# Swift Carrier Scheduling for Battery-free Sensor Tags with Sensing Chain Requirements

Lingtao Xue[1], Xuewen Dong[1], Haoben Lu[1], Jianming Zhao[1],and Weifeng Chen[1]
[1]School of Computer Science & Technology, Xidian University, Xi'an 710071, China

**Battery-free sensor tags extensively enhance sensing capabilities of IoT in a cost-effective manner, which are provided energy from unmodulated carriers of other IoT devices. Passive tag scheduling algorithms have been proposed to achieve battery-free sensor tag group scheduling through coloring active devices/nodes for carrier sharing. However, existing passive tag scheduling works realize scheduling each sensor tag once, missing chain-like scheduling requirements in scenarios such as pipeline safety detection. Besides, the existing active node coloring manner leads to frequent re-coloring since carrier conflicts change after part of tags are scheduled. In this paper, we are the first to propose the concept of sensing chain, which represents scheduling multiple battery-free sensor tags in a specific order. Then, we formulate the scheduling problem as a pure integer programming problem to jointly optimize carrier generation and energy consumption. To address this NP-hard problem, we present three types of carrier sharing coloring strategies, and develop an efficient scheduling algorithm with one-time tag (ordinary and sensing chain tag) coloring. Extensive experiments demonstrate that our proposed algorithm significantly reduces energy consumption compared to sequential scheduling. Besides, our solution is close to optimal while reducing execution time by over 40% with or without the sensing chain than state-of-arts.**

***Index Terms*—Battery-free Sensor Tags, Passive Tag Scheduling, Sensing Chain.**

## I. Introduction

With the aid of an external unmodulated carrier, backscatter communication technology could enable sensor devices to achieve two-way communication with the Internet of Things (IoT) devices [1]–[4]. These low-power sensor devices are usually deployed in complex terrain and environments, which normally face challenges for sensor maintenance and battery replacement [5]. To tackle the above problems, battery-free sensors have been proposed and widely studied due to the following reasons. On the one hand, battery-free sensors could operate without batteries through unmodulated carriers or various other energy harvesting technologies [6] [7]. On the other hand, they are directly compatible with IoT nodes, which is conducive to deployment and maintenance in wearable devices and infrastructure [8] [9] where batteries could not be installed.

**Background:** Battery-free sensor tags use backscatter communication technology that relies on an external unmodulated carrier to receive and transmit data.

- **Unmodulated carrier:** The unmodulated carrier comes from standard IoT devices in the network, which usually owns a radio test mode [10] [11]. This mode exists for regulatory certification, but is utilized as a carrier generator here. To transmit, a sensor tag employs backscatter communication techniques that selectively reflects an external Radio Frequency (RF) signal to modulate it and convey information [12] [13].
- **Carrier interference:** Because random phase and frequency offsets among carriers could cause problems for transmission and reception, a tag could not operate properly when two or more unmodulated carriers are provided [14] [15]. Thus, to avoid collisions, there is a restriction that a tag could only communicate when provided with a single unmodulated carrier.

**Scenario:** Sensor tags, as a new type of battery-free device, attracted much attention for their flexibility and adaptability. Sensor tags are usually placed around regular nodes to provide sensing functions without adding power supply devices, thereby reducing deployment and maintenance costs [16] [17]. In these applications, regular nodes could schedule sensor tags to acquire sensor data by providing the unmodulated carrier needed for tag communication. Existing studies have focused on the scheduling problem of sensor tags [14] [18], and each tag is required to be scheduled once, however, missing chain-like (orderly) scheduling requirements. in some cases, orderly collection and processing of data from multiple sensor tags could be helpful for special tasks. Currently, in scenarios such as NFV/SDN [19] [20], scheduling of service chains has been well discussed. Thus, considering our scenario and task requirements, multiple sensors could also be scheduled in a specific order to perform chained tasks. These orderly collection of sensor tags that perform sensing tasks in a specific chain sequence is called sensing chain. By scheduling a sensing chain, our regular nodes could obtain a sequence of sensor data in a specific order and perform statistical or abnormal analysis on some conditions.

The sensing chain could provide helpful analysis data for structural health detection scenarios [21] [22], two of which are as follows:

- ***Track safety detection:*** Sensor tags could be placed on the track at intervals to measure the pressure, temperature, and other parameters during the running of the train to judge the health of the track. Since the train's position is constantly moving, the sensor tags that the train moves through could form a sensing chain, and the sensor tags are interrogated in turn according to the train's position to obtain sensor data.

Manuscript received December 27, 2023; revised February 20, 2024.
Corresponding author: Xuewen Dong (email: xwdong@xidian.edu.cn).

- ***Pipeline safety detection:*** We place sensors on the pipeline at intervals to detect the leakage risk of the pipeline. As the gas or liquid enters the pipeline, the sensor tags on the flow path of these gases or liquids could form a sensing chain, and the sensor data on these paths are sequentially interrogated for safety analysis.

However, the existing research ignored requirements of sensing chain scheduling in above scenarios. Thus, discussing the scheduling problem of the sensing chain is highly necessary and extremely valuable. Furthermore, existing algorithms are designed base on optimization or graph coloring methods [18] [23]. However, with the increase of the network scale, the optimal method requires a long execution time to obtain scheduling results, which is unsuitable for the real-time scheduling of sensors. Besides, since active nodes could not provide carriers for a battery-free tag simutanously, a conflict graph could be constructed. Accoring to the conflict graph, in existing passive sensor tag scheduling algorithms, graph coloring methods group active nodes for carrier sharing. As the conflict graph of the tag scheduling network constantly changes for part of sensor tags scheduled, each round of scheduling needs to build an updated conflict graph and color the remaining nodes. This re-coloring manner results in the inefficient scheduling.

**Contribution:** In this paper, we analyze the requirements of the sensing chain in the above scenarios and focus on the problem of tag scheduling with a sensing chain. As far as we know, this is the first paper that focuses on scheduling issues in the presence of a sensing chain in IoT scenarios. We aim to obtain a tag scheduling solution by jointly optimizing carrier generation and energy consumption. Besides, we propose an efficient scheduling algorithm based on tag coloring to solve the re-coloring issue. The main contributions of this paper are summarized as follows.

- ***Sensing chain requirement analysis and formulation:*** We are the first to propose the concept of sensing chain, which means multiple passive sensor tags form a chain-like structure according to a specific sequence and to be scheduled in turn. Besides, we develop a mathematical formulation of a sensing chain with a mapping between tags and time slots;
- ***Tag scheduling problem modeling with a sensing chain:*** We propose the tag scheduling model with sensing chain in the IoT scenario. Multiple tags could be scheduled to simultaneously reuse the unmodulated carrier sent by an active node to reduce energy consumption. Besides, the tag's scheduling process must meet the sensing chain's scheduling sequence. Then, we model the scheduling problem as a pure integer programming problem to jointly optimize carrier generation and energy consumption.
- ***A swift tag scheduling algorithm with an one-time tag coloring strategy:*** We develop a swift tag scheduling algorithm with an one-time tag coloring strategy to color tags into different groups and schedule per group, while optimizing carrier generation and energy consumption. Unlike the active node coloring methods in existing passive tag scheduling work, we color the tags once according to the conflict and the scheduling order of the sensing chain, which avoids recoloring caused by conflict changes. Meanwhile, our algorithm supports requirements with and without the sensing chain.
- ***Significant performance improvement:*** Simulation experiments demonstrate that our algorithm outperforms other state-of-arts greatly and close to the optimal solution. The simulations reveal that under different network scales and the number of tags, our algorithm is far superior to the sequential scheduling algorithm in terms of energy consumption. Furthermore, our algorithm slightly outperforms the active node coloring approach without the sensing chain and reduces the execution time by more than 40 percent.

