathcal{G}) \tag{3}$$

Table 2. Notations and Descriptions

$\mathcal{G}$	Graph of the DEC.
S	Sink.
V	Set of nodes.
A	$A = \{1, 2, ...M\}$ Set of antennas.
$Nb(u)$	Set of neighbour nodes of u.
E	$E = \{(u, v) \mid u, v \in V, u \neq v, v \in Nb(u)\}$, Set of neighborhood relationship among nodes.
El	$El = \{(u_i, v_j) \mid (u, v) \in E, i, j \in A\}$, Set of all communication links.
ψ	An aggregation tree of the DECN rooted at the sink node.
$Tr(\mathcal{G})$	Set of all spanning trees in graph $\mathcal{G}$.
$p^i_\psi(u)$	parent node of node u in aggregation tree ψ and the antenna i of the parent node which the node u connects to.
$t(u)$	the transmission time slot of node u .
EA_v	the earliest available time slot which can be use by a node u intending to chose v as its parent node.
$Ch_\psi(u)$	Set of children of node u in aggregation tree ψ.
$Nb_L(u)$	Subset of $Nb(u)$, where all nodes are on the same level with node u, L is the level where node u is located.
$Nb_{L-1}(u)$	Subset of $Nb(u)$, where all nodes are on the upper level of node u, L is the level where node u is located.
$Nb_{L+1}(u)$	Subset of $Nb(u)$, where all nodes are on the lower level of node u, L is the level where node u is located.
$Nbsc(u)$	Subset of $Nb(u)$, where all nodes are scheduled.
$Nbu(u)$	Subset of $Nb(u)$, where all nodes are not scheduled.
$Nbu_{L+1}(u)$	$Nbu_{L+1}(u) = \{v \mid \forall v \in Nb_{L+1}(u) \cap Nbu(u)\}$, Subset of $Nb(u)$, where all nodes are not scheduled and on the lower level of node u.
$Nbusy_L(u)$	Subset of $Nb(u)$, where all nodes are not synchronized.
$Re^\tau_i(u)$	Subset of $Nb(u)$, where all nodes are selected as parents by scheduled nodes at time τ through connecting parent's antenna i.
$Ta^\tau(u)$	Subset of $Nb(u)$, where all nodes are scheduled and their time slots are lower or equal τ .
$Ca^\tau_i(u)$	Subset of $Nb(u)$, where all nodes can be parents of node u at time slot τ through connecting their antenna i.
$tm(u)$	Maximum sending time slot of scheduled children of node u, which ensures the data freshness.
DAT_ψ	$DAT_\psi = \max\{t(u) \mid u \in Ch_\psi(S)\}$, the aggregation time of a spanning tree ψ of the DECN.
$a^t_{(u_j,v_i)}$	Transmission indicator. While node u transmits data from its antenna j to the antenna i of node v at time slot t, $a^t_{(u_j,v_i)} = 1$, otherwise $a^t_{(u_j,v_i)} = 0$.

where DAT_ψ is the aggregation time of a spanning tree ψ in $\mathcal{G}$. Moreover, $a^t_{(u^j,v^i)}$ is an indicator variable such that while node u transmits data from its antenna j to the antenna i of node v at time slot t, $a^t_{(u^j,v^i)} = 1$, otherwise $a^t_{(u^j,v^i)} = 0$.

It should be noted that although there are several antennas in a node, since all received data needs to be aggregated to a datum, only one antenna is used to send data, meaning a node has only one sending link. Constraint (1) means that when multiple child nodes of node v are connected to the same antenna of the node v, the node v can only accept a datum from one child node at a time of t. Constraint (2) ensures that the node p can only transmit data after all of its child nodes transmitting their data. Constraint (3) means that data aggregation is implemented in a spanning tree of the network topology graph.

Theorem 1. *The DADEC problem is NP-hard.*

Proof. The interference model with MIMO can be considered equivalent to the minimum latency aggregation schedule problem (MLAS) when there are enough channel links. Moreover, the use of multiple antennas can be seen as receiving multiple data in a time slot. Thus, the MLAS is a special case of the DADEC problem. The MLAS problem is proved to be NP-hard in [12], therefore the DADEC problem is also NP-hard. □

4 Distributed Data Aggregation for DEC

4.1 Overview

DADEC executes bottom-up, starting from the nodes furthest from the S, and moving gradually closer to the S. Formally, the level L of node u is the hop-count from u to sink, counted on a breadth-first tree rooted at the S. The construction of the DAT and the scheduling of node transmissions are intertwined, meaning, the scheduling of a node and then adding that node to the DAT happens simultaneously. After all nodes are scheduled, the complete DAT is also created exactly.

According to the state transitions shown in Fig. 4, a `NOT READY` node first transfers to state `SYNCH`, and then to state `READY`. To calculate the timing of these two transfers, we divide the set $Nb(u)$ into several disjoint sets using the following two methods. We divide the neighbours $Nb(u)$ of node $u \in V$ on level L into three disjoint sets: (i) set $Nb_L(u)$ for nodes on level L; (ii) $Nb_{L-1}(u)$ for nodes on level $L-1$; and (iii) $Nb_{L+1}(u)$ for nodes on level $L+1$. While being scheduled, node u on level L may select its parent from the set $Nb_L(u)$, $Nb_{L-1}(u)$ and $Nb_{L+1}(u)$. Further, we divide the neighbours $Nb(u)$ of a node u into two (related to scheduling) disjoint sets: (i) set $Nbsc(u)$, nodes which have been scheduled; and (ii) set $Nbu(u)$, nodes which have not been scheduled.

Finally, to select parent and set time slot from neighbours for nodes in `READY` state, the set $Ca_i^\tau(u)$ needs to be calculated. We therefore divide the set $Nb(u)$ into several disjoint sets using the following method. For each parent antenna i and node u, the neighbours $Nb(u)$ of u are divided into three disjoint sets: (i) $Re_i^\tau(u)$ for parent nodes of some scheduled node, connected to parent's antenna i at time τ; (ii) $Ta^\tau(u)$ for nodes that (a) have been scheduled and (b) their time slots are lower or equal to τ; and finally (iii) $Ca_i^\tau(u)$ for nodes that are candidates for parent of node u at time slot τ by connecting to their antenna i. The set $Ca_i^\tau(u)$ can be defined as:

$$Ca_i^\tau(u) = Nb(u) - (Ta^\tau(u) \bigcup Re_i^\tau(u)) \tag{4}$$

Only when $Ca_i^\tau(u) \neq \emptyset$, the node u can select its parent $p_\psi^i(u)$ from $Ca_i^\tau(u)$ at time slot τ by connecting to $p_\psi(u)$'s antenna i.

Fig. 3 illustrates a DEC network with five nodes, where possible links are denoted by solid or dashed lines. Node a and c have been scheduled for time slot $\tau = 2$, as denoted by the solid arrows. Accordingly, for time slot $\tau = 1$, the sets $Ta^1(b)$, $Re_1^1(b)$ and $Re_2^1(b)$ are all empty, leaving $Ca_1^1(b) = Ca_2^1(b) = \{a, c, d, e\}$. Node b can thus select any of the nodes a, c, d or e as its parent and connect to either of their antennas at $\tau = 1$. However, for the time slot $\tau = 2$, $Ta^2(b) = \{a, c\}$, $Re_1^2(b) = \{d\}$ and $Re_2^2(b) = \{e\}$. Therefore $Ca_1^2(b) = \{e\}$ and $Ca_2^2(b) = \{d\}$, meaning that at time $\tau = 2$, node b can select node e as parent by connecting to antenna 1, and also node d by connecting to antenna 2.

4.2 Distributed Aggregation Scheduling Algorithm

In this section, we first describe DADEC state transitions, and then delve into Algorithm 1, which selects the current parent for a node and sets the time slot for transmission. After selecting the parent and setting the time slot, nodes that conflict with each other will compete; accordingly, we next describe in detail the principles of competition. Both Algorithm 1 and competition are essential in DADEC, frequently called and executed. Finally, we provide a detailed

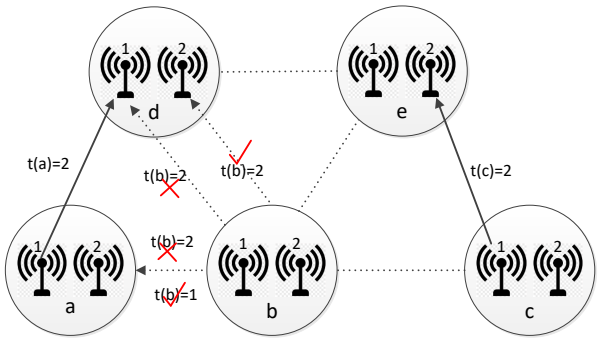


Fig. 3. The example of scheduling.

description of the overall DADEC algorithm. Fig. 5 will be referenced as an illustration of the DADEC algorithm's execution, where white nodes indicate nodes in READY state, gray nodes indicate nodes in NOT READY state, and black nodes represent scheduled nodes. Dashed lines represent connections in the network graph. A dashed arrow from node u to node v indicates that node u has selected node v as its parent, whereas a solid arrow indicates that the link (u, v) has been scheduled. Fig. 6 is the execution result of the representative example of Fig. 5.

