Hindawi
Journal of Sensors
Volume 2021, Article ID 9999403, 15 pages
https://doi.org/10.1155/2021/9999403

Research Article

An Energy-Efficient One-Shot Scheduling Algorithm for Wireless Sensor Networks

Zeng Bo,[1] **Yabo Dong**,[2] **Jie He**,[2] **and Lu Dongming**[2]

[1]*School of Information Technology, Luoyang Normal University, China*
[2]*College of Computer Science and Technology, Zhejiang University, China*

Correspondence should be addressed to Zeng Bo; wbzeng.hn@gmail.com

Received 13 March 2021; Revised 15 October 2021; Accepted 1 November 2021; Published 22 November 2021

Academic Editor: Stelios M. Potirakis

In low-load wireless sensor networks, the power consumption of the node consists mainly of two parts: data transmission and node state switching. The lower node workload causes low energy consumption on data transmission, and the state switching energy of the node cannot be ignored. This paper proposes a one-shot time division multiple access (TMDA) scheduling with unlimited channels (*SUC*) on the assumption that the number of available channels is unlimited. *SUC* combines the receiver-based consecutive slot allocation with channel allocation, which minimises the number of node state switching and optimizes energy efficiency. Theoretical analysis demonstrates that the number of channels required by *SUC* does not exceed $\log_2^{N+1}$, where N indicates the number of nodes. Seeing that the number of available wireless channels is limited in practice, the paper proposes the scheduling with limited channels (*SLC*) and uses a Lookahead Search mechanism to solve slot conflict. For the scalability of the algorithm, a distributed implementation based on the token change is proposed. The algorithm uses the depth-first-search (*DFS*) to pass the token to all nodes and terminates slot and channel assignment. The simulation results show our algorithm can reduce the energy consumption by minimizing the number of state switching and shorten the data aggregation time by reusing slots among nodes.

1. Introduction

Wireless sensor networks (WSNs) have the advantages of low power consumption, low cost, and flexible deployment and are widely used in data aggregation applications [1, 2]. A sensor network is a collection of sensing devices, each of which has limited power supply, memory, processing ability, and transmission power. The sensed data arrives at sink through one or more hop(s) to deliver the information. The routing tree is a typical form of network organization for these applications. Since the many-to-one traffic pattern of the WSNs, nodes must switch their state (that is, switch from sleep to receive/transmit or vice versa) frequently during data transmission. The main reason for this is that nodes with child need to forward their child's data in discrete time slots. In [3, 4], the authors proved that the energy consumption spent on nodes' state switching is nonnegligible. Furthermore, in low-load sensor networks, time and energy for data transmission and reception are limited. However, the energy of the node state switching is a necessary overhead, which results in considerable energy consumption on the node state switching.

A way is to minimize the number of node state switching by assigning consecutive slots to nodes based on the workload of the node. In [5], the authors research the problem of energy wastage caused by the state switching and developed a contiguous sleep scheduling algorithm, which assigns sensors with consecutive slots to reduce the times of state switching. The cost is increasing the data transmission delay. HTSAS [6] focuses on the energy consumption and delay caused by the state switching and consecutive slot assignment. The consecutive slots are assigned to each node based on their layer in the routing tree. HTSAS is a single channel scheduling algorithm and imports a multiparent node which leads to extracomplexity for the building of routing tree. These algorithms cannot always guarantee that the number of state switching of the node is the least. That is, the node wakes up from the sleep to receive the data of the child node

and then forwards all data including the data produced by itself to the parent. After that, the node goes to sleep to save energy.

In this paper, we formulate the problem as the one-shot TDMA scheduling. Figure 1 shows the difference between consecutive slot assignment and one-shot TDMA scheduling. It can be seen from the figure that the one-shot TDMA scheduling minimizes the number of state switches for all nodes by continuously allocating receiving and transmitting time slots for the nodes in the network. It is obvious that the one-shot TDMA scheduling can reduce energy consumption and delay.

The above discussion motivates us to design a one-shot TDMA scheduling algorithm, so that the energy cost of the node state switching will be minimal.

The main contributions of this paper are as follows.

(i) We consider the constraints of parallel data transmission and energy consumption and formalize the one-shot TDMA scheduling problem

(ii) A receiver-based consecutive slot assignment strategy is proposed. This strategy ensures that the number of state switches for all nodes in the network is only twice in each data aggregation cycle

(iii) In order to satisfy the one-shot TDMA scheduling, a greedy multichannel allocation method is adopted. We proved that the number of channels required by our algorithm does not exceed $\log_2^{N+1}$, where N denotes the number of nodes in the network

(iv) We take into account the constraint on the number of channels available in the actual application environment and design the one-shot TDMA scheduling with limited number of channels

Differently from the early conference papers [7], we formalized the one-shot TDMA scheduling problem and perfected the theoretical analysis and proof in this paper. Considering the challenges posed by large-scale wireless sensor networks, we also propose a distributed implementation of the algorithm. Finally, we analyze the performance of the one-shot TDMA algorithm through simulation experiments.

The paper is organized as follows: a brief discussion of related work about energy efficiency for wireless sensor networks in Section 2. Section 3 described the problem of the energy cost of node state switching and formulated the one-shot TDMA scheduling. Section 4 introduces the one-shot TDMA scheduling algorithm *SUC* and *SLC*. A mathematical analysis of their performances is included. Section 6 demonstrates the experimental results for performance evaluation by comparing the proposed method against other well-known previously designed algorithms. We end the paper in Section 7 with a conclusion and future work.

2. Related Work

In WSNs, many multichannel scheduling protocol have been proposed; they are usually used to improve the ability of parallel transmission or reduce transmission delay. Some results improve energy efficiency through reducing idle listening or overhearing.

MMSN [8] is the first multichannel MAC protocol proposed for WSNs. It is a time-sharing CSMA mechanism. In order to save energy and communication overhead, MMSN assigns different channels for nodes in two hops. It uses a snooping mechanism to detect packet transmission on different channels, which causes the nodes to switch on different channels. MMSN shows that the uniform random backoff algorithm cannot effectively avoid contention and propose a nonuniform backoff algorithm. Experimental results show that MMSN achieves good parallel transmission capability and energy efficiency. Wu and Tseng [9] adopt the top-down time slot allocation method. The time slot of each child node be assigned on the basis of the parent node time slot, thus ensuring that the time slot allocation of the parent and child nodes is continuous. The number of node state transitions is reduced. In [10], the authors take the level of the node in the routing tree as the priority and then assign the time slot to the node according to the priority. Since the priority of the parent node is higher than that of the child node, the algorithm can provide the continuity of the time slot of the parent and child nodes, thus reducing the number of state switches of the parent. Abedi and Razaghi Kariznoi [11] propose a cross-layer multichannel optimization protocol, which combines MAC and opportunistic routing. Nodes can sleep and wake up adaptively according to load and energy level. The dynamic adjustment of channel load based on the number of nodes competing for the same channel and channel utilization balances the load of each channel. The simulation results show that the network lifetime is obviously prolonged. DRCS [12] is a distributed receiver-based multichannel routing protocol, which adopts routing and channel selection dynamically. DRCS exploits asynchronous duty-cycling to avoid energy wastage from overhearing. Each node chooses a least-used channel in its neighborhood as its receiver channel and then announces to its neighbors by broadcasting in a round-robin approach. When a node needs to transmit data, then it selects the channel according to the battery-health of nodes on a channel and expected transmission count (ETX) [13] on that channel. It is worth noting that the battery-health and ETX are evaluated periodically, which helps to dynamically select the best forwarding node and channel, so as to ensure the load balance between channel and node. Wu et al. [14] analyze the difference of energy consumption of nodes in different states and propose a node active scheduling strategy based on continuous time slot allocation for energy consumption of node state switching. The paper takes node workload as weight and then ranks nodes in descending order of weight. The consecutive time slots allocated by the node match their weights, thereby reducing the node state switching energy consumption. However, the time slot allocated for the node for receiving data and the time slot for transmitting data are not connected. Ma et al. [15] consider the energy consumption of node state switching. They construct a fusion conflict graph based on the network connectivity graph and the conflict graph and allocate continuous

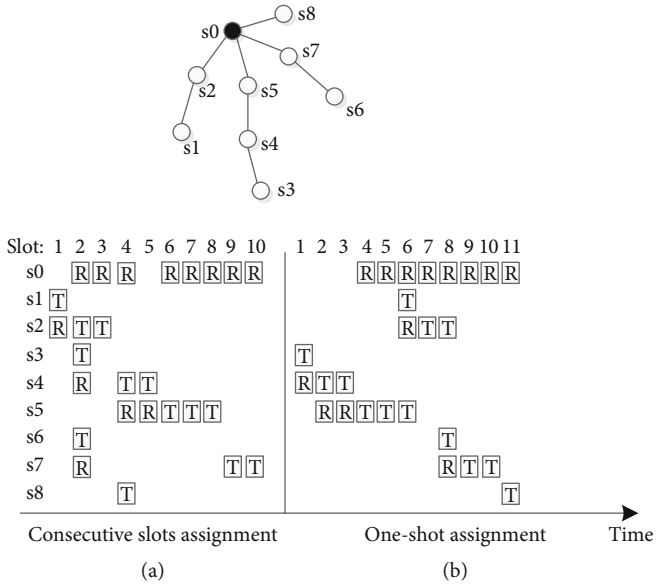

Figure 1: One-shot scheduling versus consecutive slot assignment.

time slots for each node based on the fusion conflict graph, thereby reducing the number of node state switching. However, the receive and transmit time slots allocated for a node are separate, and for most nonleaf nodes in the routing tree, they must perform two state switches to be able to complete data reception and transmission.