The remainder of the paper is structured as follows. Section 2 introduces the background knowledge of unmodulated carrier and carrier interference, which are the basis for implementing the tag schedule. In Section 3, we briefly introduces related work. Section 4 establishes the overall system model. We describe the algorithm design in detail in Section 5 and evaluate our designed algorithms in Section 6. In Section 7, we summarize this paper.

## II. Related Work

This section summarizes the related work on the development of backscatter communication enabled passive sensors to operate and communicate with unmodulated carriers.

### A. Passive Sensor Tags with Unmodulated Carrier

Some work has integrated battery-free tags into standard networks and efficiently used unmodulated carriers. The authors in Ref. [2] proposed an architecture consisting of tags, readers, and multiple carrier generators. They used WiFi routers and sensor nodes as carrier signal sources to separate carrier generation from readers. The author proposed the TunnelScatter mechanism in Ref. [4], which overcomed the limitation that the communication range is proportional to the strength of the ambient carrier signal (ACS) in the existing backscatter system. The authors in Ref. [6] designed a low-power platform that eliminated the typical energy inefficiency issues in RF backscatter downlink reception and significantly facilitates the application of battery-free tags. These works provided technical support for the application of sensor tags in sensor networks and the Internet of Things. However, these works do not consider the scheduling of passive sensor tags and the collision problem in the scheduling process.

### B. Carrier Scheduling for Passive Sensor Tags

There are a few works on the carrier scheduling problem of passive sensor tags. The authors in Ref. [14] proposed TagAlong, a medium access mechanism for interoperable sensor tags that enables multiple tags to share a carrier and synchronous communication with tags that share a carrier generator, while optimizing the carrier scheduling. The authors designed a scheduling mechanism in Ref. [18] that utilizes time slots to coordinate unmodulated carriers while minimizing delay, energy consumption, and overhead radio emissions. Besides, they proposed an scheduling algorithm that parallelizes the communication with battery-free tags where possible and simultaneously shares carriers among multiple tags.

However, these studies treat each sensor as a separate individual, ignoring sensing chain scheduling requirements, which means some passive sensor tags need to be scheduled in a specific order. Besides, existing tag scheduling methods, such as optimization algorithms and active node coloring schemes, have problems such as low efficiency, long execution time, and not supporting the sensing chain. To overcome the shortcomings of existing work, in this paper, we introduce the concept of sensing chain and propose a swift tag scheduling algorithm with an one-time tag coloring strategy to obtain scheduling results while optimizing carrier generation and energy consumption. Meanwhile, our algorithm supports the requirement with or without a sensing chain. Table I summarizes the difference and innovation of our work with existing work.

TABLE I: COMPARISION WITH RELATED WORKS

| Properities | [5] | [14] | [18] | [24] | [25] | Ours |
|---|---|---|---|---|---|---|
| Sensing chain requirement | × | × | × | × | × | ✓ |
| Carrier collision | × | ✓ | ✓ | × | × | ✓ |
| Latency overhead | × | × | ✓ | × | × | ✓ |
| Energy consumption | ✓ | × | ✓ | ✓ | × | ✓ |
| Supporting large-scale scenarios | × | × | ✓ | × | ✓ | ✓ |
| Battery-free | × | ✓ | ✓ | ✓ | ✓ | ✓ |

## III. Model and Problem Formulation

### A. System Model

As shown in Fig. 1, we study the tag scheduling problem when passive sensor tags are interrogated under two scenarios in a heterogeneous wireless sensor network. The two scenarios correspond to tag scheduling requirements with or without a sensing chain. The network contains $A$ active nodes and $M$ passive sensor tags, labeled $\mathcal{N} = \{N_1, N_2, ..., N_a, ..., N_A\}$ and $\mathcal{T} = \{T_1, T_2, ..., T_m, ..., T_M\}$, respectively. Active nodes are standard devices with radio transceivers that support

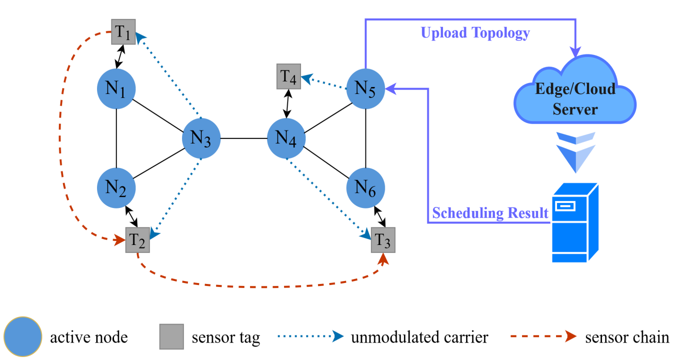



Fig. 1: Example of network topology and system model showing the scheduling of tags in a scenario with a sensing chain.

TABLE II: KEY NOTATIONS

| Notation | Definition |
|---|---|
| $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ | The topology of the network |
| $\mathcal{G}' = (\mathcal{N}, \mathcal{E}')$ | The conflict graph of the topology $\mathcal{G}$ |
| $\mathcal{N}$ | The active node set of the network |
| $\mathcal{T}$ | The sensor tag set of the network |
| $X_{s,N_a,T_m}$ | The scheduling result of the tag scheduling algorithm, where $s \in \mathcal{S}, N_a \in \mathcal{N}, T_m \in \mathcal{T}$ |
| $H_T$ | The tag-to-host mapping |
| $C_T$ | The mapping of chain's tag to slot |
| $\mathcal{S}$ | The set of scheduling result's time slots |
| $\mathcal{A}_m$ | The set of active nodes that could provide a carrier for the tag $T_m$ |
| $\mathbb{C}$ | The set of used colors |
| $\mathbb{T}_c$ | Collection of tags colored in $c \in \mathbb{C}$ |
| $\mathbb{N}_c$ | $\mathbb{N}_c$ contains many node combinations that could provide carriers for tags in $\mathbb{T}_c$ |
| $\mathbb{N}_c^k$ | $\mathbb{N}_c^k$ is a node set in $\mathbb{N}_c$ |
| $\mathbb{N}_{new}$ | $\mathbb{N}_{new}$ is used to update $\mathbb{N}_c$ |

commodity physical layer protocols such as Bluetooth [26] or IEEE 802.15.4 [27] [24] and could generate unmodulated carriers for passive sensor tags. Passive sensor tags are a class of battery-free devices with energy harvesting and sensing capabilities. Each tag should be placed near an active node which is called its host. The carrier generated by an active node $N_a$ could provide energy for tags hosted by $N_a$'s neighbor nodes. The sensor network runs on a Time Division Multiple Access (TDMA) Media Access Control (MAC) protocol [28] [29]; and the schedule consists of a set of $\mathcal{S} = \{1, 2, ..., s, ..., S\}$ time slots. To present this research more clearly, we list some important notations used in this article in Table II.