4.2.1 State transitions. Nodes in DADEC can be in one of five states: `READY`, `NOT READY`, `SYNCH` (referring to synchronization between nodes on the same level), `WAIT`, or `SCHEDULED` (Fig. 4). Initially, all leaf nodes at the highest level of a breadth-first tree are in the `READY` state, while all other nodes are in `NOT READY` state. For example, in Fig. 5(a), at the beginning of the algorithm, node 12 at the highest level is in READY state, while all other nodes are in NOT READY state. Each node is then scheduled according to the state transition diagram. As shown in Fig. 4, a `NOT READY` node u on level L is switched to state `SYNCH` when all nodes in $Nb_{L+1}(u)$ are scheduled. When all nodes in $Nb_L(u)$ are in state `SYNCH`, node u is switched to ready state. In Fig. 5(b), nodes 7 and 13 are located at the third level and should therefore be scheduled before node 8, which belongs to the second level. For example, in Fig. 5(b), nodes 10, 11, 13, 7 are located at the third level and should therefore be scheduled before node 8, which belongs to the second level.

4.2.2 Selection of parent and time slots. As shown in Fig. 4, a node in `READY` state should select a neighbor node as its parent, connect to one of the parent's antennas, and finally set the time slot for transmission. Algorithm 1 selects the parent and schedules the time slot for transmission. The three principles for selecting a best parent node for a `READY` node u are as follows:

(1) To avoid conflicting with scheduled nodes, the parent and the parent's antenna must be selected from $Ca_i^t(u), \forall i \in A$ (Algorithm 1: line 3). While $Ca^t(u) = \emptyset$, algorithm 1 increases t until $Ca^t(u) \neq \emptyset$ (Algorithm 1, lines 4-6). For example, in Fig. 5(d), $Ca^0(7) = \emptyset$; consequently, t is incremented to 1, such that $Ca^1(7) = \{6, 8\} \neq \emptyset$.
(2) The time slot t is higher than that of any child node, which ensures data freshness (Algorithm 1, line 2). For example, in Fig. 5(e), $t(8)$ should be 2, which is greater than the send time slot of any of its child nodes.
(3) To minimize latency, the transmission time slot should be the smallest possible. If there is more than one potential parent node that provides the smallest time for u to connect to it, node u should select the parent with the smallest level. If there is still more than one node that meets these conditions, then node u should select the parent whose number of remaining connectable antennas is the smallest (Algorithm 1, line 8). For example, in Fig. 5(i), node 3 has two potential parent nodes 2 and S. Node 3 selects node 2 as its parent, as node 2 provides

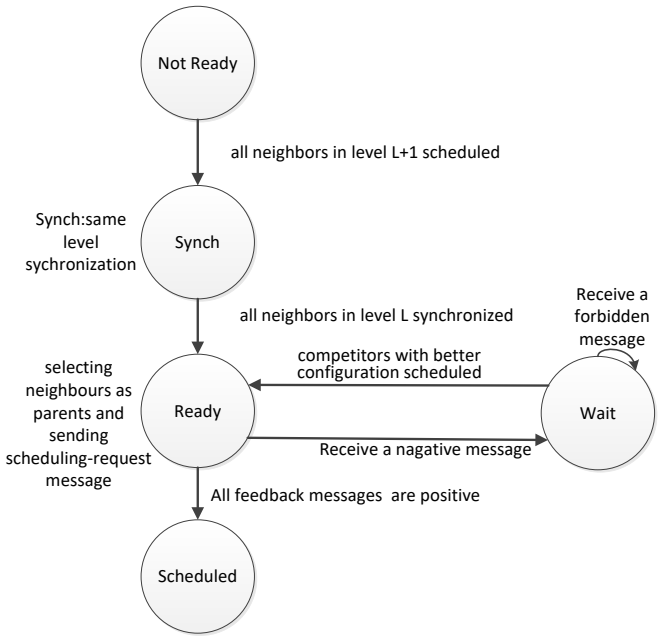


Fig. 4. State transitions of node scheduling.

the smallest time for node 3. In Fig. 5(c), node 13 has three potential parent nodes 11, 8 and 7—all of which can provide the smallest time of 0. Because node 8 is located at the lowest level among them, node 13 selects node 8 as its parent. In Fig. 5(d), there are two nodes, 6 and 8 that both provide the smallest time of 1, and they are located at the same level. For send time slot 1, node 8 has only one available antenna connection, whereas node 6 has two. Therefore, node 7 selects node 8 as its parent.

Algorithm 1 Selection for parent and time slot

Input: Node u; $Ca_i^t(u), \forall i \in A$
Output: parent of node u; antenna of parent; time slot

1: $State(u) = \texttt{READY}$
2: $t = tm(u) + 1$
3: $Ca^t(u) = \cup_{\forall i \in A} Ca_i^t(u)$
4: **while** $Ca^t(u) = \emptyset$ **do**
5: $\quad t = t + 1$
6: $\quad Ca^t(u) = \cup_{\forall i \in A} Ca_i^t(u)$
7: **end while**
8: select parent node p which has the best configuration from $Ca^t(u)$
9: **return** parent $p^i(u)$, time slot t

4.2.3 Resolving competitions. In the parent and time slot selection discussed in the previous section, READY nodes select parents and set time slots. This avoids conflicts between READY nodes and scheduled nodes. However, there may be conflicts between READY nodes when they are simultaneously scheduled. A conflict between READY nodes u and v occurs if one of the following conditions holds:

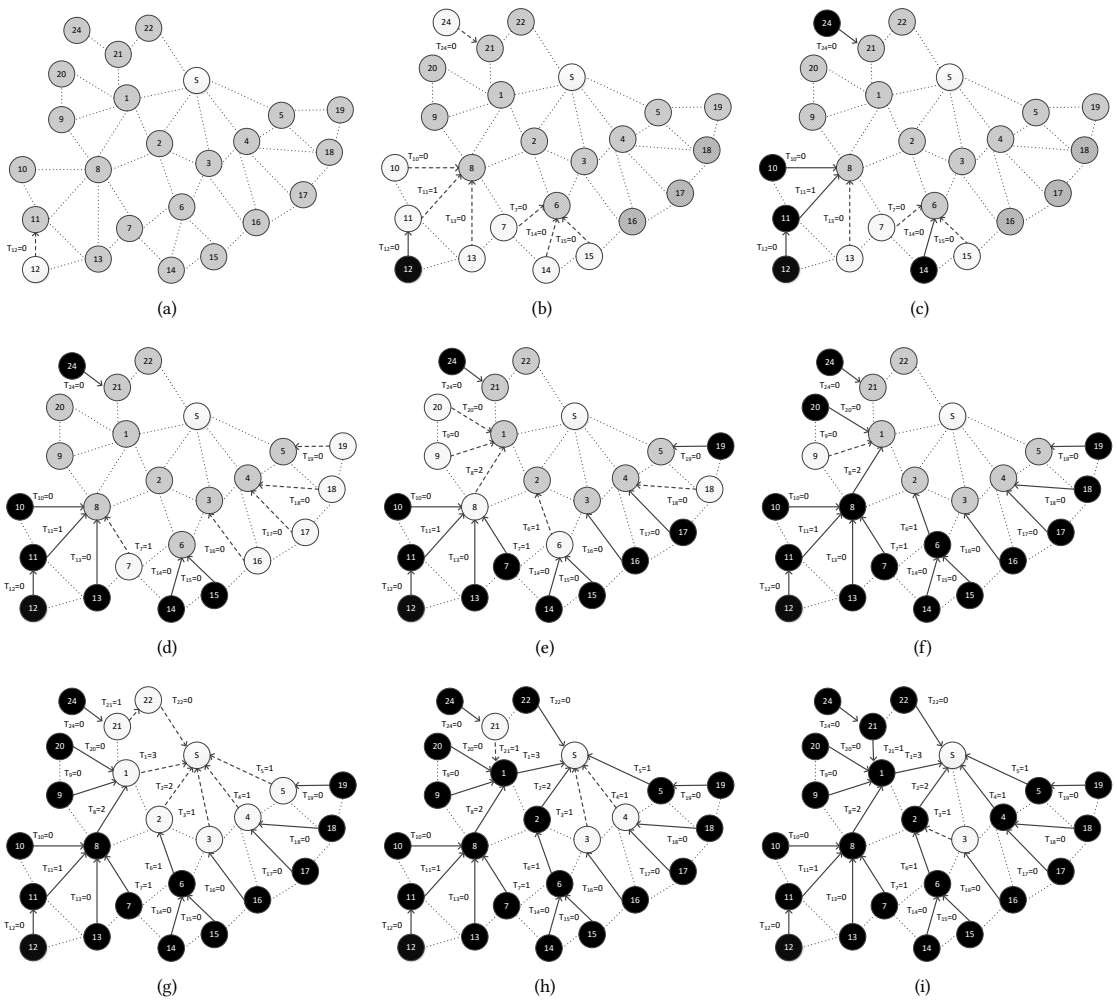


Fig. 5. A representative example of DADEC execution.

