Link-delay-aware Reinforcement Scheduling for Data Aggregation in Massive IoT

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

30-10-2021 / 01-11-2021

CITATION

Vo, Van-Vi; Nguyen, Tien-Dung; Le, Duc-Tai; Kim, Moonseong; Choo, Hyunseung (2021): Link-delay-aware Reinforcement Scheduling for Data Aggregation in Massive IoT. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.16908511.v1>

DOI

[10.36227/techrxiv.16908511.v1](https://doi.org/10.36227/techrxiv.16908511.v1)

Link-delay-aware Reinforcement Scheduling for Data Aggregation in Massive IoT

Van-Vi Vo, Tien-Dung Nguyen, Duc-Tai Le, Moonseong Kim, and Hyunseung
Choo, *Member, IEEE*

Abstract

Over the past few years, the use of wireless sensor networks in a range of Internet of Things (IoT) scenarios has grown in popularity. Since IoT sensor devices have restricted battery power, a proper IoT data aggregation approach is crucial to prolong the network lifetime. To this end, current approaches typically form a virtual aggregation backbone based on a connected dominating set or maximal independent set to utilize independent transmissions of dominators. However, they usually have a fairly long aggregation delay because the dominators become bottlenecks for receiving data from all dominatees. The problem of time-efficient data aggregation in multichannel duty-cycled IoT sensor networks is analyzed in this paper. We propose a novel aggregation approach, named LInk-delay-aware REinforcement (LIRE), leveraging active slots of sensors to explore a routing structure with pipeline links, then scheduling all transmissions in a bottom-up manner. The reinforcement schedule accelerates the aggregation by exploiting unused channels and time slots left off at every scheduling round. LIRE is evaluated in a variety of simulation scenarios through theoretical analysis and performance comparisons with a state-of-the-art scheme. The simulation results show that LIRE reduces more than 80% aggregation delay compared to the existing scheme.

Index Terms

Van-Vi Vo is with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: vovanvi@skku.edu).

Tien-Dung Nguyen is with the School of Electronics and Telecommunications, Hanoi University of Science and Technology, Ha Noi 10999, Viet Nam (e-mail: dung.nguyentien2@hust.edu.vn).

Moonseong Kim is with the Department of IT Convergence Software, Seoul Theological University, Bucheon 14754, South Korea (e-mail: moonseong@stu.ac.kr).

Duc-Tai Le and Hyunseung Choo are with the College of Computing and Informatics, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: ldtai@skku.edu, choo@skku.edu)

data aggregation, internet of things, multichannel, duty cycle, wireless sensor networks.

I. INTRODUCTION

Today's infrastructure systems, such as smart homes, smart grids, smart water networks, and intelligent transportation connect our world in ways we never imagined. Internet of Things (IoT) connects such systems all-in-one with the use of sensors. Sensors distributed in any environment connect to form a network, such a sensor network measures and collects data from terrestrial, underground, underwater, or even body area [1]. Sensor networks collect raw data, which must be analyzed and stored, but the data volume becomes enormous. By combining similar data and reducing redundancy in sensed data, the data aggregation technique is used to address this issue [2]. The technique improves energy efficiency, increases network longevity, enhances bandwidth utilization, and minimizes data transmission latency.

As they are powered by nonrechargeable batteries, sensor nodes have limited energy. Although sensor nodes are always awake to listen to the transmissions, they do not always send and receive data from their neighbors. Being awake while idling consumes a significant amount of energy. Furthermore, when aggregating data, sensors relay data in a multihop network where data are aggregated and forward by intermediate sensor nodes until it reaches the sink or the base station. Therefore, energy conservation is a major issue for extending the lifetime of a network. To this end, emerging energy-harvesting technology in the sensor nodes is continues to advance, it still faces challenges such as implementation complexity and low harvesting efficiency [3]. Another notable way to save energy is to use duty cycle technique, but this comes at the cost of increased data aggregation latency [4, 5].

To date, many aggregation scheduling schemes are using a single channel. Multiple channels for multiple frequencies can be used by sensor nodes to send and receive data. Using multiple channels increases concurrent collision-free transmissions, which improves data aggregation time efficiency [6, 7, 8, 9]. In always-on sensor networks, the proposed schemes aim to reduce data aggregation delay. Despite that the performance becomes worse if these schemes apply the problem in duty-cycled networks, energy is conserved so that the lifespan of the networks can be extended which adapts to the needs of our lives is to save or reuse the energy.

In conventional delay-efficient data aggregation (DDA) solutions, a typical aggregation structure is assumed to be built using a connected dominating set (CDS) or shortest-path tree (SPT)

[10]. After that, data scheduling is based on sender-receiver pairs in this structure. To avoid collisions, dominatees send data to their dominators sequentially using a CDS-based structure [10, 11, 12, 13]. The CDS-based approach works well in sparse and tiny networks, but it becomes a problem in dense networks. When the number of dominatees exceeds the number of dominators, the dominatees become a burden to the dominators. A bottleneck may occur at the sink or base station where many nodes concentrate in the SPT-based approach [14], the problem is more serious in dense networks. In such those types of trees, only primary collisions are covered which waste time slots. As a result, scheduling in those tree structures may not result in adequate delay performance.

The DDA problem in multichannel duty-cycled IoT sensor networks is investigated in this article. We propose a LInk-delay-aware REinforcement scheduling (LIRE), which uses the active slots of sensor nodes to explore the aggregation tree with pipeline links, to overcome the limitations of current approaches. After that, the link-delay-aware scheduling scheme augments the number of scheduling candidates by considering all leaf nodes of the network at each iteration and then invokes a reinforcement scheduling scheme. The pipeline links are used in the reinforcement scheme to maximize transmissions in a single working period. The main contributions of this paper are summarized as follows.

- 1) The wake-up time of the receivers to adopt children is used in a link-delay aggregation tree construction approach that we propose. A motivation for using the new tree construction approach is presented, followed by a step-by-step description of the tree construction process, then children distribution affected by tree construction method compared to the reference scheme is presented. We also analyze how the aggregation tree's children distribution affects the scheduling of nodes.
- 2) We propose a reinforcement scheduling scheme that enhances the nodes scheduling on unused channels and time slots at the current working period, missed by the candidate nodes schedule scheme. To obtain more scheduled nodes, the proposed scheme examines each slot of the current working period one by one. The proposed scheme reduces aggregation delay even more with this greedy scheduling strategy.
- 3) We demonstrate detailed examples throughout the article. In addition, we perform extensive simulations to evaluate the performance of our proposed scheme in terms of aggregation delay, to compare it to an existing scheme. The results show that the proposed scheme

outstandingly achieves lower aggregation delay.

The rest of the paper is organized as follows. In Section II, we go over the related work. The network model and problem formulation are presented in Section III. Then, in Section IV, we demonstrate our proposed scheme with illustrated examples and algorithm analysis. After that, the performance evaluation of our proposed scheme is provided in Section V. Finally, we conclude our work and discuss the future direction in Section VI.

II. RELATED WORK

Massive IoT applications, which require a large number of devices with low energy, low cost, and small data volumes, and critical IoT applications, which involve very high data availability with low latency and high-reliability requirements, are two types of IoT applications. IoT sensor devices are known for their low battery power, limited computation, and communication resources [15]. For IoT devices to the last longer, energy harvesting is essential. Although the energy harvesting technology is getting concerned recently, these systems have certain drawbacks, such as the amount of harvesting energy is low, the harvesting system is inefficient and the inaccessibility of the energy source where the energy intends to be harvested [16]. As a result, demand for energy-efficient scheduling methods is always high. Duty cycling, in which a sensor node is regularly put into sleep mode when there is no transmission, is a well-known approach to conserve energy [17]. However, because fewer sensor nodes are awake to transmit data, this mechanism increases transmission latency and reduces throughput.