We model the network topology as an undirected graph, denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the vertex set $\mathcal{N}$ is also the above active node set, and the edge set $\mathcal{E}$ means the communication links set between those active nodes. Each active node could host multiple tags, and the mapping of a tag to its host node is denoted as $H_T : T \in \mathcal{T} \rightarrow N \in \mathcal{N}$. If any two tags, $T_i$ and $T_j$, have a common host node, these two tags are called sibling tags. For each edge $e_{i,j} \in \mathcal{E}$, the weighted valued $w_{i,j}$ on it means the carrier's signal strength observed at node $N_i$ coming from node $N_j$. Node $N_j$ could provide an effective unmodulated carrier for node $N_i$ iff $w_{i,j}$ is greater than the threshold $w^{th}$. Similiar to [18] [25], the distance between a tag and its host is assumed to be much smaller than the distance between any two nodes. Thus, the carrier signal strength measured at a tag is assumed to equal to the signal strength at its host.

### B. Sensing Chain Model

A sensing chain in the network is denoted by a mapping set as $C_T : T \in \mathcal{T}_c \rightarrow s \in \mathcal{S}_c$, where $\mathcal{S}_c \subset \mathcal{S}$ denotes the scheduling time slot set of the sensing chain, and $T_c \subset \mathcal{T}$ represents the sensor tag set that needs to be scheduled in the sensing chain. For example, we suppose the set $\mathcal{T}_c = \{T_1, T_2, T_3\}$ and $\mathcal{S}_c = \{s_{T_1}, s_{T_2}, s_{T_3}\}$, which means in this sensing chain, sensor tags $T_1, T_2$ and $T_3$ need to be scheduled in the time slots $s_{T_1}, s_{T_2}$ and $s_{T_3}$, respectively. The tags in a sensing chain are required to satisfy the order constraint in

$\mathcal{S}_c$, which means tags in the chain should be interrogated in sequence and could not be interrogated simultaneously.

### C. Tag Scheduling Model

In the network shown in Fig. 1, at least one of the active nodes should be connected to an edge server or a cloud server. The server constructs a network topology by collecting the node information and carries out a tag scheduling algorithm based on the topology structure and the sensing chain to obtain an optimized scheduling result. Then, the scheduling result is propagated to all nodes for execution through the node connected to the server. To avoid scheduling failures caused by topology changes and link changes, the server continues to collect topology information and recalculates the tag interrogation schedule.

Based on the schedule, each node in the network performs its function (e.g., remaining off, emitting a carrier, or interrogating an indicated tag) to obtain sensor data orderly. We model the tag schedule of the scheduling algorithm as $X_{s,N_a,T_m} = \{0, 1\}$. $X_{s,N_a,T_m} = 1$ represents that active node $N_a$ provides an unmodulated carrier to tag $T_m$ in time slot $s$. $X_{s,N_a,T_m} = 0$ means active node $N_a$ remains off or queries tag $T_m$ hosted by it in time slot $s$ which further depends on whether another active node provides a carrier for its tag $T_m$.

Similar to [18], the process of an active node interrogating a tag is allocated in two consecutive time slots. The downlink for tag interrogation is on the first time slot, and the uplink is on the next time slot. Fig. 2 shows a detail process example of active node $N_1$ interrogating sensor tag $T_1$, in which $N_2$ provides $T_1$ with an unmodulated carrier. In the first time slot, $N_1$ sends a short carrier ($cg$) for a duration $T_{req}$ for requesting $N_2$ to generate an unmodulated carrier. When $N_2$ detects the short carrier ($cg$), it would transmit two unmodulated carriers ($cg_1$ and $cg_2$) on the current and the next time slot for a duration $T_{cg}$. The tag $T_1$ using $cg_1$ to receive the request from $N_1$. In the next time slot, the carrier $cg_2$ would be generated so that the tag could transmit the data to its host $N_1$.

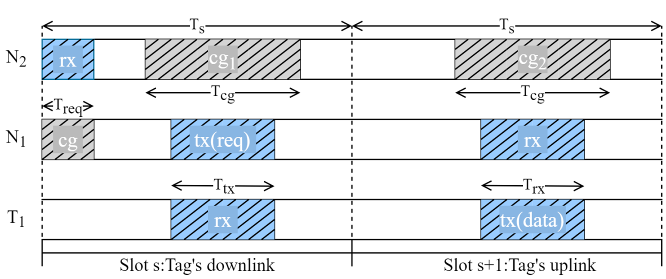



Fig. 2: The process of an active node interrogating a sensor tag distributes over two consecutive time slots. (cg:carrier generation, rx:receiving, tx:transmitting)

### D. Energy Consumption Model

To describe the average energy consumption, we define the carrier ratio as $\eta_c = \epsilon_c/M$, which is the fraction of cycles ($\epsilon_c$) of carrier generation in our solution relative to the number of cycles ($M$ cycles) required to interrogate all tags sequentially. By definition, the value of $\eta_c$ is 1 for sequential scheduling.

It could be known from Fig. 2 that the average energy that active nodes invest to interrogate all tags consists of three parts: transmission energy consumption $\widetilde{E}_{tx}$, reception energy consumption $\widetilde{E}_{rx}$, and carrier generation energy consumption $\widetilde{E}_{cg}$. The expression of the average energy is as follow:

$$\widetilde{E} = \widetilde{E}_{tx} + \widetilde{E}_{rx} + \widetilde{E}_{cg}. \tag{1}$$

The formulas of the three parts are as follows:

$$\widetilde{E}_{tx} = \frac{P_{tx} M T_{tx}}{M} = P_{tx} T_{tx}, \tag{2}$$

$$\widetilde{E}_{rx} = \frac{P_{rx}(\epsilon_c T_{req} + M T_{rx})}{M} = P_{rx}(\eta_c T_{req} + T_{rx}), \tag{3}$$

$$\widetilde{E}_{cg} = \frac{P_{tx}(M T_{req} + 2\epsilon_c T_{cg})}{M} = P_{tx}(2\eta_c T_{cg} + T_{req}), \tag{4}$$

where $P_{rx}$ and $P_{tx}$ are the power consumption of the active radios in the reception and transmission modes, respectively. $T_{tx}$ is the duration time of the interrogation message, $T_{rx}$ is the time that the host node takes to receive a reply, and $T_{req}$ is the time that the host node takes to request a carrier.