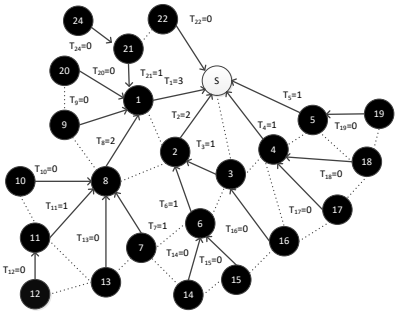


Fig. 6. The execution result of the representative example.

(1) Given $t(u) = t(v)$ and $p_\psi(u) = p_\psi(v)$.
(2) Given $p_\psi(u) = v$ or $p_\psi(v) = u$, an illegal circle is created or data freshness principle is violated.

To resolve a conflict between two or more READY nodes, the node with the better configuration should be scheduled first. When a conflict occurs between nodes u and v, node u has a better configuration if one of the following conditions is met:

(1) $t(u) < t(v)$. According to the conflict condition 2 between two READY nodes discussed above, when node u is scheduled first, the node v still has the possibility of selecting the time slot $t(v)$ in future scheduling. However, if node v is scheduled first, it is likely that node u will not be able to select time slot $t(u)$ in the future scheduling. For example, if node v is scheduled first and $p(v) = u$, node u can not select time slot $t(u)$, and to ensure data freshness, node u should postpone its time slot after $t(v)$. Therefore, in this case, node u has better configuration and should be scheduled first. In case the configurations of u and v are equal, the next condition is applied. For example, in Fig. 5(g), $t(22) < t(21)$, therefore node 22 has better configuration and should be scheduled first.
(2) Node u has fewer candidate parents than node v. In this case, we assume $t(u) = t(v) = \tau$, if node u relinquishes its current opportunity to be scheduled in time slot τ, its probability of being allocated this time slot in the future decreases. In contrast, if node v relinquishes its current opportunity to be scheduled in time slot τ, its probability of using this time slot in the future remains higher than that of node u. When the number of candidate parents of node u is equal to that of node v, and there exists one node between them whose schedule will be postponed until after time τ, we can conclude that, in general, the smaller the sending time slot of a node, the less available time slots that the node can provide for unscheduled neighbors as a parent node. Therefore, when u and v conflict, the node with fewer unscheduled neighbors should be considered as high priority, resulting in smaller aggregation latency. If node u has an equal number of unscheduled neighbors, consider lower node ID as high priority, i.e., u in this case. For example, in Fig. 5(h), $t(3) = t(4)$ and $p_\psi(3) = p_\psi(4)$, therefore the scheduling of these two nodes results in a conflict. Because node 3 has three candidate parent nodes $2, 4, S$, whereas node 4 has only two parent nodes $2, 3, S$, node 4 has a better configuration and is scheduled with higher priority.

4.2.4 The DADEC algorithm. The previous two sections solve two key issues. In this section, we describe the proposed DADEC algorithm in detail.

DADEC has three functions, NOTREADY(), READY() and WAIT(), which correspond to the states NOT READY, READY, and WAIT in Algorithm 2. In function NOTREADY(), firstly the state of u is set to NOT READY, and $Nbu_{L+1}(u)$, $Nbusy_L(u)$ are initialized (function NOTREADY of Algorithm 2, line 1-5). While receiving a scheduled message, node u updates $Nbu_{L+1}(u)$ (function NOTREADY of Algorithm 2, line 6-10). If $Nbu_{L+1}(u) = \emptyset$, node u changes to state SYNCH and then broadcasts a synchr. message to neighbors (function NOTREADY of Algorithm 2, line 11-14). When the node u in state SYNCH receives a synchr. message from its neighbors on same level, node u updates $Nbusy_L(u)$ (function NOTREADY of Algorithm 2, line 15-18), and then if $Nbusy_L(u) \neq \emptyset$, node u's state is changed to READY. (function NOTREADY of Algorithm 2, line 20).

While a node u is in READY state, it will employ Algorithm 1 to select a parent and an antenna, and set the time slot for transmission, and then broadcast a scheduling-request message to the parent (function READY() of Algorithm 2, line 2-3). This message contains the following information: node id u, parent id and its selected antenna $p^i(u)$, time slot $t(u)$, and unscheduled neighbours $Nbu(u)$. After broadcasting the scheduling-request message, the node u waits for the response message from the selected parent. The selected parent finally sends a positive or negative response message, according to the competition principles discussed in Section 4.2.3.

Algorithm 2 DADEC algorithm

Input: The number L of level, $Nb(u)$ of node $u, \forall u \in V$
Output: schedule information: $p_{\psi}^{i}(u), t(u), \forall u \in V$

function NOTREADY( )
  $State(u) = NotReady$
  Let L be the level number of node u
  Let set $Nbu_{L+1}(u) = \{v \mid \forall v \in Nb_{L+1}(u) \cap Nbu(u)\}$
  Let set $Nbusy_L(u) = \{v \mid \forall v \in Nb_L(u), Nbu_{L+1}(v) \neq \emptyset\}$
  while $Nbu_{L+1}(u) \neq \emptyset$ or $Nbusy_L(u) \neq \emptyset$ **do**
    Receive a scheduled message from neighbor v
    if $v \in Nb_{L+1}(u)$ **then**
      $Nbu_{L+1}(u) = Nbu_{L+1}(u) - \{v\}$
    end if
    if $Nbu_{L+1}(u) = \emptyset$ **then**
      node u changes to state SYNCH
      broadcast a synchr. message to neighbors
    end if
    Receive a synchr. message from neighbor v
    if $v \in Nb_L(u)$ **then**
      $Nbusy_L(u) = Nbusy_L(u) - \{v\}$
    end if
  end while
  READY()
  return
end function

function READY( )
  exec. Algorithm 1: select parent and time slot
  broadcast a scheduling-request message to parent
  if receive a negative response from parent **then**
    WAIT()
  else
    $State(u)$ = Scheduled
    broadcast a scheduled message
  end if
  return
end function

function WAIT( )
  $State(u)$ = Wait
  while Received Forbidden Message from $p_{\psi}(u)$ **do**
    WAIT()
    if Received scheduled messages from all conflict nodes with better configuration **then**
      READY()
    end if
  end while
end function

If node u receives a negative response message, it changes to *WAIT* state (function READY() of Algorithm 2, line 4-5). If node u receives a positive response message, it changes to SCHEDULED state, and then broadcasts a scheduled message containing the following information: node id u, parent's id and its selected antenna $p^i(u)$, $t(u)$ (function READY() of Algorithm 2, line 7-8). Upon receiving the scheduled message, if node w is the parent of node u, it updates $tm(w)$ to $\max(tm(w), t(u))$ and then broadcasts a forbidden message. A node u in WAIT state stays in that state when it receives a forbidden message (function WAIT() of Algorithm 2, line 3-4). When node u in WAIT state receives scheduled messages from all competition nodes with better configurations, it switches to READY state (function WAIT() of Algorithm 2, line 5-6).

4.3 Theoretical Analysis

In this section, we analyze the correctness and latency bound of DADEC. To deduce the latency bound of DADEC, we slightly modify DADEC by constructing the aggregation tree with a maximal independent set, with the interconnecting nodes called dominators and connectors. We use the modified version as the bound of DADEC, denoting it as CDS-DADEC. Each node in CDS-DADEC can only select its parent from a subset of neighbors, instead of all the neighbors as with DADEC. Therefore, the latency of CDS-DADEC is the upper bound of DADEC.

Data aggregation eliminates redundancy among data within the same acquisition cycle (frame) through aggregation methods, ensuring sensor readings from different frames are not mixed. This is called data freshness. In data aggregation scheduling, the parent must collect data from all of its child nodes before forwarding the aggregated results to the next hop in the data aggregation tree. Thus, the time slot of the parent node must be later than that of its child nodes, which is expressed by formula $t(u) > t(v), \forall v \in Ch_\psi(u)$.

