Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 512687, 13 pages
http://dx.doi.org/10.1155/2014/512687

Research Article

A Negotiation-Based TDMA Scheduling with Consecutive Slots Assignment for Wireless Sensor Networks

Bo Zeng,[1,2] Yabo Dong,[1] and Zhidan Liu[1]

[1] *College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*
[2] *Electronic Information Engineering School, Henan University of Science and Technology, Luoyang 471023, China*

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

Received 4 March 2014; Accepted 2 July 2014; Published 5 August 2014

Academic Editor: George Nikolakopoulos

TDMA protocols are recognized to save energy consumption by avoiding unnecessary idle listening and communication collision. As sensor nodes are usually equipped with limited resource supply, they are thus desirable to improve network capacity and energy efficiency by adopting TDMA scheduling for wireless sensor networks (WSNs). This paper proposes a TDMA scheduling approach, which reuses time slots among sensor nodes, for WSNs. Specifically, with the idea of consecutive slots assignment, our approach can reduce both the time and energy cost of establishing nodes' schedule. Furthermore, the energy of node's switching is also reduced. To avoid the transmission of a node's result interference to all irrelevant receivers within its interference range, we develop a slot negotiation mechanism for slot assignment. We perform extensive simulations in NS-2 to evaluate our proposed approach. The results show that our approach outperforms existing TDMA scheduling in terms of time and energy overhead.

1. Introduction

Wireless sensor networks (WSNs) are used to gather information in a variety of periodic data collection applications, such as environment sensing, structural health monitoring, and military surveillance [1]. In these applications, sensor nodes are usually self-organized to form a data collection tree. In such a multihop network, node not only periodically generates data but also relies data for other node to the sink. Therefore, each node serves as collector and router in the data gathering applications.

As nodes are equipped with limited energy supply, it is desirable to reduce the energy consumption by avoiding unnecessary idle listening and reducing the number of node's switching at the same time. TDMA protocol can reduce energy consumption by assigning time slots to nodes in the network. Nodes wakeup at their assigned time slots for data receiving and transmission and then enter into sleep mode for saving energy until the next assigned time slot. In a sensor network, it is necessary to improve network performance, such as data collection time and energy efficiency, by executing data transmission in parallel for nodes at the same slots. However, it has been proven that such a improvement is difficult to achieve, due to several constraints of WSNs, for example, energy and wireless interference [2–4].

The scheduling protocol based on TDMA (i.e., TDMA scheduling) [5] fortunately turns to be an efficient approach to realize the concurrent data transmission for nodes in WSNs. The intention behind TDMA scheduling is the idea of time slots reuse among nodes. The time slots reuse, however, is hard to implement, especially in an effective way. Many TDMA scheduling approaches [6–11] have been proposed. In these approaches, the nodes are scheduled under the protocol interference model and the nodes' slots are propagated to two-hop neighbors to eliminate wireless interference occurring in concurrent data transmission. With the scheduling generated by the existing approaches, a packet, however, may be lost because of communication collision, which is caused by the incorrect slot reuse between the receiver and its two-hop neighbors [6, 8, 9, 12]. A possible solution is that each node propagates its slot assignment to three-hop or more neighbors at the expense of extra energy cost [13, 14]. Other TDMA scheduling approaches which take the physical interference into account have been proposed in [15, 16]. The slots will be assigned to a node if the node's signal-to-interference-plus-noise-ratio (SINR) meets SINR $\leq \beta$ [17],

where β is the minimum signal-to-interference-ratio that is required for a node to successfully receive a packet. Though such an approach improves the network performance, it is uneasy to be implemented, due to the need of frequent measure of wireless interference, which is known as spatial-temporal correlated [18]. The possible failures of previous works motivate us to design a TDMA scheduling approach with guaranteed performance, for example, reliable packet delivery, yet with acceptable cost.

In this paper, we propose a TDMA scheduling with well-designed slots negotiation and consecutive slots assignment mechanisms. Furthermore, our approach adopts a local time synchronization mechanism to remove the impact of clock drift. In our approach, each node's schedule consists of slots which are used to data collection and time synchronization, respectively. Slots are assigned to nodes in the initial phase. Once all nodes establish their schedules, they receive and transmit data according to their schedules in each data collection cycle. In summary, our approach improves the network performance from the following schemes.

(i) Our slot negotiation mechanism can reduce communication collision when nodes assign time slots to themselves in the network initialization period, which makes our approach achieve high packet receiving ratio and energy efficiency.

(ii) Our consecutive slots assignment mechanism can reduce the cost of establishing nodes' schedule. Furthermore, our approach saves energy by decreasing the times of node's state switching, for example, from sleep state to transmission state.

We evaluate the performance of our approach with extensive simulation experiments in NS2. The simulation results show that, compared with existing TDMA scheduling approach, our proposed approach achieves much higher packet receiving ratio and energy efficiency. The results also demonstrate that our approach can reduce the cost of establishing nodes' schedules by assigning consecutive slots to nodes.