Cheng and Ho [16] propose a robust data transmission protocol through increasing the packet delivery ratio (PDR). Adaptive channel access (ACA) algorithm is used to overcome the interference imposed by WLAN as well as optimal channel selection. Each node contributes to the selection of the optimal channel by performing PDR measurements which is done by using channel scanning. The optimal channel which has the higher PDR value is assigned to the node. Since, each node periodically updates PDR; therefore, it may consumes additional energy. PCA-MC proposed in [17] provides quality of service (QoS) guarantee for multichannel WSNs in smart grid application. A modified receiver-based channel assignment (RBCA) [18] is adopted as the MAC protocol. In modified RBCA, the interfering degree of the nodes is calculated based on the SINR value, then a channel is assigned to the node which has a maximum interfering degree. The node which has not been assigned to channel is marked as the interfering node. The channels are assign for these interfering node in the time slot assignment phase. To realize service differentiation, a delay-aware data collection model is used for ensuring that each type of packet is marked with appropriate priority. The packets are listed in descending order by their priority and then transmit to the sink.

Tree-based multichannel protocol (TMCP) proposed in [19] divides the routing tree into multiple subtrees and assigns different channels to each subtree, thus reducing transmission interference between subtrees. After finished the channel assignment, TMCP assigns a time slot to each node. Similarly, a flow-based multichannel protocol is proposed in [20] (named MCRT). MCRT partitions the network, according to the data flow and assigns a channel to each partition, thus optimizes the end-to-end delay. Besides, MCRT also designed a well-established heuristic algorithm to optimize the number of channels. NDAS [21] optimizes the delay by exploiting the multichannel and asynchronous duty-cycle for WSNs. Two types of conflict graphs are proposed to describe the relationship of data transmission links and then present two coloring algorithms to assign time slots or channel for each link. A data aggregation tree is built based on the concept of the maximum independent set. Three types of nodes are marked with black, blue, and white, respectively. NDAS firstly schedules white nodes and then blue and black nodes. Then, NDAS schedules link at time slot t based on the following two conditions. First, the receiver of the link must be active and not be scheduled to receive data from other sender. Second, the links in the subtree rooted at the sender already have been scheduled before time slot t. Because child should always be scheduled before its parent, the delay is optimized. DMPMC [22] is a distributed multipower and multichannel scheduling algorithm for clustering WSNs. Two-level transmission power is used to support the communication intracluster and intercluster, respectively. After the nodes are clustered, a routing tree containing only cluster head is formed. Cluster head is listed in ascending order based on their ID and allocate channels in turn. Simulation results show that DMPMC achieves the low transmission delay. But, due to the use of unique channel per cluster head, DMPMC requires as many channels as cluster heads, which makes it very unsuitable for large-scale and dense WSNs. DCAS proposed in [23] is a distributed collision-free scheduling algorithm aimed at minimizing latency. DCAS assumes that each node has the 3-hop neighboring information. The node then constructs local extend relative collision graph [24] and local data-forwarding graph based on its neighborhood information. An independent set for each time slot is formed based on the previous collision graph for ensuring collision-free data transmission. MCAS-

MAC [25] is designed based on AS-MAC [26]. Unlike AS-MAC, MCAS-MAC asynchronously schedules the wake-up and sleep time of nodes and meantime has back-to-back transmission and multichannel support for WSNs with high traffic. When a node joins into the network, it first selects a least used channel as its home channel based on the gathered schedule and channel information. These information are announced by the other nodes by using Hello message. When a node needs to transmit data, sender chooses a intended receiver and then switches to receiver's home channel. When the receiver wakes up, the sender immediately sends all data in the queue to the receiver. If there are multiple queued data to be transmitted, MCAS-MAC introduces additional dwell time to transmit them. The evaluation of performance shows that MCSA-MAC improves the reliability of packet delivery and delivery latency.

Recently, machine learning-based routing [27] has become an increasingly popular direction. By analyzing empirical or historical data, machine learning-based routing can continuously improve the performance of sensor networks, such as energy efficiency, transmission reliability, and delay. However, in practice, sensor nodes only have limited computing and storage resources, which will be one of the limitations of applying machine learning algorithms.

At last, we compare these multichannel protocols and present the results in Table 1. Most multichannel protocols optimize energy efficiency and delay and use single path to aggregate data. These protocols give full play to the advantages of multichannel. However, for some asynchronous protocols [8, 12, 16, 25], how to realize efficient data transmission between sender and receiver is still a challenging problem. The multichannel protocol based on slot allocation [17, 19, 22, 23] needs to solve the problem of efficient synchronization of nodes and even the whole network.

In summary, several algorithms [14, 15] optimize the node state switching by allocating consecutive time slots. However, due to the constraint of shared channel, the node receiving and transmitting slots are not continuous. In these algorithm, the number of state switching of some nodes is at least twice. Based on this, this paper ensures that the data reception and transmission slots of each node are continuous during the data transmission process by arranging the slots of the nodes on different channels. Thus, the number of node state transitions is minimized to one.

3. System Model and Assumption

3.1. Network System Models. We consider a data collection network consists with a sink and N sensor nodes (that is, $V = v_1, v_2, \cdots, v_n$). The sensors were randomly deployed in a 2-dimensional region. Each node is equipped with a half-duplex wireless transceiver. For simplicity, we assume that nodes always use a fixed transmitting power (named E_t) to communicate with its neighbor. The neighbor is a collection of nodes that are within the effective communication distance (named R). In here, R is determined by E_t. Furthermore, the communication between node v_i and v_j is reliable if euclidean distance $d_{v_i,v_j} \leq R$.

The energy consumption of node is mainly composed of four parts: transmitting, receiving, listening radio signal, and sampling data. In this paper, we mainly consider the energy consumed by switching between state A and state B for a node. Table 2 summarizes typical values of Tmote Sky mote for different operations.

In order to use the TDMA to schedule the activity of the node, we assume that time is divided into slots with length t_s. The slots are organized in the form of *schedule frame*. The *schedule frame* is repeated in each data collection cycle.

3.2. Problem Description. In this section, we describe the problem studied in this paper. To describe the energy consumption of nodes' state switching, the energy ratio is calculated using

$$p = \frac{E_{sw}}{E_{sw} + E_t}, \tag{1}$$

where E_{sw} denotes the energy cost of node state switching and E_t denotes the energy consumption of data transmission. Using Equation (1), we illustrate the energy consumption of the Tmote sky with different initial workloads in Figure 2. The parameters in Table 2 are referred in [30]. Based to the experimental result shown in Figure 2, the energy consumption of node state switching is higher than that on data transmission when the nodes have low workload (e.g., 16 bytes). In practice, for many real-world environment monitoring applications (e.g., sensor networks for collecting temperature and humidity information), each node may produce only 20 bytes per duty-cycle. Therefore, the energy cost of node state switching is one of the main factors in energy consumption for this type of application.

We define the energy consumption of nodes in different states. Table 3 gives the symbol used in this section. For a node v_i, if it is scheduled to transmit at slot t, we denote it as $T_i^t = 1$; otherwise, we denote it as $T_i^t = 0$. Similarly, we use R_i^t to denote whether the node v_i is scheduled to receive at slot t or not. We also use $E_{P,T}$ and $E_{P,R}$ to denote energy consumption by state transition. In fact, the energy consumption from active state to sleep sate is very low compared to switch from sleep to active state. Therefore, this part of energy is often ignored.

