Maximum AoI Minimization for Target Monitoring in Battery-free Wireless Sensor Networks

Bingkun Yao, Hong Gao, Yang Zhang, Jinbao Wang and Jianzhong Li

Abstract—Age of Information (AoI) has been proposed to measure the freshness of the sensory data for IoT applications. In Battery-free WSNs (BF-WSNs), The AoI minimization data collection problem has been extensively studied. Apart from data collection, target monitoring is also an important application for BF-WSNs. To capture the freshness of the sensory data, the AoI related to targets (The AoI of targets) needs to be considered. To guarantee the performance of the target monitoring system, the maximum AoI of all targets should be minimized. However, existing works mainly investigated the AoI related to nodes (The AoI of nodes), which is different from the AoI of targets. Also, existing energy models are not practical enough for battery-free nodes. To deal with those problems, in this paper, we first propose a more practical energy model. Then the problem of Maximum AoI minimization for Target monitoring in Battery-free WSNs (MTB) is formally defined based on the proposed energy model. A two-stage algorithm is proposed to solve MTB optimally, in which all nodes in the network are scheduled collaboratively to monitor all the targets in the monitoring field. Extensive simulations and real-world experiments verify the high performance of our algorithm and energy model.

Index Terms—Battery-free wireless sensor networks (BF-WSNs), target monitoring, scheduling algorithms, age of information (AoI).

1 Introduction

In recent years, Battery-free Wireless Sensor Networks (BF-WSNs) have been proposed to tackle the limited lifetime of the WSNs. Nodes in BF-WSNs are capable of harvesting energy from the ambient environment (e.g., solar energy [1], RF energy [2] and kinetic energy [3]). Therefore, the lifetime of BF-WSN is greatly extended. Although BF-WSNs can gather energy from the surrounding environment, the ambient energy is usually not sufficient (e.g., Cloudy days for solar-powered nodes). In case of energy shortage, a node has to spend much time to get recharged before it can be scheduled to work. Thus, the energy constraint must be considered carefully in the algorithm design for BF-WSNs.

For many IoT applications, the freshness of the sensory data is very important. In recent years, Age of Information (AoI) [4] has been proposed as a new metric to measure the freshness of the sensory data in WSNs, which has attracted much attention. To guarantee the freshness of the sensory data, the AoI value should be minimized. BF-WSNs are important components for IoTs. In BF-WSNs, the problem related to AoI minimization has been studied extensively [6]- [17]. All of them considered the problem of AoI minimization data collection, in which they tackled the AoI of the sensory data related to each node. Basically, for each node n in the network, at a given time i, let $Last\_a(n, i)$ denote the most recent time that n takes samples and sends the sensed information towards the sink. Then the AoI of the sensory data related to node n at the time i is i-$Last\_a(n, i)$. In the rest of the paper, the AoI of the sensory data related to node n is referred to as ***the AoI of node n*** for simplicity.

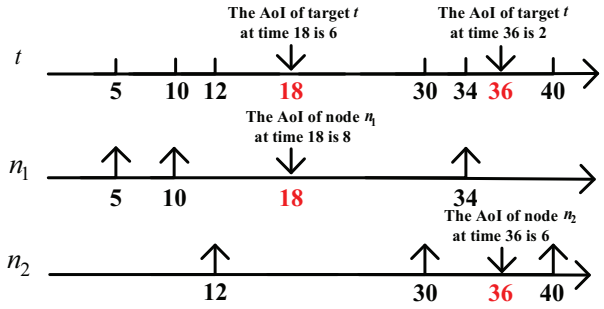


Fig. 1: An example for the AoI of the sensory data related to targets, in which target t is covered by two nodes n_1 and n_2.

Apart from data collection, target monitoring is also an important application of BF-WSNs, in which each target in the monitoring field might be covered by multiple nodes. In this scenario, to capture the freshness of sensory data, we need to consider the AoI of the sensory data related to each target. For a target t, assume that t could be covered by several nodes, whose set is denoted by $Cover(t)$. At any given time i, for a node $n \in Cover(t)$, let $Last(n, t, i)$ denote the most recent time when n takes the sample of t and transmits the sensed data of t to the sink. Therefore, at the time i, $Max\{Last(n, t, i)|n \in Cover(t)\}$ is the most recent time that t is sensed and the sensed data of t is updated. Thus, the AoI of the sensory data related to target t at the time i is i - $Max\{Last(n, t, i)|n \in Cover(t)\}$. See the example shown in Fig. 1. Fig. 1 shows the timeline of a target monitoring system, in which a target t is covered by two nodes $Cover(t)$

- *B. Yao, H. Gao and J. Wang are with the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150000, China (e-mail: bingkunyao@stu.hit.edu.cn, {honggao, wangjinbao}@hit.edu.cn).*
- *Y. Zhang is with the Department of Computer Science and Technology, Harbin Institute of Technology, and also with the Key Laboratory of Mechatronics, Heilongjiang University, Harbin 150000, China (e-mail: zhang_yang@hlju.edu.cn).*
- *J. Li is with the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150000, China, and also with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, China (e-mail: lijzh@hit.edu.cn).*

Manuscript received April 19, 2005; revised August 26, 2015.

= $\{n_1, n_2\}$. The sensory data of t is sampled and updated by n_1 at times 5, 10, 34, and n_2 at times 12, 30, 40 respectively. At time 18, the most recent update by n_1 and n_2 is at time 10 and 12 respectively so that $Last(n_1, t, 18) = 10$ and $Last(n_2, t, 18) = 12$. Thus, the AoI of the sensory data of t at time 18 is 18 - $Max\{10, 12\} = 6$. Similarly, at time 36, $Last(n_1, t, 36) = 34$ and $Last(n_2, t, 36) = 30$. Hence, the AoI of the sensory data of t at time 36 is 2. In the rest of this paper, for simplicity, we refer the AoI of the sensory data related to a target t as ***the AoI of target t***. Besides, Fig.1 also shows the difference between the AoI of the target and the AoI of the node. For example, at time 18, the most recent time that node n_1 takes samples and updates the sensory data is 10. Thus, the AoI of node n_1 at time 18 is 18-10 = 8, which is different from the AoI of target t. This is also true for the AoI of node n_2 at time 36. It is not hard to see, the AoI of the node can only represent the age of the sensory data captured by a single node while the AoI of the target stands for the age of the sensory data that is jointly captured by multiple nodes.

In BF-WSNs, the AoI of targets has not been investigated yet. All existing works [6]- [17] studied the AoI of the node rather than the AoI of the target. However, as mentioned above, the AoI of the node is quite different from the AoI of the target. Thus, they are not fit for the problem of the AoI of the target. In the scenario of target monitoring, for each target in the monitoring field, its AoI is determined by all nodes that can cover it. Also, nodes in the monitoring field differ in their energy harvesting capability. In this situation, how to schedule multiple battery-free nodes cooperatively to optimize the AoI of all targets in the monitoring field while satisfying the energy constraints, is still an open issue.

Obviously, during the whole working duration of a target monitoring system using BF-WSNs, to capture the interested event in real-time, the maximum AoI of all targets should be as small as possible. Otherwise, the AoI of some targets is likely to be too large during some time periods. As a result, some events of interest may not be captured [5]. Theoretically, the maximum AoI of all targets indicates a theoretical guarantee on event capture. According to [18], assume that each event of interest that occurs at each target lasts for at least n units of time. If the maximum AoI of all targets is m units of time, each target is sensed for at last every m time-units. Then for each interested event, the possibility of capturing it is at least n/m. Thus, **the maximum AoI of all targets** should be minimized to guarantee the performance of the target monitoring system.

To tackle the above issues, in this paper, we investigate the problem of **M**aximum AoI Minimization problem for **T**arget monitoring in **B**F-WSNs (**MTB** for short), in which an algorithm is proposed to find a monitoring schedule for all nodes in the network to minimize the maximum AoI of all targets.

In general, when dealing with the problem of algorithm and protocol design (e.g., Data collection, routing, in-network processing, etc.) for energy-harvesting WSNs, researchers need to assume an energy model for energy-harvesting nodes in their problem definitions. However, the assumed energy models in existing works are usually not practical enough for battery-free sensor nodes. Firstly, most of the works including [6]- [17] evaluated their proposed protocols by simulation. In their assumed energy model, usually, the energy consumption of a certain operation is fixed (e.g., The energy consumed by sending a packet is always 1 in [6]- [17]). This is not true on real-world testbeds. If this kind of model is applied to real-world nodes, the node's energy level may not be estimated correctly, which may cause node death. Secondly, some works have their proposed protocols implemented on real-world testbeds, which can be classified into two types. (a) Some existing works are based on rechargeable batteries or large capacitors (In the size of F). However, battery-free nodes usually apply small capacitors (In the size of mF and μF) as the energy storage. (b) Other works on battery-free nodes with small capacitors usually set two threshold voltages for the node. Once the voltage increases to the higher threshold, the node will become active, and if the voltage decreases to the lower threshold, the node will switch to sleep mode. Under this model, we do not know how much work the node can do when the node reaches an arbitrary voltage. Also, this model indicates a fixed working schedule for each node. Therefore, in order to better implement our algorithm for MTB, a more practical energy model for battery-free nodes is highly required. In this paper, we propose a more practical energy model for battery-free nodes with small capacitors, which can avoid node death while supporting more flexible working scheduling on the node. The proposed energy model is not limited to the problem of MTB and can be used for any timeslot-based algorithm and protocol design problems in BF-WSNs.

To summarize, the main contributions of this paper are as follows.

1) We propose a practical energy model for the battery-free node with small capacitors with no need of knowing any detailed electrical characteristics of the testbed. This model can be applied for any timeslot-based algorithm and protocol design problems in BF-WSNs.

2) Based on the proposed energy model, the problem of MTB is formally defined, in which the AoI of the target in target monitoring of BF-WSNs is firstly investigated.

3) We propose a pseudo-polynomial algorithm to solve the MTB problem optimally. The algorithm consists of two phases. First, an upper bound of the optimal solution is calculated by a dynamic programming method, in which each target is assigned to a sensor node and each node is scheduled to monitor its assigned targets in a round-robin manner. Second, a binary-search-based method is applied to find the optimal solution of the MTB. The complexity of the algorithm is analyzed.

4) Extensive simulations based on read-world energy data are carried out. Further, we conduct empirical studies on real-world testbeds. The result shows that our energy model and scheduling algorithm have high performance.

The remainder of the article is as follows. Section 2 surveys the related work. Section 3 presents the system model. Section 4 defines the problem of MTB formally. Section 5 describes the algorithm for MTB and its theoretical analysis in detail. Section 6 discusses some issues of our energy model and scheduling algorithm. Section 7 and 8 evaluate the performance of our protocol by simulation and real-world experiment respectively. Section 9 concludes the paper and introduces the future work.

2 Related Works

AoI is a hot issue in recent years and there have been some works on AoI minimization for battery-powered WSNs [5], [33]- [38], which can be classified into two categories. The first category [34]- [38] studied AoI minimization under various network environments. The second [5], [33] optimized other metrics under AoI constraints. The above works mainly focused on WSNs with non-renewable batteries, and none of them considers the energy constraints of BF-WSNs. Therefore, they can not solve the problem of AoI minimization in BF-WSNs.

In recent years, the problem of (maximum) AoI minimization has been studied extensively in BF-WSNs [6]- [17]. They all focused on the data collection problem, in which the AoI of the sensory data related to each node is investigated. [6]- [17] can be divided into two categories. The first category [6]- [15] investigated time-averaged or the maximum AoI minimization. Among them, [11] and [12] tried to minimize the time-average AoI of the sensory data related to a battery-free sensor that continuously senses the environment and transmits the sensed information to the sink. [6], [7] and [10] focused on the scenario in which a battery-free sensor monitors a stochastic process and the energy units arrive according to Poisson processes. Compared with [11] and [12], their AoI metric also takes the state change of the monitored process into consideration. To deal with transmission failure, [8] utilized partial information from unsuccessful transmissions to help decoding at the receiver by a reinforcement learning method. [9] proposed several online algorithms to minimize the time-average AoI of the sensory data for an energy harvesting sensor with finite batteries. [13] tried to minimize the time-average AoI related to the secondary user that is an energy harvesting sensor in a cognitive network. [14] focused on a two-hop network, in which both the source node and relay node are powered by ambient energy. [15] proposed both offline and online algorithms to minimize the average maximum AoI of the sensory data of all nodes in a one-hop network, with the restriction that only one node can transmit to the sink in one timeslot. The second category [16], [17] explored the AoI-Reliability tradeoff for an energy-harvesting node. The above works all studied the AoI of nodes. However, as mentioned in Section 1, since a target is usually covered by several nodes in the target monitoring scenario, the AoI of nodes is different from the AoI of targets. Thus, existing works on (maximum) AoI minimization is not suitable for the problem of MTB. Existing works can be used to solve a special case of MTB, in which the number of nodes equals the number of targets and the coverage relationship is a fixed one-to-one matching between nodes and targets.

Now there have been many works on algorithm and protocol design for energy-harvesting WSNs. Their assumed energy models can be classified into two types. Firstly, most of the existing works, including [6]- [17], evaluated the proposed algorithms by simulation. In their assumed energy model, usually, the energy consumption of a certain operation is fixed. This is not true on real-world testbeds. For example, [6]- [17] assumed that the energy cost of sending a packet is always 1. However, when the actual capacitor voltage of the node is high and the charging voltage is relatively low, the actual energy cost is higher than 1. In this situation, the energy level of the node is over-estimated, so the node may be assigned with excessive workloads, which may lead to node death. Secondly, some works have implemented their protocols on real-world testbeds, which consists of two categories: (a) The first category (e.g., [21]- [24]) is based on rechargeable batteries or large capacitors (In the size of F). However, battery-free nodes usually apply small capacitors (In the size of μF and mF) to store the harvested energy. The electrical characteristics are different between batteries/large capacitors and small capacitors. (b) The second category (e.g., [25]- [28]) that is implemented on small-capacitor nodes usually sets two threshold voltages for the testbed. Once the voltage increases to the higher threshold, the node will become active, and if the voltage decrease to the lower threshold, the node will switch to the sleep mode. Nevertheless, this type of energy model can not describe the energy level of the node (Namely, we do not know how much work the node can do when the node reaches an arbitrary voltage). Also, it is hard to compute a flexible working schedule under this model since this model indicates a fixed working schedule for each node.

3 System Model

In this section, we describe the network model and the energy model. In particular, for any timeslot-based algorithm and protocol design problems in BF-WSNs, we propose a practical and realistic energy model for a battery-free node with a small capacitor as energy storage.

3.1 Network Model

We assume that there are multiple battery-free nodes $N = \{n_1, n_2, ...\}$ and targets $T = \{t_1, t_2, ...\}$ in the monitoring field. We use mapping $\Gamma : N \rightarrow 2^T/\emptyset$ to represent the relationship between each node n and its covering targets $\Gamma(n)$. Correspondingly, for each target $t \in T$, we use $Cover(t)$ to denote the set of nodes that can cover t. The whole working duration is divided into discrete timeslots. The length of a timeslot is fixed and the same for each node. At each timeslot, node n can either: (a) be ***active*** if n takes samples of a target $t \in \Gamma(n)$ and sends t's monitoring information to the sink through one-hop transmission.[1] or (b) be ***idle*** if n shutdowns sensing and communication modules and stay idle. All nodes share a common channel.