The authors in [18] prove the Delay-efficient Data Aggregation scheduling (DDA) problem is NP-hard and propose an algorithm with latency bound $(\Delta - 1)R$ - approximation algorithm for that problem, in which Δ equals to the maximum node degree of a sensor node and R is communication range. Another algorithm based on maximal independent sets (MIS) with a latency bound $23R + \Delta - 18$ is proposed in [19], this algorithm offers a near-constant approximation with a much lower latency than the one in [18]. Xiaohua Xu et al. [20] present three algorithms with latency $15R + \Delta - 4$, $2R + O(\log R) + \Delta$ and $(1 + O(\log R / \sqrt[3]{R}))R + \Delta$. Another time-efficient approach is proposed in [21], the latency of the proposed scheme is upper-bounded by $(\lfloor \frac{2\pi}{\arccos(\frac{1}{1+\epsilon})} \rfloor + 4)R + \Delta - 4$, where $0.05 < \epsilon \leq 1$.

All of the preceding methods are centralized algorithms. The distributed algorithms have also been extensively studied for the DDA problem. The authors in [22] present a distributed algorithm based on MIS with the latency bound $24D + 6\Delta + 16$, where D is the network diameter

($D \leq 2R$) and Δ is the maximum node degree. Later that same year, XiaoHua Xu et al. [23] propose an improved algorithm with an upper-bounded data aggregation latency $16R + \Delta - 14$, where R is the communication range. Another distributed algorithm is proposed in [24] named DICA with the time latency is at most $(\lfloor \frac{2\pi}{\arccos(\frac{1}{1+\epsilon})} \rfloor + 3)R + \Delta - 4$, DICA intertwines the tree construction and nodes scheduling to lower the aggregation delay. In [25], the authors propose the FAST algorithm, which aims to reduce time latency $12R + \Delta - 2$.

However, all the works discussed above are solving the DDA problem using one channel for each sensor node to communicate. Using multichannel, several studies have been conducted to address this issue. The authors in [26] design a scheduling algorithm with a schedule length of $O(\Delta(T)\log n)$ and the smallest number of channels required to remove all collisions, where $\Delta(T)$ is the maximum node degree on the tree T and n is the number of nodes in the network. Another method is studied to solve the DDA problem in multichannel multihop sensor networks is proposed in [27], the approximation algorithm has a latency upper bound of nearly $(\alpha + 11\beta)$, where α, β are constants along with communication r formed collision range αr and carrier sensing range βr . Ghods et al. [28] design a scheme to reduce the data aggregation delay, named MC-MLAS, the scheme uses multichannel to execute the aggregation tree construction and scheduling simultaneously. A distributed algorithm is proposed in [29], a cluster-based distributed data aggregation scheduling scheme that aims to reduce data aggregation delay in multichannel and multipower sensor networks. Nguyen et al. [30] employ a distributed collision-avoidance scheduling algorithm, named DCAS, to minimize latency for data aggregation in WSNs.

In always-on networks, the studies listed above are attempting to reduce the data aggregation latency for IoT sensors. Even though they do not collect data all of the time, the sensor nodes are always awake. The sensor nodes waste energy due to the wake-up time without performing any mission, the network lifetime becomes lower. By putting sensors into sleep mode when they are in idle time, the duty cycle mechanism is used to conserve energy. The DDA problem in multichannel duty-cycled sensor networks is investigated in [13, 31, 32]. Yu and Li [13] consider the DDA problem in duty-cycled WSNs for the first time, they show an approximation algorithm based on CDS with aggregation latency nearly-constant not exceed $(15R + \Delta - 3)T$, where R is a lower bound on the aggregation delay, T is the working period length and Δ is a maximum degree in communication graph G . [31] proposes the first distributed algorithm for duty-cycled WSNs without considering routing structure, the algorithm simultaneously does

nodes scheduling as well as generates an aggregation tree. To reduce aggregation delay, Jiao et al. [32] propose two maximal independent set-based algorithms based on two new conflict graph concepts. The authors in [33] claim that routing structure algorithms greatly affect aggregation latency. Neither SPT nor CDS are sufficient to construct aggregation trees because of delay efficiency. Therefore, in this research, we propose a new aggregation tree construction approach where the active slots of nodes are applied. We minimize the latency value between senders and receivers when constructing the aggregation tree to increase the number of transmissions in a working period so that reducing the total aggregation delay.

To date, revolutionary technologies such as machine learning, blockchain are greatly applied to most applications automatically and efficiently operate and manage IoT sensor data. The Authors in [34] use Fuzzy logic to find a shortest path for data aggregation, the proposed method improves throughput and conserves energy for the network. Data aggregation scheduling schemes for sensor nodes based on Q-learning are proposed in [35, 36] aiming to improve energy and obtain a longer network lifetime. Wang et al. [37] design a Blockchain-based scheme for edge computing empowered IoT obtaining high throughput, low transaction latency and energy efficiency with the restriction on a security level. However, in this study, we use a traditional approach to investigate the DDA problem in multichannel duty-cycled IoT sensor networks. So far, the DDA problem in multichannel duty-cycled IoT sensor networks is only studied in [32] where we compare our approach with their proposed scheme with intensive simulation scenarios.

III. PRELIMINARIES

A. Network model and assumptions

We model a WSN as an undirected graph $G = (V, E)$ where V is the set of sensor nodes and E is the set of communication links. An omnidirectional antenna with a fixed transmission range is installed on each node in the network. The sensor nodes operate in a half-duplex mode so that they can only send or receive the data from others in a certain slot. A communication link (u, v) between sender u and receiver v belonging to E exists when the Euclidean distance between them is less than or equal to the transmission range R , and u and v are neighbors.

For WNS, duty-cycled mechanism is proposed, in which sensor nodes switch between active and dormant modes. We divide the scheduling time into working periods that have the same lengths. A working period is further divided into $\mathcal{L}$ slots, $\mathcal{L} \in \mathbb{N}$, i.e., indexing from 0 to $\mathcal{L} - 1$,

Fig. 1: Collisions in wireless networks.

in which a slot is a sufficient time duration for one transmission. In duty-cycled networks, a sensor node is active in some slots and sleeps in the others during the same working period to conserve energy. In this mechanism, a node can receive data from others only at its active slots but can wake up at any time slot for data transmission. In this paper, each sensor node is randomly active at α one or multiple slots per working period, $\alpha \in \mathbb{N}, 0 < \alpha < \mathcal{L}$. The duty-cycled ratio of a node is $\alpha/\mathcal{L}$.

Two types of data collisions occur in every transmission in the WSN. When two nodes transmit their data to a common neighbor at the same time, a primary collision occurs at the common neighbor. When a node receives data from a sender, but it also overhears data from another sender at the same time, a secondary collision occurs at the received node. Collisions in wireless networks are depicted in Fig. 1. In both cases (a) and (b), if nodes B and C transmit data at the same time slot, then collision happens at nodes A and A cannot receive any data from B or C . To avoid such collision, transmissions from B and C can be assigned different time slots or allocated different channels if it is a secondary collision. We assume that the network has a set of available channels, $\mathcal{F} = \{1, 2, \dots, f\}$, where f is the number of channels.

B. Problem formulation

The data aggregation problem aims to collect data from all nodes to the sink. The data aggregation structure is typically modeled as a tree rooted at the sink, with each child node transmitting data to its parent in the tree. An intermediate node receives data from its children then aggregates with its data. To send to the sink node, the aggregated data is compressed into a single packet. The data aggregation scheduling process, which assigns channels and time slots

to nodes in the network, completes when all sensor nodes except the sink node are allocated channels and assigned time slots.