For node v_i, its energy consumption during a scheduling period is $\Sigma_{t=1}^{T}(T_i^t * E_T + R_i^t * E_R + T_i^t * E_{P,T} + R_i^t * E_{P,R})$. The energy cost for state-transition is $\Sigma_{t=1}^{T} C_{i,T}^t * E_{P,T} + C_{i,R}^t * E_{P,R})$. The objective of a schedule S is to minimize these two energy consumption. If the node has fully consecutive slots, we have $\Sigma_{t=1}^{T}(C_{i,T}^t + C_{i,R}^t) = 1$.

The one-shot TDMA scheduling problem is defined as follows: given a data collection tree T, how to ensure that each node in T wakes up only once during each data collection cycle, and the wireless channel used by the algorithm is the least (named CN). The optimization problem is stated as

Table 1: Comparison of multichannel scheduling protocol for WSNs.

Protocols	Special focus	Simulation platform	No. of paths	Network size	Topology
MMSN [8]	Throughput, energy efficiency	GloMoSim [28]	Single	Small	Tree
PCA-MC [17]	QoS	Matlab-based simulator	Single	Small	Tree
Wu et al. [9]	Energy efficiency	Matlab	Single	Small	Tree
DRCS [12]	Network lifetime, good PDR	MICAz motes, Castalia simulator	Single	Small and large	Hybrid
TMCP [19]	Throughput, packet losses	GloMoSim	Single	Large	Tree
MCRT [20]	Delay	Tmote motes, NS-2	Multiple	Large	Tree
NDAS [21]	Delay	Matlab	Single	Small and large	Tree
DMPMC [22]	Delay, energy efficiency	Matlab	Single	Small and large	Clustering
MCAS-MC [25]	Packet delivery ratio, delay	Mica2 motes, RePTEX [29]		Small	Hybrid
DCAS [23]	Delay	Matlab	Single	Small and large	Hybrid

Table 2: Parameters of Tmote Sky mote.

Parameter	Value
Transmission current	45 mW
Reception current	79 mW
Start-up current	9 mW
Duration of radio module start-up	5 ms

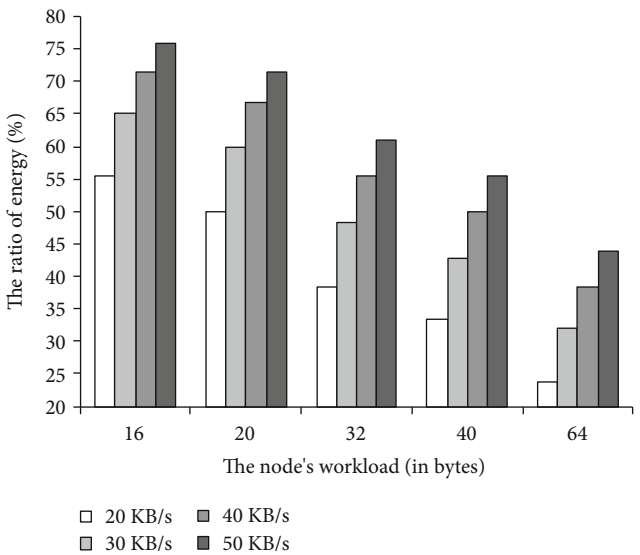

Figure 2: The energy ratio when node has different workload. We use Tmote sky mote as an example.

follows:

$$\begin{aligned} &\text{Minimize } \Sigma_{t=1}^{T}\left(C_{i,T}^{t}+C_{i,R}^{t}\right), \text{Minimize } CN,\\ &\text{s.t } C1: T_i^t = R_i^t + 1, 1 \le i \le n,\\ &C2: T_i^t + R_i^t = 1, 1 \le i \le n,\\ &C3: T_i^t + T_k^t > 1, 1 \le i \le n, 1 \le k \le n. \end{aligned} \tag{2}$$

The first condition ensures that the data produced by itself and delivered by its child is reliably forwarded to next hop. The second condition ensures that the node only be in transmitting or receiving at any slot t. The third condition ensures that the transmission of node v_i and node v_k is interference-free only when two nodes use different channels at slot t.

4. One-Shot TDMA Scheduling Algorithm

The first proposed in this section is the one-shot TDMA scheduling algorithm with no limit on the number of available channels. However, considering that the number of available channels is limited in an real-word wireless communication system, another scheduling algorithm with a limited number of available channels is proposed. They are all centralized algorithms. In the next section, we will present a distributed implementation of our algorithm.

4.1. One-Shot TDMA Scheduling with Unlimited Channels (SUC). The *SUC* algorithm consists of two phases: the workload collection phase is used to calculate the workload of each node in the network; the receiver-based transmission slot allocation phase is used to assign consecutive slots to each node based on their workload. In order to ensure that the slot arrangement of each node meets the one-shot scheduling requirements, and the scheduling of the nodes is collision-free, these consecutive slots are arranged to different wireless channels.

Phase one: workload collection. In a data aggregation sensor network, nodes periodically generate data and deliver data to the sink in a multihop manner. In this data transfer process, we assume that each hop passes the raw data, that is, the data is not aggregated by the node. Based on this, we can use $W_i = W_{si} + w_i$ to compute the node v_i's workload, where W_i is the node v_i's total workload, w_i denotes the amount of data produced by node v_i during a data collection cycle, and W_{si} denotes the total amount of raw data generated by a subtree with node v_i as the root node in one data collection cycle. The workload collection is completed using depth-first-search (*DFS*) which is widely used to resolve the node traversal problem. Figure 3 illustrates the node workload collection. Each node produces one raw data and delivers it to the parent in one slot. Notice that our algorithm can be easily applied to scenarios where the initial workload of node is different.

Table 3: Symbol notation.

Symbol	Meaning
T_i^t	Indicator whether node v_i transmitting at slot t
R_i^t	Indicator whether node v_i receiving at slot t
$C_{i,T}^t$	Indicator whether node v_i switches to transmit at slot t
$C_{i,R}^t$	Indicator whether node v_i switches to receive at slot t
E_T	Energy consumption of transmitting in a slot
E_R	Energy consumption of receiving in a slot
$E_{P,T}$	Energy consumption from sleep to transmitting
$E_{P,R}$	Energy consumption from sleep to receiving

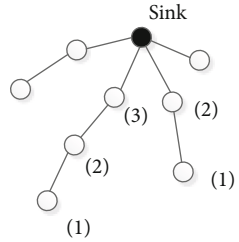

Figure 3: Workload example.

Phase two: receiver-based transmission slot assignment. The receiver-based transmission slot assignment starts from the sink. First, according to the workload of the child, all the children of the sink are allocated consecutive transmission slots for transmitting the collected raw data to the sink. Subsequently, in order to ensure that the slot assignment of the nodes meets the one-shot scheduling requirement, each node allocates consecutive transmission slots to its children based on its own transmission slot. For any node v_i, the number of slots assigned to it is determined by its workloads W_i calculated in the front. We assume node v_i turns to assign slots, and it has j children. For node v_i, its first transmission slot is t_i, then its last transmission slot is $t_i + W_i - 1$. Node v_i works as follows to guarantee the slot assignment meets the requirement of one-shot TDMA scheduling.

(i) Step one: node v_i assigns consecutive transmission slots to its children. In order to guarantee one-shot scheduling of node v_i, the slot assigned to the child must be smaller than the first slot of node v_i. Therefore, for node v_i's child k $(1 \le k \le j)$, its first slot is $v_i - \sum_{m=1}^{k-1} W_m$, and the last slot is $v_i - \sum_{m=1}^{k-1} W_m - 1$, where W_m denotes the workload of the child m of node v_i. If node k is the first child of node v_i, e.g., $k = 1$, then we have $v_i - \sum_{m=1}^{c} W_m - 1 = v_i - 1$. As the time slot allocation is performed layer by layer from top to bottom, there may be a case where the first slot number of the node is less than 0. Therefore, when the slot allocation of all nodes is completed, the slot number of the node needs to be adjusted. The slot number adjustment method is as follows: the slot number less than 0 is defined as an *offset*, and the maximum *offset* is the absolute value of the minimum slot number (less than 0). Then, each slot of the node is added to the *offset*. This operation ensures that the slot number of the node is a positive number greater than 0

(ii) Step two: channel allocation. To minimize the number of wireless channels, *SUC* uses a greedy assignment mechanism. We define the candidate channel list as *CCL*. When a node v_i assigns a channel to child k, it first checks the assigned slot information on each channel in *CCL*. If there is a channel (for example, ch_i) in *CCL* that satisfies the requirement of collision-free slot allocation for child k, then channel ch_i will be assigned to child k; otherwise, a new channel ch_{i+1} will be assigned to child k and added to the *CCL*. Hence, we have $CCL = CCL \cup ch_{i+1}$

Algorithm 1 is the pseudocode of *SUC*. In this algorithm, t_j is the last slot assigned to node v_j, and W_j denotes the workload of node v_j. AS_{ch_j} includes all the slots active in channel ch_j.