3.2 Energy Model

As we have mentioned in Section 2, for nodes with small capacitors as the energy storage, existing energy models have two drawbacks: (a) Can not describe the energy level of the node. (b) Do not support flexible working scheduling on the node.

Apart from the above issues, we also characterize the challenge of designing an energy model for battery-free testbeds with small capacitors and short working timeslots (In the size of milliseconds to seconds). In general, on a small-capacitor node, the amount of energy consumed by

1. In our system, to ensure the successful reception of the sensed data, each packet is transmitted multiple times.

a certain operation (e.g., send a packet with 10 bytes) is related to the following factors: (a) The operation itself; (b) The charging voltage and the voltage of the capacitor when the operation performs; (c) The electrical characteristic of the node. (d) Other simultaneous operations. Among all factors mentioned above, Factors (b) and (d) are time-varying. Factor (c) may differ between nodes (e.g., To control the manufacturing cost, the uniformity of solar panels equipped on different nodes may not be so good). Thus, it is difficult to measure the actual energy cost of a certain operation accurately. Also, usually, there are multiple simultaneous operations in one timeslot (e.g., Sensing the environment, sending a packet, and monitoring the voltage of the capacitor). For the reason of Factor (d), it is very hard to compute the total energy consumption in one active slot accurately.

To tackle the above issues, in this subsection, we propose a novel energy model for battery-free nodes equipped with small capacitors. Notice that our model is not specially designed for the problem of MTB and can be used for any timeslot-based algorithm and protocol design problems in BF-WSNs. We assume that the ambient energy source remains stable for some time. [2] Also, the ambient energy is enough for a node to always stay idle but not enough to always keep active. This subsection consists of two parts. Firstly, we introduce some properties for battery-free nodes. Secondly, based on these properties, our energy model is proposed.

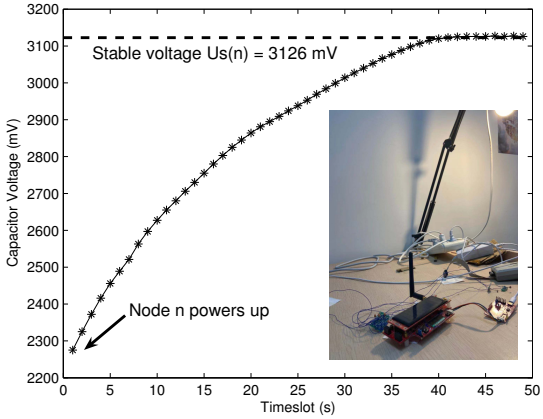


Fig. 2: The charging process of a solar-powered battery-free IEA node n under the light condition of 458 Lux.

3.2.1 Some Properties of Battery-free Nodes

Before we describe our energy model in detail, we introduce some properties on the charging process and discharging process of a battery-free node. For clarity, these properties are illustrated by our own-developed battery-free testbed IEA [19] shown in Fig. 2. In our example, IEA is powered by ambient light and a commercial LED lamp [20] is placed 0.8 meters above the node as the energy source (The light intensity at the node is 458 Lux). IEA is equipped with a 1mF capacitor as energy storage. If the node is active in one timeslot, it will send four 40-byte packets. Mention that our analysis is not restricted to solar-powered nodes. Let the node in Fig. 4 be denoted by n. In the following, we discuss the charging and discharging process of n respectively.

First, consider the charging process of node n. As shown by the charging curve in Fig. 2, after n powers up, if it remains idle, the voltage of its capacitor [3] will increase monotonously for a certain period of time. Let $C(n)$ denote the length of this period (in timeslots). As shown in Fig. 1, $C(n)$ is about 43. After $C(n)$ idle slots, the voltage of n holds steady. This is because after $C(n)$ timeslots, the capacitor voltage will be equal to the charging voltage and both voltages will keep unchanged. We refer to this voltage as the ***stable voltage*** of n, which is denoted as $U_s(n)$. For the example shown in Fig. 2, $U_s(n)$ = 3126 mV. Apparently, $U_s(n)$ is the highest voltage of n. Further, let $U_c(n, i)$ denote the voltage of n at the beginning of the i-th ($1 \leq i \leq C(n)$) timeslot after n powers up. By Observation, we have the following property on the charging process of n:

Property 1. *For a battery-free node n, if n is powered by a stable ambient energy source, $C(n)$, $U_s(n)$, and $U_c(n, i)$ for $1 \leq i \leq C(n)$ can be regarded as fixed values.*

By Property 1, we can see that the charging process of a node n is roughly fixed when n is powered by a stable ambient energy source.

Secondly, we focus on the discharging process of n, as shown in Fig. 3. Since the ambient energy is not enough for n to always remain active, if n is active continually, the voltage of n will decrease monotonously. Now assume that n switches to active when its voltage is u and let $U_d(n, u, i)$ denote the voltage of n after n is active for continuous i timeslots. For example, as the red arrows shown in Fig. 3, n switches to active when its voltage reaches to $U_s(n)$ = 3126 mV and we have $U_d(n, U_s(n), 1)$ = 2922 mV, $U_d(n, U_s(n), 2)$ = 2669 mV and $U_d(n, U_s(n), 3)$ = 2431 mV, respectively. In practice, the node may die due to energy overuse. To avoid the death of nodes, we set a *threshold voltage* U_{th}, which means the capacitor voltage of each node should not be lower than U_{th}. For example, in Fig. 3 we set U_{th} = 2250 mV. Under the restriction of U_{th}, let $Df(n, u)$ denote the maximum i such that $U_d(n, u, i) \geq U_{th}$, which means that if n switches to active when its capacitor voltage is u, n can be active for at most $Df(n, u)$ continuous timeslots. As shown by the red arrows in Fig. 3, if n switches to active when its voltage is $U_s(n)$, after 4 consecutive active slots, n's voltage will drop below U_{th}. [4] Thus, we have $Df(n, U_s(n))$ = 3. By Observation, we have the following properties on the discharging process of n:

Property 2. *For a certain battery-free node n, given a threshold voltage U_{th}, if n is powered by a stable ambient energy source, for each voltage u such that $U_{th} < u \leq U_s(n)$, $Df(n, u)$ and $U_d(n, u, i)$ for $1 \leq i \leq Df(n, u)$ can be regarded as fixed values.*

Property 3. *For a certain battery-free node n, given a threshold voltage U_{th}, if n is powered by a stable ambient energy source, for any $u_1 \geq u_2 > U_{th}$, we have (1) $Df(n, u_1) \geq Df(n, u_2)$; (2) Given integer i, $U_d(n, u_1, i) \geq U_d(n, u_2, i)$.*

2. This assumption holds for many circumstances. e.g., For a solar-powered node deployed in the corridors of a large building, the ambient light remains relatively stable for the whole night.

3. In the rest of the paper, "the capacitor voltage of node n" and "the voltage of node n" are used interchangeably.

4. In practice, n may die directly due to low voltage.

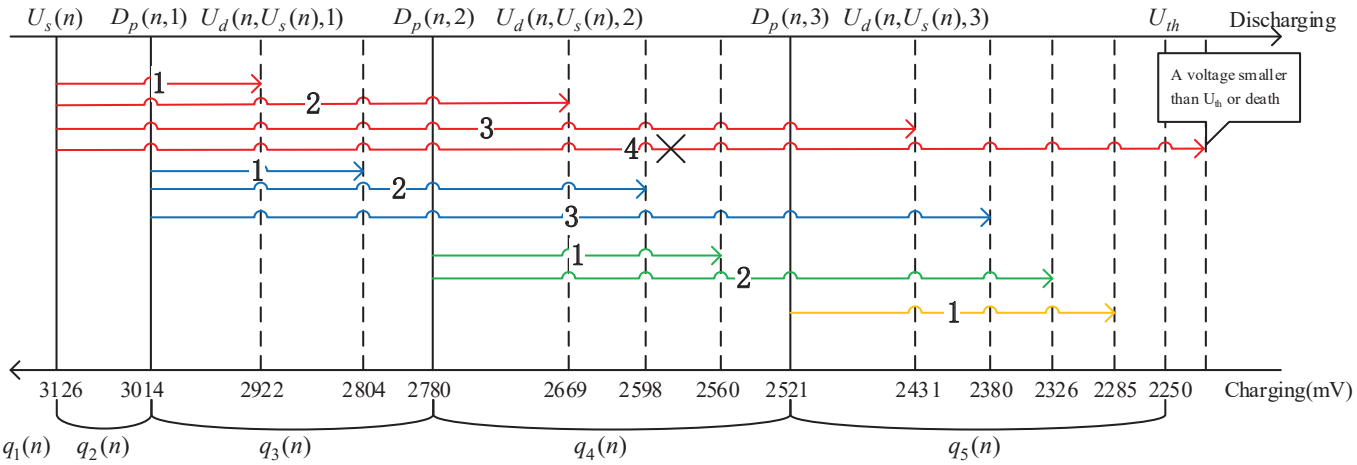


Fig. 3: The discharging process of n. The energy levels of n are represented by states set $Q(n) = \{q_1(n), q_2(n), q_3(n), q_4(n), q_5(n)\}$. The threshold voltage U_{th} = 2250 mV, $U_s(n)$ = 3126 mV and $Df(n, U_s(n)) = 3$.

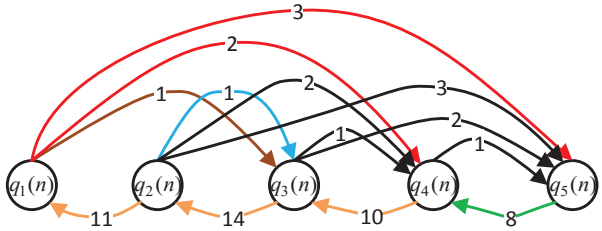


Fig. 4: The energy model of n is formulated as a state transition graph $EN(n) = (Q(n), \Delta(n))$.

Like Property 1, Property 2 indicates that the discharging process of a node n is roughly fixed when n is powered by a stable ambient energy source.

3.2.2 The Proposed Energy Model

Based on the properties mentioned in Section 3.2.1, now we are ready to propose our energy model. The energy model of n is formulated as a discrete state transition diagram $EN(n) = (Q(n), \Delta(n))$, whose meaning is explained as follows:

1) Each state in states set $Q(n) = \{q_1(n), q_2(n), ...q_{|Q(n)|}(n)\}$ represents an energy level of node n.

2) The set of state transition edges $\Delta(n)$ represents the action that n can take and the corresponding change on n's energy level. In detail, each edge δ in $\Delta(n)$ is defined as a quaternion $\delta = (q_1, q_2, action, i)$ in which $q_1, q_2 \in Q(n)$, $action \in \{0, 1\}$ and $i > 0$. $\delta = (q_1, q_2, action, i)$ means that when n's state is q_1, if n is active for consecutive i idle ($action$ = 0) or active ($action$ = 1) slots, n's state will change to q_2. If $action$ = 0 (1), we say that δ **represents the charging (discharging) process of** n.

$EN(n)$ is constructed in the following three steps. In the first step, we derive the state set $Q(n)$ by dividing the voltage interval $[U_{th}, U_s(n)]$ into some subintervals. In the second and the third step, $\Delta(n)$ is computed according to the discharging and charging process of n analyzed in Section 3.2.1. These three steps are illustrated by constructing $EN(n)$ shown in Fig. 4, which follows the charging and discharging process shown in Fig. 2 and Fig. 3.

Step 1 of constructing $EN(n)$: In this step, we compute $Q(n)$. Firstly, the first state of $Q(n)$, $q_1(n)$ denotes the highest voltage of n. As n's voltage is at most $U_s(n)$, state $q_1(n)$ represents the voltage of $U_s(n)$. Secondly, the voltage interval $[U_{th}, U_s(n)]$ can be divided into some subintervals, in which each voltage subinterval is denoted by a state in $Q(n)$. By Property 3, n can be active continuously for the most number of timeslots when n's voltage reaches $U_s(n)$. Thus, a natural idea is to divide voltage interval $[U_{th}, U_s(n)]$ into $Df(n, U_s(n))+1$ subintervals. For $1 \le i \le Df(n, U_s(n))+1$, the i-th subinterval is denoted by $q_{i+1}(n) \in Q(n)$. To compute these subintervals, we need to find $Df(n, U_s(n))$ dividing points in $[U_{th}, U_s(n)]$. We use $D_p(n, i)$ to denote the i-th dividing point. Mention that $D_p(n, 1) > D_p(n, 2) > ... > D_p(n, Df(n, U_s(n)))$, $D_p(n, 0) = U_s(n)$ and $D_p(n, Df(n, U_s(n))+1) = U_{th}$. Hence, the i-th subinterval is $[D_p(n, i), D_p(n, i\text{-}1))$.

Now we discuss how to find these dividing points. According to Property 3, for a voltage u such that $U_d(n, U_s(n), i) \le u \le U_d(n, U_s(n), i\text{-}1)$, we have $Df(n, U_s(n))\text{-}i \le Df(n, u) \le Df(n, U_s(n))\text{-}i+1$. Thus, we pick a voltage between $U_d(n, U_s(n), i)$ and $U_d(n, U_s(n), i\text{-}1)$ as the i-th dividing point $D_p(n, i)$. In detail, $D_p(n, i)$ is computed as:

1) Given a voltage u that $U_{th} \le u \le U_s(n)$, let $R(u)$ denote the first slot that n reaches voltage u after n powers up, namely, $R(u) = \underset{1 \le i \le C(n)}{argMin} \{U_c(n, i) \ge u\}$.

2) $D_p(n, i) = U_c(n, p)$, where $p = \lceil \alpha \times R(U_d(n, U_s(n), i-1)) + (1-\alpha) \times R(U_d(n, U_s(n), i)) \rceil$. α is a constant parameter that belongs to [0, 1].

Now we illustrate the computation of $Q(n)$ by the example in Fig. 2, 3 and 4. Firstly, n's stable voltage $U_s(n)$ = 3126 mV so that $q_1(n)$ represents the voltage of 3126 mV. Then, as $Df(n, U_s(n)) = 3$, the interval [2250mV, 3126mV] is divided into 4 subintervals by 3 dividing points. Take the first dividing point $D_p(n, 1)$ as an example. As shown in Fig. 3, $D_p(n, 1)$ is picked between voltage $U_d(n, U_s(n), 1)$ = 2922 mV and $U_d(n, U_s(n), 0) = U_s(n)$ = 3126 mV. According to the charging curve in Fig. 2, R(2922 mV) = 26 and R(3126 mV) = 43. Let α = 0.33. Since $\lceil 43 * 0.33 + 26 * (1 - 0.33) \rceil$ = 32, $U_c(n,32)$ = 3014 mV is picked as $D_p(n, 1)$. Similarly, as shown in Fig. 3, we have $D_p(n, 2)$ = 2780 mV and $D_p(n, 3)$ = 2521 mV. The first, second, and third subintervals are [3014mV, 3126mV), [2780mV, 3014mV), [2521mV, 2780mV)

and they are denoted by states $q_2(n)$, $q_3(n)$, $q_4(n)$ respectively in Fig. 3 and 4. In the end, $q_5(n)$ represents the last voltage subinterval [2250mV, 2521mV).