Let $G = (V, E)$ be the communication graph of a sensor network where V is a set of vertices and E is a set of edges between vertices. We construct an aggregation tree $T = (V_T, E_T)$ consisting of V_T sensor nodes and E_T edges based on the communication graph. Let $s(u) = [f(u), t(u)]$ denote a transmitting schedule of node u . Herein, node u is scheduled to transmit data at time slot $t(u)$ on channel $f(u)$. Given the number of available channels limited to f and a working period divided into $\mathcal{L}$ slots. The minimum latency aggregation schedule in a multichannel duty-cycled network is formulated as follows.

Finding a data aggregation schedule $\mathcal{S}$ that minimizes the total data aggregation delay $\mathcal{D}$ such that:

- 1) $f(u) \in \{1, 2, \dots, f\}, \forall u \in V \setminus \{s\}$;
- 2) $s(u) \neq, \forall u \in V \setminus \{s\}$. All nodes in the network except sink s must be scheduled;
- 3) $t(u) < t(p(u)), \forall u \in V \setminus \{s\}$. A parent node must send data after its child nodes;
- 4) If $p(u) \equiv p(v)$ then $t(u) \neq t(v), \forall u, v \in V \setminus \{s\}$. Two nodes that have the same parent cannot be scheduled at the same time to avoid a primary collision;
- 5) If $p(u) \neq p(v)$ and $(u \in N(p(v))$ or $v \in N(p(u)))$ then $f(u) \neq f(v)$ or $t(u) \neq t(v), \forall u, v \in V \setminus \{s\}$. Allocating different available channels, or different working periods to avoid the secondary collision while doing scheduling.

IV. LINK-DELAY DATA AGGREGATION SCHEDULING

Table I presents all notation and abbreviations used throughout the paper.

A. Motivation

The recent solution uses an MIS to route sensory data to the sink for sensors with multiple active slots. Each dominator must collect data from all of its neighbors. Because they have so many neighbors, these dominators become bottlenecks in aggregating data in a dense network. If nodes with asynchronous wake-up slots are improperly scheduled, the total aggregation delay may increase. The duty-cycled awareness is proved much more effective than MIS or CDS [32]. The calculation of waiting time between two nodes is non-deterministic when nodes have multiple active slots. In this paper, we propose an algorithm that allows us to take advantage of duty-cycled wake-up slots, in which the link-delay value is calculated based on a differential of

TABLE I: Notations and abbreviations

Notation	Explanation
$G = (V, E)$	Communication graph G includes set of vertices V and set of edges E
$T = (V_T, E_T)$	Aggregation tree T consists of set of vertices V and set of edges E_T
$\mathcal{S}$	Number of transmissions of all nodes in $V \setminus \{s\}$
$\mathcal{D}$	Total aggregation delay (in time slots)
$\mathcal{N}$	Number of nodes in a network
$\mathcal{L}$	Number of slots in a working period
R	Communication range between two sensor nodes
$\mathcal{F}$	Set of limited channels
w	Current working period
$N(u)$	Set of neighbor nodes of node u in G
$p(u)$	Parent of node u
$C(u)$	Set of children of node u in G
$A(u)$	Set of active slots of node u
$f(u)$	Channel of node u
$t(u)$	Transmitting time slot of node u
$s(u)$	$[f(u), t(u)]$: Transmitting schedule of node u
(u, v)	A link connects between transmitter u and receiver v

active slots between nodes.

Using MIS and link-delay approaches, Fig. 2 shows a partial tree construction. Sink node S considers adopting children from its neighbors. Each node is active at two slots in a working period length $\mathcal{L} = 10$. The MIS-based approach (Fig. 2a) assigns S as a parent of all nodes A, B , and C . Since node S is active at two slots, and it can only receive data from one node per slot, it will take two working periods for S to collect data from the other nodes. Fig. 2b shows that node S has two children, and node A has one child. Intuitively, node S collects data from its children in one working period when child node of B transmits its data in a time slot ahead.

In subsection IV-B, we show how to build the aggregation tree, and in subsection IV-C, we show how to create the scheduling scheme.

Sender u finishes its aggregation at slot τ_u and wants to transmit data at slot τ_v which is an active slot of receiver v , the minimum time it should wait, denoted as $w(\tau_u, \tau_v, \mathcal{L})$ defined as

Fig. 2: Comparing MIS-based and Link-delay-based approaches in terms of the tree construction method.

follows.

$$w(\tau_u, \tau_v, \mathcal{L}) = \begin{cases} \tau_v - \tau_u, & \text{if } \tau_v > \tau_u \\ \tau_v - \tau_u + \mathcal{L}, & \text{otherwise.} \end{cases} \quad (1)$$

The $w(\tau_u, \tau_v, \mathcal{L})$ means that if the active slot of u is ahead of the receiver's receive slot, u can transmit data to v in the same working period, otherwise it needs to wait until the next working period.

Such waiting time on a link is an important factor for an aggregation tree construction, as shown in [38]. However, determining the waiting time for a link is not deterministic in a duty-cycled network where a node is active at several slots per working period. We define the link-delay on a link (u, v) with respect to the active slot $\tau_v \in A(v)$ of node v as follows.

Definition 1. Consider slot $\tau_v \in A(v)$ of a working period, the link-delay between sender u and receiver v is defined as follows:

$$d(u, \tau_v, \mathcal{L}) = \min_{\tau_u \in A(u)} \{w(\tau_u, \tau_v, \mathcal{L})\}, \quad (2)$$

Where $A(u)$ and $A(v)$ are active slots of nodes u and v , respectively. $\mathcal{L}$ denotes the length of a working period.

Herein, $d(u, \tau_v, \mathcal{L})$ is the minimum of all possible waiting time $w(\tau_u, \tau_v, \mathcal{L})$ between u and v if v wants to receive data at slot τ_v . Assuming that the sender u has data ready all of the time,

it must wait until the receiver v is active to send its data. If the active slot of receiver τ is larger than the active slot of the sender τ_u , i.e., $\tau_v > \tau_u$, the sender transmits its data to the receiver in the same working period. The sender must wait until the next working period to send data to the receiver at the receiver's active slot if the receiver's active slot is smaller than the sender's.

Since the sensor nodes are active at multiple slots in a working period, $d(u, \tau_v, \mathcal{L})$, which is the minimum link delay for sender u to wait until receiver v is active to send its data, needs to be determined between two nodes. At v 's active slot τ_v , all the link delay values with sender u 's active slots $A(u)$ are calculated, then the minimum value selected among those values is the final delay value of the link. Using the link-delay as a strategy, we can construct pipeline transmissions where we maximize the number of nodes transmitting data in a working period. To establish a parent-child pair between two neighbor nodes during the tree construction phase, the algorithm uses link-delay.

B. Link-delay tree construction

Algorithm 1: Link-delay tree construction

Input : $G(V, E)$, $\mathcal{L}$, sink node $s \in V$

Output: An aggregation tree $T = (V_T, E_T)$

```

1  $V_T \leftarrow \{s\}$ 
2  $E_T \leftarrow \emptyset$ 
3 while  $V \neq V_T$  do
4   for each  $\tau \in [0, \mathcal{L} - 1]$  in decreasing order do
5     for each  $v \in V_T$  and  $\exists u \in N(v) \mid p(u) = \text{None}$  do
6       if  $\tau \in A(v)$  then
7          $x \leftarrow \underset{u \in N(v) \setminus V_T}{\operatorname{argmin}} \{d(u, \tau, \mathcal{L})\}$ 
8          $p(x) \leftarrow v$ 
9          $E_T \leftarrow E_T \cup \{(x, v)\}$ 
10         $V_T \leftarrow V_T \cup \{x\}$ 

```

Algorithm 1 presents an aggregation tree construction process. Take the communication graph $G(V, E)$ including sink s and working period length $\mathcal{L}$ as inputs. Output is the aggregation tree T with V_T sensor nodes and $|E_T|$ links. V_T is the set of nodes that belong to the tree and

Fig. 3: Link-delay tree construction process.

initially, V_T contains only the sink s (line 1). The tree will continue to grow until it contains all of the graph's sensor nodes (line 3). Starting from the highest slot in $\mathcal{L}$, for every $v \in V_T$, if v is active at the current slot τ and there is at least one neighbor $u \in N(v)$ (lines 4-6), the algorithm assigns a child whose link-delay value d between active slot τ of node v and node u is smallest. Then the algorithm selects the node that forms the smallest link-delay value as the child of v . The parent of that child node is v and that child node x is added to the set V_T (lines 7-10). So forth, the algorithm ends when V_T includes all sensor nodes.