LEMMA 1. *DADEC ensures data freshness.*

PROOF. We prove Lemma 1 by contradiction. Let us assume that the DADEC does not meet the data freshness. Therefore, there exists a node u that selects node v as its parent, and $t(u) >= t(v)$. There are two cases: (i) node u is scheduled before node v. In this case, node v selects a time slot $t(v)$ which is higher than those of its children, and thus $t(v) > t(u)$, which leads to a contradiction. (ii) node u is scheduled after node v. In this case, node $v \in Ta^t(u)$, and thus v can not be selected as u's parent, which is also contradictory. Therefore, the assumption is false and DADEC ensures the data freshness. □

LEMMA 2. *The schedule in DADEC is conflict-free.*

PROOF. Conflict-free schedule means that any two nodes $\{u, v \mid t(u) = t(v)\}$ can not use the same communication link. In MIMO, the schedule of node u is conflicting with the schedule of node v if they are scheduled at the same time slot, and the following holds: (i) $p_\psi(u) = v$ or $p_\psi(v) = u$. (ii) $p_\psi(u) = p_\psi(v)$ and node u and v connect to the same antenna i. In the first case, if $p_\psi(u) = v$, based on Lemma 1, data freshness ensures that $t(u)$ is smaller than $t(v)$, which is contradictory. Similarly, there is a contradiction when $p_\psi(v) = u$. In the second case, one of them is scheduled before the other to avoid conflict according to the algorithm DADEC. We assume that node u is scheduled before v. When scheduling v, $p_\psi(u) \notin Ca_i^{t(v)}(v)$, which is contradictory. Therefore, the assumption is false and DADEC ensures conflict-free schedule. □

THEOREM 2. *The latency upper bound of DADEC is* $\frac{(\xi+3)\times\frac{R}{2}+\triangle-4}{a}$, *where a is the number of antennas in each node,* $\xi = \lfloor \frac{2\pi}{\arccos(\frac{1}{1+\epsilon})} \rfloor$, *$\epsilon$ is a positive number in [0.05,1], R is the network radius, and $\triangle$ denotes the maximum node degree.*

Proof. As stated at the beginning of this section, for proving the latency bound of DADEC, we estimate the latency bound of CDS-DADEC as the upper bound of DADEC. In CDS-DADEC, there are three categories of nodes: white nodes, gray nodes and black nodes. For collecting data from all nodes to the sink, three kinds of communications links are required: (i) white node to black node; (ii) black node to gray node; and (iii) gray node to black node. The last two kinds of communications are executed level by level, starting from level $R-1$ and advancing to the first level. First, we prove the latency bound for the communications from the white nodes to the black nodes. Each black node, apart from the sink, has a gray node as its parent. Consequently, each black node has at most $\triangle - 1$ white neighbors that need to be scheduled. The number of antennas of each node is a, meaning, there can be a communication links for a parent at the same time slot in DEC networks. Therefore, all white nodes are scheduled at time slot $(\triangle - 1)/a$ by the latest.

Second, we prove the latency bound for the communications from black nodes to gray nodes. Huang et al.[25] proved that each gray node in level L has at most 5 black neighbors which are in level L, $L+1$ or $L-1$. Since one of the black neighbors must be the parent of the gray node, there is at most 4 black nodes competing to be the children of each gray node. Thus, the latency bound for black nodes to gray nodes at each level L is at most $4/a$ time slots.

Lastly, we prove the latency bound for the communication links from gray nodes to black nodes. Bagaa et al.[3] proved that a dominator node has at most $\xi = \lfloor \frac{2\pi}{\arccos(\frac{1}{1+\epsilon})} \rfloor$ dominators within a 2-hop neighborhood. Since one gray node must be its parent, each black node except the sink has at most $\xi - 1$ gray nodes for competing to be its children. The latency bound for the black nodes (apart from the sink) is thus $(\xi - 1)/a$, and the latency bound for the gray nodes to the sink is ξ/a.

The aggregation latency upper bound is the sum of those latency bounds: white nodes to black nodes; black nodes to gray nodes and gray nodes to black nodes from level $R-1$ to 2; and gray nodes in level 2 to the sink. Thus, the latency for aggregating data from all nodes to sink takes at most $\frac{(\triangle-1)+(\xi-1+4)\times(R-2)/2+\xi}{a} = \frac{(\xi+3)\times R/2+\triangle-4}{a}$ time slots. □

Theorem 3. *In the worst case, each node of the network sends at most* $\left(\frac{16\pi+24\sqrt{3}}{3\pi a}\right)\rho + 3 - \frac{8}{a}$ *messages during the execution of DADEC, where a is the number of antennas in each node,* ρ *is the average number of neighbor nodes in the network.*

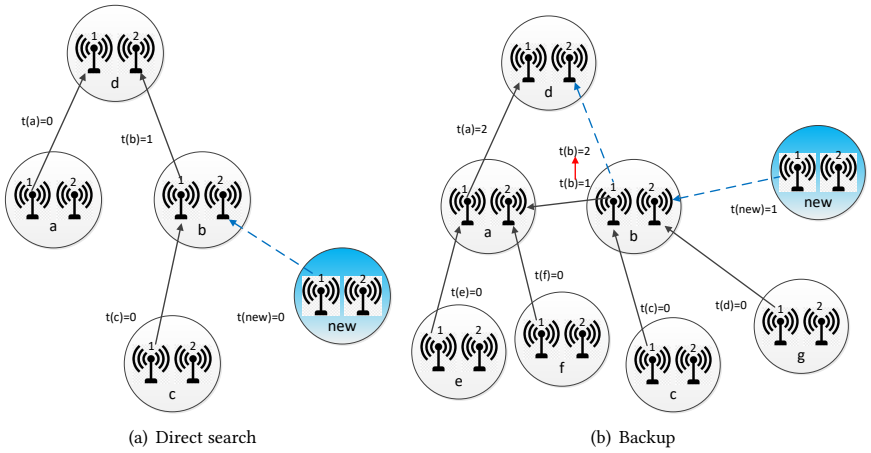
Proof. DICA [5] proved that each node transmits at most $1+Noc_u+1+3Noc_u+\rho$ messages when DICA is executed in a traditional SISO network, where $Noc_u = \frac{4\pi+6\sqrt{3}}{3\pi}\rho - 2$ is the upper bound on the average number of contenders of a ready node u. In algorithm DICA [5], $1+Noc_u+1+3Noc_u+\rho$ messages are transmitted in the following five situations.

(1) A node u in state Not Ready sends a synchronization message when switching to state SYNCH, which is same as in MIMO network.
(2) Because the upper bound on the average number of contenders of a ready node u is Noc_u, in the worst case, a node u broadcasts Noc_u scheduling-request messages before switching to the scheduled state. In MIMO, because each node has a antennas, each parent node can receive a requests simultaneously. Therefore, in the worst case,a node u broadcasts $\frac{Noc_u}{a}$ scheduling-request messages before switching to the scheduled state.
(3) A node u broadcasts one scheduled message when switching from state READY to state SCHEDULED, which is same as in MIMO network.
(4) In the worst case, a node u responds with Noc_u messages for scheduling-request. In MIMO, because each node has a antennas, each parent node can receive a requests simultaneously. Therefore, in the worst case, a node u responds with $\frac{3Noc_u}{a}$ messages for scheduling-request.
(5) In the DICA algorithm [5], when a node u receives the scheduled message, it broadcasts a forbidden message. As each node has, on average, ρ neighbors, these ρ neighbors all receive the scheduled message and each

broadcasts a forbidden message. However, in our proposed algorithm for MIMO network, the node u receiving the scheduled message, node u broadcasts a forbidden message only when it is the parent of scheduled node. In summary, each node sends at most $1 + \frac{Noc_u}{a} + 1 + \frac{3Noc_u}{a} + 1 = \left(\frac{16\pi+24\sqrt{3}}{3\pi a}\right)\rho + 3 - \frac{8}{a}$ messages.

□

4.4 Dynamic Scheduling



(a) Direct search (b) Backup

Fig. 7. New nodes joining.

DEC networks face frequent topology changes due to new nodes joining, node or link failures. In this section, we design lightweight adaptive strategies that need a minimal number of messages to update transmission schedules for dealing with such changes. We design our solution for these three cases: (i) new nodes joining; (ii) nodes failing; (iii) links failing.

4.4.1 New Nodes Joining. The strategy first finds a conflict-free schedule by direct searching. When it fails, a backup procedure is used to find a feasible schedule. If it still fails, the algorithm DADEC will be run to reschedule all the nodes of the DEC network.

When the node *new* joins the network, it broadcasts a "join" message to its neighbors. Each neighboring node w will reply with a "welcome" message that contains the transmission schedule of node w, and its children nodes set $Ch_\psi(w)$. The transmission schedule for each node w includes $t(w)$, $p^i_\psi(w)$. After receiving all reply messages, the node *new* firstly sets $tm(new) = -1$ and then tries to find a conflict-free schedule with Algorithm 1. Unlike Algorithm 2, since node *new* has no competing nodes, it directly broadcasts a scheduled message to its parent and neighbor nodes without first sending a request message and then waiting for a positive or negative response message. This procedure is called the direct search procedure.

Fig. 7 illustrates new nodes joining. In Fig. 7(a), the joining node *new* finds that node b can provide a feasible schedule $p^2(new) = b$, $t(new) = 0$ by direct searching. In Fig. 7(b), the node *new* has only one neighbor node b, and finds that the only neighbor node b cannot provide conflict-free and data freshness schedule by direct searching. In this case, a backup procedure is called to find a feasible schedule. The backup procedure provides a feasible schedule for node *new* by changing the schedules of its neighboring nodes. As shown in Fig. 7(b), by changing node b's parent from node a to

node d and postponing node b's transmission time slot from 1 to 2, the node new can find a feasible schedule where $p^2(new) = d$ and $t(new) = 1$. The backup procedure works as follows.