4.2. An Illustration. This section uses an example to illustrate the *SUC* algorithm. Figure 4 shows the subtree used in the example; the set of nodes is $\{s1, s2, s3, s4, s5\}$. According to the *SUC* algorithm, the time slots and channels of these nodes are allocated as follows:

Step 1. The workload collection: the node workload is collected using DFS technology and the results are as follows (in slots):

(i) s1: $W_1 = W_{s1} + W_{s2} + W_{s3} + W_{s4} + W_{s5} = 5$

(ii) s2: $W_2 = W_{s2} + W_{s3} + W_{s4} = 3$

(iii) s3: $W_3 = W_{s3} = 1$

(iv) s4: $W_4 = W_{s4} = 1$

(v) s5: $W_5 = W_{s5} = 1$

Step 2. The receiver-based transmission slot assignment: the transmission slot allocation starts from the node $s1$. The slot scheme of the node is as follows:

(i) s1: (t_1, t_5);

(ii) s2: $(t_5 + 1, t_5 + W_2) = (t_6, t_8)$;

(iii) s3: $(t_8 + 1, t_8 + W_3) = (t_9, t_9)$;

(iv) s4: $(t_8 + 1 + W_3, t_8 + W_3 + W_4) = (t_{10}, t_{10})$;

(v) s5: $(t_5 + 1 + W_2, t_5 + W_2 + W_5) = (t_9, t_9)$;

Next, we will adjust the slot assignment. According to the allocated slot, the largest slot number is t_{10}. Therefore, the slots of each node are adjusted as follows:

(i) s1: $(t_{10-5}, t_{10-1}) = (t_5, t_9)$;

(ii) s2: $(t_{10-8}, t_{10-6}) = (t_2, t_4)$;

Input:
Data Collection Tree: T
The node set: V
Output:
Scheduling frame for T
1: **while** $V \neq \varnothing$ **do**
2: Finding the node v_j by using *DFS*
3: Assigning slots $AS_j = [t_j - W_j + 1, t_j]$ to v_j
4: ch_j=1;
5: $t_l = t_j - W_j + 1$;
6: **while** $(t_l < = t_j)$ **do**
7: **if** $t_l \in AS_{ch_j}$ **then**
8: $ch_j = ch_j + 1$;
9: $t_l = t_j - W_j + 1$;
10: **if** $t_l == t_j + 1$ **then**
11: $AS_{ch_j} = AS_{ch_j} \cup AS_j$
12: Assigning AS_j and ch_j to v_j
13: $V = V - v_j$

Algorithm 1: Scheduling With Unlimited Channels (*SUC*).

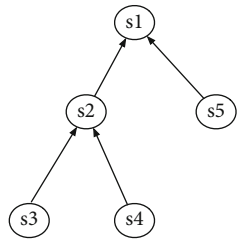

Figure 4: An example to illustrate the slot and channel allocation.

(iii) s3: $(t_{10-9}, t_{10-9}) = (t_1, t_1)$;

(iv) s4: $(t_{10-10}, t_{10-10}) = (t_0, t_0)$;

(v) s5: $(t_{10-9}, t_{10-9}) = (t_1, t_1)$;

Step 3. Channel allocation: note that in the slot allocation of all nodes, only slot t_1 is simultaneously allocated to $s3$ and $s5$. Therefore, in order to ensure that the data transmission between $s3$ and $s5$ does not conflict, the algorithm specifies different channels for $s3$ and $s5$, for example, $s3$ uses channel $ch1$, and $s5$ uses channel $ch2$. Since slots t_0, $t_2 - t_9$ are not reused, in order to minimize the number of channels used, all nodes are forced to use channel $ch1$. Therefore, the final channel allocation is as follows:

(i) t0: ch1

(ii) t1: ch1(s3), ch2(s5);

(iii) t2-t9: ch1

4.2.1. Analysis of Algorithm 1. In this section, we first prove the energy efficiency of *SUC* and then give the boundary of the number of channels required by our algorithm.

Theorem 1. *Given any data collection tree T, Algorithm 1 is always able to find an one-shot TDMA schedule S that enables energy optimization (say E_{opt}).*

Proof. For the application of raw data collection without data fusion, the energy consumption of a node can be divided into two parts: energy consumption of transmitting and receiving data (say E_D) and energy consumption of state-transition (say E_S). Since the node needs to wake-up at least once during the data collection process, the optimal energy consumption of state-transition $E^S_{\text{opt}} = n * e_s$, where e_s denotes the energy for state-transition and n denotes the number of nodes in the network.□ □

According to Algorithm 1, since the slot allocated for the child of the node always takes precedence over the node itself, the node will forward the data to its parent or sink only after the data of the child is completely collected. In this process of aggregating data, the node only needs to wake up once, that is $E_S = E^S_{\text{opt}}$. Let E be the total energy consumption in a one-shot TDMA scheduling. We get $E = E_D + E_S = E_D + E^S_{\text{opt}}$. Because E_D is fixed for a given data collection tree, and it only depends on the structure of data collection tree, we have $E^D_{\text{opt}} = E_D$, where E^D_{opt} is the optimum energy consumption for data transmit and receive. At least, we get $E = E^D_{\text{opt}} + E^S_{\text{opt}} = E_{\text{opt}}$.

For the TDMA scheduling algorithm, there is a *scheduling conflict* when different nodes reuse the same slot. To ensure the reliability of data convergence and the continuity of node slots, *scheduling conflict* problem must be solved. Algorithm 1 solves it by assigning different channels to these nodes. As a result, our algorithm can ensure that slot is interference-free. Based on this, we can get the following conclusion for the data collection tree T.

Theorem 2. *For data collection tree T composed with N sensor nodes, the number of channels CN is used by Algorithm 1 to establish the one-shot TDMA scheduling of T less than* $\log_2^{N+1}$.

Proof. According to the receiver-based transmission slot allocation algorithm, the transmission slot allocation process starts from the sink. Therefore, for each node in the tree, the transmission slot number of the child is always smaller than the slot number of the parent. For the channel allocation, Algorithm 1 assigns a new wireless channel to the node and adds it to the *CCL* if and only if there is a *scheduling collision* and the *scheduling collision* cannot be resolved using the wireless channel in the *CCL*. Therefore, the number of channels is equal to the maximum number of times a slot is reused among nodes. We discuss the channel requirements of Algorithm 1 in two different topologies.□ □

Type I: N-ray regular tree (such as binary tree and ternary tree): Considering the N-tree T of height h (counting from top to bottom begin with the root node), according

to the time slot assignment based on the receiver, the slot interval (denoted by T_1) assigned by each node in the child node set $\{s_{11}, s_{12}, \cdots, s_{1N}\}$ of level 1 in the tree T (remember for) is shown in

$$\begin{cases} \left[1, \frac{N^h-1}{N-1}\right], \\ \left[\frac{N^h-1}{N-1}+1, 2\frac{N^h-1}{N-1}\right], \\ \cdots\cdots, \\ \left[(N-2)\frac{N^h-1}{N-1}+1, N^h-1\right], \\ \left[N^h, N\frac{N^h-1}{N-1}\right]. \end{cases} \tag{3}$$

The total number of slots required is $N((N^h-1)/(N-1))$; the slot interval (recorded as T_2) assigned by each node in the set of child $\{s_{21}, s_{22}, \cdots, s_{2N^2}\}$ of level 2 is shown in

$$\begin{cases} \left[1-\frac{N^{h-1}-1}{N-1}, 0\right], \\ \left[1-2\frac{N^{h-1}-1}{N-1}, -\frac{N^{h-1}-1}{N-1}\right], \\ \cdots\cdots \\ \left[\frac{(N-1)(N^h-1)-(N^{h-1}-1)}{N-1}+1, N^h-1\right], \\ \cdots\cdots, \\ \left[\frac{(N-1)(N^h-1)-N(N^{h-1}-1)}{N-1}+1, (N-1)N^{h-1}\right]. \end{cases} \tag{4}$$

From the above derivation of the slot assignment of each node in the N-tree T with levels 1 and 2, respectively, it is known that at least one slot in T_1 is included to T_2, for example, N^h-1. Thus, according to the receiver-based slot assignment, it can be inferred that slot T_i assigned to the node set $\{s_{i1}, s_{i2}, \cdots, s_{iN^i}\}$ at level i in T always contains at least one slot in T_1. Therefore, for tree T, there is at least one slot k used by nodes located at different level in the tree. That is to say, there is a node set $S_k = \{s_1, s_2, \cdots, s_h\}$ corresponding to the slot k which consists of node located at different level in T. The size of the node set is equal to h.