### E. Problem Formulation

The tag scheduling optimization problem aims to find a time slot assignment, so that all passive sensor tags in the network could be queried once under sensing chain requirements in the shortest scheduled time slot without carrier collision. Thus, the total number of scheduled time slots is expressed as

$$\min \sum_{s\in\mathcal{S}} \left( \bigvee_{N_a\in\mathcal{N}} \bigvee_{T_m\in\mathcal{T}} \right) X_{s,N_a,T_m}. \tag{5}$$

Considering the conflict of carrier scheduling, we list the problem's constraints as follows. Firstly, all sensor tags must be scheduled in a specified time slot, and we could yield

$$\sum_{s\in\mathcal{S}} \sum_{N_a\in\mathcal{N}} \sum_{T_m\in\mathcal{T}} X_{s,N_a,T_m} = M. \tag{6}$$

Next, each sensor tag could only be scheduled once, and the following constraints are obtained

$$\sum_{s\in\mathcal{S}} \sum_{N_a\in\mathcal{N}} X_{s,N_a,T_m} = 1, \forall T_m \in \mathcal{T}. \tag{7}$$

Each active node could not provide carriers for the tags it hosts. Then we have

$$X_{s,H_{T_m},T_m} = 0, \forall s \in \mathcal{S}, \forall T_m \in \mathcal{T}. \tag{8}$$

In each time slot, an active node could only perform one function, scheduling its own tag or providing a carrier for neighbor tags. The following constraints are obtained

$$\bigvee_{T_m\in\mathcal{T}} X_{s,N_a,T_m} + \bigvee_{N_i\in\mathcal{N}_a, T_j\in\mathcal{T}_a} X_{s,N_i,T_j} \leq 1, \forall s \in \mathcal{S}, \forall N_a \in \mathcal{N}, \tag{9}$$

where $\mathcal{N}_a$ represents the set of neighbor nodes of node $N_a$, and $\mathcal{T}_a$ denotes the set of tags hosted by node $n$.

At most, one tag of an active node could be scheduled in each slot. Then we yield

$$\sum_{T_i \in \mathcal{T}_a} ( \bigvee_{N_j \in \mathcal{N}_a} X_{s,N_j,T_i}) \leq 1, \forall s \in \mathcal{S}, \forall N_a \in \mathcal{N}. \quad (10)$$

To avoid scheduling failure due to collision, only one neighbor node could provide it with a carrier when a tag is scheduled. Thus, we have

$$\sum_{N_i \in \mathcal{N}_m} X_{s,N_i,T_m} \leq 1, \forall s \in \mathcal{S}, \forall T_m \in \mathcal{T}, \quad (11)$$

where $\mathcal{N}_m$ represents the set of nodes that could provide carrier for tag $T_m$.

For an active node, since only its neighbor nodes could provide carriers for the tags it hosts, the following constraints are yield

$$X_{s,N_i,T_m} = 0, \forall s \in \mathcal{S}, \forall T_m \in \mathcal{T}, \forall N_i \in \mathcal{N}_m. \quad (12)$$

The tags of the sensing chain must be scheduled in order based on the sensing chain $C_t$. Then we have

$$\sum_{N_j \in \mathcal{N}_i} X_{s,N_j,T_i} = 1, \forall T_i \in \mathcal{T}_c, \forall s \in \mathcal{S}_c. \quad (13)$$

Our goal is to minimize the time slots spent in scheduling all tags. Thus, the optimization problem could be expressed as

$$\mathcal{P}1: \quad \min \sum_{s \in \mathcal{S}} \left( \bigvee_{N_a \in \mathcal{N}} \bigvee_{T_m \in \mathcal{T}} \right) X_{s,N_a,T_m}. \quad (14)$$

s.t. constraints (7)-(13).

## IV. Tag Scheduling Design

The tag scheduling problem $\mathcal{P}1$ aims to provide all battery-free tags with effective unmodulated carriers in the fewest time slots, while minimizing the energy consumption. This pure integer programming problem is NP-hard, which is hard to be solved optimally in polynomial time. Thus, this paper proposes an efficient tag scheduling algorithm, which is close to the optimal solution validated by experiments. The algorithm includes three phases (1) the conflict graph construction phase, which computes the topology's conflict graph to obtain the collision situations of the active node's carriers. (2) tag coloring phase, which colors all tags once according to the conflict graph. (3) tag scheduling result generation phase, which constructs the schedule based on the one-time coloring results of tags.

### A. Conflict Graph Construction

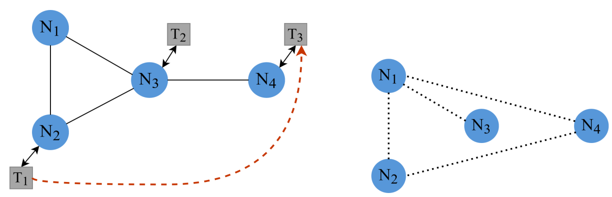
Due to each tag could only accept the unmodulated carrier provided by one active node in our scheduling, we first built a conflict graph to describe conflicting relationships between nodes. In the conflict graph $\mathcal{G}'$, there is an edge between nodes $N_i$ and $N_j$ if they have at least one common neighbor with associated tags, which means they could not generate carriers simultaneously. Otherwise, scheduling would fail because multiple active nodes provide carriers for one tag. For example, in Fig. 3, $N_1$ and $N_3$ have a common neighbor $N_2$ with associated tag $T_1$. Thus, $N_1$ and $N_3$ are in conflict. Figures 3(b), 4(b), and 5(b) show the conflict graphs of Figures 3(a), 4(a), and 5(a) according to the above rules, respectively.

### B. Tag Coloring

Contrast to existing passive tag scheduling scheme with re-coloring on active nodes, we design a one-time coloring method on all tags. Tags with the same color mean that they could be scheduled simultaneously without interruption. Due to the tags in the sensing chain needing to be scheduled in order, we first color them differently and record the carrier node set for each color. Then, we determine the color of the remaining tags sequentially. In the process of tag coloring, three situations would be encountered, as shown in Fig. 3, Fig. 4 and Fig. 5.

#### 1) Carrier Sharing

The first case is that an uncolored tag could be scheduled by an existing carrier, which already provides energy for a colored tag. Thus the tag could be colored in an existing color. Fig. 3 shows an example of carrier sharing. We first color the tags $T_1$ and $T_3$ in the sensing chain in red and yellow, respectively, and obtain the nodes set that provide them with carriers, as shown in Table III (before coloring $T_2$). Then, we find that $T_1$ and $T_2$ could share the same carrier provided by $N_1$. Thus we could color $T_2$ in red and update the carrier node set, as shown in Table III (after coloring $T_2$).




(a) Carrier sharing. (b) Conflict graph.

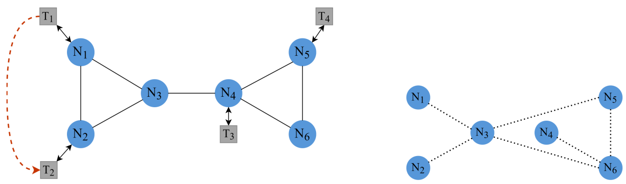
Fig. 3: Example topology 1 shows the case of carrier sharing.