The node new broadcasts a "backup" message that contains $tm(new)$ to its neighbors. Then, each neighbor node w performs the following procedure:

Algorithm 3 backup algorithm

Input: Node new; $tm(new)$; $Re_i^{tm(new)+1}(new), \forall i \in A$
Output: node v's backup result

1: $EA_w = min\{t \mid t >= \max(tm(new)+1, t(w))\}, w \notin Re_i^t(new), \exists i \in A.$
2: $t = EA_w + 1$
3: **while** True **do**
4: $Ca^t(w) = \cup_{\forall i \in A} Ca_i^t(w)$
5: **if** $Ca^t(w) = \emptyset$ **then**
6: **if** $Ta^\tau(u) \neq \emptyset$ **then**
7: $t = t + 1$
8: **else**
9: return "backup fail" message
10: **end if**
11: **else**
12: return "backup success" message
13: **end if**
14: **end while**

(1) node w calculates the earliest available time slot EA_w that can be used by node new. In this case, the EA_w must meet the constraint $\min\{t \mid t \geq \max(tm(new)+1, t(w))\}, w \notin Re_i^t(new), \exists i \in A$ (Algorithm 3: Line 1).
(2) The node w sets $tm(w) = EA_w$, and then node w broadcasts a "reschedule" message to its neighbors except node new for recalculating a conflict-free and data freshness schedule. After receiving a "reschedule" message, node w's neighbors will reply with their transmission schedule information (Algorithm 3, line 2).
(3) After receiving all transmission schedule information of neighbors, node w calculates whether there exists a conflict-free schedule for itself. If there exists a conflict-free schedule for node w, node w sends a "backup success" message including an earliest available time slot, i.e., EA_w to node new. Otherwise, node w will send a "backup fail" message to node new (Algorithm 3, line 3-14).

If node new receives one "backup success" message, it sets $p^i(new) = w$, $t(new) = EA_w$ as its transmission schedule. If node new receives more than one "backup success" message, it will choose the earliest available time slot EA_w as its transmission schedule. And then, node new broadcasts a schedule message to its neighbors. However, if node new did not receive any one "backup success" message, all the neighbors cannot provide a conflict-free and data freshness schedule for node new by postponing their transmission time slots. In this situation, the algorithm DADEC will be run to reschedule all the nodes in the DEC network.

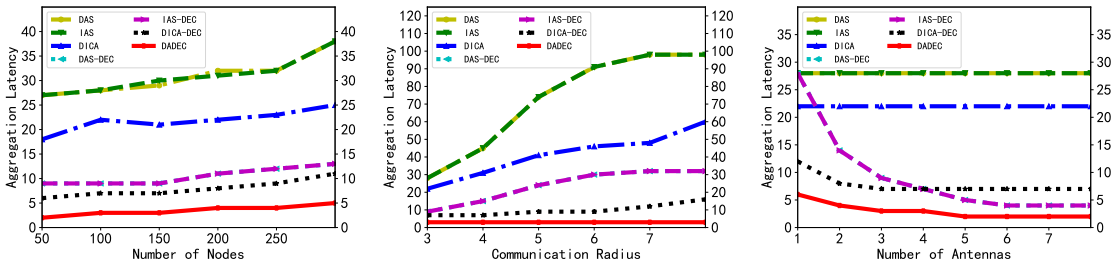
In summary, the strategy calculates an available time slot among the neighbors for the new joining node. If it succeeds, the strategy will not increase aggregation latency. The experimental results show that the success rate in providing a feasible transmission time slot for a new node by the strategy reaches 90%.

4.4.2 Handling Link Failure and Node Failure. When a link fails, there are two cases to consider. Let's consider the broken link to be (u, v). In the case of (u, v) not in the data aggregation tree, it has no effect on the data aggregation

process and can be ignored. If the broken link (u, v) is in the data aggregation tree $u \in Ch_{\psi}(v)$, the node u needs to be rescheduled. Node u first removes node v from its neighbor node set $Nb(u)$, and then node u broadcasts a "repair link" message to its neighbors. Upon receiving the "repair link" message, neighbors of node u except its children will reply with their transmission schedule. Next, except for not having to reset $tm(u)$ to -1, the processing strategy is the same as joining a new node (as discussed before).

When a node fails, this case is equivalent to that all links from broken nodes to neighbors fail. Let the broken node be u. The nodes in set $Ch_{\psi}(u)$ need to be rescheduled. Firstly, each node $w \in Ch_{\psi}(u)$ removes node u from its neighbor set $Nb(w)$ and then runs the above strategy of processing link failure to calculate a feasible transmission time slot for node u. When node u has a large number of children, it is highly probable that the above strategy cannot find a feasible transmission for all children. In this case, the algorithm DADEC will be run again to reschedule all the nodes of DEC network.

5 Simulation Results



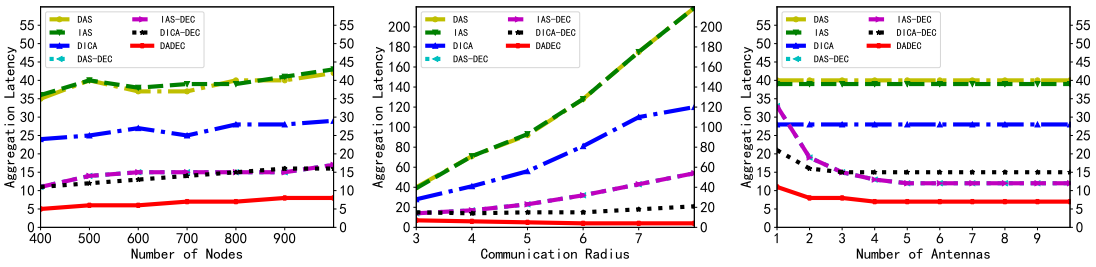
(a) $cr = 3$ and $na = 3$. (b) $n = 100$ and $na = 3$. (c) $n = 100$ and $cr = 3$.

Fig. 8. Aggregation latency in small DEC networks.

In this section, the performance of the proposed DADEC algorithm is evaluated with extensive simulations and compared with existing works such as DAS [58], IAS [46], and DICA [5]. We verify the effectiveness of the proposed DADEC with modified versions of DAS, IAS, and DICA, adapting them to DEC networks and inference models. We refer to these modified versions of DAS, IAS, and DICA as DAS-DEC, IAS-DEC, and DICA-DEC, respectively.

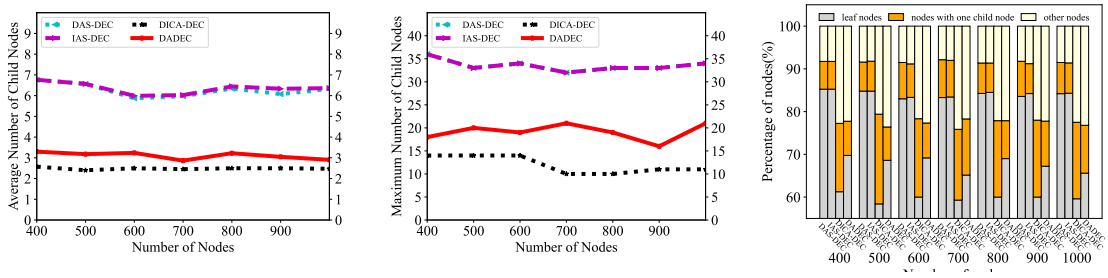
To evaluate the performance of these algorithms, we use Networkx [1] to generate different network topologies. In the simulations, we focus on aggregation latency in small and massive DEC networks. We randomly deploy nodes in square fields. Further, we analyze DADEC's effectiveness by comparing node distributions in aggregation trees. In addition, we simulate topology changes in the network to verify DADEC's robustness. The reported results of each plotted point are the average of 100 executions.

We use the following notation to describe the simulations: (i) n denotes the network size, i.e., the number of nodes; (ii) na denotes the number of antennas each node has (identical for all nodes); and (iii) cr is the communication radius of a node. The cr determines the connection density of the network. Different connection density determines the different average number of neighbor nodes.



(a) $cr = 3$ and $na = 4$. (b) $n = 800$ and $na = 4$. (c) $n = 800$ and $cr = 3$.

Fig. 9. Aggregation latency in massive DEC networks.



(a) Average number of child nodes (b) Maximum number of child nodes (c) Distribution of leaf nodes

Fig. 10. Node distribution in an aggregation tree.

5.1 Aggregation Latency in Small DEC Networks

Fig. 8 shows the data aggregation latency as a function of n, cr and na in small DEC networks. We notice that the proposed DADEC aggregation latency is lower than DAS, IAS, and DICA. This is because of MIMO, which reduces transmission interference and increases parallel transmission capability, thus greatly reducing aggregation latency. Compared with DAS-DEC and IAS-DEC, DADEC decreases aggregation latency by 4.4 × on average. In DAS-DEC and IAS-DEC, a CDS-based tree is constructed for data aggregation, resulting in a large average number of children and a high aggregation latency. Compared with DICA-DEC, DADEC decreases aggregation latency by 2.8 × on average. In DICA-DEC, the aggregation tree is constructed with a traditional SISO network, which generates a deeper tree than DADEC, which is based on MIMO. A deeper tree results in larger aggregation latency, since nodes are scheduled based on the aggregation tree level by level.