Based on the above discussion, it can be concluded that for any N-tree T, the number of channels required for one-shot TDMA scheduling of T is $CN = h$.

Type II: random data collection tree: for a random data collection tree T, we assume that the sink has n children, denoted by $st_1, st_2, \cdots, st_n$, respectively, and uses N_i to represent the workload of node st_i. Since each node produces a workload during a data collection cycle, we have $N_1 + N_2 + \cdots + N_n = N$. For Algorithm 1, since the slots of the parent and child are contiguous, slot reuse can only occur between subtrees, for example, two subtrees rooted at st_1, st_2. We use δ to represent the maximum number of time slot reuses between subtrees. Then, the value of δ may be {1,2,3,4, ...}. Considering that different subtree combinations (for example, $\{st_1, st_2\}$, $\{st_1, st_3\}$, etc.) result in different δ, we can get a sequence of values of δ: $\{\delta_1, \delta_2, \cdots, \delta_m\}$. For Algorithm 1, we have $CN = \max(\delta_i), (1 \le i \le m)$. Based on the previous discussion of the N-ray regular tree, it can be known that δ is maximum when each layer of the tree reuses the same time slot. According to the graph theory, the number of nodes in a tree with h layer is $2^h - 1$. For the n subtrees of tree T, we have $2^h - 1 \le \max(N_1, N_2, \cdots, N_n)$, and so, $h \le log_2^{\max(N_1, N_2, \cdots, N_n)+1}$. Since $N_i \le N$, we have $h \le \log_2^{N+1}$. Based on the discussion for the N-ray regular tree, we have $\delta_i = h$. Moreover, we have $CN = \max(\delta_i)$, and so, $CN \le \log_2^{N+1}$.

4.3. Scheduling with Limited Channels. In Section 4.1, *SUC* implements one-shot TDMA scheduling with enough available channels. However, in practice, there is a limitation on the number of wireless channels over which a transceiver can use. Typically, the number of orthogonal channels in the 2.4 GHz band is less than 6. Therefore, we take into account this constraint on the number of channels and design an algorithm for the one-shot TDMA scheduling with given number of channels.

Compared with *SUC*, when we develop scheduling algorithm with limited channels, the major challenge is how to assign slots to nodes when the available channels are insufficient to ensure that the state switching times is optimized. In Algorithm 2, we use another simple greedy strategy to overcome this challenge. We assume that the number of channels is k.

The *Lookahead Search* described in Algorithm 2. Note that the slot included in a slot slice is consecutive.

4.4. Complexity Analysis. The main process of Algorithms 1, 2, and 3 is to collect the workload of node and assign consecutive slot to node. These two processes are completed by using DFS algorithm. The time is mainly consumed in finding adjacency nodes in routing tree, so the time complexity of these algorithms is $O(n^2)$, where n denotes the number of nodes in the networks.

5. A Distributed Implementation

In the algorithm *SUC*, the time slot and channel assignment of a node are assigned by the aggregation node. The advantage is that the energy consumption is saved. However, as the network scale increases, the probability of redundant exchange of node time slot and channel assignment information in the information distribution process will be greatly increased, and frequent exchange of redundant information between nodes will be lead to the waste of energy. One of the methods of adapting the *SUC* to a large-scale network is to adopt a distributed implementation. In this paper,

Input:
Data Collection Tree: T
The node set: V
the number of available channel k $(1 \le k)$
Output:
Scheduling frame for T
1: **while** $V \neq \varnothing$ **do**
2: Finding the node v_j by using *DFS*
3: Assigning slots $AS_j = [t_j, t_j - W_j + 1]$ to v_j
4: ch_j=1;
5: $t_l = t_j - W_j + 1$;
6: **while** $t_l \le t_j$ and $ch_j \le k$ **do**
7: **if** $t_l \in AS_{ch_j}$ **then**
8: $ch_j = ch_j + 1$;
9: $t_l = t_j - W_j + 1$;
10: **if** $t_l \le t_j$ and $ch_j > k$ **then**
11: Calling Algorithm 3
12: **else**
13: $AS_{ch_j} = AS_{ch_j} \cup AS_j$
14: Assigning AS_j and ch_j to v_j
15: $V = V - v_j$

Algorithm 2: Scheduling With Limited Channels (*SLC*).

a distributed implementation of *SUC* algorithm is described as follows.

The TOKEN for slot and channel assignment is first generated by the Sink, and TOKEN is passed to each node in the data aggregation tree by using DFS to accomplish the statistics of the total workload of the node. When TOKEN returns Sink, the total workload of any node i in the network can be calculated by $W_i = W_{si} + w_i$, where W_{si} is the total workload of the subtree rooted at node i, w_i indicating the amount of work generated by the node itself.

In the node slot and channel assignment phase, there are two types of information that need to be carried in TOKEN: (1) node's slots, expressed as (the workload: W, the ID of first slot: t); (2) channels' slot, expressed as (channel number: c, slot list: T_L). The process consists of three steps.

Step 1. Sink passes the TOKEN to the first leaf node, such as node i. To simplify the problem, we assume that the node i has the most workload. Next, node i begins to assign slot and channels to itself and stores this information to the T_L of node's schedule S, while storing the slot information into the TOKEN for return to the parent node. Since the phase of slot and channel assignment begins with node i, there is no conflict no matter slot or channel. After node i completes the assignment of slot and channel, the TOKEN is then passed to its parent.

Step 2. When the parent p receives the TOKEN from the child i, it first stores the slot information (W, t) and child's ID carried in the TOKEN into the R_L of its schedule, also named S. If the parent p does not complete the assignment of slot and channel, it will continue to perform slot and channel assignment. Otherwise, the parent p skips the slot and channel assignment. The slot and channel assignment of the parent p is as follows.

Based on the slot information (W, t) carried in the TOKEN, p can calculate its slot interval as $[t + W, t + W + W_p - 1]$. In the process of channel assignment, p looks for the T_L related with each channel to find a channel that can place its slot $[t + W, t + W + W_p - 1]$. If the channel cannot be found, an idle channel is added and the slot of p is associated with the new channel. After completing the slot and channel assignment, p stores the slot into its' T_L and then calls the *TOKEN processing routine*.

TOKEN processing routine: when p completes or skips the slot and channel assignment, the TOKEN may be delivered to p's child or parent, which are handled as follows:

(1) If TOKEN is passed to the child, p selects the child with the largest workload from the unassigned child set and passes the TOKEN. Before passing the TOKEN, p needs to specify the minimum slot number $t_{\min}$ in the schedule S to t and initialize w to 0

(2) If TOKEN is passed to the parent, and $t = t_p$, $w = w_p$, where t_p is the first transmitted slot number of p, and w_p denotes the total workload of p

Step 3. Assign slot and channel to child: when child, such as k, receives TOKEN, node k first calculates node's first slot number $t_k = t - w_k$ according to t in TOKEN and determines that node's slot interval is $[t_k, t_k + w_k - 1]$. In the process of assigning channels, node k looks for each T_L associated to channel to find the channels that can satisfy node k's slot requirements. If not found, a new idle channel is added, and the slot interval $[t_k, t_k + w_k - 1]$ of node k is arranged to the channel. After the slot and channel assignment is completed, node k first stores the slot assignment into the T_L of k's schedule S and then calls the aforementioned *TOKEN processing routine*.

After all the nodes in the network have completed the slot and channel assignment, TOKEN will return to the sink. Note that the above distributed implementation completes the slot and channel assignment only by token transfer. The overhead of the algorithm is only caused by the TOKEN transfer between the nodes. The algorithm has a low complexity.

6. Simulation Results

This section summarizes the performance of proposed algorithm. The MATLAB is used as the simulation platform, and implements one-shot TDMA scheduling and two comparison algorithms centralized TDMA scheduling (named *centralized*) [5] and multihop-TDMA. The *centralized* is proposed to reduce the energy consumption of nodes' state transition. It reduces the number of slots used for scheduling by reusing slots. The main difference between our algorithms and *centralized* is that *centralized* uses only one channel. Therefore, when the *centralized* reduces the number of node state switching, it cannot guarantee that the transmitting slot and the receiving slot of the node are completely

Input: an initial slot slice $[t_j, t_j + W_j - 1]$ assigned to node j;
the channel ch_j which has the minimum traffic load
Output: a new slot slice $[t_m - W_j, t_m]$ for node v_j
1: **if** $t_l \in [t_j, t_j + W_j - 1]$ and t_l is the slot assigned to nodes by all k channels **then**
2: Calculating the maximum unassigned slot number $t_m = t_l - 1$ and a new slot slice $[t_m - W_j, t_m]$
3: **for** each slot $t_k \in [t_m - W_j, t_m]$ **do**
4: t_k is unassigned in channel ch_j and so could be allocated to node v_j
5: **if** t_k cannot be assigned to node v_j **then**
6: $t_m = t_k$
7: Recalculating slot slice $[t_m - W_j, t_m]$
8: Assigning slot slice $[t_m - W_j, t_m]$ and the channel ch_j to node v_j

Algorithm 3: Lookahead Search.

continuous. In addition, we also implemented a *multihop-TDMA* without slot reuse as the baseline. Since the distributed scheduling is the same as the centralized scheduling on the core mechanism, we only report the results of the *centralized* scheduling.