TABLE III: COLORING EXAMPLE: Carrier Sharing

| Phase | Color | Tag Set | Carrier Set |
|---|---|---|---|
| Before Coloring $T_2$ | red | $\{T_1\}$ | $\{\{N_1\},\{N_3\}\}$ |
| | yellow | $\{T_3\}$ | $\{\{N_3\}\}$ |
| Before Coloring $T_2$ | red | $\{T_1, T_2\}$ | $\{\{N_1\}\}$ |
| | yellow | $\{T_3\}$ | $\{\{N_3\}\}$ |

#### 2) Parallel Carriers

The second case is that when an uncolored tag could not share a carrier with any colored tag, there is a new node's carrier providing for this tag and being in parallel with the exiting carriers without conflict. Thus, the tag could color in an existing color and schedule with the colored tag in parallel. Fig. 4 shows an example of parallel carriers. We first color the tags $T_1$ and $T_2$ in red and yellow, as shown in Table IV (before coloring $T_3$ and $T_4$). Next, the tag $T_3$ meets the first case and is colored red. Then we color the tag $T_4$, which could be provided with a carrier by $N_4$ or $N_6$. However, neither $N_4$ nor $N_6$ are in the valid carrier set of red and yellow. Thus, we judge whether $N_4$ or $N_6$ could generate a carrier in parallel with carrier node sets of existing colors. We find that the combination of $\{N_3, N_4\}$, $\{N_1, N_4\}$, $\{N_1, N_6\}$ could generate carriers in parallel. Thus, $T_4$ could be colored in yellow, and the carrier node set is updated, as shown in Table IV (after coloring $T_3$ and $T_4$).




(a) Parallel carriers. (b) Conflict graph.

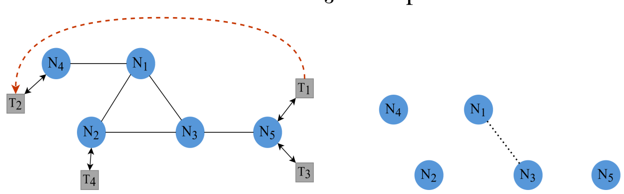
Fig. 4: Example topology 2 shows the case of parallel carriers.

TABLE IV: COLORING EXAMPLE: Parallel Carrier

| Phase | Color | Tag Set | Carrier Node Set |
|---|---|---|---|
| Before coloring $T_3$ and $T_4$ | red | $\{T_1\}$ | $\{\{N_2\},\{N_3\}\}$ |
| | yellow | $\{T_2\}$ | $\{\{N_1\},\{N_3\}\}$ |
| After coloring $T_3$ and $T_4$ | red | $\{T_1, T_3\}$ | $\{\{N_3\}\}$ |
| | yellow | $\{T_2, T_4\}$ | $\{\{N_1, N_4\},\{N_1, N_6\},\{N_3, N_4\}\}$ |

### 3) Carrier Collision

The third case indicates that carrier collision exists when coloring an uncolored tag in the above two cases. That is, this uncolored tag could neither share a current existing carrier with any colored tag nor be provided a carrier by a new node in parallel with existing carriers. Thus, a new color needs to be added to color it. Fig. 5 shows an example of carrier collision. We first color the sensing chain tags in red and yellow, as shown in Table V (before coloring $T_3$ and $T_4$). Since the same node's tags could not be scheduled simultaneously, $T_3$ could not be colored in red. Then, the node $N_3$ providing the carrier for $T_3$ conflicts with $N_1$, which provides a carrier for for the tag set colored in yellow. Thus, $T_3$ could not be colored in yellow either. Since the existing color could not colored $T_3$, we add a new color to color $T_3$ and update the Table V.




(a) Carrier collision. (b) Conflict graph.

Fig. 5: Example topology 3 shows the case of adding color.

TABLE V: COLORING EXAMPLE: Carrier Collision

| Phase | Color | Tag Set | Carrier Node Set |
|---|---|---|---|
| Before coloring $T_3$ and $T_4$ | red | $\{T_1\}$ | $\{\{N_3\}\}$ |
| | yellow | $\{T_2\}$ | $\{\{N_1\}\}$ |
| After coloring $T_3$ and $T_4$ | red | $\{T_1, T_4\}$ | $\{\{N_3\}\}$ |
| | yellow | $\{T_2\}$ | $\{\{N_1\}\}$ |
| | yellow | $\{T_3\}$ | $\{\{N_3\}\}$ |

### C. Tag Scheduling Result Generation

After all tags are colored, the set of tags for each color and the set of nodes that could provide valid carriers for them are obtained. Next, we schedule the tags following the color order, and the color order satisfies the scheduling order of the tags in the sensing chain. A schedule is generated by randomly selecting one case from the set of nodes that provide valid carriers. For example, Table IV shows three combinations that could provide carriers for set $\{T_2, T_4\}$, which are $\{N_1, N_4\}$, $\{N_1, N_6\}$, and $\{N_3, N_4\}$, respectively. We choose any one of these sets to generate carriers.

**Algorithm 1:** Tag scheduling algorithm

**Input:** $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, $\mathcal{T}$, $H_T$, $C_T$
▷ **Step 1:Initializing:**
$\mathbb{C} = \{\}, \mathbb{T}_c = \{\}, \mathbb{N}_c = \{\}$, time slot $s = 1$.
$\forall T_m \in \mathcal{T}$, update $\mathcal{A}_m$ based on $\boldsymbol{W}$ and $w^{th}$
$X_{s,N_a,T_m} \leftarrow$ new empty schedule
Computes $\mathcal{G}' = CONFLICT\_GRAPH(\mathcal{G})$
▷ **Step 2:Sensing chain tags coloring:**
Color tags of the sensing chain with different colors, and update $\mathbb{C}$, $\mathbb{T}_c$, and $\mathbb{N}_c$.
▷ **Step 3:Remaining tags coloring:**
**for** *tag* $T_m \in \mathcal{T}/\mathcal{T}_c$ **do**
  boolean $colored = false$
  **for** *color* $c \in \mathbb{C}$ **do**
    **if** $T_m$*'s each sibling tag* $\notin \mathbb{T}_c$ **then**
      ▷ Determine whether $T_m$ could be colored in $c$
      $colored = Algorithm2(T_m, c, \mathcal{G}', \mathbb{T}_c, \mathbb{N}_c, \mathcal{A}_m)$
    **if** $colored = true$ **then**
      break
▷ Carrier collision:
**if** $\forall c \in \mathbb{T}_c$ *could not color tag* $T_m$ **then**
  Choose a new color $c$ to color tag $T_m$, update $\mathbb{C}$, $\mathbb{T}_c$, and $\mathbb{N}_c$.
▷ **Step 4:Obtaining the schedule result:**
**for** $c \in \mathbb{C}$ **do**
  Choose a set $\mathbb{N}_c^k \in \mathbb{N}_c$
  **for** $T_m \in \mathbb{T}_c$ **do**
    Choose a node $N_a$ :$\{N_a, H_{T_m}\} \in \mathcal{E} \wedge N_a \in \mathbb{N}_c^k$
    Update the schedule $X_{s,N_a,T_m} = 1$
  $s = s + 1$
**Output:** $X_{s,N_a,T_m}$.