Step 2 of constructing *EN(n)*: In this step, edges in $\Delta(n)$ that represent the discharging process of n are derived. Firstly, we focus on the discharging process of n starting from state $q_1(n)$. From Step 1 above we can see that apart from $q_1(n)$, each state stands for a subinterval of voltage. Starting from $q_1(n)$, after n is active for i consecutive slots, n's voltage will drop to $U_d(n, U_s(n), i)$, the state of n will change to the q such that $U_d(n, U_s(n), i)$ is within the interval that q represents. For example, as Fig. 3 shows, $U_d(n, U_s(n), 1)$ = 2922 mV $\in$ [2780mV, 3014mV) represented by $q_3(n)$. Therefore, we use an edge ($q_1(n)$, $q_3(n)$, 1, 1) to represent this state change of n. This edge is shown as the brown edge in Fig. 4, in which the number of continuous active slots is marked on the edge. Similarly, as shown by red edges in Fig. 4, edges ($q_1(n)$, $q_4(n)$, 1, 2) and ($q_1(n)$, $q_5(n)$, 1, 3) are also added into $\Delta(n)$.

Secondly, we study the discharging process that starts from other states except for $q_1(n)$. Consider a state $q_i(n)$ where $i \geq 2$ and $q_i(n)$ stands for voltage subinterval [a, b). When n is in state $q_i(n)$, the voltage of n is at least a. When n's voltage reaches a, for $1 \leq j \leq Df(n, a)$, after j continuous active timeslots, n's voltage will drop to $U_d(n, a, j)$. Assume that voltage $U_d(n, a, j)$ belongs to the voltage subinterval represented by state $q_k(n)$. Correspondingly, edge ($q_i(n)$, $q_k(n)$, 1, j) is added into $\Delta(n)$. For example, consider the discharging process starting from $q_2(n)$ as shown by blue edges in Fig. 3. The minimum voltage when n is in $q_2(n)$ is 3014 mV. Also, $U_d(n, 3014\text{mV}, 1)$ = 2804 mV $\in$ [2780mV, 3014mV) that is denoted by state $q_3(n)$. Hence, we have an edge ($q_2(n)$, $q_3(n)$, 1, 1) $\in \Delta(n)$ (The blue edge in Fig. 4). Edges that represent the discharging process starting from $q_3(n)$ and $q_4(n)$ are derived similarly.

Finally, the last state in $Q(n)$ indicates the lowest energy level of n. In this state, n will not be scheduled to be active. Thus, as shown in Fig. 4, there are no edges that represent the discharging process starting from $q_5(n)$.

Step 3 of constructing *EN(n)*: In this step, edges that stand for the charging process of n are computed. According to Step 1, $|Q(n)| = Df(n, U_s(n))+2$. For each state $q_i(n)$ such that $2 \leq i \leq Df(n, U_s(n))+2$, let the voltage subinterval represented by $q_i(n)$ to be denoted by [a_i, b_i) and $a_1 = U_s(n)$. Now assume that n needs to be idle for consecutive $Rc(i)$ slots to recharge from voltage a_i to voltage a_{i-1}. It is not hard to see, n needs to be idle for at most $Rc(i)$ consecutive slots to change from state $q_i(n)$ to $q_{i-1}(n)$. Thus, an edge ($q_i(n)$, $q_{i-1}(n)$, 0, $Rc(i)$) is added into $\Delta(n)$ to denote this state change. For example, as in Fig. 4, $q_5(n)$ and $q_4(n)$ denote the voltage interval [2250mV, 2521mV) and [2521mV, 2780mV) respectively. According to the charging curve in Fig. 2, continuous 8 idle slots are needed for n to recharge from 2250mV to 2521mV. Hence, edge ($q_5(n)$, $q_4(n)$, 0, 8) is added into $\Delta(n)$ (The green edge in Fig. 4). Edges that denote the transition from $q_4(n)$ to $q_3(n)$, $q_3(n)$ to $q_2(n)$, $q_2(n)$ to $q_1(n)$ are derived similarly (3 orange edges in Fig. 4).

After the above three steps, the energy model of a node n, $EN(n)$ = ($Q(n)$, $\Delta(n)$) is derived. As mentioned by Step 1 and 3, to compute $Q(n)$ and the edges that represent the charging process of n, we need the charging curve of n. To compute the edges that stand for the discharging process, we need to record the discharging process starting from each state. Thus, to compute $EN(n)$, n needs to recharge and discharge for $Df(n, U_s(n))+1$ times, which takes $O(Df(n, U_s(n))C(n))$-time. Algorithm 1 summarizes the procedure of computing $EN(n)$.

The path on *EN(n)*: Now we introduce an important concept on $EN(n)$, namely, the meaning of the ***path*** on $EN(n)$. We describe this by an example. Consider a path $P = q_2(n) \rightarrow q_4(n) \rightarrow q_3(n)$ on $EN(n)$ shown in Fig. 4. As illustrated in Fig. 4, from $q_2(n)$ to $q_4(n)$, n is active for 2 timeslots firstly. Then n is idle for 10 slots to change from $q_4(n)$ to $q_3(n)$. It is not hard to see, path P indicates an idle/active schedule of n whose length is 12 slots, in which n is active for slot 1-2 and idle for slot 3-12. Correspondingly, in a time period whose length is no smaller than 12 slots, if n is scheduled to be active for the slot 1-2 and idle for slot 3-12, we say that ***in this time period, n assigns its active/idle slots according to the path P***. The ***length*** of a path is defined as the number of timeslots included in it. Hence, the length of the path P is 2+10 = 12, which is denoted by $|P| = 12$. We also say that ***P contains* 2 active slots and 10 idle slots**. Further, a path is also called a ***ring path*** if its starting state and ending state are the same.

Theorem 1. *If n assigns its active/idle slots according to a path on EN(n), n's capacitor voltage will not be lower than U_{th}.*

Proof. This can be verified easily by Property 1 and 2 and we omit the proof here. □

Algorithm 1 Compute the energy model of a node n

Input: The threshold voltage U_{th};
Output: The energy model of n $EN(n)$=($Q(n)$, $\Delta(n)$).
1: Remain idle until the voltage reach stable. Record $U_s(n)$, $C(n)$ and $U_c(n, i)$ for each $1 \leq i \leq C(n)$;
2: Being active continuously until the voltage dropped below U_{th} or death. Record $Df(n, U_s(n))$, $U_d(n, U_s(n), i)$ for $1 \leq i \leq Df(n, U_s(n))$;
3: Compute $Q(n)$ and edges that outcoming from $q_1(n)$;
4: **for** $1 \leq i \leq Df(n, U_s(n))$ **do**
5: Remain idle until the voltage reach $D_p(n, i)$;
6: Being active continuously until the voltage dropped below U_{th} or death. Record $Df(n, D_p(n, i))$, $U_d(n, D_p(n, i), j)$ such that $1 \leq j \leq Df(n, D_p(n, i))$.
7: Compute edges outcoming from state $q_{i+1}(n)$;
8: **end for**

4 Problem Definition

In this part, the problem of MTB is formalized based on the system model above. For all the nodes in N, we aim to find a monitoring schedule, which is denoted by a mapping $\theta : (n, i) \rightarrow \Gamma(n) \cup \{0\}$ where $n \in N$, $i \in \mathbb{Z}^+$. This means that if $\theta(n, i) = t \in \Gamma(n)$, at the timeslot i, node n is scheduled to take samples of target t and send the sampled data of t to the sink. Otherwise, if $\theta(n, i) = 0$, node n will be idle at timeslot i. Under a working schedule θ, for each target $t \in T$, at any timeslot i, for each node $n \in Cover(t)$, let $Last(n, t, i)$ denote the most recent time when n takes the sample of t and transmits the sensory data of t to the sink, namely, $Last(n, t, i) = Max\{j | \theta(n, j) = t \ \& \ j \leq i\}$. Obviously, at timeslot i, the most recent time when the sensory data of target t

is sampled and updated is $Max\{Last(n,t,i)|n \in Cover(t)\}$. Therefore, the age of information (AoI) of t at the i-th slot is defined as:

$$A(t,i) = i - Max\{Last(n,t,i)|n \in Cover(t)\} \quad (1)$$

As we have no restriction on the length of the whole working duration, for each node $n \in N$, the size of its monitoring schedule $\theta(n,i)$ is unbounded, which may result in a mass of control messages. We solve this problem as follows. Consider a monitoring schedule in a time period with the length of L timeslots, during which the sensory data of each target is sampled and updated for at last once. Also, the monitoring schedule of these L slots is repeated in the whole working duration. In this way, the size of the monitoring schedule of each node can be within $O(L)$ and the maximum AoI of all targets is at most L. As we aim to find the monitoring schedule to minimize the maximum AoI of all targets, we will find a minimum L and derive the monitoring schedule of each node within the time period of L timeslots.

Further, to prevent the node from running out of all the energy (death), the voltage of each node should not be lower than the threshold voltage U_{th}. To this end, in a period of L slots, for each node $n \in N$, n should assign its active/idle slots according to a path on $EN(n)$ according to Theorem 1. Also, to ensure that the monitoring schedule in L slots can be repeated, the length of the path should be no longer than L, and the starting and ending state of the path should be the same.

To sum up, the problem of **M**aximum AoI minimization for **T**arget monitoring in **B**F-WSNs (**MTB** for short) is defined as follows:

Input:

1) The set of battery-free nodes N and the set of targets in the monitoring field T.

2) A mapping Γ: $N \rightarrow 2^T/\emptyset$ that indicates each node and the targets within its coverage range.

3) The energy model $EN(n)$ for each node $n \in N$.

Output: A positive integer L and a mapping $\theta : (n, i) \rightarrow \Gamma(n) \cup \{0\}$ for each $n \in N$ and $1 \leq i \leq L$, which satisfy the following conditions:

1) For each target $t \in T$, there exists $n \in N$ and $1 \leq i \leq L$ such that $\theta(n,i) = t$.

2) Each node $n \in N$ assigns its active/idle slots according to a ring path on $EN(n)$, and the length of the path is no larger than L.

3) L is minimized.

Condition (1) above means that the sensory data of each target is sampled and updated at least once in each time period of length L. Condition (2) ensures that the voltage of each node will never be lower than the threshold voltage, and the monitoring schedule in the period of length L can be repeated. Mention that for a positive integer l, if there exists a mapping $\theta : (n, i) \rightarrow \Gamma(n) \cup \{0\}$ that satisfies Condition (1) and (2) above, l is referred to as a ***feasible solution***.

Table 1 lists key symbols and notations in this article.

TABLE 1: Symbols and Notations

Notation	Description
$n = \{n_1, n_2, ...n_m\}$	The set of all battery-free nodes
$T = \{t_1, t_2, ...t_n\}$	The set of all monitored targets
$\Gamma(n)$	The set of targets that can be covered by node n
$Cover(t)$	The set nodes that can cover target t
$EN(n) = (Q(n), \Delta(n))$	The energy model of node n
$q_i(n)$	The i-th state in $Q(n)$
$(q_1, q_2, action, i)$	An edge in $\Delta(n)$
$U_s(n)$	The stable voltage of node n
$U_d(n, u, i)$	If n switches to active at voltage u and n is active for i consecutive slots, the voltage of n
U_{th}	The threshold voltage
$Df(n, u)$	If n switches to active at voltage u, the most number of consecutive slots that node n can be active for
$D_p(n, i)$	The i-th voltage dividing points of node n
$\theta(n, i)$	The target to be sampled and updated by node n at timeslot i
$A(t, i)$	The AoI of target t at i-th timeslot
$Ap(t)$	The node to which t is assigned in Phase 1
$F(n)$	The set of targets assigned to n in Phase 1
$P_l(n, i)$	The shortest ring path on $EN(n)$ that contains at least i active slots
$G(j) = (U(j),V(j),E(j))$	The bipartite graph used for judging whether j is a feasible solution
$M(j)$	A maximum matching of $G(j)$
$V(j)[n]$	Vertexes in $V(j)$ that represent all active slots of n in the period of length j
$P_s(n, j)$	The ring path with a maximum length of j on $EN(n)$ that contains the maximum number of active slots
L_{upper}	An upper bound of the feasible solution computed by Phase 1
L_{opt}	The minimum (optimal) feasible solution

5 THE ALGORITHM

In this subsection, we describe the proposed algorithm for the problem of MTB in detail. According to the definition of the problem, given an instance of MTB, for an arbitrary positive integer l, if l is a feasible solution, for each $j \geq l$, j is also a feasible solution. Otherwise, for each $k \leq l$, k is not a feasible solution. Therefore, the minimum feasible solution and the corresponding monitoring schedule can be computed in the following two phases: In the first phase, we compute an upper bound of the feasible solution denoted by L_{upper} by assigning each target to a node using a dynamic-programming method, in which each node is scheduled to sample and update the sensory data of its assigned target in a round-robin manner. In the second phase, we use a binary-search-based method to compute the minimum feasible solution $l \in [1, L_{upper}]$, while the monitoring schedule for each node is derived accordingly.

5.1 Phase 1: Compute an Upper Bound L_{upper}

In this phase, an upper bound of the feasible solution L_{upper} is computed. According to the problem definition, as L_{upper} is a feasible solution to the problem, there exists a mapping $\theta(n, i)$ that satisfies Condition (1) and (2). Since $1 \leq i \leq L_{upper}$, we only need to discuss the monitoring schedule from timeslot 1 to L_{upper}. Firstly, to satisfy Condition (1), from timeslot 1 to L_{upper}, nodes are scheduled to be active as follows:

1) For each target $t \in T$, t is assigned to a node in $Cover(t)$, which is denoted by $Ap(t)$. Accordingly, for each node $n \in N$, the set of targets assigned to n is denoted as $F(n)$.

2) During the period from the first slot to the L_{upper}-th slot, for each node $n \in N$, n is active for at least $|F(n)|$ slots,

and n samples and updates the sensory data of each target in $F(n)$ in a round-robin manner.

It is not hard to see, from the first slot to the L_{upper}-th slot, for each target t, the sensory data of t is sampled and updated at least once. Next, according to Condition (2), for each node n, during the period from the first slot to L_{upper}-th slot, n assigns its active/idle slots according to a ring path on its energy model $EN(n)$, and the ring path is no longer than L_{upper}. Let $P_1(n)$ denote this ring path of n. To meet Condition (1), $P_1(n)$ should contain at least $|F(n)|$ active timeslots. Also, $L_{upper} = Max\{|P_1(n)|\,|n \in N\}$ and the value of L_{upper} should be as small as possible. Therefore, in this phase, we need to compute $P_1(n)$ and $F(n)$ simultaneously for each $n \in N$ to minimize $L_{upper} = Max\{|P_1(n)|\,|n \in N\}$ satisfying the following conditions:

1) $P_1(n)$ is a ring path on $EN(n)$ and $P_1(n)$ contains at least $|F(n)|$ active timeslots.