The typical topology used in our all experiments shown in Figure 5. The experimental results are the mean value of 10 random network topology. All nodes are randomly placed in a 2D region of no more than 100 m × 100 m. The transmission distance is 12 m. Each node produces only a unit data which can be transmitted completely in a slot in a data collection cycle. In other words, the initial workload of the node is one packet. We form a shortest-path data collection tree to study these algorithms. In our simulations, each node produces one packet in a data collection cycle. Our simulations use the Tmote Sky mote, and its parameters are shown in Table 2.

6.1. Impact of the Network Scale. Figure 6 shows the number of channels used by our algorithm in a network topology with different hop counts. We can conclude that the number of channels used by our algorithm is increasing as the number of hops in the network increases. Furthermore, N-ray regular tree uses the most channels to guarantee the one-shot scheduling in the most simulation experiments. The main reason is that nodes with different hops in the N-tree will reuse the same time slot. Therefore, in order to avoid wireless interference between these nodes, each node must use a different channel, resulting in the number of channels being equal to the height of the tree. In a random shortest path tree, the probability that the same time slot is reused by nodes located at different hop counts is low, and therefore, less time slot reuse between nodes reduces the number of channels.

The number of slots required for scheduling generated by different algorithm is shown in Figure 7. Compared with *SUC*, the number of slots is large due to the lack of slot reuse when nodes use *multihop-TDMA* algorithm. As shown in the figure, as the number of nodes continues to increase, the number of slots required for scheduling generated by all algorithms is also significantly increased. Since *multihop-TDMA* does not employ the slot reuse mechanism, it generated the scheduling with the largest number of slots. Correspondingly, the data delivery delay is also the largest. As for the *SUC* and *centralized*, since both algorithms have a slot reuse mechanism, the number of slots they need is not much different. Since data transmission occurs on orthogonal channels, *SUC* can provide the reliable data delivery. However, the *centralized* uses the protocol interference model. Since the protocol interference model only models the wireless interference between nodes in the two-hop range, it cannot fully represent the wireless interference between nodes in the actual environment. The scheduling produced by the *centralized* may not easy to guarantee the reliable data transmission.

The energy consumption of nodes' state switching is shown in Figure 8. Since the number of orthogonal channels that a node can use is unlimited, the scheduling generated by *SUC* ensures that each node only need to switch states once and thus has the lowest energy consumption. For *multihop-TDMA* and *centralized*, since the receiving slots and transmitting slots of the node are discontinuous, the node has to switch from sleep to receiving or from sleep to the transmitting state many times, thereby increasing the energy consumption of node.

6.2. Impact of the Number of Channels. In a real-world network, the number of channels that a node can use is very limited. Therefore, we evaluate the performance of *SLC* in this section. We first measure the energy consumption for state switching and then evaluate the data collection time derived by *SLC*. In all experiments, we use the *multihop-TDMA* and *centralized* as the benchmark.

Figure 9 shows the total energy consumption of state switching during a data collection cycle. It can be concluded that as the number of available channels increases, more nodes can complete one-shot TDMA scheduling, thereby reducing the energy consumption of sate switching. After the number of channels is 6, all the nodes in the network need only switch the state once to complete the data collection, and the energy consumption of state switching is the least.

The relationship between the number of available channels and the data collection time is shown in Figure 10. Among the three algorithms, the scheduling generated by *multihop-TDMA* requires the most slots. When the number of available channels is less than 4, the scheduling derived by *SLC* requires more slots than the *centralized* algorithm.

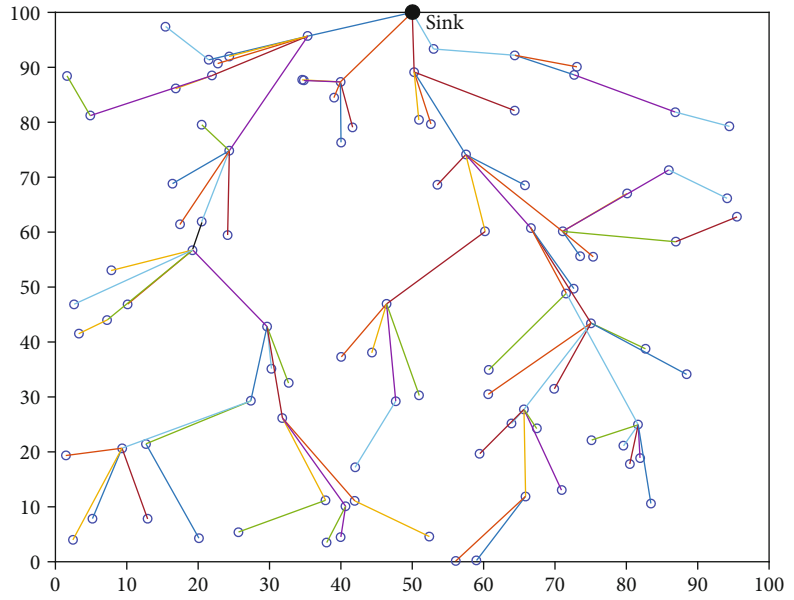

Figure 5: The typical topology used in our all experiments. The transmission distance is 12 m. The initial workload of the node is 1 packet.

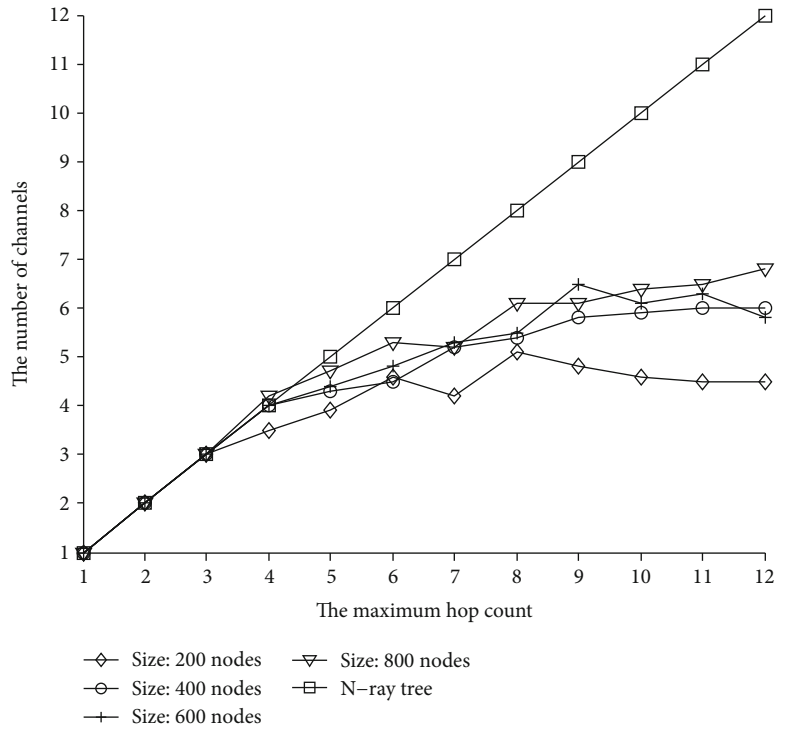

Figure 6: The number of channels versus the maximum hop count in different network scale.

When the number of available channels exceeds 4, the *SLC* is equivalent to the slot requirement of the *centralized* algorithm. In addition, when the number of nodes increases to 800, the slots difference between the two algorithms is significantly reduced. The reason for this is more slots are reused by nodes on different channel when the sufficient channels are available and then shortens the data collection time.

The number of state-transition is shown in Figure 11. It can be concluded from the figure that although the single channel *centralized* can reduce the number of state-transition by allocating consecutive slots for nodes, in fact, only a small part can be reused between nodes due to the existence of wireless interference. *Centralized* cannot guarantee that the nodes have consecutive slot assignment. For *SLC*, when the number of available channels is 2, the number of state transition is not much different from that of *centralized*. However, when the number of channels is increased to 4, the number of state-transitions is significantly different.