We take Fig. 3 as an example to illustrate the link-delay tree construction process. A sample network topology (Fig. 3a) consists of 12 sensor nodes in which sink S collects data from other nodes in the network. In a working period of length $\mathcal{L} = 10$, each node is assumed to be active at two slots i . $|\mathcal{F}| = 1$ is the maximum number of channels that can be used. The algorithm first adds node S to the aggregation tree. It starts with one of the slots that S is active, e.g., slot 7. At active slot 7 of S , the link-delay values are calculated one by one from the neighbors to S , nodes A , B , and C . As C is active at slots 2 and 6, the link-delay values between C and S at the two active slots are $7 - 2 = 5$ and $7 - 6 = 1$, respectively. So that $d(C, S_7, 10) = 1$ is the link-delay value of the link CS . Similarly, the link-delay values of links AS and BS are 2 and 6, respectively. Node C is chosen as a child of S and added to the aggregation tree because it has the shortest link-delay value among neighbors of S (Fig. 3b).

In decreasing order, the algorithm moves on to slots 5 and 6. At each slot, nodes G and F

are sequentially added to the tree. The aggregation tree has four nodes at slot 4, two of them are active at this slot which is S and F . The algorithm checks one by one node, as S adopts A to be its child, F adopts K to be its child since these links have smallest link-delay values compared to other links as shown in Fig. 3c. The algorithm works in the same process until all nodes in the network are added to the tree (Fig. 3d).

C. Aggregation scheduling

After constructing the aggregation tree, the candidate nodes scheduling scheme (Algorithm 3) takes the tree T consists of V sensor nodes and E_T links connecting children and parent nodes, a working period length $\mathcal{L}$ and set of available channels $\mathcal{F}$ can be used as inputs. All nodes except the sink are scheduled once the algorithm has finished scheduling. The algorithm starts with the first working period w set as 1, all leaf nodes in V_{leaf} which have no children are scheduled first. To find viable scheduling candidates, the algorithm examines all of the tree's leaf nodes. While the set V_{leaf} is not empty, a set of candidate nodes V_c is initialized as an empty set. A set P adopts all parent nodes of nodes in V_{leaf} (line 4). With each node $v \in P$, if a child $u \in C(v)$ such that $u \in V_{leaf}$, candidate set V_c embraces u then V_{leaf} removes u from it (lines 6-9). When two children nodes with the same parent are scheduled at the same time, this strategy helps to avoid a primary collision. After determining a set of candidate V_c is possibly scheduled in a working period, Candidate_nodes_schedule (Algorithm 3) is applied to schedule nodes in V_c , then it updates the aggregation tree by removing scheduled nodes.

After Algorithm 3, the current working period w is updated (line 11). Because Algorithm 3 leaves some available time slots and channels where nodes can transmit data, Algorithm 4 is used to enhance nodes scheduling in the current working period w . This process runs until no more feasible nodes can be found to be scheduled in the current working period (lines 13-15). Then the algorithm moves to the next working period (line 16) and updates the set of leaf nodes V_{leaf} (line 17).

Algorithm 3 does scheduling to nodes in set V_c . The algorithm takes the communication graph $G(V, E)$, the aggregation tree $T(V_T, E_T)$, candidate set V_c , the current working period w , the number of slots in a working period $\mathcal{L}$, and a set of channels $\mathcal{F}$ as input. The algorithm returns an updated tree in which scheduled nodes are removed from the tree. A selected node-set for each slot $\tau \in [0, \mathcal{L} - 1]$ as SN^τ is initiated empty (line 1). Interference set I_u^τ is created for $u \in C$ at each $\tau \in [0, \mathcal{L} - 1]$ to include nodes that occur secondary conflict with node u at slot

Algorithm 2: Link-delay-aware schedule**Input :** $T(V_T, E_T), \mathcal{L}, \mathcal{F}$ **Output:** Scheduled nodes

```

1   $w \leftarrow 1$ 
2   $V_{leaf} \leftarrow \{u \in V_T \mid C(u) = \emptyset\}$ 
3  while  $V_{leaf} \neq \emptyset$  do
4       $P \leftarrow \{p(u) \mid u \in V_{leaf}\}$ 
5       $V_c \leftarrow \emptyset$ 
        // Select set of candidate nodes  $V_c$  from  $V_{leaf}$ 
6      for each  $v \in P$  do
7           $u \leftarrow$  A random node in  $C(v) \cap V_{leaf}$ 
8           $V_c \leftarrow V_c \cup \{u\}$ 
9           $V_{leaf} \leftarrow V_{leaf} \setminus \{u\}$ 
10      $T(V_T, E_T) \leftarrow \text{Candidate\_nodes\_schedule}(G, T, V_c, w, \mathcal{L}, \mathcal{F})$ 
11      $w \leftarrow$  Current working period after Algorithm 3
        // Schedule more nodes on unused channels and time slots
        // at the current working period
12      $V_{temp} \leftarrow \emptyset$ 
13     while  $|V_{temp}| \neq |V_T|$  do
14          $V_{temp} \leftarrow V_T$ 
15          $T(V_T, E_T) \leftarrow \text{Reinforcement\_schedule}(G, T, w, \mathcal{L}, \mathcal{F})$ 
        // Update  $t_s$  to start a new working period
16      $w \leftarrow w + 1$ 
17     Update  $V_{leaf} \leftarrow \{u \in V_T \mid C(u) = \emptyset\}$ 

```

τ (lines 2-3).

The algorithm finds the node u in candidate set V_c to add into selected nodes set SN_u^τ at slot $\tau \in [0, \mathcal{L} - 1]$ if $|I_u^\tau|$ is smallest. Because we want to minimize secondary collisions in a particular slot, we choose a node that has smallest number of occurred secondary collision

Algorithm 3: Candidate nodes schedule

Input : $G(V, E), T(V_T, E_T), V_c, w, \mathcal{L}, \mathcal{F}$
Output: Selected nodes to be scheduled at each time slot

```

1  $SN^\tau \leftarrow \emptyset, \forall \tau \in [0, \mathcal{L} - 1]$ 
2 for each  $u \in V_c$  do
3    $I_u^\tau \leftarrow \{\text{Set of nodes that occurs secondary conflict with } u \text{ at slot } \tau\}, \tau \in [0, \mathcal{L} - 1]$ 
4 while  $V_c \neq \emptyset$  do
5    $min \leftarrow \underset{u \in V_c}{\operatorname{argmin}} |I_u^\tau|, \tau \in [0, \mathcal{L} - 1]$ 
6    $slot \leftarrow \tau$ 
7    $node \leftarrow u$ 
8   for each  $\tau \in [0, \mathcal{L} - 1]$  do
9     if  $\tau \neq slot$  then
10       for each  $u \in V_c$  do
11          $I_u^\tau \leftarrow I_u^\tau \setminus \{node\}$ 
12    $V_c \leftarrow V_c \setminus \{node\}$ 
13    $SN^{slot} \leftarrow SN^{slot} \cup \{node\}$ 
14 for each  $\tau \in SN^\tau$  do
15   Sort  $SN^\tau$  based on  $|I_u^\tau|$  (Non-increasing order)
16   for each  $u \in SN^\tau$  do
17      $c(u) \leftarrow \text{Color of node } u \text{ with respect to } SN^\tau$ 
18      $f(u) \leftarrow (c(u) \bmod |\mathcal{F}|) + 1$ 
19   for each  $f \in \mathcal{F}$  do
20      $Q \leftarrow \{u \in SN^\tau | f(u) = f\}$ 
21     for each  $u \in Q$  do
22        $c(u) \leftarrow \text{Color of node } u \text{ with respect to } Q$ 
23        $t(u) \leftarrow (w - 1 + c(u)) \cdot \mathcal{L} + \tau$ 
24        $V_T \leftarrow V_T \setminus \{u\}$ 
25        $E_T \leftarrow E_T \setminus \{(u, p(u))\}$ 