2) $F(n) \subseteq \Gamma(n)$, $\bigcup_{n\in N} F(n) = T$ and $F(n_1) \cap F(n_2) = \emptyset$ for any $n_1, n_2 \in N$.

As shown by the subproblem above, for each node n, the assigned target set $F(n)$ and the ring path $P_1(n)$ need to be derived jointly. We notice that since $F(n) \subseteq \Gamma(n)$, $|F(n)| \leq |\Gamma(n)|$. Therefore, we can solve this problem in two stages. In the first stage, for each node $n \in N$ and $1 \leq i \leq |\Gamma(n)|$, we compute the ring path on $EN(n)$ with the minimum length that contains at least i active timeslots, which is denoted by $P_l(n, i)$. In the second stage, $F(n)$ is computed to make $L_{upper} = Max\{|P_l(n, |F(n)|)|\,, n \in N\}$ as small as possible, in which $P_1(n) = P_l(n, |F(n)|)$. Next, we discuss these two stages in detail.

Stage 1: In this stage, we compute $P_l(n, i)$ for each node $n \in N$ and $1 \leq i \leq |\Gamma(n)|$. The starting state of $P_l(n, i)$ could be any state in $Q(n)$. For a state $q \in Q(n)$, let $P_l(n, i, q)$ denote the shortest ring path on $EN(n)$ containing at least i active slots with q as both the starting and ending state. Obviously, the $P_l(n, i, q)$ ($q \in Q(n)$) with the minimum length is $P_l(n, i)$. Next, for each state $q \in Q(n)$, we describe the computation of $P_l(n, i, q)$ in the following two steps:

Step 1 of computing $P_l(n, i, q)$: First of all, we define $P_l(n, q_1, j, q_2)$ as the shortest path on $EN(n)$ satisfying the following: (a) The starting and ending state of are q_1 and q_2 respectively, in which $q_1, q_2 \in Q(n)$; (b) $P_l(n, q_1, j, q_2)$ contains at least j active timeslots.

Apparently, $P_l(n, i, q)$ is equal to $P_l(n, q, i, q)$. Now we discuss how to compute $P_l(n, q, i, q)$. For any $q_1, q_2 \in Q(n)$ and $j \geq 0$, let $Mt(n, q_1, j, q_2)$ denote the length of $P_l(n, q_1, j, q_2)$. Then we have the following recursive relation:

$$Mt(n, q_1, j, q_2) = Min\,\{Mr(n, q_1, j, q_3) \mid edge\ (q_3, q_2, action, m) \in \Delta(n)\}, \quad (2)$$

in which $Mr(n, q_1, j, q_3) = Mt(n, q_1, j\text{-}m, q_3) + m$ if $action = 1$ and $Mt(n, q_1, j, q_3) + m$ if $action = 0$.

Step 2 of computing $P_l(n, i, q)$: As indicated by Equation (2), $P_l(n, q, i, q)$ can be computed using a dynamic-programming method in a bottom-to-top manner. As (2) shows, each hop in $P_l(n, q, i, q)$ can be represented by a binary (j, q') in which $1 \leq j \leq i$ and $q' \in Q(n)$. For example, in the computation of $P_l(n, q, i, q)$, for $q_1 \in Q(n)$ and $j \leq i$, if we have $Mt(n, q, j, q_1) = Mt(n, q, j\text{-}1, q_2) + 1$, the last hop of (j, q_1) is marked as $(j\text{-}1, q_2)$, which is denoted as $Last\_hop(j, q_1) = (j\text{-}1, q_2)$. The initial conditions, i.e., $Mt(n, q, 0, q')$ for each $q' \in Q(n)$ can be computed by the shortest path algorithm, in which for any edge $\delta = (q_1, q_2, action, m) \in \Delta(n)$, the weight of δ is m. Mention that $P_l(n, q, 0, q')$ may contain some active slots. Thus, after the computation of $Mt(n, q, 0, q')$, if there are m ($m > 0$) active slots on $P_l(n, q, 0, q')$, we have $Mt(n, q, j, q') = Mt(n, q, 0, q')$ for $1 \leq j \leq m$.

Algorithm 2 Phase 1: Compute an upper bound L_{upper}

Input: The set of nodes N, the set of targets T, Γ: $N \rightarrow 2^T/\emptyset$, $EN(n) = (Q(n), \Delta(n))$ for each $n \in N$.
Output: An upper bound L_{upper} of the feasible solution.
1: $P_l(n, |\Gamma(n)|, q) \leftarrow Compute\_Pl(EN(n), |\Gamma(n)|, q)$ for each $n \in N$ and $q \in Q(n)$;
2: $P_l(n, i) \leftarrow \underset{P_l(n,i,q), q\in Q(n)}{argMin} |P_l(n, i, q)|$ for each $n \in N$ and $1 \leq i \leq |\Gamma(n)|$;
3: $Ap(t) \leftarrow \underset{n\in Cover(t)}{argMax} \mid P_l(n, |\Gamma(n)|)\mid$, $F(Ap(t)) \leftarrow F(Ap(t)) \cup \{t\}$ each target $t \in T$;
4: Change $Ap(t)$ of a target t to reduce L_{upper} the most until L_{upper} can not be reduced;
5: **function** $Compute\_Pl(EN(n) = (Q(n), \Delta(n)), i, q)$
6: Sort all states in $Q(n)$ in the ascending order of energy level;
7: $Mt(n, q, j, q') \leftarrow \infty$, $Last\_hop(j, q') \leftarrow null$ for $1 \leq j \leq i$ and each state $q' \in Q(n)$;
8: Compute $P_l(n, q, 0, q')$ by Dijkstra Algorithm for each $q' \in Q(n)$;
9: Update $Mt(n, q, j, q')$ and $Last\_hop(j, q')$ according to (2) for $1 \leq j \leq i$ and each state $q' \in Q(n)$;
10: **end function**

Function *Compute_Pl* in Algorithm 2 summarizes the computation of $P_l(n, i, q)$, which has a time complexity of $O(|Q(n)|i + |Q(n)|^3 log(|Q(n)|))$.

Theorem 2. *$P_l(n, i, q)$ can be derived correctly by function Compute_Pl in Algorithm 2.*

Proof. Since *Compute_Pl* is a dynamic-programming method, we only need to prove that the recursive relation (i.e., Equation (2)) is correct by contradiction. For this purpose, we can assume that there exists another path $P_l'(n, q_1, j, q_2)$ such that its length $Mt'(n, q_1, j, q_2) < Mt(n, q_1, j, q_2)$. For details, please see Appendix A in the supplement material [32]. □

In the above, we discuss how to compute $P_l(n, i, q)$ for a given n, i and state $q \in Q(n)$. It is not hard to see, in the computation of $P_l(n, i, q)$, for all $1 \leq j \leq i$, $P_l(n, j, q)$ is also derived. Thus, for all $n \in N$ and $1 \leq i \leq |\Gamma(n)|$, $P_l(n, i)$ is computed in the following two steps. Firstly, for each $n \in N$ and $q \in Q(n)$, compute $P_l(n, |\Gamma(n)|, q)$ by function *Compute_Pl* in Algorithm 2. Then, $P_l(n, i)$ equals the $P_l(n, i, q)$ with the minimum length.

Stage 2: In Stage 1, we have derived $P_l(n, i)$ ($n \in N$ and $1 \leq i \leq |\Gamma(n)|$). In this stage, each target is assigned to a node $Ap(t)$ that $Ap(t) \in Cover(t)$. We try to minimize $L_{upper} = Max\{|P_l(n, |F(n)|)|\,, n \in N\}$ by a simple greedy strategy in the following two steps: Firstly, assign each target to a node that can cover it; Secondly, change the assigned node of a target to reduce the value of L_{upper} until L_{upper} can not be reduced. The details for this stage are as follows:

1) To start with, for each $t \in T$, $Ap(t)$ is initialized as $\underset{n\in Cover(t)}{argMax} \mid P_l(n, |\Gamma(n)|)\mid$. Add t into $F(Ap(t))$.

2) For the node $n' = \underset{n\in N}{argMax} \mid P_l(n, |F(n)|)\mid$, select a $t' \in F(n')$, change $Ap(t')$ to reduce L_{upper} the most. Repeat this step until L_{upper} can not be reduced.

After the above two stages, the computation of L_{upper} is finished. Algorithm 2 summarizes the two stages.

5.2 Phase 2: Compute the Optimal Solution by Binary-search-based Method

In the last subsection, an upper bound L_{upper} of the feasible solution is derived. In this subsection, by binary searching between [1, L_{upper}], we compute the minimum feasible solution L_{opt} and the corresponding monitoring schedule of each node in the period from the first slot to the L_{opt}-th slot (i.e. $\theta(n, i)$ for each $n \in N$ and $1 \le i \le L_{opt}$). We only need to consider the following question: **For an arbitrary j such that $1 \le j \le L_{upper}$, whether j is a feasible solution? If j is a feasible solution, how to compute the corresponding monitoring schedule $\theta(n, i)$?** Next, we show that this problem is equal to a maximum match problem on a bipartite graph $G(j) = (U(j), V(j), E(j))$, which is constructed in the following three steps. This is illustrated by a simple example shown in Fig. 5, in which $N = \{n_1, n_2\}$, $T = \{t_1, t_2, t_3, t_4\}$, $\Gamma(n_1) = \{t_1, t_3, t_4\}$, $\Gamma(n_2) = \{t_2, t_4\}$ and $j = 12$.

Step 1 of constructing $G(j)$: The left part of bipartite graph G, $U(j)$ has $|N|$ vertexes, in which each vertex represents a target in T. Thus, we have $U(j) = T$, the vertex that stands for the target $t \in T$ is also denoted by t. For example, as Fig. 5 shows, the left part $U(12)$ equals to $T = \{t_1, t_2, t_3, t_4\}$.

Step 2 of constructing $G(j)$: The right part of G, namely $V(j)$ represents all active slots of all nodes from the first slot to j-th slot. Specifically, a vertex $v \in V(j)$ is denoted by a binary (n, k), which represents that node n is scheduled to be active at timeslot k ($1 \le k \le j$). $V(j)$ also has the following property: Let $V(j)[n]$ denote the set of vertexes that stand for node n's active slots from 1st slot to j-th slot. Then for each $n \in N$, $V(j)[n]$ represents all active slots contained in a ring path on $EN(n)$ whose length is no more than j. Also, $|V(j)[n]|$ is maximized. In Fig. 5, assume that when $|V(12)[n_1]|$ and $|V(12)[n_2]|$ are maximized, from timeslot 1 to 12, n_1 and n_2 are active at slot 1, 4, 7, 10 and slot 6, 12, respectively. Then we have $V(12) = \{(n_1, 1), (n_1, 4), (n_1, 7), (n_1, 10), (n_2, 6), (n_2, 12)\}$. How to compute $V(j)[n]$ to maximize $|V(j)[n]|$ is discussed later in this subsection.

Step 3 of constructing $G(j)$: $E(j)$ denotes the coverage relationship between nodes and targets. In detail, for a vertex $v = (n, k) \in V(j)$ and for each target $t \in \Gamma(n)$, an edge $(t, (n, k))$ is added into $E(j)$. This means that node n can choose to sample and update the sensory data of target t at timeslot k. For example, as illustrated by red lines in Fig. 5, since $\Gamma(n_1) = \{t_1, t_3, t_4\}$ and n_1 is scheduled to be active at timeslot 1, three edges $[t_1, (n_2, 1)]$, $[t_3, (n_2, 1)]$ and $[t_4, (n_2, 1)]$ are added into $E(12)$.

The following theorem gives a necessary and sufficient condition on whether j is a feasible solution:

Theorem 3. *If $G(j)$ has a matching M of size $|T|$, j is a feasible solution to the problem and vice versa.*

Proof. Firstly, we prove the sufficiency of the theorem. To prove that j is a feasible solution, we construct a monitoring schedule $\theta(n, i)$ for each $n \in N$ and $1 \le i \le j$ as follows: for each edge $e = [t, (n, k)] \in M$, we have $\theta(n, k) = t$. We only need to prove that this schedule satisfies Condition (1) and (2). Firstly, $|M| = |T|$ and the left part $U(j) = T$. Hence, for each target $t \in T$, there must exist an edge $e = [t, (n, k)] \in M$. Thus, Condition (1) is satisfied. Secondly, from the second step of constructing $G(j)$, for each $n \in N$, $V(j)[n]$ represents all active slots contained in a ring path on $EN(n)$, and the path is no longer than j. Therefore, Condition (2) holds.

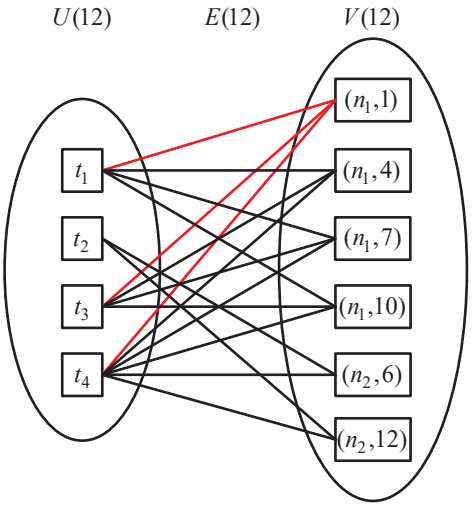


Fig. 5: An example for constructing the bipartite graph $G(j) = (U(j), V(j), E(j))$, where $N = \{n_1, n_2\}$, $T = \{t_1, t_2, t_3, t_4\}$, $\Gamma(n_1) = \{t_1, t_3, t_4\}$, $\Gamma(n_2) = \{t_2, t_4\}$ and $j = 12$.

Then we prove the necessity of the theorem. Assume that j is a feasible solution. According to the problem definition, there exists a mapping $\theta(n, i)$ satisfying Condition (1) and (2). Then we construct a bipartite graph $G'(j) = (U'(j), V'(j), E'(j))$ similar to $G(j)$ as follows: (a) $U'(j)=T$; (b) $V'(j)= \bigcup_{n \in N} V'(j)[n]$, in which $V'(j)[n]$ denotes all active slots of n from timeslot 1 to timeslot j, namely, for each $n \in N$ and $1 \le i \le j$, if $\theta(n,i) \neq 0$, the vertex $(n,i) \in V'(j)[n]$; (c) For each vertex $(n,k) \in V'(j)$, there exists an edge $[t, (n,k)] \in E'(j)$ for each target $t \in \Gamma(n)$. A maximum matching M' on $G'(j)$ can be derived as: for each $t \in T$, select a $(n,k) \in V'(j)$ such that $\theta(n,k) = t$, and add edge $[t, (n,k)]$ into M'. Since θ satisfies Condition (1), $|M'| = |T|$. Also, as θ satisfies Condition (2), for each $n \in N$, $V'(j)[n]$ stands for all active slots contained in a ring path with a length no more than j on $EN(n)$. According to the Step 2 in constructing $G(j)$, since $|V(j)[n]|$ is maximized, we have $|V(j)[n]| \ge |V'(j)[n]|$. Thus, $G(j)$ also has a maximum matching whose size is $|T|$. □

By Theorem 3 and its proof, in order to judge whether j is a feasible solution, we can compute a maximum matching $M(j)$ on $G(j)$. Then, if j is the minimum feasible solution, the corresponding monitoring schedule for all nodes is derived according to $M(j)$, in which an edge $[t, (n, k)] \in M(j)$ indicates $\theta(n, k) = t$. The Hungarian algorithm is applied to find the maximum matching on $G(j)$.