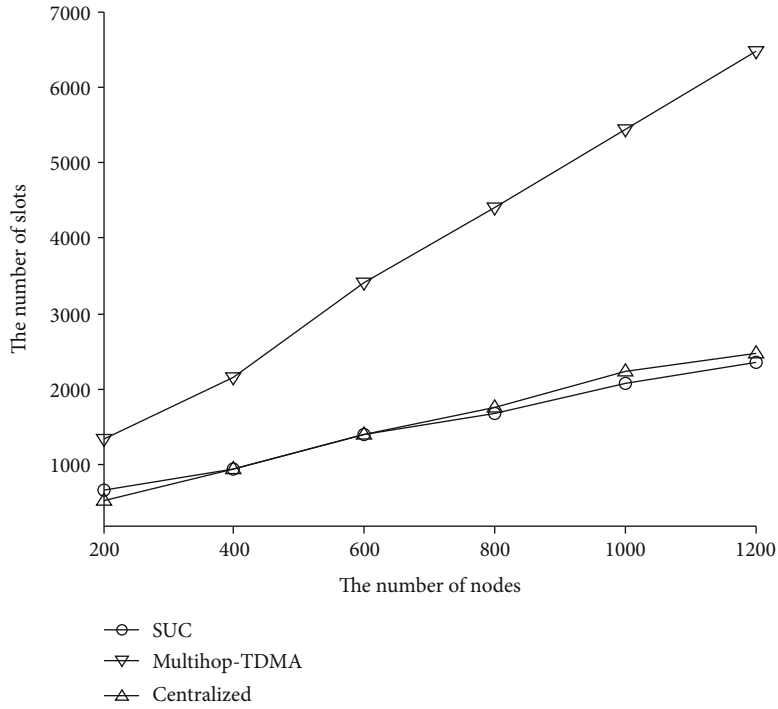

Figure 7: The number of slots versus network scale.

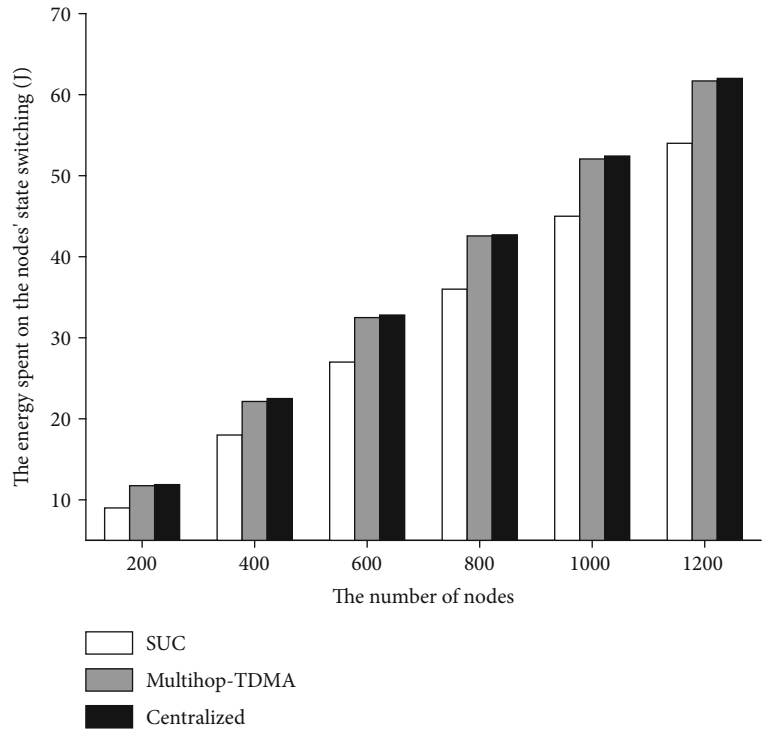

Figure 8: The energy consumption for state switching versus network scale in one data collection cycle.

When the number of channels is increased to 6, each node needs only one state transition to complete the receiving and transmitting data, thereby obtaining optimal energy efficiency.

Based on the experimental results, it can be found that when the available channels increase, the receiver-based slot assignment makes the slot of the node as continuous as possible, so as to minimize the number of state switching of the

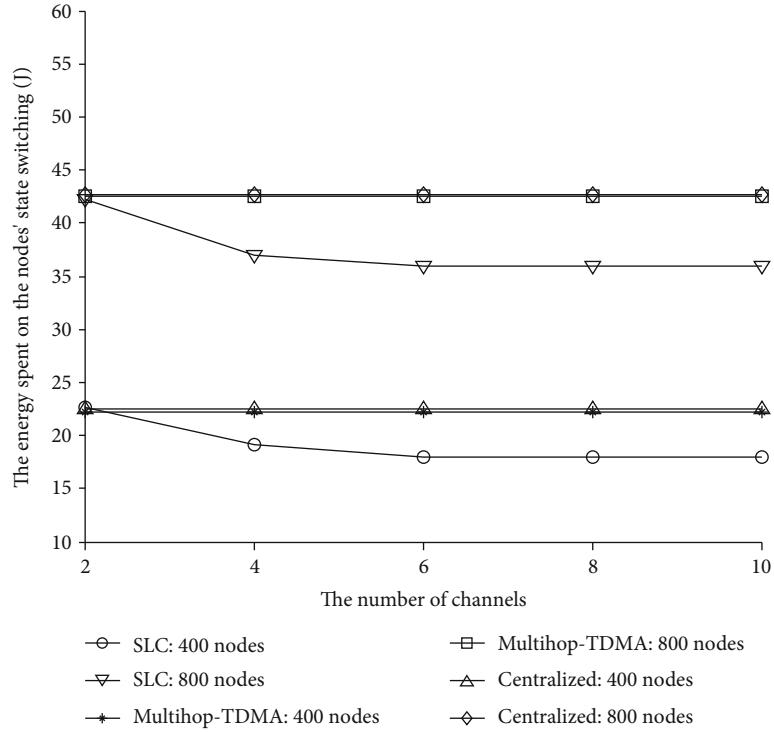

Figure 9: The energy consumption for state switching versus the number of channels in a data collection cycle.

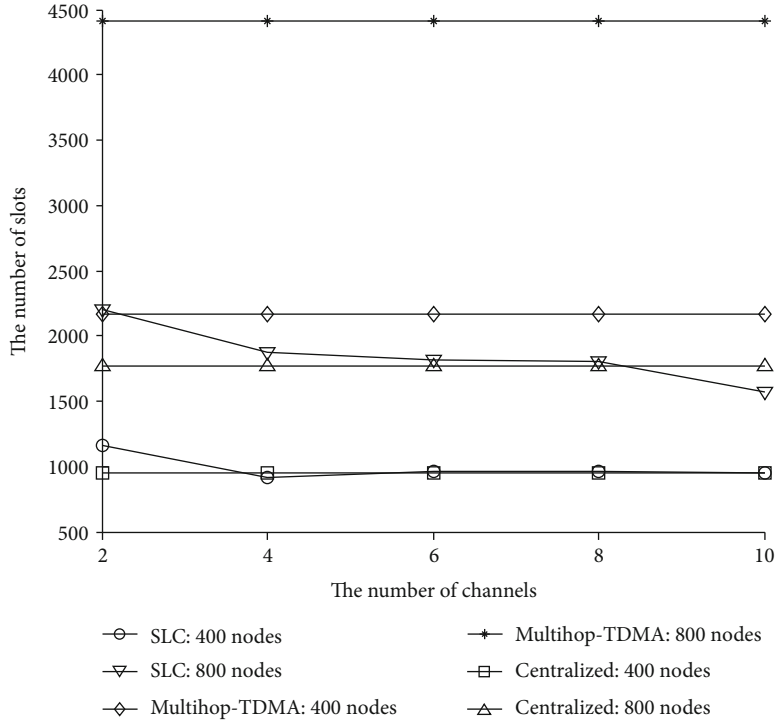

Figure 10: The number of slots versus the number of channels.

node. In fact, because the node always receives child's data through continuous slots firstly, and then forwards all data to its parent, the data transmission delay has also been optimized. Therefore, our algorithm is helpful to design delay-aware routing protocol. Considering that IEEE 802.15.4e [31] has supported the multichannel communication technology, our algorithm is feasible to a certain extent. However, because the slots allocated to nodes can be reused