```

nodes. As a result, the number of nodes that can transmit data simultaneously without colliding is maximized. The node added to SN is removed from V_C and $I^{slot}, slot \in \mathcal{L}$ and $slot \neq \tau$. This process operates until all the nodes in set V_C are removed (lines 4-13). Following that, selected nodes in SN are assigned channels and time slots one by one. For each τ , nodes in SN^τ are sorted in nonincreasing order then channels are assigned using a coloring method. At slot τ , if a node occurs a secondary collision with others (adjacent nodes) in set SN^τ , they are assigned different colors with its conflict nodes as $c(u)$. The channel assigned to that node $f(u)$ is calculated as $(c(u) \bmod |\mathcal{F}|) + 1$ (lines 14-19). At each iteration, the algorithm checks each channel f at slot τ , from set $\mathcal{F}$, nodes having the same channel are added into a set Q at each iteration (lines 20-21). After that, another coloring method is applied to nodes in set Q , nodes in this set are marked in different colors if those nodes conflict with each other, they are assigned in different time slots, i.e, the time slot of node u is calculated as $(w - 1 + c(u)) \cdot \mathcal{L} + \tau$, where $c(u)$ is the coloring method, and w is the current working period (lines 22-24). The algorithm then removes the scheduled nodes from the tree T and the link connecting from the scheduled nodes to their parents (lines 25-26).

Fig. 4 illustrates the process of scheduling nodes after tree construction phase. As shown in Fig. 4a, the tree has three leaf nodes, from which we can find three links, namely DH , IF and JK . The algorithm, then, finds the interference set of each link at each receiver node's active slot. Herein, at the active slot $\tau = 0$, nodes D and K are active. A secondary collision occurs at D if H and J transmit data at the same time. Therefore, interference set I_H^0 contains $\{J\}$. There is no collision happened when J transmits data so that interference set $I_J^0 = \emptyset$. Similarly, the interference sets are shown in Fig. 4b for other active slots. Calculating the number of interference nodes in the corresponding set I yields the selected nodes set SN regarding to receiver nodes' active slots. Herein, starting with node H in the candidate set V_C , as $|I_H^0| = 1$ and $|I_H^6| = 0$, the algorithm adopts selected node H into set SN^6 . Interference set I_J^0 is updated, but it remains as the empty set because link HD does not interfere with link JK . Set V_C removes H from it.

A similar process to select nodes at specific slots to schedule in candidate set V_C , nodes J and I are chosen at slots 0 and 1, respectively, then added into SN^0 and SN^1 . If a node has the same number of nodes in the interference sets at slots, the algorithm selects the node at an earlier active slot to add into the selected node set SN . When more available time slots are remain after nodes have been scheduled, then more nodes can be scheduled. Selected nodes in

Fig. 4: Leaf nodes scheduling process in the first working period.

SN are scheduled from the earliest slot to a later one. Since SN^0 contains only node J , the algorithm applies coloring method c to nodes in SN^0 , i.e., $c(J) = 0$. The channel of node J is $f(J) = (0 \bmod 2) + 1 = 1$. Similar to nodes I and H , their channels are both 1. With channel 1, at slot 0, node J is added to the set Q since $J \in SN^0$. Another coloring method is applied to node $J \in Q$ with value $c(J) = 0$, the time slot assigned to J is $t(J) = (1 - 1 + 0) \cdot 10 + 0 = 0$. Then, node J is removed from the tree. Similarly, the time slots of I and H are $t(I) = 1$ and $t(H) = 6$, respectively. Those nodes I and H are removed from the tree after they are scheduled.

Although Algorithm 3 completes nodes scheduling in candidates set V_c at the current working period w , it leaves some available channels and time slots where nodes can transmit data. Algorithm 4 explores again all slots τ in the current working period w to allocate more nodes in the tree $T(V_T, E_T)$ on those available channels and time slots. At each slot, Algorithm 4 checks if there is any available channel, the algorithm exploits the feasible node(s) among the candidates in reinforcement nodes set to schedule them. Algorithm 4 takes graph $G(V, E)$, aggregation tree $T(V_T, E_T)$, current working period w , working period length $\mathcal{L}$ and set of channels $\mathcal{F}$ as inputs. After removing scheduled nodes from unused channels and time slots during the current working period, the algorithm returns a tree. At each slot $\tau \in w$, set $\bar{F}$ is initially created containing all unused channels at slot τ . If there is at least one unused channel available, the algorithm creates

a reinforcement nodes set V_R can be scheduled at slot τ of working period w (lines 1-6). The nodes added to V_R are satisfied the three conditions:

- 1) These are leaf nodes.
- 2) The receiver of a node in this set is active at slot τ .
- 3) The receiver of a node in this set is not yet be scheduled to receive data from any other node in this time slot τ .

With each node u in set V_R , an interference set I_u is obtained, consisting of nodes occur secondary collision with u at time slot t (lines 7-12). Then the algorithm selects the nodes in a set I that have the fewest interference nodes to schedule as many nodes as possible. With this selection, we can schedule the most concurrent nodes possible. The selected node is, then, added to selected nodes set SN and removed from V_R . The nodes are added to SN as long as $|I|$ is not larger than $|\bar{F}|$ (lines 13-22). After that, we apply a coloring method c to nodes in selected nodes SN , the channel of node u is $f(u)$ calculated as $(c(u) \bmod |\bar{F}|) + 1$, and the transmitting time slot of node u is $t(u)$ calculated as $w \cdot \mathcal{L} + \tau$ (lines 24-27). Then node u and the link from u to its parent $(u, p(u))$ are removed from the tree $T(V_T, E_T)$ (lines 28-29).

After applying algorithm 3 in the first working period, Fig. 5 shows the reinforcement schedule on an updated tree. After they are scheduled, nodes H, I and J are removed from the tree. The current working period is $w = 1$. Algorithm 4 starts checking unused channels at each time slot in this first working period. At slots $\tau = 0$ and $\tau = 1$, there are no unused channels since there are some scheduled nodes at these time slots. At slots $\tau = 2$ and $\tau = 3$, the channel is unused, but there are no nodes that satisfy the three conditions to add to the reinforcement nodes set V_R . The algorithm continues with slot $\tau = 4$ at the current working period $w = 1$, there is no channel being used and link KF is satisfied the three conditions which K is a leaf node, the receiver F is active at this slot and it does not receive any data from other nodes in this time slot as shown in the Fig. 5a. Node K is added to the reinforcement nodes set V_R , the interference set of node K at slot 4 is $I_K^4 = \emptyset$ since only K is in V_R . Then node K is added to selected nodes set SN^4 at slot 4. A coloring method c is applied to set SN^4 in which $c(K) = 0$, so that the channel of node K is $f(K) = (0 \bmod 1) + 1 = 1$ and the transmitting time slot of K is $t(K) = (1 - 1) \cdot 10 + 4 = 4$. Then, node K and link KF are removed from the tree. In the current working period, the same procedure is followed for later time slots. The sample network's final schedule is shown in Fig. 5b; the scheduling takes two working periods to complete.