Another observation from Fig. 8(c) is that as the number of antennas increases, the performance of DICA-DEC is surpassed by IAS-DEC. DICA is the most efficient existing algorithm for minimizing the aggregation latency in traditional WSNs. However, when DICA and IAS are modified for the DEC network, IAS-DEC performance exceeds DICA-DEC, which indicates that DICA does not scale well and DICA-DEC lacks stability. From Fig. 8, we can also see that DADEC not only obtains the lowest aggregation latency but also is the most stable.

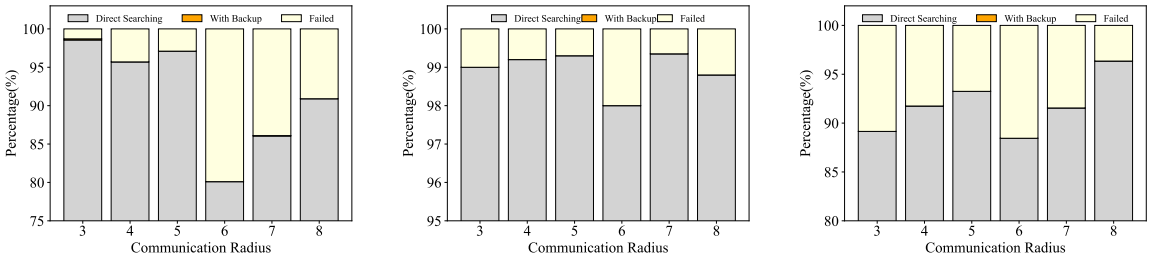
5.2 Aggregation Latency in Massive DEC Networks

In DEC networks, there are often a large number of nodes. Fig. 9 shows that the data aggregation latencies of DADEC are again far below those of DAS, IAS, and DICA. Compared with DAS-DEC and IAS-DEC, DADEC decreases the aggregation latency by 3.3 × on average. Compared with DICA-DEC, DADEC decreases the aggregation latency by 2.4 × on average. The performance improvement rate is slightly lower than that of small DEC networks. This is because the aggregation latency base in a massive DEC network is larger. Specifically, when there are more nodes, the aggregation tree will have more branches. However, children on different branches do not compete for time slots because there is no conflict according to the interference model. Therefore, only adding nodes on the same branch will lead to an increase in data aggregation delay, and adding nodes on different branches will not lead to an increase in data aggregation latency. When the number of branches is more, the number of nodes that can be connected is more for each additional time slot. The results of Fig. 9 show also that DADEC not only obtains the lowest latency but also is the most stable. DADEC has the maximum stability because it uses each available time slot more fully and evenly. The simulation results in both small and massive DEC networks show high performance for DADEC, indicating good node scalability.

5.3 Node Distribution in Aggregation Trees

In this section, we investigate the node distribution in the generated data aggregation tree. Since the aggregation trees of DAS-DEC, IAS-DEC, and DICA-DEC are the same as those of DAS, IAS, and DICA, we only compare DADEC with DAS-DEC, IAS-DEC, and DICA-DEC. We set the number of antennas to be 3 for each node. The simulation results indicate that the aggregation tree constructed by DADEC has the most suitable average number and maximum number of child nodes, which fully and evenly utilizes communication resources, resulting in minimal aggregation latency.

As shown in Fig. 10(a) and Fig. 10(b), the average and maximum number of child nodes of DAS-DEC and IAS-DEC are much larger than those of DADEC and DICA, which corresponds to a higher proportion of leaf nodes with DAS-DEC and IAS-DEC compared to DICA-DEC and DADEC (Fig. 10(c)). The large average and maximum number of child nodes lead to large aggregation latency. However, the average number of child nodes and the maximum number of child nodes of DICA-DEC is smaller than that of DADEC, which indicates that the aggregation tree of DICA-DEC is deeper than that of DADEC. Because nodes are scheduled based on aggregation tree level by level, a deeper tree also leads to larger aggregation latency.



(a) Node joining. (b) Link failure. (c) Node failure.

Fig. 11. Result analysis of handling dynamic changes in network topology.

5.4 Handling Network Topology Changes

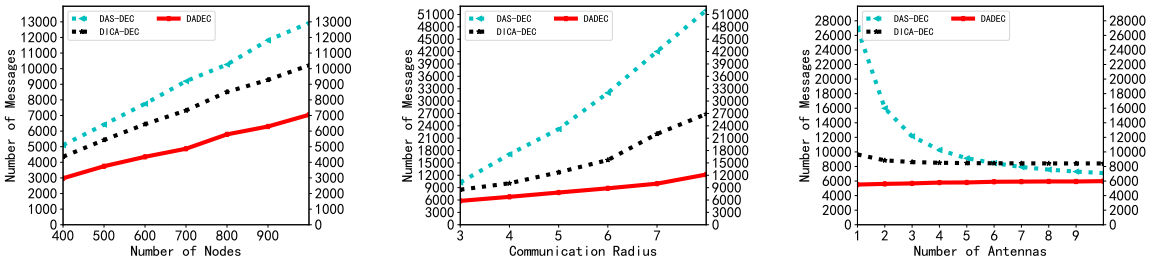
In this section, we evaluate the performance of DADEC for handling network topology changes, namely, node joining, link failure, and node failure. Fig. 11 shows the results. The grey part represents the success ratio of the direct search strategy, the orange part represents the success ratio of the backup strategy and the yellow part denotes the failure ratio.

Fig. 11(a) shows the result of a new node joining the DEC network. The direct search strategy returns a feasible transmission schedule for a new joining node in an average of 91% cases, illustrating the strong scalability of the proposed strategy to find available time slots for the new node among its neighbors after they execute DADEC. Fig. 11(b) shows the results of handling link failure. The direct search strategy successfully handles link failure in an average of 99% of cases. Finally, Fig. 11(c) shows the results of handling node failure. The direct search strategy returns a feasible transmission schedule for all children of the broken node in an average of 91% cases.

The success ratio of the strategy for node joining is equal to that for node failure, and lower than that for link failure. This is because the new node increases the number of communications, which decreases the available communication resources, thereby reducing the probability of finding a feasible available time slot among neighbors. When a node fails, the available communication resources are reduced, reducing the possibility of finding a viable time slot among neighbors. However, after direct searching fails, the probability of backup strategy finding available time slots is almost 0%. This indicates that the proposed DADEC algorithm schedules every node fairly and evenly. Therefore, while direct searching failing, the nodes involved in the backup also can not provide available time slots at this time.

5.5 Messages Overhead

Fig. 12 shows the number of messages generated by distributed methods DAS-DEC, DICA-DEC and DADEC. As shown in Fig. 12(a), Fig. 12(b) and Fig. 12(c), the number of messages generated by DADEC is significantly smaller than that of the other methods across all scenarios. This is because DADEC employs MIMO technology for data aggregation tree construction and scheduling, and it only needs to coordinate with its direct neighbors. In contrast, the DAS-DEC uses a two-step disjoint procedure in which each node must communicate with nodes located two hops away. DICA-DEC first adopts DICA's method for constructing the aggregation tree and then applies MIMO-based scheduling—also using a two-step disjoint process—which leads to a greater number of control messages than DADEC. Furthermore, when the number of antennas exceeds seven, the message overhead of DICA-DEC even surpasses that of DAS-DEC.



(a) $cr = 3$ and $na = 4$. (b) $n = 800$ and $na = 4$. (c) $n = 800$ and $cr = 3$.

Fig. 12. The number of exchanged messages.

5.6 Discussion

Our simulation results confirm the following performance improvements:

(1) DADEC has excellent performance in aggregation latency not only in small scale DEC networks but also in massive scale networks. This indicates that DADEC has high adaptability.
(2) Compared with existing methods, the aggregation tree generated by DADEC has the best structure in terms of average number and maximum number of child nodes.
(3) DADEC demonstrates strong robustness for handling network topology changes.
(4) Compared with existing distributed algorithms, DADEC has the smallest message overhead.

DADEC is an efficient distributed algorithm that only needs to send and exchange messages between adjacent nodes. The information in each node does not need to be transmitted to the whole DEC network, which saves communication overhead. However, DADEC has its limitations: when both the direct search and the backup fail, DADEC will be run again to reschedule all the nodes of the whole DEC network. This is because the strategies for handling network topology changes just search neighbors within one to two levels of the damaged nodes or links. However, according to simulation results, the full DADEC process needs to run rarely for topology changes. This is because the direct searching and backup procedures are generally able to find a conflict-free schedule, owing to the robust performance of the algorithms and the ample number of antennas available at neighboring nodes.