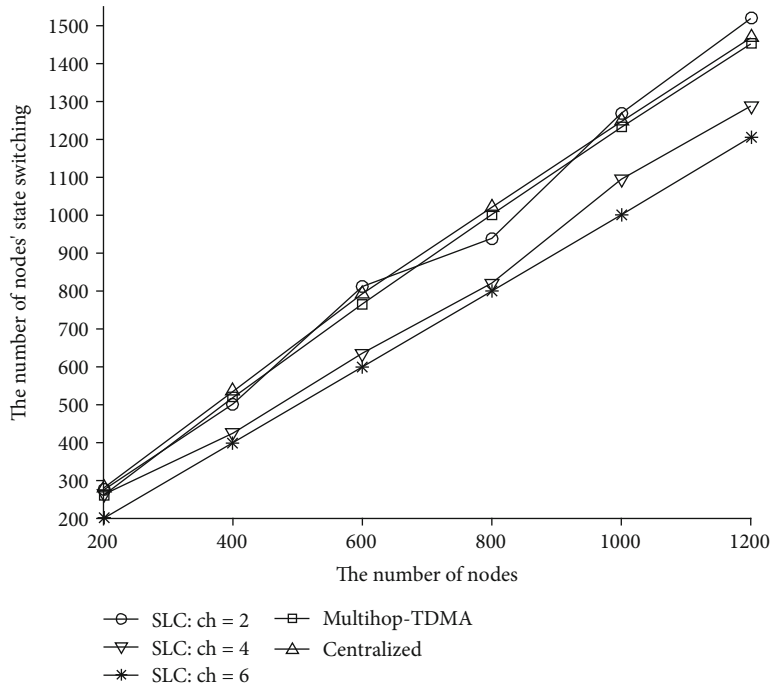

Figure 11: The number of nodes' state switching versus the number of nodes.

only when nodes use different channels, the algorithm in this paper does not fully realize the optimization of data transmission scheduling. In addition, our algorithm does not consider the performance of nodes with different initial workload or under different network topology [32]. These problems are also worthy of further consideration.

7. Conclusion

In this paper, the one-shot TDMA scheduling algorithm is proposed for the energy consumption of state switching. The one-shot TDMA scheduling algorithm uses a continuous slot allocation algorithm based on the receiver to ensure that each node only needs to wake up once during data aggregation, and at the same time, by arranging the slots on different channels, the data transmission of the node is guaranteed to be interference-free. We considered the case where the number of available channels is limited and developed the *SLC* algorithm. The extensive simulation results show that the proposed algorithm can effectively reduce the number of node state switching and obtain the optimal data collection time.

However, our algorithm does not provide optimal data collection time when the number of available channels is small. Therefore, in our future research work, we plan to implement intrachannel and interchannel slot reuse based on the wireless interference model to further reduce data collection time.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

An earlier version of this paper was presented at the "2013 IEEE/CIC International Conference on Communications in China (ICCC 2013)."

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors appreciate the helpful comments and suggestions of anonymous reviewers. This research was funded by the CERNET Innovation Project (no. NGII20180313).

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: a survey," *Journal of Network and Computer Applications*, vol. 60, pp. 192–219, 2016.

[3] L. Van Hoesel and P. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks: reducing preamble transmissions and transceiver state switches," in *1st International Workshop on Networked Sensing Systems, INSS 2004*, pp. 205–208, Japan, 2004.

[4] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless, "Minimum power configuration in wireless sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pp. 390–401, New York, NY, USA, 2005.

[5] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *IEEE INFOCOM 2009*, pp. 630–638, Rio de Janeiro, Brazil, 2009.

[6] Z. Wang, J. Li, L. Kang, C. Wang, and Y. Zhang, "Low-latency TDMA sleep scheduling in wireless sensor networks," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, Shenzhen, China, 2015.

[7] B. Zeng, Y. Dong, J. He, and D. Lu, "An energy-efficient TDMA scheduling for data collection in wireless sensor networks," in *2013 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 633–638, Xi'an, China, 2013.

[8] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher, "Mmsn: multi-frequency media access control for wireless sensor networks," *Infocom*, vol. 6, 2006.

[9] F.-J. Wu and Y.-C. Tseng, "Distributed wake-up scheduling for data collection in tree-based wireless sensor networks," *IEEE Communications Letters*, vol. 13, no. 11, pp. 850–852, 2009.

[10] V. L. Hai and X. Tang, "An efficient scheduling algorithm for data collection through multi-path routing structures in wireless sensor networks," in *2010 Sixth International Conference on Mobile Ad-hoc and Sensor Networks*, pp. 68–73, Hangzhou, China, 2010.

[11] O. Abedi and S. Razaghi Kariznoi, "Load-balanced and energy-aware opportunistic routing with adaptive duty cycling for multi-channel WSNs," *The Journal of Supercomputing*, vol. 77, no. 2, pp. 1038–1058, 2021.

[12] A. Pal and A. Nasipuri, "Distributed routing and channel selection for multi-channel wireless sensor networks," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, p. 10, 2017.

[13] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pp. 1–14, New York, NY, USA, 2009.

[14] Y. Wu, X. Y. Li, Y. H. Liu, and W. Lou, "Energy-efficient wake-up scheduling for data collection and aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 275–287, 2010.

[15] J. Ma, W. Lou, and X. Y. Li, "Contiguous link scheduling for data aggregation in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1691–1701, 2014.

[16] C.-h. Cheng and C.-c. Ho, "Implementation of multi-channel technology in ZigBee wireless sensor networks," *Computers and Electrical Engineering*, vol. 56, pp. 498–508, 2016.

[17] M. Yigit, V. C. Gungor, E. Fadel, L. Nassef, N. Akkari, and I. F. Akyildiz, "Channel-aware routing and priority-aware multi-channel scheduling for WSN-based smart grid applications," *Journal of Network and Computer Applications*, vol. 71, 2016.

[18] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2011.

[19] Y. Wu, J. A. Stankovic, H. Tian, and L. Shan, "Realistic and efficient multi-channel communications in wireless sensor networks," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pp. 1193–1201, Phoenix, AZ, USA, 2008.

[20] X. Wang, X. Wang, X. Fu, and G. Xing, "MCRT," *ACM Transactions on Sensor Networks*, vol. 8, no. 1, pp. 1–30, 2011.

[21] D.-c. Wsns, X. Jiao, W. Lou et al., "Delay efficient scheduling algorithms for data aggregation in multi-channel asynchronous," *IEEE Transactions on Communications*, vol. 67, pp. 6179–6192, 2019.

[22] M. Ren, J. Li, L. Guo, X. Li, and W. Fan, "Distributed data aggregation scheduling in multi-channel and multi-power wireless sensor networks," *IEEE Access*, vol. 5, pp. 27887–27896, 2017.

[23] N. T. Nguyen, B. H. Liu, S. I. Chu, and H. Z. Weng, "Challenges, designs, and performances of a distributed algorithm for minimum-latency of data-aggregation in multi-channel WSNs," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 192–205, 2019.

[24] N.-T. Nguyen, B.-H. Liu, V.-T. Pham, and T.-Y. Liou, "An efficient minimum-latency collision-free scheduling algorithm for data aggregation in wireless sensor networks," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2214–2225, 2018.

[25] J. B. Lim, B. Jang, and M. L. Sichitiu, "MCAS-MAC: a multi-channel asynchronous scheduled MAC protocol for wireless sensor networks," *Computer Communications*, vol. 56, pp. 98–107, 2015.

[26] B. Jang, J. B. Lim, and M. L. Sichitiu, "AS-MAC: an asynchronous scheduled MAC protocol for wireless sensor networks," in *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 434–441, Atlanta, GA, USA, 2008.

[27] D. Praveen Kumar, T. Amgoth, and C. S. R. Annavarapu, "Machine learning algorithms for wireless sensor networks: a survey," *Information Fusion*, vol. 49, pp. 1–25, 2019.

[28] X. Zeng, R. L. Bagrodia, and M. Gerla, "GloMoSim," *ACM SIGSIM Simulation Digest*, vol. 28, no. 1, pp. 154–161, 1998.

[29] J. B. Lim, R. Jang, R. Yoon, R. L. Sichitiu, and A. G. Dean, "Raptex: rapid prototyping tool for embedded communication systems," *ACM transactions on sensor networks*, vol. 7, 2010.

[30] I. Marín, J. Arias, E. Arceredillo, A. Zuloaga, I. Losada, and J. Mabe, "LL-MAC: a low latency MAC protocol for wireless self-organised networks," *Microprocessors and Microsystems*, vol. 32, no. 4, pp. 197–209, 2008.

[31] D. De Guglielmo, S. Brienza, and G. Anastasi, "Ieee 802.15.4e: a survey," *Computer Communications*, vol. 88, pp. 1–24, 2016.

[32] D. K. Sah, K. Cengiz, P. K. Donta, V. N. Inukollu, and T. Amgoth, "EDGF: empirical dataset generation framework for wireless sensor networks," *Computer Communications*, vol. 180, pp. 48–56, 2021.