Now we discuss the issue mentioned in Step 2 of constructing $G(j)$. That is, for each node $n \in N$, how to compute the set $V(j)[n]$ to maximize $|V(j)[n]|$? As $V(j)[n]$ consists of vertexes representing the active slots of n from the first timeslot to the j-th timeslot, the number of n's active timeslots from timeslot 1 to j should be maximized. Also, $V(j)[n]$ stands for all active slots contained in a ring path

Algorithm 3 Phase 2: Computation of L_{opt} and the corresponding monitoring schedule

Input: N, T, $\Gamma(n)$ and $EN(n) = (Q(n), \Delta(n))$ for each $n \in N$, L_{upper}.
Output: The minimum feasible solution L_{opt} and $\theta(n, k)$ for each $n \in N$ and $1 \leq k \leq L_{opt}$.
1: $P_s(n, L_{upper}, q) \leftarrow Compute\_Ps(EN(n), L_{upper}, q)$ for each $n \in N$ and each $q \in Q(n)$;
2: $P_s(n, j) \leftarrow P_s(n, j, q)$ with the maximum number of active slots that $q \in Q(n)$ for each $n \in N$ and $1 \leq j \leq L_{upper}$;
3: $L_{opt} \leftarrow Min\{i | i \in [1, L_{upper}]\ and\ |M(i)| == |T|\}$, in which $M(i)$ is the maximum matching of the bipartite graph $G(i)$ and $G(i)$ is computed according to N, T, $\Gamma(n)$ and $P_s(n, i)$ for each $n \in N$;
4: Compute $\theta(n, k)$ for each $n \in N$ and $1 \leq k \leq L_{opt}$ according to $M(L_{opt})$;
5: **function** $Compute\_Ps(EN(n) = (Q(n), \Delta(n)), j, q)$
6: $Mt_2(n, q, i, q) \leftarrow 0$ for $1 \leq i \leq j$;
7: $Mt_2(n, q, i, q') \leftarrow$ -∞ for $1 \leq i \leq j$ and each state $q' \in Q(n)$ in which $q' \neq q$;
8: $Last\_hop_2(i, q') \leftarrow null$ for $1 \leq i \leq j$ and each state $q' \in Q(n)$;
9: Update $Mt_2(n, q, i, q')$ and $Last\_hop_2(i, q')$ according to (3) for $1 \leq i \leq j$ and each state $q' \in Q(n)$;
10: **end function**

no longer than j on $EN(n)$. Thus, to maximize $|V(j)[n]|$, we need to compute the ring path on $EN(n)$ that contains the maximum number of active slots while the path is no larger than j. Let $P_s(n, j)$ denote this path. Then $V(j)[n]$ represents all active slots contained in $P_s(n, j)$.

Now we discuss how to calculate $P_s(n, j)$ for each node n and $1 \leq j \leq L_{upper}$. Like the computation of $P_l(n, j)$, the starting state of $P_s(n, j)$ could be any state in $Q(n)$. For a state $q \in Q(n)$, let $P_s(n, j, q)$ denote the ring path on $EN(n)$ that contains the maximum number of active timeslots, which satisfies the following conditions: (a) The length of $P_s(n, j, q)$ is no larger than j and (b) the starting and ending state of $P_s(n, j, q)$ are both q. Obviously, $P_s(n, j)$ is the $P_s(n, j, q)$ ($q \in Q(n)$) that contains the maximum number of active slots. To compute $P_s(n, j, q)$, we define $P_s(n, q_1, i, q_2)$ ($q_1, q_2 \in Q(n)$, $i \geq 0$) as the path containing the maximum number of active slots on $EN(v)$, which satisfies the following: (a) The starting and ending state of the path is q_1 and q_2 respectively; (b) The length of the path can not exceed i. Obviously, $P_s(n, q, j, q) = P_s(n, j, q)$. Let $Mt_2(n, q_1, i, q_2)$ denote the number of active slots contained in $P_s(n, q_1, i, q_2)$. Then we have the following recursive relation:

$$Mt_2(n, q_1, i, q_2) = Max\{Mr_2(n, q_1, i, q_3) \mid edge\ (q_3, q_2, action, m) \in \Delta(n)\}, \quad (3)$$

in which $Mr_2(n, q_1, i, q_3) = Mt_2(n, q_1, i\text{-}m, q_3) + m$ if $action = 1$ and $Mt_2(n, q_1, i\text{-}m, q_3)$ if $action = 0$.

According to Equation (3) above, similar to $P_l(n, q, j, q)$, $P_s(n, q, j, q)$ can also be computed using a dynamic-programming algorithm. In terms of the initial condition of the dynamic programming, since $P_s(n, q, j, q)$ starts from q, before the computation, $Mt_2(n, q, i, q)$ is initialized as 0 for $0 \leq i \leq j$. Function *Compute_Ps* in Algorithm 3 shows the detail of computing $P_s(n, j, q)$, which has a time complexity of $O(|Q(n)|j)$. Like Theorem 2, it can be verified easily that $P_s(n, j, q)$ can be derived correctly by function *Compute_Ps*.

Similar to $P_l(n, j, q)$, by simply computing $P_s(n, j, q)$, $P_s(n, k, q)$ can be derived for each $k \in [1, j]$. Therefore, for all $n \in N$ and $1 \leq j \leq L_{upper}$, $P_s(n, j)$ is calculated as follows: Firstly, for each $n \in N$ and $q \in Q(n)$, $P_s(n, L_{upper}, q)$ is computed, in which $P_s(n, j, q)$ for $1 \leq j \leq L_{upper}$ can also be derived simultaneously. Then for each $n \in N$ and $1 \leq j \leq L_{upper}$, $P_s(n, j, q)$ that contains the maximum number of active slots is selected as $P_s(n, j)$.

Algorithm 3 summarizes the procedure of finding the minimum feasible solution L_{opt} between $[1, L_{upper}]$ and the corresponding monitoring schedule for all the nodes.

Algorithm 4 The algorithm for MTB

Input: The set of nodes N, the set of targets T, $\Gamma: N \rightarrow 2^T/\emptyset$, $EN(n) = (Q(n), \Delta(n))$ for each $n \in N$.
Output: The minimum feasible solution L_{opt} and $\theta(n, k)$ for each $n \in N$ and $1 \leq k \leq L_{opt}$.
1: Compute an upper bound of the feasible solution L_{upper} by Algorithm 2;
2: Compute L_{opt} and $\theta(n, k)$ for each $n \in N$ and $1 \leq k \leq L_{opt}$ by Algorithm 3;

Algorithm 4 summarizes the proposed algorithm for MTB, which consists of two phases. In Phase 1, an upper bound of the feasible solution L_{upper} is calculated by Algorithm 2. Then, in Phase 2, by binary searching between $[1, L_{upper}]$, the optimal feasible solution L_{opt} and corresponding monitoring schedule are derived using Algorithm 3.

5.3 The Time Complexity

Now we analyze the time complexity of the algorithm for MTB.

Lemma 1. *The time complexity of Algorithm 2 is* $O(|N||\overline{Q}|^2(|\overline{Q}|^2 log(|\overline{Q}|)+|\overline{\Gamma}|)+|N|L_{most})$, *where* $|\overline{Q}| = Avg(|Q(n)|, n \in N)$, $|\overline{\Gamma}| = Avg(|\Gamma(n)|, n \in N)$ *and* $L_{most} = Max\{|P_l(n, 1)| \times |\Gamma(n)|, n \in N\}$.

Proof. As function *Compute_Pl* in Algorithm 2 consumes $O(|Q(n)|i+|Q(n)|^3 log(|Q(n)|))$-time, the time complexity of Line 1 is $O(|N||\overline{Q}|^2(|\overline{Q}|^2 log(|\overline{Q}|)+|\overline{\Gamma}|))$. Both Line 2 and 3 take $O(|N||\overline{\Gamma}|)$-time. It is not hard to see that L_{upper} is upper bounded by L_{most}. Thus, the time consumption of Line 4 is $O(|N|L_{most})$. This ends the proof. □

Lemma 2. *The time complexity of Algorithm 3 is* $O(|N||\overline{Q}|^2 L_{most} + \frac{|N|^2|\overline{\Gamma}|L_{most}{}^2 log(L_{most})}{|\overline{p}|^2})$, *where* $|\overline{\Gamma}| = Avg(|\Gamma(n)|, n \in N)$ *and* $|\overline{p}| = Avg(|P_l(n, 1)|, n \in N)$.

Proof. The time complexity of function *Compute_Ps* in Algorithm 3 is $O(|Q(n)|L_{upper})$ in Line 1. Also, $L_{upper} \leq L_{most}$. Thus, Line 1 consumes at most $O(|N||\overline{Q}|^2 L_{most})$-time. The time consumption of Line 2 is $O(|N||\overline{Q}|log(|\overline{Q}|)L_{most})$. In each loop of Line 3, for the bipartite graph $G(j) = (U(j), V(j), E(j))$ to be considered, it can be proved easily that $|U(j)| = O(|T|)$, $|V(j)| = O(\frac{|N|L_{most}}{|\overline{p}|})$ and $|E(j)| = O(\frac{|N|L_{most}|\overline{\Gamma}|}{|\overline{p}|})$. Thus, it takes $O((|U(j)|+|V(j)|)E(j)) = O(\frac{|N|^2|\overline{\Gamma}|L_{most}{}^2}{|\overline{p}|^2})$-time for the Hungarian algorithm, which is repeated for $O(log(L_{most}))$ times by binary searching. Thus, Line 3 takes $O(\frac{|N|^2|\overline{\Gamma}|L_{most}{}^2 log(L_{most})}{|\overline{p}|^2})$-time. Line 4 takes $O(|N|L_{most})$-time. Summarizing Line 1-4 ends the proof. □

By summarizing the time complexities of Algorithm 2 (Lemma 1) and Algorithm 3 (Lemma 2), the theorem below shows the overall time complexity of the algorithm for MTB.

Theorem 4. *The time complexity of the proposed algorithm for MTB (Algorithm 4) is* $O[|N||\overline{Q}|^2(|\overline{Q}|^2 log|\overline{Q}| + L_{most}) + \frac{|N|^2|\overline{\Gamma}|L_{most}{}^2 log(L_{most})}{|\overline{p}|^2}]$.

6 Discussion on Some Design Issues

In this section, we discuss some design issues for our energy model and scheduling algorithm.

6.1 Parameters in the Energy Model

In this subsection, we discuss how to set the parameters in the computation of the energy model. First of all, under our energy model, the voltage of the node can not be lower than a threshold U_{th}. To keep each node working normally, U_{th} is set to be slightly larger than the critical voltage points of the node. For example, the IEA node [19] has two critical voltage points. One is power-off voltage (2.13 V). Also, if the capacitor voltage is lower than 2.2 V, the node can not measure the capacitor voltage accurately. Thus, we set the threshold voltage U_{th} = 2250 mV, which is 0.05 V higher than the maximum critical voltage point.

Next, according to Step 1 of constructing $EN(n)$ in Section 3.2.2, for a node n, the state $q_{i+1}(n) \in Q(n)$ ($1 \leq i \leq Df(n, U_s(n))$) stands for the voltage subinterval $[D_p(n, i), D_p(n, i\text{-}1))$. $D_p(n, i)$ is selected from $[U_d(n, U_s(n), i), U_d(n, U_s(n), i\text{-}1)]$, in which the parameter $\alpha \in [0, 1]$ determines the value of $D_p(n, i)$. Also, the voltage $D_p(n, i)$ is regarded as the energy level of n when n is in state $q_{i+1}(n)$. Thus, α has a crucial influence on the energy model $EN(n)$.

Now we briefly study the impact of α on the performance of $EN(n)$. For a node n, it is not hard to see, when n is scheduled under $EN(n)$, $|P_l(n, 1)|$ can be regarded as the minimum time needed for node n to harvest enough energy for being active for at least one timeslot. Obviously, the smaller $|P_l(n, 1)|$ is, the node can more fully utilize the ambient energy. Therefore, we evaluate $|P_l(n, 1)|$ in terms of α by a simple empirical study on an IEA node powered by ambient light, in which the light intensity is 270 Lux at the node's solar panel. Under this setting, the node has to spend over 25 idle slots to harvest the energy needed for one active slot, which is fit for most practical scenarios where the ambient energy is not sufficient.

TABLE 2: The impact of α on $|P_l(n, 1)|$

α	0.5	0.4	0.3	0.2	0.1	0.0
$\lvert P_l(n,1)\rvert$	36	32	27	26	60	62

Table 2 shows the impact of α on $|P_l(n, 1)|$. We can see that at the beginning, $|P_l(n, 1)|$ decreases with α. When α drops below a value, $|P_l(n, 1)|$ increases sharply. The reason for this is analyzed as follows. Usually, when $|P_l(n, 1)|$ is minimized, the path $P_l(n, 1) = q_{|Q(n)|}(n) \rightarrow q' \rightarrow q_{|Q(n)|}(n)$, in which q' is $q_{(|Q(n)|-1)}(n)$ or $q_{(|Q(n)|-2)}(n)$. When α is large, $q' = q_{(|Q(n)|-1)}(n)$. According to Step 1 of constructing $EN(n)$ in Section 3.2.2, the energy level of state q' decreases with α. Thus, it takes fewer idle slots for n to charge from state $q_{|Q(n)|}(n)$ to state q'. This reduces $|P_l(n, 1)|$. After α descends to a certain value, the energy level of $q_{(|Q(n)|-1)}(n)$ will be too low such that n can not be scheduled to be active. Under this condition, $q' = q_{(|Q(n)|-2)}(n)$. Since the energy level of $q_{(|Q(n)|-2)}(n)$ is higher than that of $q_{(|Q(n)|-1)}(n)$, the number of idle slots needed for charging from state $q_{|Q(n)|}(n)$ to q' suddenly increases, causing $|P_l(n, 1)|$ to rise sharply.

As analyzed above, in the real-world implementation, to determine an appropriate value for α, we can deploy a node at a place where the ambient energy is insufficient, then $|P_l(n, 1)|$ is evaluated with varying α and we choose α that can minimize $|P_l(n, 1)|$.