Data aggregation has long been a key research focus in traditional WSNs due to its potential to significantly reduce energy consumption and resource overhead. With the rapid development of DEC technologies, the number of terminal devices has increased substantially. As a foundational technology for 5G/6G, MIMO enhances wireless network communication capabilities and improves the utilization of communication resources in DEC networks. Data aggregation in MIMO-enabled DEC networks not only expands communication capacity but also lowers energy consumption, making it a pressing and valuable research topic. With the rising demand for real-time applications, minimizing data aggregation latency has become a central issue in recent studies. However, the introduction of MIMO technology leads to a fundamentally different interference model, rendering traditional minimum-latency aggregation methods designed for SISO-based WSNs inadequate. To address this gap, we propose a novel latency-minimizing data aggregation scheme DADEC specifically designed for MIMO-enabled DEC environments.

Compared to SISO systems, MIMO significantly improves communication efficiency and reduces interference. Nevertheless, MIMO antennas typically consume more power, posing challenges for deployment in battery-powered terminal edge devices. Recent advancements in Multiple Directional Antenna Wireless Sensor Networks [31, 50] have demonstrated the feasibility of incorporating MIMO into WSNs. In data aggregation research, real-world deployment remains difficult due to discrepancies between theoretical models and practical environments. Compared to previously studied network architectures, MIMO-enabled networks exhibit greater complexity. Consequently, recent work on MIMO-based data aggregation [61] has primarily relied on simulated experimental setups. Acknowledging the importance of practical validation, we plan to evaluate our proposed approach on testbeds and in real-world applications in future work.

6 Conclusions

In this paper, we provided a first study on minimizing latency in an aggregation schedule for modern DEC networks. Due to the use of multiple antennas and MIMO, the interference model of a modern DEC network is different from that of traditional WSNs. We proposed a distributed and efficient method, named DADEC, which intertwines the construction of an aggregation tree and the scheduling of node transmissions. In DADEC, each node may select its

parent from its neighbors on the same level, upper level, and lower level, which fully utilizes transmission time slots, thereby reducing data aggregation latency. Additionally, we proposed lightweight adaptive strategies to update the schedule in case of network topology changes without increasing data aggregation latency. Simulation results showed that DADEC significantly reduces aggregation latency by at least 2.6× on average compared to the state-of-the-art methods.

In future works, we will investigate the problem of minimizing data aggregation latency in multi-sink DEC networks and in mobile sink DEC networks. Initially, we will investigate the efficiency of centralized and distributed methods. Second, for a mobile sink, the mobility law is an important factor affecting aggregation latency. Therefore, the impact of sink node mobility on data aggregation latency will also be investigated.

Acknowledgments

This work was supported by the Natural Science Research Project of the Anhui Educational Committee (No. KJ2020A0250) and the Postdoctoral Funding Program (2025M771581, GZC20251097).

References

[1] Networkx. http://networkx.lanl.gov.

[2] Alia Asheralieva, Dusit Niyato, and Yoshikazu Miyanaga. Efficient dynamic distributed resource slicing in 6g multi-access edge computing networks with online admm and message passing graph neural networks. *IEEE Transactions on Mobile Computing.*, 23(4):2614–2638, 2023.

[3] Miloud Bagaa, Abdelouahid Derhab, Noureddine Lasla, Abdelraouf Ouadjaout, and Nadjib Badache. Semi-structured and unstructured data aggregation scheduling in wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications*, pages 2671–2675, 2012.

[4] Miloud Bagaa, Mohamed Younis, and Nadjib Badache. Efficient data aggregation scheduling in wireless sensor networks with multi-channel links. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pages 119–124, 2013.

[5] Miloud Bagaa, Mohamed Younis, Djamel Djenouri, Abdelouahid Derhab, and Nadjib Badache. Distributed low-latency data aggregation scheduling in wireless sensor networks. *ACM Transactions on Sensor Networks.)*, 11(3):1–36, 2015.

[6] Kunyi Chen, Hong Gao, Zhipeng Cai, Quan Chen, and Jianzhong Li. Distributed energy-adaptive aggregation scheduling with coverage guarantee for battery-free wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications*, pages 1018–1026, 2019.

[7] Quan Chen, Zhipeng Cai, Lianglun Cheng, and Hong Gao. Low-latency data aggregation scheduling for cognitive radio networks with non-predetermined structure. *IEEE Transactions on Mobile Computing.*, 20(7):2412–2426, 2020.

[8] Quan Chen, Zhipeng Cai, Lianglun Cheng, Feng Wang, and Hong Gao. Joint near-optimal age-based data transmission and energy replenishment scheduling at wireless-powered network edge. In *Proceedings of IEEE International Conference on Computer Communications*, pages 770–779, 2022.

[9] Quan Chen, Hong Gao, Zhipeng Cai, Lianglun Cheng, and Jianzhong Li. Energy-collision aware data aggregation scheduling for energy harvesting sensor networks. In *Proceedings of IEEE International Conference on Computer Communications*, pages 117–125, 2018.

[10] Quan Chen, Hong Gao, Siyao Cheng, Jianzhong Li, and Zhipeng Cai. Distributed non-structure based data aggregation for duty-cycle wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications*, pages 1–9, 2017.

[11] Quan Chen, Song Guo, Kaijia Wang, Wenchao Xu, Jing Li, Zhipeng Cai, Hong Gao, and Albert Y Zomaya. Towards real-time inference offloading with distributed edge computing: the framework and algorithms. *IEEE Transactions on Mobile Computing.*, 23(7):7552–7571, 2023.

[12] Xujin Chen, Xiaodong Hu, and Jianming Zhu. Minimum data aggregation time problem in wireless sensor networks. In *Proceedings of Springer Mobile Ad-hoc and Sensor Networks*, pages 133–142, 2005.

[13] Jong-Moon Chung, Joonhyung Kim, and Donghyuk Han. Multihop hybrid virtual MIMO scheme for wireless sensor networks. *IEEE Transactions on vehicular Technology.*, 61(9):4069–4078, 2012.

[14] Shuguang Cui, Andrea J Goldsmith, and Ahmad Bahai. Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks. *IEEE Journal on selected areas in communications.*, 22(6):1089–1098, 2004.

[15] Penglin Dai, Biao Han, Xiao Wu, Huanlai Xing, Bingyi Liu, and Kai Liu. Distributed convex relaxation for heterogeneous task replication in mobile edge computing. *IEEE Transactions on Mobile Computing.*, 23(2):1230–1245, 2022.

[16] Indrakshi Dey, Hemdutt Joshi, and Nicola Marchetti. Space-time spreading aided distributed MIMO-WSNs. *IEEE Communications Letters.*, 25(4):1338–1342, 2020.

[17] Praveen Kumar Donta, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. Delay-aware data fusion in duty-cycled wireless sensor networks: A Q-learning approach. *Elsevier Sustainable Computing: Informatics and Systems.*, 33:100642, 2022.

[18] Yunquan Gao, Xiaoyong Li, Jirui Li, and Yali Gao. Distributed and efficient minimum-latency data aggregation scheduling for multichannel wireless sensor networks. *IEEE Internet of Things Journal.*, 6(5):8482–8495, 2019.

[19] Xiaowen Gong. Delay-optimal distributed edge computing in wireless edge networks. In *Proceedings of IEEE Conference on Computer Communications*, pages 2629–2638, 2020.

[20] Mohammad Goudarzi, Maria A Rodriguez, Majid Sarvi, and Rajkumar Buyya. μ-DDRL: A qos-aware distributed deep reinforcement learning technique for service offloading in fog computing environments. *IEEE Transactions on Services Computing.*, 17(1):47–59, 2023.

[21] Longjiang Guo, Yingshu Li, and Zhipeng Cai. Minimum-latency aggregation scheduling in wireless sensor network. *Journal of Combinatorial Optimization.*, 31:279–310, 2016.

[22] Vini Gupta and Swades De. An energy-efficient edge computing framework for decentralized sensing in wsn-assisted iot. *IEEE Transactions on Wireless Communications.*, 20(8):4811–4827, 2021.

[23] Abhishek Hazra, Praveen Kumar Donta, Tarachand Amgoth, and Schahram Dustdar. Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications. *IEEE Internet of Things Journal.*, 10(5):3944–3953, 2023.

[24] Biao Hu, Yinbin Shi, Gang Chen, Zhengcai Cao, and Mengchu Zhou. Workload-aware scheduling of real-time jobs in cloud computing to minimize energy consumption. *IEEE Internet of Things Journal.*, 11(1):638–652, 2023.

[25] SC-H Huang, P-J Wan, Chinh T Vu, Yingshu Li, and Frances Yao. Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications*, pages 366–372, 2007.

[26] Baofeng Ji, Ying Han, Shuwen Liu, Fazhan Tao, Gaoyuan Zhang, Zhumu Fu, and Chunguo Li. Several key technologies for 6G: Challenges and opportunities. *IEEE Communications Standards Magazine.*, 5(2):44–51, 2021.