Algorithm 4: Reinforcement schedule**Input** : $G(V, T), T(V_T, E_T), w, \mathcal{L}, \mathcal{F}$ **Output:** Nodes scheduled on unused channels and time slots in a working period

```

1 for  $\tau \in [0, \mathcal{L} - 1]$  do
2    $X \leftarrow \{u \in V \mid t(u) = (w - 1) \cdot \mathcal{L} + \tau\}$ 
3    $\bar{F} \leftarrow \mathcal{F} \{f(u) \mid u \in X\}$ 
4   if  $|\bar{F}| \neq 0$  then
5      $\bar{P} \leftarrow \{p(u) \mid u \in X\}$ 
6      $V_R \leftarrow \{u \in V_T \mid C(u) = \emptyset \text{ and } \tau \in A(p(u)) \text{ and } p(u) \notin \bar{P}\}$ 
7     for each  $u \in V_R$  do
8        $I_u \leftarrow \emptyset$ 
9       for each  $x \in V_R$  do
10        if  $\tau \in A(p(x))$  then
11          if  $p(x) \in N(u)$  or  $x \in N(p(u))$  then
12             $I_u \leftarrow I_u \cup \{x\}$ 
13     while  $V_R \neq \emptyset$  do
14        $min \leftarrow \mathcal{L}$ 
15        $SN \leftarrow \emptyset$ 
16       for each  $u \in V_R$  do
17         if  $min \geq |I_u|$  then
18            $min \leftarrow |I_u|$ 
19            $node \leftarrow u$ 
20        $V_R \leftarrow V_R \setminus \{node\}$ 
21       if  $|SN| < |\bar{F}|$  then
22          $SN \leftarrow SN \cup \{node\}$ 
23     Sort  $SN$  based on  $|I_u|$  in a non-increasing order
24     for each  $u \in SN$  do
25        $c(u) \leftarrow$  Color of node  $u$  with respect to  $SN$ 
26        $f(u) \leftarrow (c(u) \bmod |\bar{F}|) + 1$ 
27        $t(u) \leftarrow (w - 1) \cdot \mathcal{L} + \tau$ 
28        $V_T \leftarrow V_T \setminus \{u\}$ 
29        $E_T \leftarrow E_T \setminus \{(u, p(u))\}$ 

```

Fig. 5: Reinforcement schedule on updated tree and final schedule of the sample tree.

D. Analysis

Theorem 1. *LIRE allows the sink to aggregate data from all other sensor nodes.*

Proof: LIRE constructs an aggregation tree based on active slots of sensor nodes. The tree construction process is performed in a top-down manner from the sink, and it completes when all sensor nodes, except the sink, adopt their parents. Then, the algorithm iteratively schedules nodes in a bottom-up manner from leaf nodes. In each iteration, scheduled nodes are removed from the tree. The next step is to determine and schedule new leaf nodes. Repeat the process until the sink is the only node left. As a result, data from all sensor nodes in the network is aggregated at the sink. ■

Theorem 2. *The nodes scheduled by LIRE are collision-free.*

Proof: LIRE schedules all nodes in the aggregation tree by Algorithm 3 and Algorithm 4. In each iteration, candidate nodes V_c , which are leaf nodes in the tree, are selected by Algorithm 2 such that none of them has the same parent. Primary collisions are eliminated using this method of selection. The candidate nodes are scheduled using Algorithm 3. Algorithm 3 enhances the nodes to select nodes set SN to increase the number of scheduled nodes in one working period

based on the constructed interference sets I and active slots of nodes in the candidate set. Then, two coloring methods c are applied to nodes in the selected nodes set SN to ensure that two interfered links, which start from two nodes in the selected nodes set to their parents, are assigned different colors. The colors are then used to assign different channels and time slots to these nodes. Scheduled nodes are removed from the tree at the end of each iteration.

Algorithm 4 schedules nodes in the tree on available channels at each specific time slot. First, the algorithm finds reinforcement leaf nodes set V_R to be scheduled. Then, for each node in the reinforcement nodes set V_R , it constructs an interference set I . A selected node-set SN is obtained based on the interference set I . Finally, a coloring method c is applied to the selected nodes set SN to assign two nodes that form two interfered links in different colors. Different channels are assigned to these nodes. As a result, the LIRE-scheduled nodes are collision-free. ■

Theorem 3. *The time complexity of the LIRE scheme is at most $O(\mathcal{L}n^2 + \mathcal{L}|\mathcal{F}|)$.*

Proof: LIRE first constructs the aggregation tree, with each slot of the working period length $\mathcal{L}$, Algorithm 1 grows the tree by adding neighbors that form the smallest link-delay values with nodes in the tree, the whole this process takes $O(\mathcal{L}n^2)$ time. Then Algorithm 2 consequentially finds leaf nodes V_{leaf} in the tree, their parent nodes, and candidate nodes V_c from parent nodes, the process takes $O(n^2)$ time. Next, the algorithm schedules candidate nodes in Algorithm 3. This step takes at most $O(\mathcal{L}n^2 + \mathcal{L}|\mathcal{F}|)$. Finally, Algorithm 4 handles the scheduling of reinforcement nodes. This step takes at most $O(\mathcal{L}n^2)$. We combine all steps of LIRE and express the total time complexity $O(\mathcal{L}n^2 + \mathcal{L}|\mathcal{F}|)$. ■

V. PERFORMANCE EVALUATION

We use Python to evaluate the performance of our proposed scheme, LIRE (Link-delay-aware REinforcement scheduling). We compare our scheme with the existing algorithm named NDAS (Novel Data Aggregation Scheduling) in [32] in various network settings. By varying the number of channels, the number of nodes, the lengths of working periods, and the number of active slots, we compare LIRE's performance to that of the reference scheme in terms of data aggregation time.

TABLE II: Simulation Settings

Parameter	Value
Number of nodes	200, 500, and 1000 nodes
Area size	$100 \times 100m^2$
Communication range	$20m$
Number of channels	$1 \sim 7$
Length of working periods	$10 \sim 70$
Number of active slots	$2 \sim 7$

Fig. 6 shows the distribution of number of children using MIS-based and Link delay-based tree construction methods. In this sample topology, we deploy 500 nodes in an area of $100 \times 100m^2$. Each circle's central point represents a nonleaf node, and the size of the circle relatively presents the number of children that the node has. The node (the circle's central point) has a high degree if the circle is large, and vice versa. Many nodes have high degrees in the MIS-based tree, whereas nodes in the link delay-based tree have much lower degrees and a higher density. As a result, at a certain point in time, the number of concurrent data transmissions increases. Each child is scheduled to send data to the node in different working periods for a node with many children. The primary collision occurs when more than one child send data to the same parent at the same time. As a result, when compared to the link delay-based tree construction method, the MIS-based tree construction method takes longer to schedule.

The simulation settings are presented in Table II. We deploy 200, 500, and 1000 nodes randomly in a network area fixed as $100 \times 100m^2$. Each node has the same random number of active slots in each network topology. We conduct 100 random networks for each setting and average values. The communication range R is fixed as $20m$. The number of available channels varies from 1 to 7. The working period length varies from 10 to 70. Sensor nodes can be active from 2 to 7 slots.

On a sample network of 500 sensor nodes, we investigate the distribution of transmitting time slots for LIRE and NDAS schemes. The number of available channels $|\mathcal{F}|$ is 2. Each node is active at 2 slots in a working period of length $\mathcal{L} = 10$. Fig. 7 shows the number of transmissions of LIRE and NDAS schemes over time (in working periods). LIRE requires 45

Fig. 6: Number of children distribution of tree construction based on MIS and link-delay approaches with $\mathcal{L} = 10$, $N = 500$, $|A(u)| = 2 : u \in V$.

working periods to aggregate data from all nodes whereas NDAS takes 73 working periods. The transmissions in the LIRE scheme appears to be concentrated on some first working periods, up to 9 transmissions in a working period. In later working periods, from 35% to 80% the total working period, every slot almost has 3 transmissions on average. With the NDAS scheme, the transmissions concentrate in the first 40% of total working periods, i.e., up to 6 transmissions. NDAS scheme leaves many unused slots in each working period since it uses only 1 or 2 slots to schedule nodes. Because a node has a high density of children and it is active at 2 slots, it can receive data from two children maximum in each working period. So that other slots in that working period are unused. To receive more data from its other children, the node must wait until the next working period.