6.2 Leveraging Unused Active Timeslots

In Section 5, we have derived the monitoring schedule in the form of L_{opt} and $\theta(n, i)$ for each node $n \in N$ and $1 \leq i \leq L_{opt}$. In Algorithm 3, $\theta(n, i)$ is calculated from a maximum matching $M(L_{opt})$ on the bipartite graph $G(L_{opt}) = (U(L_{opt}), V(L_{opt}), E(L_{opt}))$. According to the proof of Theorem 3, for each edge $e = (t, (n, k)) \in M(L_{opt})$, we have $\theta(n, k) = t$. Obviously, for a vertex $v = (n, k) \in V(L_{opt})$, if v is not involved in $M(L_{opt})$, at the k-th timeslot of the time period from timeslot 1 to L_{opt}, no target is assigned for n. Thus, this active slot of n is wasted. We refer to these unused active slots of n as ***spare slots of n***. For example, assume that L_{opt} = 12, $G(12) = (U(12), V(12), E(12))$ is the bipartite graph shown in Fig. 5. Also, the monitoring schedule $\theta(n, i)$ is derived from a maximum matching $M(12) = \{(t_1, (n_1, 1)), (t_2, (n_2, 6)), (t_3, (n_1, 4)), (t_4, (n_1, 7))\}$. Under this condition, in $V(12)$, two vertexes (n_1, 10) and (n_2, 12) are not involved in $M(12)$. Therefore, the spare slot of n_1 and n_2 is 10 and 12, respectively.

To leverage spare slots, each node n is scheduled to sample and update the sensory data of each target in $F(n)$ in a round-robin manner in n's spare slots, in which $F(n)$ is the set of targets assigned to n in Phase 1 (i.e., Algorithm 2). If $F(n) = \emptyset$, $F(n)$ is set as $\Gamma(n)$.

7 Simulations

In this section, the performance of the proposed monitor scheduling algorithm is evaluated by simulations under different energy conditions and distributions of nodes/targets.

7.1 Simulation Setups

In this subsection, we describe the detailed simulation setups. The performance of our algorithm is evaluated in three aspects: (a) the maximum AoI of all targets during the whole working duration (i.e., $Max\{A(t, i) | t \in T \,\&\, 1 \leq i \leq W\}$, in which W is the length of the whole working duration), (b) the average AoI of all targets during the whole working duration (i.e., $Avg\{A(t, i) | t \in T \,\&\, 1 \leq i \leq W\}$, in which W is the length of the whole working duration) and (c) event capture [5]. Overall, in each simulation, for each node n, its simulated energy model $EN_s(n) = (Q_s(n), \Delta_s(n))$ is derived based on energy models computed on real-world testbeds firstly. Then, our algorithm is evaluated based on the simulated energy model, and we assume that nodes will never die. Further, we also compare our algorithm with a baseline algorithm. Data of each simulation is the average of 200 executions, in which for each execution, the length of the whole working duration is 10000 timeslots.

5. In our evaluation, for each target t, some events occur at t and each event lasts for some time. The performance of event capture is evaluated by the number of events captured at all targets during the whole working duration. An event is considered to be captured if the sensory data of the corresponding target is sampled and updated at least once during the event occurrence.

7.1.1 Nodes and Targets

As we focus on one-hop BF-WSNs, a large network scale is not needed. By default, 8 nodes are deployed in the monitoring field to cover 10 targets. For each target, $|Cover(t)|$ is 3 on average. As for the evaluation on the performance of event capture, at each target, each event lasts for 30 timeslots on average and the average interval between two consecutive events is 200 timeslots.

7.1.2 Energy Model

In this part, the simulated energy model for each node in the simulated network is derived. The IEA node [19] is used in our simulation. We assume that nodes are powered by ambient light and we use a commercial LED lamp [20] as the energy source, in which the light intensity in the monitoring field varies from 270Lux to 830Lux. Also, for each node n in the simulated network, n's simulated energy model is determined by the light intensity at n. Let the simulated energy model under light intensity $I \in$ (270Lux, 830Lux) be denoted by $EN_s(I) = (Q_s(I), \Delta_s(I))$. As there are an infinite number of Is between 270Lux to 830Lux, to compute $EN_s(I)$ for any $I \in$ (270Lux, 830Lux), we firstly compute $EN_s(I)$ under 8 different light intensities between 270Lux and 830Lux on an IEA testbed using Algorithm 1, in which the step size is 80Lux. Then for arbitrary $I \in$ (270Lux, 830Lux), $EN_s(I)$ is approximated by polynomial interpolation on energy models under 8 different light intensities in the following three steps:

Step 1: Compute $Q_s(I)$. According to the 8 energy models computed on an IEA testbed using Algorithm 1, $|Q_s(I)|$ can be estimated as the following table.

light intensity I (Lux)	$\lvert Q_s(I)\rvert$
270∼400	3
401∼480	4
481∼620	5
621∼720	6
721∼820	7
821∼830	8

Step 2: Compute edges in $\Delta_s(I)$ representing the charging process. Let $Q_s(I)$ be denoted by $\{Q_s(I,1), Q_s(I,2), ...Q_s(I,|Q_s(I)|)\}$. First of all, for any $1 \le k < j \le |Q_s(I)|$, we assume that the energy level of $Q_s(I, j)$ is higher than $Q_s(I, k)$. According to Step 2 of constructing $EN(n)$ in Section 3.2.2, the node can not be scheduled to be active in state $Q_s(I, 1)$. For $1 < k \le |Q_s(I)|$, let $Re(I, k)$ denote the number of idle slots needed to change from state $Q_s(I, 1)$ to state $Q_s(I, k)$. Obviously, under light intensity I, it takes $Re(I, k)$ - $Re(I, k\text{-}1)$ idle slots to change from state $Q_s(I, k\text{-}1)$ to $Q_s(I, k)$. Thus, edge ($Q_s(I, k\text{-}1)$, $Q_s(I, k)$, 0, $Re(I, k)$ - $Re(I, k\text{-}1)$) is added into $\Delta_s(I)$ for each $2 \le k \le |Q_s(I)|$. Then, how to approximate $Re(I, k)$? As shown in Fig. 6, triangle points represent the actual values of $Re(I, k)$ for $2 \le k \le 8$ under 8 different light intensities from 270Lux to 830Lux with a step size of 80Lux. Then for arbitrary $I \in$ (270Lux, 830Lux), $Re(I, k)$ can be approximated using polynomial interpolation on these actual values, which is shown by star lines in Fig. 6.

Step 3: Compute edges in $\Delta_s(I)$ that stand for the discharging process. First of all, for an energy model $EN(n) = (Q(n), \Delta(n))$, we introduce a theorem on edges in $\Delta(n)$ that represent the discharging process of n:

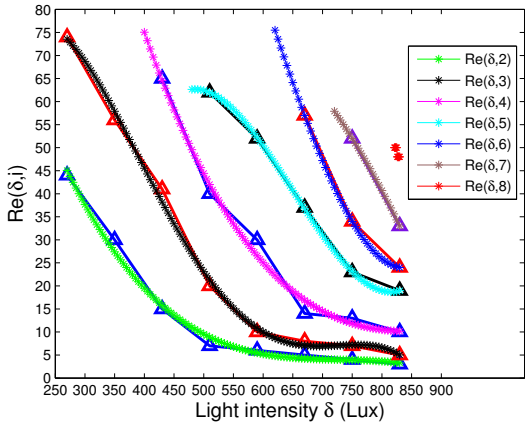


Fig. 6: $Re(I, k)$ for 270Lux $\le I \le$830Lux and $2 \le k \le 8$ is approximated by polynomial interpolation.

TABLE 3: Simulation environment parameters.

Parameter	value
The number of battery-free nodes	4∼44, default: 8
The number of targets	4∼24, default: 10
Average $\lvert Cover(t)\rvert$	1∼7, default: 3
Average light intensity (Lux)	300∼660, default: 350
The length of the working duration (s)	10000
The duration of each event (s)	30
The average interval between two consecutive events	100∼800, default: 200

Theorem 5. *Given $EN(n) = (Q(n), \Delta(n))$ that $Q(n) = \{q_1(n), q_2(n), ...q_{|Q(n)|}(n)\}$ and for any $k < j \le |Q(n)|$, the energy level of $q_k(n)$ is higher than that of $q_j(n)$, we have:*

1) *For each edge $(q_1(n), q_j(n), 1, m) \in \Delta(n)$, we have $1 \le m \le |Q(n)|$-2 and $j = m+2$.*

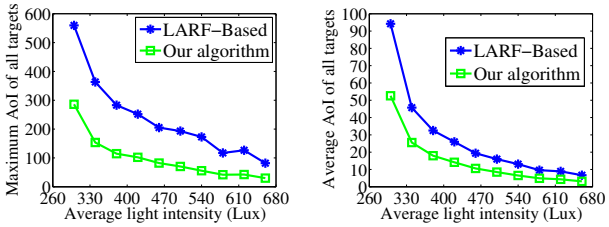
2) *For each edge $(q_k(n), q_j(n), 1, m) \in \Delta(n)$, given k, we have $m \le d$ in which $d \in \{|Q(n)| - k - 1, |Q(n)| - k\}$. Further, given m, we have $j \in \{k + m, k + m + 1\}$.*

Proof. The proof consists of two steps. Consider an edge $(q_k(n), q_j(n), 1, m) \in \Delta(n)$ such that $k < j \le |Q(n)|$. Firstly, compute the possible range of m by assuming that k is fixed. Then, the possible range of j can be derived by assuming that both k and m are fixed. For details, please see Appendix B in the supplement material [32]. □

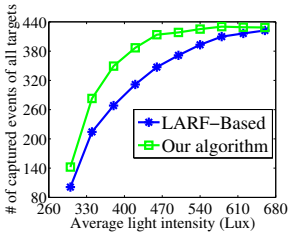
According to the Theorem 5 above, we firstly rearrange all states in $Q_s(I)$ by $Q_s(I, k) = Q_s(I, |Q_s(I)|+1\text{-}k)$ so that for any $k < j \le |Q(n)|$, the energy level of $Q_s(I, k)$ is higher than that of $Q_s(I, j)$. Then, edges that represent the discharging process are added into $\Delta_s(I)$: For each $1 \le m \le |Q_s(I)|$-2, edge ($Q_s(I, 1)$, $Q_s(I, m+2)$, 1, m) is added into $\Delta_s(I)$. For edges that outcome from $Q_s(I, k)$ ($k \neq 1$), randomly select $d \in \{|Q_s(I)| - i - 1, |Q_s(I)| - k\}$. Then, for each $1 \le m \le d$, edge ($Q_s(I, k)$, $Q_s(I, j)$, 1, m) is added into $\Delta_s(I)$, in which j is randomly selected from $\{k + m, k + m + 1\}$.

In our simulation, to simulate harsh energy conditions, the average light intensity at each node is set to be 350 Lux by default, in which each node needs at least 30 idle timeslots on average to harvest enough energy for one active slot. Mention that the simulated energy model for other types of energy sources can also be derived similarly.

All simulation environment parameters are summarized in Table 3.

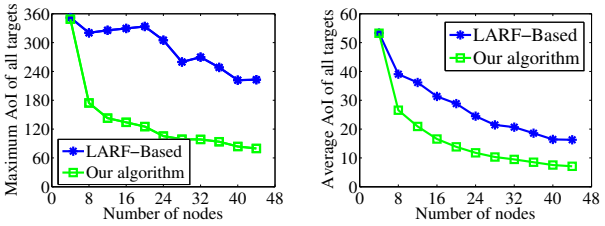


(a) Average light intensity vs maximum AoI. (b) Average light intensity vs average AoI.

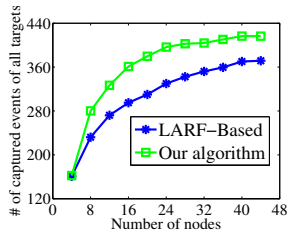


(c) Average light intensity vs event capture.

Fig. 7: The impact of ambient energy conditions.



(a) Number of nodes vs maximum AoI. (b) Number of nodes vs average AoI.



(c) Number of nodes vs event capture.

Fig. 8: The impact of nodes number.

7.1.3 The Baseline Algorithm

In [15], the author presents an algorithm named LARF to minimize the sum of each node's maximum AoI in a one-hop BF-WSN. For comparison, we need to make some modifications to LARF. First of all, in [15], in order to avoid transmission conflict, at each timeslot, only one node can transmit the data to the sink. For the sake of fairness, we slightly modify LARF, in which the transmission conflict is no longer taken into account. Secondly, [15] assumes that a node always consumes one unit of energy for one active slot. To make this assumption compatible with our energy model, in our simulation, a node n is assumed to have $|Q(n)|$-i units of energy when n is in state $q_i(n)$. This is generous to some extent since by Theorem 6, it is possible that n can only be active for at most $|Q(n)|$-i-1 consecutive slots at state $q_i(n)$. Thirdly, as [15] does not consider the coverage between nodes and targets, in the evaluation of LARF, each node n is assumed to sample and update the sensory data of each target in $\Gamma(n)$ in a round-robin manner.

7.2 Simulation Results

7.2.1 The Impact of Energy Conditions

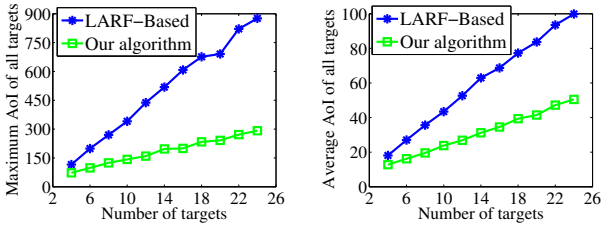
In this subsection, we study the performance of both algorithms in terms of energy conditions. To be specific, as the energy harvesting profile is determined by the ambient light condition, the performance is evaluated as function of the average light intensity at each node.

As shown in Fig. 7(a) and (b), both the maximum and average AoI of all targets decrease as the ambient light intensity grows. The reason is obvious. As the light intensity rises, nodes spend fewer idle slots on recharging. Also, our algorithm has high performance in terms of the maximum AoI and average AoI, in which our algorithm reduces 55% of the maximum AoI and 50% of the average AoI compared with the LARF-based algorithm. This is because our algorithm considers the coverage between nodes and targets, in which multiple nodes are scheduled cooperatively in the computation of the monitoring schedule. In this way, all active slots are distributed uniformly among all targets. As for Fig. 7(c), the number of captured events for both algorithms grows with the ambient light, and our algorithm can capture 30% more events than the LARF-based when the average light intensity is below 500 Lux. The reason for this is the same as that for Fig. 7(a) and (b). As the light intensity continues to increase, the advantage of our algorithm is decreased. The reason is as follows. As the average time duration of an event is fixed, when the ambient energy is relatively sufficient, the monitoring frequency is large enough to capture most of the events. Thus, the number of captured events of our algorithm ceases to increase after the average light intensity reaches some level (Larger than 500 Lux).