**Algorithm 2:** Color judgement algorithm

**Input:** $T_m, c, \mathcal{G}', \mathbb{T}_c, \mathbb{N}_c, \mathcal{A}_m$
Initialize $colored = false, \mathbb{N}_{new} = \{\}$
▷ **Step 1:Carrier sharing judgement:**
**for** *nodeset* $\mathbb{N}_c^k \in \mathbb{N}_c$ **do**
  **if** $\mathbb{N}_c^k \cap \mathcal{A}_m \neq \emptyset$ **then**
    ▷ $\mathbb{N}_c^k$ could provide a carrier for $T_m$
    $\mathbb{N}_{new}.add(\mathbb{N}_c^k)$
**if** $\mathbb{N}_{new} \neq \emptyset$ **then**
  ▷ Tag $T_m$ could be colored in $c$
  $colored = true$, $\mathbb{T}_c.add(T_m)$, $\mathbb{N}_c = \mathbb{N}_{new}$
  **return** $colored$
▷ **Step 2:Parallel carrier judgement:**
**for** *nodeset* $\mathbb{N}_c^k \in \mathbb{N}_c$ **do**
  **for** *node* $N_i \in \mathcal{A}_m$ **do**
    **if** *node* $N_i$ *and* $\forall N_j \in \mathbb{N}_c^k$ *not conflict* **then**
      $\mathbb{N}_{new}.add(N_i \cup \mathbb{N}_c^k)$
**if** $\mathbb{N}_{new} \neq \emptyset$ **then**
  ▷ Tag $T_m$ could be colored in $c$
  $colored = true$, $\mathbb{T}_c.add(T_m)$, $\mathbb{N}_c = \mathbb{N}_{new}$
**return** $colored$.

### D. Tag Scheduling Algorithm

Based on the analysis of the above three phases, our approximate scheduling algorithm is shown in Algorithm 1.

The algorithm takes $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, $\mathcal{T}$, $H_T$, and $C_T$ as input. The algorithm is divided into four steps:

*Step 1 (Initialization):* We initialize $\mathbb{C}$, $\mathbb{T}_c$ and $\mathbb{N}_c$ as empty sets (Line 2), where $\mathbb{C}$ represents the set of colors that are used to color tags, $\mathbb{T}_c$ means the tag set colored in $c \in \mathbb{C}$, and $\mathbb{N}_c$ is the set of node combinations that could provide carriers for the tags in $\mathbb{T}_c$. Then, we update $\mathcal{A}_m$ based on carrier strength Matrix $\boldsymbol{W}$ and threshold $w^{th}$, where $\mathcal{A}_m$ represents the set of nodes that could provide a carrier for tag $T_m$ (line 3). Line 4 initializes the schedule result $X_{s,N_a,T_m}$ and line 5 computes the conflict graph $\mathcal{G}'(\mathcal{N}, \mathcal{E}')$ of $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.

*Step 2 (Sensing chain tags coloring):* We first color the tags in the sensing chain in different colors and update $\mathbb{C}$, $\mathbb{T}_c$ and $\mathbb{N}_c$ (line 7).

*Step 3 (Remaining tag coloring):* Then, we need to color the remaining tags other than the tags in the sensing chain (lines 9-19). For each $T_m$ which belongs to $\mathcal{T}/\mathcal{T}_c$, we need to judge whether it could be colored in a color $c \in \mathbb{C}$ (lines 9-16). If all sibling nodes of tag $T_m$ do not color in $c$, then Algorithm 2 would judge whether the tag $T_m$ could be colored in $c$ (line 14)(i.e., carrier sharing case or parallel carrier case). If all $c \in \mathbb{C}$ could not color the tag $T_m$, a new color (i.e., carrier collision case) is chosen to color it, and the sets $\mathbb{C}$, $\mathbb{T}_c$, and $\mathbb{N}_c$ are updated (lines 18-19).

*Step 4 (Schedule result generating):* Eventually, the schedule $X_{s,N_a,T_m}$ would be updated by scheduling tags of each color in turn based on the $\mathbb{T}_c$ and $\mathbb{N}_c$ (lines 21-26). The algorithm would terminate until each color $c \in \mathbb{C}$ be chosen.

Algorithm 2 is the coloring judgment algorithm. The primary step of the algorithm is to judge the first two coloring cases in the section V in turn.

*Step 1 (Carrier sharing judgement):* Lines 3 to 6 determine whether node set $\mathbb{N}_c^k \in \mathbb{N}_c$ could provide a carrier for tag $T_m$. If yes, the sets $\mathbb{T}_c$ and $\mathbb{N}_c$ are updated. Tag $T_m$ could be colored in $c$ (lines 7-10).

*Step 2 (Parallel carrier judgement):* When the case of carrier sharing is not satisfied, we need to judge further whether the case of parallel carriers is satisfied. If node set $\mathbb{N}_c^k \in \mathbb{N}_c$ has no conflict with a node $N_i \in \mathbb{A}_m$, we merge $\mathbb{N}_c^k$ and $N_i$ into a new set and add it to $\mathbb{N}_{new}$ as a carrier generation situation (lines 12-15). $\mathbb{N}_{new}$ is not an empty set, indicating that there are parallel carriers. Thus, tag $T_m$ could be colored in $c$, and the sets $\mathbb{T}_c$ and $\mathbb{N}_c$ are updated (lines 16-18). Otherwise, tag $T_m$ could not be colored in $c$.

If all the coloring cases do not satisfy the carrier sharing and parallel carrier cases, then $T_m$ satisfies the carrier collision case, as shown in Algorithm 1 (lines 18-19). Thus we color $T_m$ in a new color.

## V. EVALUATION

In order to verify the performance of our scheme, we compare our proposed algorithm with other solutions in terms of energy consumption and execution time in scenarios with or without a sensor chain. Besides, we conduct experiments under different active node numbers and tag densities to prove the applicability and correctness of our algorithm The details of the comparison algorithms are as follows.

- Sequential scheduling algorithm: The sensor tags are arranged in different time slots and only a sensor tag is scheduled per time slot.
- TagAlong [18]: This algorithm is the only existing passive sensor tag scheduling algorithm that colors active nodes to schedule sensor tags.
- Optimal solution: We utilize the $Gurobi$ solver to solve the proposed pure integer programming problem to obtain the optimal solution.

TABLE VI: EVALUATION TOPOLOGY DETAILS

| Topology | $HurricaneElectric$ | $SwitchL3$ | $Topology3$ | $Topology4$ |
|---|---|---|---|---|
| Number of Active Nodes | 24 | 42 | 75 | 100 |
| Average Node Degree | 3.1 | 3.0 | 10 | 13 |

### A. Simulation Setting

We conduct experiments on four topologies with different scales of active nodes. The first two topologies are real datasets from the Topology-zoo website [30] [31], named $HurricaneElectric$ and $SwitchL3$, and the remaining large-scale topologies are randomly generated. Subsequently, we randomly generate sensor tags for these topologies for different tag densities and randomly select some tags for the sensing chain. For each tag density, ten sets of different tag-to-host assignments are generated for each topologies. Active nodes are equipped with antennas with an output power of 12dBm to transmit unmodulated carriers and messages. The specific topology information is shown in Table VI. Due to the solver's solution speed limitation under large-scale topology, we only utilize $Gurobi$ in topology $HurricaneElectric$ to obtain the optimal solution for comparison.