A. Impact of duty cycle

By increasing the number of active slots from 2 to 7, we can compare the aggregation delay between NDAS and LIRE schemes by changing the number of active slots from 2 to 7 as shown in Fig. 8. The working period length is fixed as 10. The experiments are conducted with the networks of 200, 500, and 1000 nodes. LIRE achieves lower aggregation delay comparing to NDAS in both single-channel and two-channel scenarios. Moreover, when the network is

Fig. 7: Number of transmissions contour diagrams of (a) LIRE and (b) NDAS schemes, respectively, with $\mathcal{L} = 10$, $N = 500$, $m = 2$, $|A(u)| = 2 : u \in V$.

denser, the gap of aggregation delay between LIRE and NDAS gets bigger, i.e, up to 62.09% when $N = 1000$ and $m = 1$.

The main reason is that LIRE constructs the aggregation tree based on the link delay between neighbor nodes that makes the number of children of nonleaf nodes not much different. NDAS, on the other hand, is based on CDS, which means that some nonleaf nodes have many children while others have only a few. LIRE has more concurrent transmissions at the same time slot when doing scheduling. In NDAS, nodes with many children have to wait for later working periods to collect data from their children otherwise the primary collision will occur. In addition, in each scheduling round, LIRE scans all leaf nodes in the network to obtain a candidate set. NDAS, on the other hand, considers all of the network's leaf nodes first, then nonleaf nodes layer by layer in each scheduling round. LIRE's scheduling strategy increases the number of nodes that can be scheduled in a working period, resulting in a faster scheduling completion time.

The data aggregation delay of LIRE increases as the number of active slots in a working period increases. While NDAS has a minor impact, when using two channels, it achieves a lower aggregation delay in all cases of network density. The reason for this is that when LIRE builds the aggregation tree, it creates pipeline transmissions from the top down. LIRE then

schedules nodes from the bottom up. When the number of active slots is small, the algorithm schedules nodes from the leaf nodes, it takes advantage of pipeline transmissions from the tree construction phase by selecting the same nodes in the pipelines, then the number of transmissions in a working period increases. However, when the nodes are active at more slots, nodes are selected to schedule becoming randomly, not related to the pipelines. So that the number of transmissions in a working period decreases, which means that the aggregation delay is higher when the number of active slots increases.

Fig. 8: Impact of number of active slots in different network sizes on data aggregation delay.

B. Impact of number of nodes and number of channels

Fig. 9: Impact of number of nodes and number of channels on data aggregation delay.

We conduct experiments where nodes in a network are active 2 slots in a working period length $\mathcal{L} = 10$. Fig. 9a presents the impact of the number of nodes on the aggregation delay. We change the network size from 50 to 1000 nodes, and the number of channels $m = 1$ (NDAS-1 and LIRE-1) and $m = 2$ (NDAS-2 and LIRE-2). When the network gets denser, both NDAS and LIRE need more time to schedule nodes so that the aggregation delay of both schemes gets higher. However, LIRE achieves a lower aggregation delay than NDAS and its slopes are gentler than NDAS' when the number of nodes increases from 50 to 1000. The reason

for this is that LIRE scheduling algorithm takes advantage of pipeline transmissions created during the tree construction phase, which affects the nodes with fewer active slots. Moreover, when the scheduling algorithm schedules more nodes on unused channels and time slots of the whole working period, that helps reduce the aggregation delay further, i.e., LIRE achieves better aggregation delay than NDAS up to 59.23% (1 channel) and 55% (2 channels).

The impact of the number of channels on the aggregation delay is shown in Fig. 9b. The number of channels varies from 2 to 7. We compare NDAS and LIRE with the networks of 200, and 500 nodes. The results show that increasing the number of channels has little effect on the aggregation delay of NDAS, while the aggregation delay of LIRE reduces. Comparing the aggregation delay with NDAS, LIRE achieves better than NDAS up to 79.05% and 81.40% in the network of 200 and 500, respectively. Since NDAS constructs the aggregation tree based on MIS which makes the dominators concentrate lots of dominatees. When a dominator has a large number of dominatees, it can only receive data from them in different slots during a working period or at different working periods. Therefore, this solution does not leverage the increasing number of channels. LIRE, on the other hand, schedules nodes from a tree with a relatively even distribution of children. Therefore, different channels are useful to allocate to these nodes, which occur the secondary collisions. When the network is denser, say 500 nodes, it is even more effective .

C. Impact of working period length

We perform the simulations with varying the network size 200, 500, and 1000 nodes. With each network size, we compare the aggregation delay of NDAS and LIRE using 1 channel (NDAS-1 and LIRE-1) and 2 channels (NDAS-2 and LIRE-2) by changing the working period length from 10 to 70. The number of active slots of each node is fixed as 2. Fig. 10 shows how the length of the working period affects the aggregation delay of LIRE and NDAS. LIRE has a lower aggregation delay in all cases when compared to NDAS. The aggregation delay increases as the working period length increases. However, the aggregation delay of LIRE increases slightly, while the aggregation delay of NDAS increases dramatically. The results show that LIRE using 2 channels performs better aggregation delay than NDAS up to 82.65%, 84.88%, and 85.75% when $N = 200, 500$, and 1000 nodes, respectively. Because the working period length is large, nodes are active at a few slots randomly in the working period. In a single working period, pipeline transmissions in LIRE become longer. It means that in LIRE more nodes are scheduled

in one working period while NDAS waits for the next working periods to schedule nodes having the same parents.

Fig. 10: Impact of working period length in different network sizes on data aggregation delay.

D. Impact of starting active slot in aggregation tree construction

The aggregation delay is also influenced by the starting active slot chosen when building the aggregation tree. We conduct the simulations to compare the aggregation delay when starting from the highest active slot and the lowest active slot values with 1 channel (From high active slot-1, From low active slot-1) and 2 channels (From high active slot-2, From low active slot-2) in varying the network sizes 200, 500, and 1000 as shown in Fig. 11. In this experiment, the number of active slots is changed from 2 to 7 in a working period length $\mathcal{L} = 10$, nodes in the network are active at 2 slots. The results show that starting from the highest active slot of the sink achieves lower aggregation delay than from the lowest active slot, i.e., at most 6.13% (for 1 channel), 10.5% (for 2 channels), and 11.88% (for 2 channels).

Fig. 11: Impact of the number of active slots of LIRE by active slot selecting strategy to construct the tree in different network sizes.

VI. CONCLUSION AND FUTURE WORK

In this paper, we study the DDA problem in multichannel duty-cycled IoT sensor networks. LIRE scheduling is a novel aggregation approach that consists of two phases. First, we present a new top-down approach to construct the aggregation tree which leverages the gap between nodes' active slots. Second, scheduling algorithms schedule nodes in the tree in a bottom-up manner from leaf nodes. The reinforcement algorithm improves node scheduling by utilizing unused channels and time slots left by the main algorithm after each scheduling round. With extensive simulation scenarios, the results show that LIRE performs vastly superior comparing to the only best algorithm that pays attention on this problem in multichannel duty-cycled IoT networks, i.e., up to 81%. We intend to use a machine learning concept to solve this problem in the future, as well as consider energy efficiency together with the delay efficiency of the IoT networks in our research.