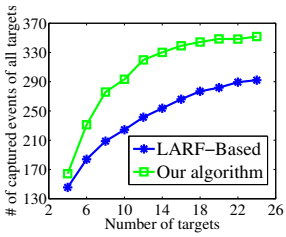
7.2.2 The Impact of the Number of Nodes

In this part, we study the impact of nodes number on the performance of our algorithm, in which each target is covered by 25% of all the nodes on average.

From Fig. 8, we can see that the performance of both algorithms improves with the increase in nodes number. The reason is very obvious that more nodes bring higher coverage quality for targets. For the same reason of Fig. 7, our algorithm outperforms the LARF-based algorithm in all three metrics, in which our algorithm reduces 46%∼65% and 34%∼57% on the maximum and the average AoI compared with the LARF-based algorithm. Also, in terms of the performance on event capture, according to Fig. 8(c), our algorithm is 17% higher than the LARF-based on average. Further, we notice that the advantage of our algorithm is not very obvious when the number of nodes is relatively small. This is because when the network scale is small, it is very likely that each target is within the coverage range of only a few nodes, which limits the optimization space of the MTB problem. As the optimization space grows with the number of nodes, our algorithm has more advantages over the baseline algorithm. Moreover, as shown in Fig. 8(c), the performance on event capture of our algorithm increases

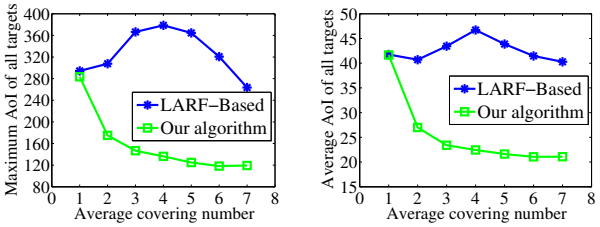


(a) Targets number vs maximum AoI. (b) Targets number vs average AoI.

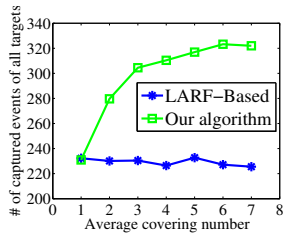


(c) Targets number vs event capture.

Fig. 9: The impact of targets number.



(a) Average $|Cover(t)|$ vs maximum AoI. (b) Average $|Cover(t)|$ vs average AoI.



(c) Average $|Cover(t)|$ vs event capture.

Fig. 10: The impact of node's covering range.

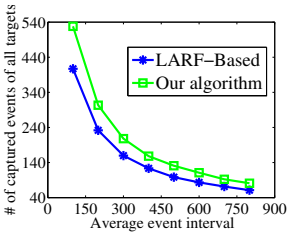


Fig. 11: Average event interval vs event capture.

more slowly when the number of nodes is larger than 24. The reason for this is the same as that for Fig. 7(c).

7.2.3 The Impact of Targets Number

Now we analyze the performance of both algorithms as functions of targets number. Since the ambient energy and the number of nodes in the monitoring field remain unchanged, as the number of targets grows, the number of active timeslots assigned to each target is decreased. Therefore, as shown in Fig. 9(a) and (b), both the maximum and average AoI of all targets increase. Compared with the LARF-based algorithm, our algorithm still has better performance, and the advantage grows with the number of targets. The percentage of our algorithm's maximum AoI and average AoI on the baseline decrease from 62% to 33% and 66% to 50% respectively. The reason for this is that fewer targets cause a smaller optimization space of the MTB problem. Besides, there is another reason for the growing tendency of our algorithm's advantage in Fig. 9(a). As the number of targets in the monitoring field increases, it is more likely that there exist some targets that can only be covered by a few nodes with very low energy harvesting capability. In this situation, the maximum AoI of these targets may become very large without careful consideration of the cooperation among nodes. As our algorithm addresses this problem, the advantage of our algorithm is more obvious when there are many targets in the monitoring field. In terms of event capture shown in Fig. 9(c), the performance of our algorithm is 22% better than the baseline on average. As the targets number increases, the number of all events also rises so that both algorithms can capture more events. However, the overall monitoring resources are limited. Thus, after the targets number reaches some value (As shown in Fig. 9(c), 20 for our algorithm and 22 for LARF-based algorithm), the captured events number remains stable.

7.2.4 The Impact of Node's Covering Range

Now we illustrate the impact of node's covering range on the performance of both algorithms. As the coverage range of each node enlarges, a target in the network can be covered by more nodes. Thus, the performance is evaluated as function of the average $|Cover(t)|$ of all targets.

As shown in Fig. 10(a)-(c), our algorithm outperforms the baseline in all three metrics. Also, the performance of our algorithm is increased as the average $|Cover(t)|$ rises. This is because our algorithm considers the cooperation among multiple nodes in target monitoring, and the optimization space induced by the increasing average $|Cover(t)|$ can be exploited. Moreover, as illustrated in Fig. 10(a), as the average $|Cover(t)|$ rises, the maximum AoI of the LARF-based algorithm increases firstly and then decreases. The reason for this is as follows. When the average $|Cover(t)|$ is relatively low, as LARF-based algorithm schedules each node n to sample and update the sensory data of each target in $\Gamma(n)$ in a round-robin manner, if there is only small rise in $|Cover(t)|$, the resource assigned to those targets that can only be covered by a few nodes is reduced. For example, consider a target t that is within the coverage range of only one node n, which can be active for one slot in every 40 slots. When the average $|Cover(t)|$ increases slightly, n can cover an additional target t' but t may still be covered by only one node n. Under this condition, the maximum AoI of t is increased by 40 slots. As average $|Cover(t)|$ continues to grow, when the average $|Cover(t)|$ is relatively high, each target is covered by many nodes. Therefore, the maximum AoI decreases after the average $|Cover(t)|$ reaches some value. The above reason can also explain the tendency of the average

AoI of the LARF-based algorithm shown in Fig. 10(b). In Fig. 10(c), the performance of the LARF-based algorithm remains steady with the varying average $|Cover(t)|$. This is because when the average $|Cover(t)|$ grows, the overall monitoring resource remains unchanged.

7.2.5 The Impact of Event Frequency

In this part, we study the impact of event frequency on the performance of event capture. As shown in Fig. 11, as the average event interval grows, there are fewer events to be captured. Thus, the number of captured events of both algorithms decreases. Our algorithm still outperforms the baseline, which captures 30% more events.

7.2.6 Execution Time of the Scheduling Algorithm

As shown by Theorem 4, the monitoring scheduling algorithm is pseudo-polynomial. To verify its feasibility, in Table 4 below, we provide the average execution time of the scheduling algorithm as functions of nodes number. As shown in Table 4, the scheduling algorithm can finish in a few milliseconds even when the network scale is large (i.e., The number of nodes is 100). Thus, the scheduling algorithm is practical in terms of its execution time.

TABLE 4: Nodes number vs execution time of the monitoring scheduling algorithm (Simulation).

Nodes number	10	20	40	60	80	100
Execution time (ms)	1.03	2.07	7.37	18.57	30.58	52.98

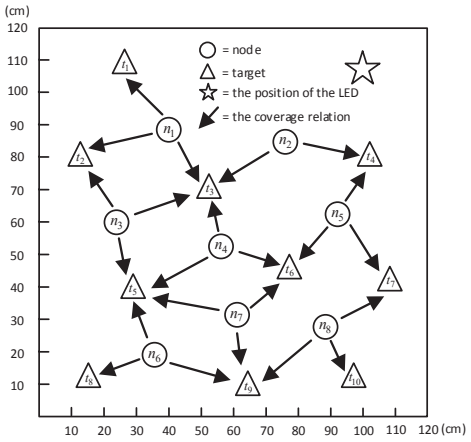
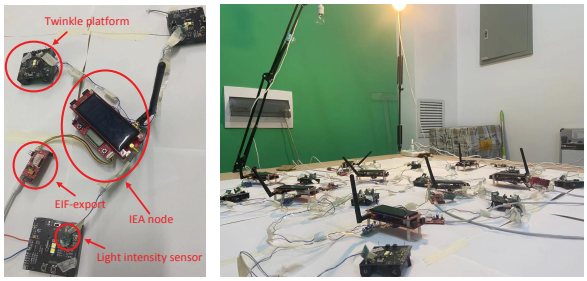


Fig. 12: The position of hardwares. Arrows denotes the coverage relationship between nodes and targets.



(a) An IEA node in the deployed network that covers 3 Twinkle platforms.

(b) Our testbed is deployed according to Fig. 12.

Fig. 13: Experimental hardwares.

8 Experiments

In this section, we conduct empirical studies on real-world testbeds to evaluate the performance of our energy model and scheduling algorithm. Each execution lasts for 3000 timeslots, and each data point is the average of 5 executions.

8.1 Experiment Setups

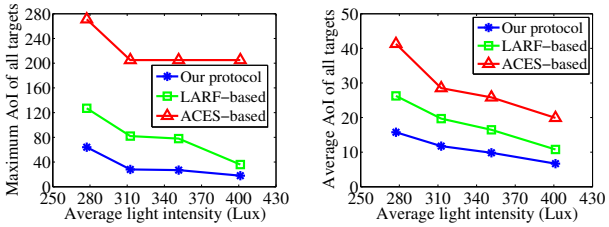
8.1.1 Hardware Setups

In this part, we briefly describe the hardware used in our experiment. First of all, we deploy 8 IEA nodes in a 120cm $\times$ 120cm area. Each node is equipped with an 8cm $\times$ 3.5cm solar panel for energy harvesting and a 1mF capacitor for energy storage. Besides, there are several directional light intensity sensors on each node, and each sensor can measure the light intensity near its facing direction. Secondly, we develop the Twinkle platform as the target. Each Twinkle has two small LEDs whose brightness can be adjusted in three levels. We deploy 10 Twinkles as targets, in which each Twinkle stands for a target. The switching on of both LEDs at a Twinkle indicates the occurrence of an event at the corresponding target. When the event ends, both LEDs are switched off. Thirdly, a commercial LED lamp [20] with adjustable luminance is placed 90cm above the testbed as the energy source for all IEA nodes. Fourthly, in terms of the coverage between nodes and targets, for an IEA node n and a Twinkle platform t, if $t \in \Gamma(n)$, one of the n's light intensity sensors will be used for monitoring the light intensity at t. As each IEA node has at most three directional light intensity sensors, $|\Gamma(n)|$ is not larger than 3. Further, to track the real-time working state of an IEA node, EIF-export [29] is used to transfer the debugging information of the node to the PC. As EIF-export is powered by external sources, it has no influence on the energy level of the node. Fig. 13(a) shows an IEA node that can cover 3 Twinkle platforms and an EIF-export is connected to the node for tracking the real-time working status of the node. Our testbed is shown in Fig. 13(b). The positions of all nodes, targets, the LED lamp and the coverage relationship between all nodes and targets are shown in Fig. 12.

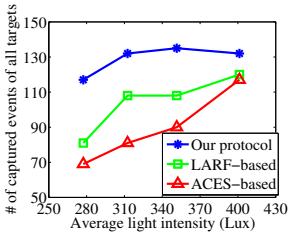
Apart from the above, an IEA node powered by DC-AC is used as the sink. In each execution, the sink receives the required information of the scheduling algorithm from each node (e.g., The energy model) and transfers the information to PC through EIF-export. The PC runs the monitor scheduling algorithm and transmits the derived monitoring schedule back to the sink, which is later responsible for sending the derived schedule to each node.

8.1.2 The Operation in an Active Slot

For each node, the length of one timeslot is one second. In each active slot, the node and the sink work as follows: (a) From 0 to 600ms, the node will take 10 samples on the light intensity of the target that the node is scheduled to monitor, in which each sample is a 4-byte float number. These 10 samples are packed into a 40-byte data packet; (b) From 600ms to 1000ms, the node will send the data packet



(a) Average light intensity vs maximum AoI. (b) Average light intensity vs average AoI.



(c) Average light intensity vs event capture.

Fig. 14: The impact of energy conditions.

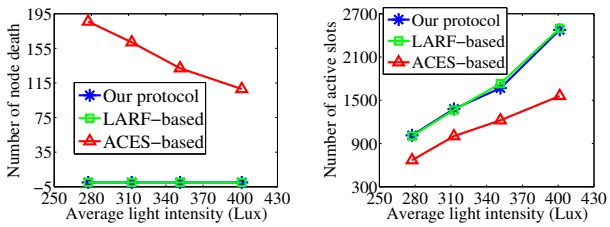
with a transmit power of -10 dBm 5 times. In this way, the transmission success rate of the packet is over 99.5%. (c) After receiving the packet from the node, the sink relays the sampling data contained in the packet to the PC, and the PC updates the AoI information. Also, the PC will use a simple outlier detection method [30] to judge whether the event occurs at the corresponding target.

8.1.3 Our Protocol and Baseline Protocols

In real-world experiments, our proposed energy model and scheduling algorithm are evaluated together. Therefore, in each execution of our protocol, each node will compute its energy model firstly by Algorithm 1, then based on energy models of all nodes, the monitoring schedule is derived by our scheduling algorithm. In terms of baselines, the following two protocols are compared:

1) LARF-based protocol: The setting is just similar to Section 7.1.3. However, [15] does not consider the death of the node. In real-world experiments, a node may die due to the energy shortage. Therefore, for the sake of fairness, in the evaluation of the LARF-based protocol, each IEA node is equipped with a battery. Once the capacitor voltage is lower than 2200mV, the node will trigger the battery discharge. In this way, node death can be avoided.

2) ACES-based protocol: In [31], the author proposed ACES protocol for event monitoring in energy-harvesting WSNs. ACES reduces the duty-cycle period of each node and maximizes the sensing quality by a q-learning-based algorithm. We implement ACES on each IEA node, in which the parameters of q-learning are set just the same as in [31]. In particular, [31] uses 24 hours as the working duration. For each time period of 15 minutes, the node observes the state transition in q-learning, updates the parameters and determines the operation for the next 15 minutes. In each period of 15 minutes, the node can choose to sleep for 15, 60, 300, or 900 seconds respectively. In our experiment, one execution lasts for 3000 seconds. Correspondingly, in our implementation of the ACES-based protocol, the updating cycle is set to be 40 seconds, during which the node can



(a) Average light intensity vs the number of node death. (b) Average light intensity vs the number of active slots.

Fig. 15: The impact of energy condition on nodes' working.

choose to be active for one slot in every 5, 15, 30 seconds or just sleep for 40 seconds. Besides, similar to LARF-based protocol, ACES does not consider the cooperation of multiple nodes on target monitoring. Thus, in the experiment for ACES-based protocol, each node n is scheduled to sample and update the sensory data of each target in $\Gamma(n)$ in a round-robin manner.