[27] Minhyuk Kang and Sang-Woon Jeon. Energy-efficient data aggregation and collection for multi-uav-enabled iot networks. *IEEE Wireless Communications Letters.*, 13(4):1004–1008, 2024.

[28] Virve Karsisto and Lauri Lovén. Verification of road surface temperature forecasts assimilating data from mobile sensors. *Weather and Forecasting.*, 34(3):539–558, 2019.

[29] Henna Kokkonen, Susanna Pirttikangas, and Lauri Lovén. EISim: A platform for simulating intelligent edge orchestration solutions. *arXiv preprint arXiv:2311.01224.*, 2023.

[30] D Praveen Kumar, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. Machine learning algorithms for wireless sensor networks: A survey. *Elsevie Information Fusion.*, 49:1–25, 2019.

[31] Tan D. Lam and Dung T. Huynh. Target coverage and connectivity in directional wireless sensor networks. In *Proceedings of IEEE Conference on Computer Communications*, pages 1–10, 2023.

[32] Erik G. Larsson, Ove Edfors, Fredrik Tufvesson, and Thomas L. Marzetta. Massive MIMO for next generation wireless systems. *IEEE Communication Magazine.*, 52(2):186–195, 2014.

[33] Matti Latva-Aho, Kari Leppänen, et al. Key drivers and research challenges for 6G ubiquitous wireless intelligence, 2019.

[34] Ji Li, Siyao Cheng, Zhipeng Cai, Jiguo Yu, Chaokun Wang, and Yingshu Li. Approximate holistic aggregation in wireless sensor networks. *ACM Transactions on Sensor Networks.*, 13(2):1–24, 2017.

[35] Jing Li, Song Guo, Weifa Liang, Quan Chen, Zichuan Xu, Wenzheng Xu, and Albert Y Zomaya. Digital twin-assisted, sfc-enabled service provisioning in mobile edge computing. *IEEE Transactions on Mobile Computing.*, 23(1):393–408, 2022.

[36] Jing Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, Yuchen Li, and Xiaohua Jia. Service home identification of multiple-source iot applications in edge computing. *IEEE Transactions on Services Computing.*, 16(2):1417–1430, 2022.

[37] Xiuhua Li, Zhenghui Xu, Fang Fang, Qilin Fan, Xiaofei Wang, and Victor CM Leung. Task offloading for deep learning empowered automatic speech analysis in mobile edge-cloud computing networks. *IEEE Transactions on Cloud Computing.*, 11(2):1985–1998, 2022.

[38] Onel LA López, Osmel M Rosabal, David E Ruiz-Guirola, Prasoon Raghuwanshi, Konstantin Mikhaylov, Lauri Lovén, and Sridhar Iyer. Energy-sustainable iot connectivity: Vision, technological enablers, challenges, and future directions. *IEEE Open Journal of the Communications Society.*, 4:2609–2666, 2023.

[39] Lauri Lovén, Virve Karsisto, Heikki Järvinen, Mikko J Sillanpää, Teemu Leppänen, Ella Peltonen, Susanna Pirttikangas, and Jukka Riekki. Mobile road weather sensor calibration by sensor fusion and linear mixed models. *PloS one.*, 14(2):e0211702, 2019.

[40] Lauri Lovén, Tero Lähderanta, Leena Ruha, Ella Peltonen, Ilkka Launonen, Mikko J Sillanpää, Jukka Riekki, and Susanna Pirttikangas. EDISON: An edge-native method and architecture for distributed interpolation. *Sensors.*, 21(7):2279, 2021.

[41] Junte Ma, Sihao Xie, and Jin Zhao. Flexible offloading of service function chains to programmable switches. *IEEE Transactions on Services Computing.*, 16(2):1198–1211, 2022.

[42] Baljeet Malhotra, Ioanis Nikolaidis, and Mario A Nascimento. Aggregation convergecast scheduling in wireless sensor networks. *Wireless Networks.*, 17:319–335, 2011.

[43] Andreas F Molisch, Vishnu V Ratnam, Shengqian Han, Zheda Li, Sinh Le Hong Nguyen, Linsheng Li, and Katsuyuki Haneda. Hybrid beamforming for massive MIMO: A survey. *IEEE Communications magazine.*, 55(9):134–141, 2017.

[44] Ella Peltonen, Ijaz Ahmad, Atakan Aral, Michele Capobianco, Aaron Yi Ding, Felipe Gil-Castineira, Ekaterina Gilman, Erkki Harjula, Marko Jurmu, Teemu Karvonen, et al. The many faces of edge intelligence. *IEEE Access.*, 10:104769–104782, 2022.

[45] Ella Peltonen, Mehdi Bennis, Michele Capobianco, Merouane Debbah, Aaron Ding, Felipe Gil-Castiñeira, Marko Jurmu, Teemu Karvonen, Markus Kelanti, Adrian Kliks, et al. 6G white paper on edge intelligence. *arXiv preprint arXiv:2004.14850.*, 2020.

[46] Meirui Ren, Longjiang Guo, and Jinbao Li. A new scheduling algorithm for reducing data aggregation latency in wireless sensor networks. *Scientific Research Publishing International Journal of Communications, Network and System Sciences.*, 3(8):679–688, 2010.

[47] Chenguang Shi, Yijie Wang, Sana Salous, Jianjiang Zhou, and Junkun Yan. Joint transmit resource management and waveform selection strategy for target tracking in distributed phased array radar network. *IEEE Transactions on Aerospace and Electronic Systems.*, 58(4):2762–2778, 2021.

[48] G Thirumal, Chiranjeev Kumar, and Praveen Kumar Donta. Low discrepancy on non-linear sensor deployment in a time-critical linear IIoT network. *Internet of Things.*, page 101165, 2024.

[49] PC Thirumal and L Shylu Dafni Agnus. Forest fire detection and prediction–survey. In *2022 International Conference on Inventive Computation Technologies*, pages 1295–1302, 2022.

[50] Tien Tran and Dung T. Huynh. Symmetric connectivity algotirthms in multiple directional antennas wireless sensor networks. In *Proceedings of IEEE Conference on Computer Communications*, pages 333–341, 2018.

[51] Ali Akbar Vali, Sadoon Azizi, and Mohammad Shojafar. RESP: A recursive clustering approach for edge server placement in mobile edge computing. *ACM Transactions on Internet Technology.*, 24(3):1–25, 2024.

[52] Peng-Jun Wan, Scott C-H Huang, Lixin Wang, Zhiyuan Wan, and Xiaohua Jia. Minimum-latency aggregation scheduling in multihop wireless networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 185–194, 2009.

[53] Hao Wang, Shenglan Ma, Chaonian Guo, Yulei Wu, Hong-Ning Dai, and Di Wu. Blockchain-based power energy trading management. *ACM Transactions on Internet Technology.*, 21(2):43:1–43:16, 2021.

[54] Liantao Wu, Peng Sun, Zhibo Wang, Yanjun Li, and Yang Yang. Computation offloading in multi-cell networks with collaborative edge-cloud computing: A game theoretic approach. *IEEE Transactions on Mobile Computing.*, 23(3):2093–2106, 2023.

[55] Li Xiong, Subramanyam Chitti, and Ling Liu. Preserving data privacy in outsourcing data aggregation services. *ACM Transactions on Internet Technology.*, 7(3):17, 2007.

[56] XiaoHua Xu, Xiang Yang Li, XuFei Mao, Shaojie Tang, and ShiGuang Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems.*, 22(1):163–175, 2011.

[57] Zichuan Xu, Dapeng Zhao, Weifa Liang, Omer F Rana, Pan Zhou, Mingchu Li, Wenzheng Xu, Hao Li, and Qiufen Xia. HierFedML: aggregator placement and UE assignment for hierarchical federated learning in mobile edge computing. *IEEE Transactions on Parallel and Distributed Systems.*, 34(1):328–345, 2022.

[58] Bo Yu, Jianzhong Li, and Yingshu Li. Distributed data aggregation scheduling in wireless sensor networks. In *Proceedings of IEEE Conference on Computer and Communicaitons*, pages 2159–2167.

[59] Qiyang Zhang, Xiangying Che, Yijie Chen, Xiao Ma, Mengwei Xu, Schahram Dustdar, Xuanzhe Liu, and Shangguang Wang. A comprehensive deep learning library benchmark and optimal library selection. *IEEE Transactions on Mobile Computing.*, 23(5):5069–5082, 2023.

[60] Qiyang Zhang, Xiang Li, Xiangying Che, Xiao Ma, Ao Zhou, Mengwei Xu, Shangguang Wang, Yun Ma, and Xuanzhe Liu. A comprehensive benchmark of deep learning libraries on mobile devices. In *Proceedings of the ACM Web Conference*, pages 3298–3307, 2022.

[61] Miao Zhao and Yuanyuan Yang. A framework for mobile data gathering with load balanced clustering and MIMO uploading. In *Proceedings of 30th IEEE International Conference on Computer Communications*, pages 2759–2767, 2011.