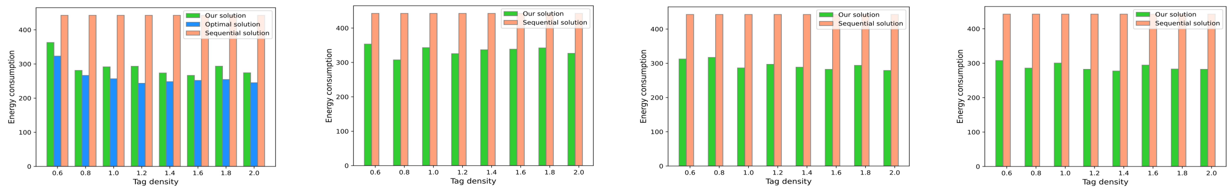
### B. Comparison Scenario with Sensing Chain

This set of experiments compares our solution with comparison solutions at different tag densities in scenarios with a sensing chain. The experiments compare the average energy consumption and carrier ratio $\eta_c$, respectively.

#### 1) Average Energy Consumption

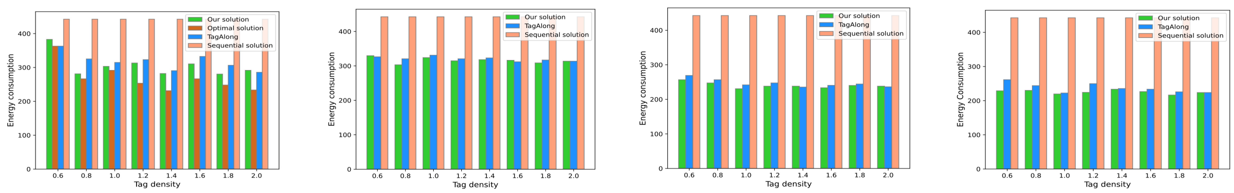
Fig. 6 compares the average energy consumption at a chain density of 0.3. Fig. 6(a) compares the energy consumption ($\mu j$) among our solution, sequential solution, and the optimal solution on the topology $HurricaneElectric$. Fig. 6(b), Fig. 6(c), and Fig. 6(d) compare the energy consumption among our solution and sequential solution on the other three topologies, respectively. Due to the speed limitation, these three topologies could not be solved by $Gurobi$.

According to formulas (1)-(4), we know that more energy is saved in receiving the sensor data and generating the carriers due to the sharing and parallelization of the carrier. The experiment results validate that our algorithm obtains a higher degree of carrier sharing and parallelization than sequential scheduling, thus saving more energy. As shown in Fig. 6(a), our algorithm is close to optimal scheduling in terms of average energy consumption.



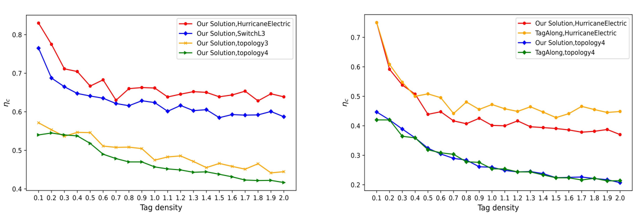

(a) $HurricaneElectric$. (b) $SwitchL3$. (c) $Topology3$. (d) $Topology4$.

Fig. 6: Our solution performs well in scenarios with sensing chain on four topologies. Most energy savings are achieved due to a reduced need for carrier generation.




(a) $HurricaneElectric$. (b) $SwitchL3$. (c) $Topology3$. (d) $Topology4$.

Fig. 7: Our solution performs well in scenarios without sensing chain on four topologies. Most energy savings are achieved due to a reduced need for carrier generation.




(a) Scenarios with Sensing chain. (b) Scenarios without Sensing chain.

Fig. 8: With the increase of tag density, the carrier ratio $\eta_c$ gradually improves. As the size of the network expands, this metric also becomes better (a). Our solution outperforms TagAlong in carrier ratio $\eta_c$ (b).

*2) Carrier Ratio $\eta_c$*

In this set of experiments, we investigate the correlation of scheduling result with network topology. From equations (2)-(4), it could be seen that for different $\eta_c$, there will be changes in energy consumption. From the definitions $\eta_c$, we know that this metric depends on the specific network topology and tag deployment. Fig. 8(a) shows the trend of the metric $\eta_c$ with increasing tag density for the four topologies. The experiment results show that this metric decreases with tag density and eventually stabilizes. This means that with the increased number of tags in the network, the carrier sharing situation becomes better and stabilizes. Besides, Fig. 8(a) shows that $\eta_c$ decreases as the number of active nodes increases, which means that carrier sharing and carrier parallelism would not be affected by the network size.

### C. Comparison Scenario without Sensing Chain

This set of experiments compares our algorithm with other algorithms at different tag densities in scenarios without a sensing chain. The experiments compare the average energy consumption, excution time and carrier ratio $\eta_c$, respectively.

*1) Average Energy Consumption*

To verify the superiority and effectiveness of our algorithm in the scenario without a sensing chain, we introduce another tag scheduling algorithm, which is suitable for the tag scheduling problem without a sensing chain. Thus, we compare the energy comsumption with the other algorithms in Fig. 7 on four topologies. According to the experimental results, we could know that our algorithm is far better than the sequential scheduling algorithm and slightly better than the TagAlong in the scenario without a sensing chain. Besides, the performance of our algorithm could be approximately close to the optimal scheduling scheme obtained by the solver.

*2) Carrier Ratio $\eta_c$*

In this set of experiments, we select two topologies (i.e., $HurricaneElectric$ and $Topology4$ ) to compare our algorithm with TagAlong on the metric $\eta_c$. As seen from Fig. 8(b), in the scenarios without a sensing chain, this metric's trend is the same as in the scenarios with a sensing chain. Furthermore, our algorithm outperforms or approximates TagAlong on this metric in two topologies, which means our algorithm obtains higher scheduling efficiency.

## VI. Conclusion

In this paper, we propose the concept of sensing chains for the first time, which means multiple battery-free sensor tags are scheduled in a specific order. Then, we formulate the tag scheduling problem as a pure integer programming problem to jointly optimize carrier generation, and energy consumption. To address this NP-hard problem, we develop three types of carrier sharing strategies and design an efficient one-time tag coloring scheduling algorithm. Extensive experiments demonstrate that our proposed algorithm significantly reduces energy consumption compared to sequential scheduling. Furthermore, our solution is close to optimal in less execution time with or without the sensing chain than state-of-arts.

## REFERENCES


[1] K. Han and K. Huang, "Wirelessly powered backscatter communication networks: Modeling, coverage, and capacity," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2548–2561, 2017.

[2] A. Varshney, O. Harms, C. Pérez-Penichet, C. Rohner, F. Hermans, and T. Voigt, "Lorea: A backscatter architecture that achieves a long communication range," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–14.

[3] X. Xu, Y. Shen, J. Yang, C. Xu, G. Shen, G. Chen, and Y. Ni, "Passivevlc: Enabling practical visible light backscatter communication for battery-free iot applications," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 180–192.