REFERENCES

- [1] Soroush Abbasian Dehkordi et al. "A survey on data aggregation techniques in IoT sensor networks". In: *Wireless Networks* 26.2 (2020), pp. 1243–1263.
- [2] Sabrina Boubiche et al. "Big data challenges and data aggregation strategies in wireless sensor networks". In: *IEEE access* 6 (2018), pp. 20558–20571.
- [3] Mohammed H Alsharif, Sunghwan Kim, and Nuri Kuruoğlu. "Energy harvesting techniques for wireless sensor networks/radio-frequency identification: a review". In: *Symmetry* 11.7 (2019), p. 865.
- [4] Long Cheng et al. "Collision-Free Dynamic Convergecast in Low-Duty-Cycle Wireless Sensor Networks". In: *IEEE Transactions on Wireless Communications* (2021).
- [5] Lijie Xu et al. "Optimizing energy efficiency for minimum latency broadcast in low-duty-cycle sensor networks". In: *ACM Transactions on Sensor Networks (TOSN)* 11.4 (2015), pp. 1–31.
- [6] Ngoc-Tu Nguyen et al. "Challenges, designs, and performances of a distributed algorithm for minimum-latency of data-aggregation in multi-channel WSNs". In: *IEEE Transactions on Network and Service Management* 16.1 (2018), pp. 192–205.
- [7] Miloud Bagaa et al. "Multi-path multi-channel data aggregation scheduling in wireless sensor networks". In: *2013 IFIP Wireless Days (WD)*. IEEE, 2013, pp. 1–6.
- [8] Ozlem Durmaz Incel et al. "Fast data collection in tree-based wireless sensor networks". In: *IEEE Transactions on Mobile computing* 11.1 (2011), pp. 86–99.
- [9] Baobing Wang and Xiaohua Jia. "Reducing data aggregation latency by using partially overlapped channels in sensor networks". In: *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009, pp. 1–6.

- [10] Quan Chen et al. “Distributed low-latency data aggregation for duty-cycle wireless sensor networks”. In: *IEEE/ACM Transactions on Networking* 26.5 (2018), pp. 2347–2360.
- [11] Duc Tai Le, Taewoo Lee, and Hyunseung Choo. “Delay-aware tree construction and scheduling for data aggregation in duty-cycled wireless sensor networks”. In: *EURASIP Journal on Wireless Communications and Networking* 2018.1 (2018), pp. 1–15.
- [12] Xianlong Jiao et al. “Data Aggregation Scheduling in Uncoordinated Duty-Cycled Wireless Sensor Networks under Protocol Interference Model.” In: *Ad Hoc Sens. Wirel. Networks* 15.2-4 (2012), pp. 315–338.
- [13] Bo Yu and Jian-Zhong Li. “Minimum-time aggregation scheduling in duty-cycled wireless sensor networks”. In: *Journal of Computer Science and Technology* 26.6 (2011), pp. 962–970.
- [14] Nguyen Phan Khanh Ha, Vyacheslav Zalyubovskiy, and Hyunseung Choo. “Delay-efficient data aggregation scheduling in duty-cycled wireless sensor networks”. In: *Proceedings of the 2012 ACM research in applied computation symposium*. 2012, pp. 203–208.
- [15] Emma Fitzgerald, Michał Pióro, and Artur Tomaszewski. “Energy-Optimal Data Aggregation and Dissemination for the Internet of Things”. In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 955–969.
- [16] Hassan Elahi et al. “Energy Harvesting towards Self-Powered IoT Devices”. In: *Energies* 13.21 (2020).
- [17] Yuqun Zhang et al. “Energy-Efficient Duty Cycle Assignment for Receiver-Based Convergecast in Wireless Sensor Networks”. In: *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. 2010, pp. 1–5.
- [18] Xujin Chen, Xiaodong Hu, and Jianming Zhu. “Minimum data aggregation time problem in wireless sensor networks”. In: *International conference on mobile ad-hoc and sensor networks*. Springer. 2005, pp. 133–142.
- [19] SC-H Huang et al. “Nearly constant approximation for data aggregation scheduling in wireless sensor networks”. In: *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE. 2007, pp. 366–372.
- [20] Peng-Jun Wan et al. “Minimum-Latency Aggregation Scheduling in Multihop Wireless Networks”. In: New York, NY, USA: Association for Computing Machinery, 2009, pp. 185–194. ISBN: 9781605586243.
- [21] Miloud Bagaa et al. “Semi-structured and unstructured data aggregation scheduling in wireless sensor networks”. In: *2012 Proceedings IEEE INFOCOM*. 2012, pp. 2671–2675.
- [22] Bo Yu, Jianzhong Li, and Yingshu Li. “Distributed data aggregation scheduling in wireless sensor networks”. In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 2159–2167.
- [23] XiaoHua Xu et al. “An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks”. In: *Proceedings of the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*. 2009, pp. 47–56.
- [24] Miloud Bagaa et al. “Distributed low-latency data aggregation scheduling in wireless sensor networks”. In: *ACM Transactions on Sensor Networks (TOSN)* 11.3 (2015), pp. 1–36.
- [25] Hamed Yousefi et al. “Fast aggregation scheduling in wireless sensor networks”. In: *IEEE Transactions on Wireless Communications* 14.6 (2015), pp. 3402–3414.

- [26] Amitabha Ghosh et al. "Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks". In: *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*. 2009, pp. 363–372.
- [27] Deying Li et al. "Conflict-Free Many-to-One Data Aggregation Scheduling in Multi-Channel Multi-Hop Wireless Sensor Networks". In: *2011 IEEE International Conference on Communications (ICC)*. 2011, pp. 1–5.
- [28] Fatemeh Ghods et al. "MC-MLAS: Multi-channel minimum latency aggregation scheduling in wireless sensor networks". In: *Computer Networks* 57.18 (2013), pp. 3812–3825.
- [29] Meirui Ren et al. "Distributed Data Aggregation Scheduling in Multi-Channel and Multi-Power Wireless Sensor Networks". In: *IEEE Access* 5 (2017), pp. 27887–27896.
- [30] Ngoc-Tu Nguyen, Bing-Hong Liu, and Hao-Zhe Weng. "A Distributed Algorithm: Minimum-Latency Collision-Avoidance Multiple-Data-Aggregation Scheduling in Multi-Channel WSNs". In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–6.
- [31] Quan Chen et al. "Distributed non-structure based data aggregation for duty-cycle wireless sensor networks". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.
- [32] Xianlong Jiao et al. "Delay efficient scheduling algorithms for data aggregation in multi-channel asynchronous duty-cycled WSNs". In: *IEEE Transactions on Communications* 67.9 (2019), pp. 6179–6192.
- [33] Chen Tian et al. "Neither shortest path nor dominating set: Aggregation scheduling by greedy growing tree in multihop wireless sensor networks". In: *IEEE Transactions on Vehicular Technology* 60.7 (2011), pp. 3462–3472.
- [34] A BabuKaruppiyah and S Kannadhasan. "A novel approach to detect the shortest path for secure data aggregation using fuzzy logic in wireless sensor networks". In: *International Journal Of Engineering And Computer Science* 2.2 (2013), pp. 506–510.
- [35] Surender Redhu, Pratyush Garg, and Rajesh Hegde. "Joint mobile sink scheduling and data aggregation in asynchronous wireless sensor networks using Q-learning". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 6438–6442.
- [36] Wan-Kyu Yun and Sang-Jo Yoo. "Q-Learning-Based Data-Aggregation-Aware Energy-Efficient Routing Protocol for Wireless Sensor Networks". In: *IEEE Access* 9 (2021), pp. 10737–10750.
- [37] Xiaoding Wang et al. "A secure data aggregation strategy in edge computing and blockchain empowered Internet of things". In: *IEEE Internet of Things Journal* (2020).
- [38] Tien-Dung Nguyen et al. "Fast Sensory Data Aggregation in IoT Networks: Collision-Resistant Dynamic Approach". In: *IEEE Internet of Things Journal* (2020).