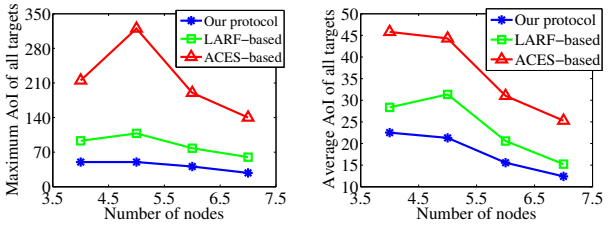
8.2 Experimental Results

In this subsection, we compare our protocol with the LARF-based and ACES-based protocol in three aspects as in Section 7. Further, to obtain deeper insight into the experimental results, two additional metrics are discussed: (a) The number of deaths of all nodes during the whole working duration and (b) The number of active slots of all nodes during the whole working duration.

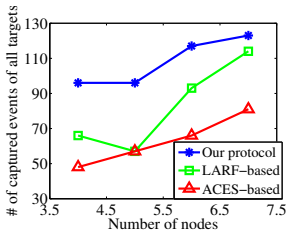
8.2.1 The Impact of Energy Conditions

In this part, we evaluate the performance of three protocols in terms of ambient energy conditions, in which the average light intensity at each node varies from 277Lux to 401Lux.

The experimental results shown in Fig. 14 meet well with the simulation results in Fig. 7, in which our protocol outperforms two baselines in all three metrics. Compared with the LARF-based protocol and ACES-based protocol, on average, our protocol reduces the maximum AoI of all targets by 58% and 80%, the average AoI by 40% and 62%, and increases the number of captured events by 23% and 44%. The advantage of our protocol over the LARF-based protocol is due to the same reason as in Fig. 7. Compared with ACES-based protocol, our protocol not only considers the cooperation of multiple nodes on target monitoring, more importantly, it also utilizes the ambient energy more sufficiently. As shown in Fig. 15(a), while our protocol exhibits zero node death, node death occurs for some times under the ACES-based protocol. This is because some node deaths are allowed in ACES. The impact of node death is shown in Fig. 15(b): since much energy is consumed for node reboot after node death, the number of active slots of the ACES-based protocol is only 65% of our protocol. Fundamentally speaking, ACES protocol is designed for energy-harvesting nodes with relatively large capacitors (In the size of F) as the energy storage, in which the energy harvesting and storage capability is relatively large and the energy consumed for node reboot is little compared with the energy for node operation (e.g., Sampling, receiving and sending packets). However, battery-free nodes are usually equipped with small capacitors (In the size of mF and μF), whose energy gathering and storage capability is relatively



(a) Nodes number vs maximum AoI. (b) Nodes number vs average AoI.



(c) Nodes number vs event capture.

Fig. 16: The impact of nodes number.

low. For each node, once it is dead, the energy consumption for node reboot is a huge waste. Thus, when designing protocols for battery-free WSNs, it is important to consider the avoidance of node death.

Apart from the above, as shown in Fig. 15(b), in terms of the number of all active slots, our protocol has the same performance as the LARF-based protocol. In [15], to minimize the sum of the maximum AoI of all the nodes, for each node, LARF maximizes the number of its active slots during the whole working duration. Thus, our protocol can fully exploit the harvested energy.

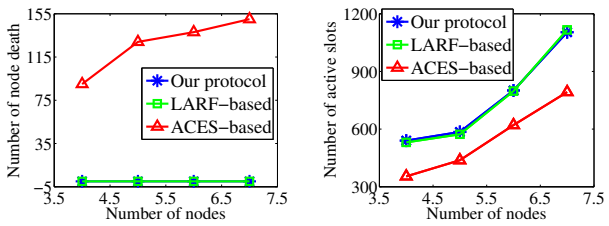
8.2.2 The Impact of Nodes Number

Now we analyze the performance of three protocols with varying nodes number. To this end, we deploy 8 Twinkle platforms as targets in a 120cm $\times$ 120cm area, and the number of nodes varies from 4 to 7. The average light intensity at each node is set to be 289Lux. Also, in each execution, each target is covered by at least one IEA node.

As shown in Fig. 16, as more nodes are deployed in the monitoring field, better coverage can be obtained. Thus, the performance on three metrics of all protocols increases. Due to the same reason analyzed in subsection 8.2.1, our protocol still has the advantage over the two baselines. Compared with the LARF-based and ACES-based protocol, our protocol is 51% and 81% lower in the maximum AoI, 26% and 52% lower in the average AoI, 30% and 71% higher in the event capture, respectively. As for Fig. 17, for the ACES-based protocol, as the number of nodes increases, there are more node deaths. At the same time, the number of total active slots of all the nodes is increased. Similar to Fig. 15, Fig. 17 shows that our protocol can fully exploit the ambient energy while avoiding the death of nodes.

8.2.3 Execution Time of the Scheduling Algorithm

Similar to Section 7.2.6, since the monitoring scheduling algorithm is pseudo-polynomial, in this subsection, we provide the execution time of the monitoring scheduling algorithm in the real-world implementation. As shown by



(a) Nodes number vs the number of node death. (b) Nodes number vs the number of active slots.

Fig. 17: The impact of nodes number on nodes' working.

TABLE 5: Nodes number vs execution time of the scheduling algorithm (Real-world experiment).

Nodes number	2	4	6	8
Execution time (ms)	0.46	0.50	0.52	0.65

Table 5, in our testbed experiments, it takes less than 1ms to execute the scheduling algorithm. This verifies the feasibility of our scheduling algorithm in terms of execution time.

9 Conclusion and Future Work

In this paper, we first investigate the problem of maximum AoI minimization for target monitoring in BF-WSNs (MTB). For any timeslot-based algorithm and protocol design problems in BF-WSNs, we propose a simple and practical energy model for real-world battery-free nodes with small capacitors as energy storage. Based on the energy model, the problem of MTB is formally defined. A two-stage algorithm based on dynamic programming and binary search is proposed to solve MTB optimally, in which all nodes are scheduled cooperatively to minimize the maximum AoI of all targets. Finally, the simulation and experimental results reveal the high performance of our proposed energy model and scheduling algorithm in terms of AoI metrics, event capture, and energy efficiency. As for future work, we aim to evaluate our proposed protocol under the scenario where nodes are powered by other energy sources (e.g., Wireless power, wind power). We will also extend our scheme to multi-hop BF-WSNs.

References

[1] S. Ulukus *et al.*, "Energy Harvesting Wireless Communications: A Review of Recent Advances," in *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 360-381, 2015.
[2] Y. Yang, L. Su, Y. Gao and T. F. Abdelzaher, "SolarCode: Utilizing Erasure Codes for Reliable Data Delivery in Solar-powered Wireless Sensor Networks," in *Proceedings of IEEE INFOCOM*, 2010, pp. 1-5.
[3] J. Wenck, J. Collier, *et al.*, "Scaling self-timed systems powered by mechanical vibration energy harvesting," in *ACM J. Emerg. Technol. Comput. Syst.*, vol. 6, no. 2, pp. 5:1-5:24, 2010.
[4] S. Kaul, R. Yates and M. Gruteser, "Real-time status: How often should one update?" in *Proceedings of IEEE INFOCOM*, 2012, pp. 2731–2735.
[5] C. Li, S. Li, Y. Chen, Y. Thomas Hou and W. Lou, "AoI Scheduling with Maximum Thresholds," in *Proceedings of IEEE INFOCOM*, 2020, pp. 436-445.
[6] G. Stamatakis, N. Pappas and A. Traganitis, "Control of Status Updates for Energy Harvesting Devices That Monitor Processes with Alarms," in *IEEE Globecom Workshops*, 2019, pp. 1-6.

[7] S. Farazi, A. G. Klein and D. R. Brown, "Average age of information for status update systems with an energy harvesting server," in *IEEE INFOCOM Workshops*, 2018, pp. 112-117.
[8] E. T. Ceran, D. Gündüz and A. György, "Reinforcement Learning to Minimize Age of Information with an Energy Harvesting Sensor with HARQ and Sensing Cost," in *IEEE INFOCOM Workshops*, 2019, pp. 656-661.
[9] A. Arafa, J. Yang *et al.*, "Age-Minimal Transmission for Energy Harvesting Sensors With Finite Batteries: Online Policies," in *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 534-556, 2020.
[10] X. Zheng, S. Zhou, Z. Jiang and Z. Niu, "Closed-Form Analysis of Non-Linear Age of Information in Status Updates With an Energy Harvesting Transmitter," in *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, pp. 4129-4142, 2019.
[11] B. T. Bacinoglu, Y. Sun, E. Uysal–Bivikoglu and V. Mutlu, "Achieving the Age-Energy Tradeoff with a Finite-Battery Energy Harvesting Source," in *IEEE ISIT*, 2018, pp. 876-880.
[12] S. Feng and J. Yang, "Optimal status updating for an energy harvesting sensor with a noisy channel," in *IEEE INFOCOM Workshops*, 2018, pp. 348-353.
[13] S. Leng and A. Yener, "Age of Information Minimization for an Energy Harvesting Cognitive Radio," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 2, pp. 427-439, 2019.
[14] A. Arafa and S. Ulukus, "Timely Updates in Energy Harvesting Two-Hop Networks: Offline and Online Policies," in *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, pp. 4017-4030, 2019.
[15] T. Zhu, J. Li, H. Gao, Y. Li and Z. Cai, "AoI Minimization Data Collection Scheduling for Battery-Free Wireless Sensor Networks," in *IEEE Transactions on Mobile Computing*, 2021. (Early access)
[16] Y. Dong, P. Fan and K. B. Letaief, "Energy Harvesting Powered Sensing in IoT: Timeliness Versus Distortion," in *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10897-10911, 2020.
[17] N. Nouri, D. Ardan and M. M. Feghhi, "Age of information-reliability trade-offs in energy harvesting sensor networks," *arXiv preprint*, arXiv:2008.00987, 2020.
[18] DK. Y. Yau et al., "Quality of monitoring of stochastic events by periodic and proportional-share scheduling of sensor coverage," in *ACM Trans. Sen. Netw.*, vol. 7, no. 2, pp. 18:1-18:49, 2010.
[19] Z. Yang, H. Gao, S. Cheng, Z. Cai and J. Li, "IEA: An Intermittent Energy Aware Platform for Ultra-Low Powered Energy Harvesting WSN," in *Springer WASA*, 2017, pp. 185-197.
[20] LED lamp. https://item.taobao.com/item.htm?spm=a1z09.2.0.0.7e3d2e8dCuYkIp&id=596556502253&_u=j253garl099d
[21] H. Yang and Y. Zhang, "Analysis of Supercapacitor Energy Loss for Power Management in Environmentally Powered Wireless Sensor Nodes," in *IEEE Transactions on Power Electronics*, vol. 28, no. 11, pp. 5391-5403, 2013.
[22] T. L. Porta, C. Petrioli, C. Phillips and D. Spenza, "Sensor Mission Assignment in Rechargeable Wireless Sensor Networks," in *ACM Trans. Sen. Netw.*, vol. 10, no. 4, pp. 60:1-60:39, 2014.
[23] X. Lin *et al.*, "Concurrent Task Scheduling and Dynamic Voltage and Frequency Scaling in a Real-Time Embedded System With Energy Harvesting," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 11, pp. 1890-1902, 2016.
[24] C. Lin, F. Gao, H. Dai, J. Ren, L. Wang and G. Wu, "Maximizing Charging Utility with Obstacles through Fresnel Diffraction Model," in *Proceedings of IEEE INFOCOM*, 2020, pp. 2046-2055.
[25] J. Hester and J. Sorber, "Flicker: Rapid Prototyping for the Battery-less Internet-of-Things," in *ACM SenSys*, 2017, pp. 1–13.
[26] A. Y. Majid, P. Schilder and K. Langendoen, "Continuous Sensing on Intermittent Power," in *ACM/IEEE IPSN*, 2020, pp. 181-192.
[27] M. Afanasov, N. A. Bhatti *et al.*, "Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus," in *ACM SenSys*, 2020, pp. 368–381.
[28] K. Geissdoerfer and M. Zimmerling, "Bootstrapping Battery-free Wireless Networks: Efficient Neighbor Discovery and Synchronization in the Face of Intermittency," in *USENIX NSDI*, 2021, pp. 439-455.
[29] Y. Zhang, H. Gao, H. Chen, Y. Li and D. Tien, "EIF-Export: An energy-interference-free data export approach for EH-WSN Networks," in *IEEE ICSEng*, 2018, pp. 1-6.
[30] M. Oded and L. Rokach, "Data mining and knowledge discovery handbook," *Springer*, 2010.
[31] F. Fraternali, B. Balaji *et al.*, "Aces: Automatic configuration of energy harvesting sensors with reinforcement learning," in *ACM Transactions on Sensor Networks*, vol. 16, no. 4, pp. 36:1-36:31, 2020.
[32] Supplement Material of the paper "Maximum AoI Minimization for Target Monitoring in Battery-free Wireless Sensor Networks". https://github.com/bingkunyao/aoipaper/blob/main/aoi-appendix.pdf
[33] A. Valehi and A. Razi, "Maximizing Energy Efficiency of Cognitive Wireless Sensor Networks With Constrained Age of Information," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 643-654, 2017.
[34] I. Kadota and E. Modiano, "Minimizing the Age of Information in Wireless Networks with Stochastic Arrivals," in *Proceedings of ACM Mobihoc*, 2019, pp. 221–230.
[35] S. Leng and A. Yener, "Age of Information Minimization for Wireless Ad Hoc Networks: A Deep Reinforcement Learning Approach," in *Proceedings of IEEE Globecom*, 2019, pp. 1-6.
[36] I. Kadota, A. Sinha and E. Modiano, "Scheduling Algorithms for Optimizing Age of Information in Wireless Networks With Throughput Constraints," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1359-1372, 2019.
[37] A. M. Bedewy, Y. Sun and N. B. Shroff, "The Age of Information in Multihop Networks," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1248-1257, 2019.
[38] Y. P. Hsu, E. Modiano and L. Duan, "Scheduling Algorithms for Minimizing Age of Information in Wireless Broadcast Networks with Random Arrivals," in *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2903-2915, 2020.

Bingkun Yao received the B.S. degree in computer science from the Harbin Institute of Technology, Harbin, China, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology.

His current research interests include energy harvesting wireless sensor networks and in-network data processing.

Hong Gao is a Professor in the School of Computer Science and Technology, Harbin Institute of Technology. She received the BS degree in computer science from Heilongjiang University, the MS degree from Harbin Engineering University, China, and the PhD degree in computer science from Harbin Institute of Technology. Her research interests include graph data management, sensor network, and massive data management.

Yang Zhang received the BS and MS degrees from Heilongjiang University, and he is currently working in Key Laboratory of Mechatronics, Heilongjiang University. His research areas focus on wireless sensor networks.

Jinbao Wang received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, 2006, 2008, 2013, respectively. He is currently an Associate Professor in Database with the School of Computer Science and Technology, Harbin Institute of Technology. His research interests include big data analytic, wireless sensor networks and data privacy.

Jianzhong Li was a Visiting Scholar with the University of California at Berkeley, USA, as a Staff Scientist with the Information Research Group, Lawrence Berkeley National Laboratory, USA, and as a Visiting Professor with the University of Minnesota, USA. He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, China. His research interests include data management systems, sensor networks, and data intensive computing.