[4] A. Varshney, A. Soleiman, and T. Voigt, "Tunnelscatter: Low power communication for sensor tags using tunnel diodes," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–17.

[5] P. Cheng, S. He, F. Jiang, Y. Gu, and J. Chen, "Optimal scheduling for quality of monitoring in wireless rechargeable sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 3072–3084, 2013.

[6] A. Galisteo, A. Varshney, and D. Giustiniano, "Two to tango: Hybrid light and backscatter networks for next billion devices," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 80–93.

[7] J. Ensworth, "Ultra-low-power bluetooth low energy (ble) compatible backscatter communication and energy harvesting for battery-free wearable devices," Ph.D. dissertation, 2016.

[8] A. Varshney, A. Soleiman, L. Mottola, and T. Voigt, "Battery-free visible light sensing," in *Proceedings of the 4th ACM Workshop on Visible Light Communication Systems*, 2017, pp. 3–8.

[9] H. Truong, S. Zhang, U. Muncuk, P. Nguyen, N. Bui, A. Nguyen, Q. Lv, K. Chowdhury, T. Dinh, and T. Vu, "Capband: Battery-free successive capacitance sensing wristband for hand gesture recognition," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 54–67.

[10] D. Piumwardane, C. Rohner, and T. Voigt, "Reliable flooding in dense backscatter-based tag-to-tag networks," in *2021 IEEE International Conference on RFID (RFID)*. IEEE, 2021, pp. 1–8.

[11] C. Pérez-Penichet, F. Hermans, A. Varshney, and T. Voigt, "Augmenting iot networks with backscatter-enabled passive sensor tags," in *Proceedings of the 3rd Workshop on Hot Topics in Wireless*, 2016, pp. 23–27.

[12] J. Simonjan, B. D. Unluturk, and I. F. Akyildiz, "In-body bionanosensor localization for anomaly detection via inertial positioning and thz backscattering communication," *IEEE Transactions on NanoBioscience*, vol. 21, no. 2, pp. 216–225, 2021.

[13] I. Mathews, S. N. R. Kantareddy, S. Sun, M. Layurova, J. Thapa, J.-P. Correa-Baena, R. Bhattacharyya, T. Buonassisi, S. Sarma, and I. M. Peters, "Self-powered sensors enabled by wide-bandgap perovskite indoor photovoltaic cells," *Advanced Functional Materials*, vol. 29, no. 42, p. 1904072, 2019.

[14] C. Pérez-Penichet, D. Piumwardane, C. Rohner, and T. Voigt, "Tagalong: efficient integration of battery-free sensor tags in standard wireless networks," in *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2020, pp. 169–180.

[15] H. Park, H. Roh, and W. Lee, "Tagora: A collision-exploitative rfid authentication protocol based on cross-layer approach," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3571–3585, 2020.

[16] G. P. Hancke, B. de Carvalho e Silva, and G. P. Hancke Jr, "The role of advanced sensing in smart cities," *Sensors*, vol. 13, no. 1, pp. 393–425, 2012.

[17] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, computing, and communications for energy harvesting iots: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1222–1250, 2019.

[18] C. Pérez Penichet, D. Piumwardane, C. Rohner, and T. Voigt, "A fast carrier scheduling algorithm for battery-free sensor tags in commodity wireless networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 994–1003.

[19] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware vnf chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 267–276.

[20] W. Ding, W. Qi, J. Wang, and B. Chen, "Openscaas: an open service chain as a service platform toward the integration of sdn and nfv," *IEEE Network*, vol. 29, no. 3, pp. 30–35, 2015.

[21] M. Abdulkarem, K. Samsudin, F. Z. Rokhani, and M. F. A Rasid, "Wireless sensor network for structural health monitoring: a contemporary review of technologies, challenges, and future direction," *Structural Health Monitoring*, vol. 19, no. 3, pp. 693–735, 2020.

[22] H. Mei, M. F. Haider, R. Joseph, A. Migot, and V. Giurgiutiu, "Recent advances in piezoelectric wafer active sensors for structural health monitoring applications," *Sensors*, vol. 19, no. 2, p. 383, 2019.

[23] C. Pérez-Penichet and T. Voigt, "Carrier scheduling in iot networks with interoperable battery-free backscatter tags," in *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2019, pp. 329–330.

[24] P. D. P. Adi and A. Kitagawa, "Quality of service and power consumption optimization on the ieee 802.15. 4 pulse sensor node based on internet of things," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 5, pp. 144–154, 2019.

[25] Y. Ma, C. Tian, and Y. Jiang, "A multitag cooperative localization algorithm based on weighted multidimensional scaling for passive uhf rfid," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6548–6555, 2019.

[26] M. Zimmerling, L. Mottola, and S. Santini, "Synchronous transmissions in low-power wireless: A survey of communication protocols and network services," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–39, 2020.

[27] S. Yi, H. Wang, W. Xue, X. Fan, L. Wang, J. Tian, and R. Matsukura, "Interference source identification for ieee 802.15. 4 wireless sensor networks using deep learning," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–7.

[28] K. H. Mohammadani, K. A. Memon, I. Memon, N. N. Hussaini, and H. Fazal, "Preamble time-division multiple access fixed slot assignment protocol for secure mobile ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 16, no. 5, p. 1550147720921624, 2020.

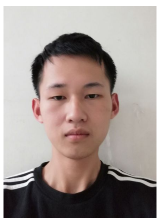
[29] A. B. Tambawal, R. M. Noor, R. Salleh, C. Chembe, M. H. Anisi, O. Michael, and J. Lloret, "Time division multiple access scheduling strategies for emerging vehicular ad hoc network medium access control protocols: a survey," *Telecommunication Systems*, vol. 70, no. 4, pp. 595–616, 2019.

[30] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[31] A. Alshamrani, S. Guha, S. Pisharody, A. Chowdhary, and D. Huang, "Fault tolerant controller placement in distributed sdn environments," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.


**Lingtao Xue** received the B.E. degree in computer science and technology from the Xidian University of China, in 2021. He is now pursuing his Ph.D. degree in Computer Science and Technology in Xidian University, Xi'an, China. His research interests include federated learning and wireless network security.

**Xuewen Dong** received the B.E., M.S. and Ph.D. degrees in computer science and technology from the Xidian University of China, in 2003, 2006 and 2011, respectively. From 2016 to 2017, he was with the Oklahoma State University of USA as a visiting scholar. He is currently a professor in the School of Computer Science and Technology, Xidian University. His research interests include wireless network security and Blockchain.



**Haoben Lu** received the bachelor's degree in cyberspace security from Xiamen University in China and is currently studying for a master's degree in cyberspace security from Xidian University in China. His research interests include fuzz testing, blockchain, etc.

**Jianming Zhao** received his B.E. degree in computer science and technology from Xidian University, China. His research interest includes consortium blockchain, network optimization and Software-Defined Networking.

**Weifeng Chen** graduated with a bachelor's degree in Mechanical Design, Manufacturing, and Automation from Xidian University of China in 2021. Currently, he is pursuing further studies in the Electronic Information program at the School of Computer Science and Technology, Xidian University.