The rest of the paper is organized as follows. We discuss related work in Section 2. Our approach is elaborated in Section 3. In Section 4, we present the evaluation results. Section 5 concludes this paper.

2. Related Work

Many TDMA scheduling approaches have been proposed for data collection in wireless sensor networks. The distributed coloring method has been proposed in [8, 19–22]. When nodes are assigned colors with the distributed coloring method, each node assigns itself with the minimum color number so that the whole network can minimize the number of colors. In [12, 19, 23], node picks up a color by running a lottery game. In [6, 8], node can color itself only when it receives the token from its parent node in the data collection tree. For most of these algorithms, however, slot may not be fully utilized because nodes can only be active at their assigned time slots that are related to their color. In addition, it is a complicated problem to assign slots based on nodes' coloring in a many-to-one communication pattern. In another proposed method, nodes assign collision-free slots based on the k-hop slot information exchange [13, 14, 24]. This method allows two or more nodes to assign slots concurrently. However, the huge message overhead, due to slots information exchange among nodes, results in unacceptable energy consumption.

A TDMA scheduling based on the physical interference has been proposed in [15, 16]. Taking the physical interference into consideration will lead to a better scheduling with spatial reuse. Brar et al. [15] present a distributed algorithm which uses a global primitive to exchange schedule information among nodes, to provide guaranteed collision-free slot assignment. A PHY-aware distributed algorithm for wireless ad hoc networks is proposed in [16]. The authors use the optimal stopping theory to balance the tradeoff between network throughput and the overhead of channel probing. However, the real-time calculation of external interference is prohibited for resource constrained sensor nodes.

Some works aim at improving energy efficiency for data collection with TDMA scheduling [7, 25–29]. Zhao and Tang [7] propose an energy-efficient TDMA schedule to accommodate periodic data collection with dynamic traffic patterns. The energy consumption self-adapts to the node's traffic load produced in traffic pattern when node has a consecutive slots assignment. However, this schedule cannot achieve high energy conservation if nodes have inconsecutive slots assignment. TRAMA [25] is an energy-efficient scheduling protocol that uses distributed election scheme to assign slots to nodes. Nodes use a hash function to determine their priorities among contending neighbors. The drawback of this hashing based method is that the priority for a node is only effective in a specific slot, and thus frequent node switching due to priority changes is needed. Dynamically adjusting the node's duty cycle when the network changes is an effective method to reduce the energy consumption. In [26], nodes can reduce energy consumption through maintaining a low duty cycle during their lifetime. Only part of nodes are scheduled in each data collection cycle. This method, however, is not suitable for periodical data collection applications. Another work in [27] uses the similar method to minimize the energy consumption. They first construct multiple connected dominating sets (CDS) for the network and then propose a heuristic algorithm to find an optimal combination of CDSs. But, the proposed algorithm is not practically applicable to wireless sensor networks, since it requires sink to compute schedule and disseminate schedule to node.

To reduce the cost of establishing nodes' schedules is an important objective for TDMA scheduling. In [30], a constant time scheduling policy is developed for ad hoc networks. However, this policy has not considered the energy efficiency. In [31], two low-complexity randomized scheduling algorithms based on the random access technique are developed for wireless networks. They achieve the low overhead implementation at the cost of performance loss on network capacity. Sanghavi et al. [32] propose a scheduling algorithm which can achieve any required fraction of wireless capacity

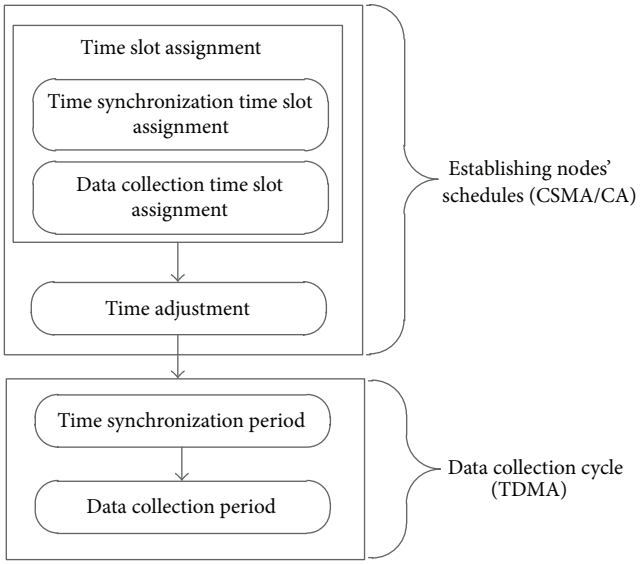


Figure 1: The order of two phases.

region in arbitrary wireless networks with low overhead. The algorithm only considers the one-hop interference model that is not proper in the practical WSNs.

3. TDMA Scheduling with Consecutive Slots Assignment

3.1. Overview. Our approach consists of two phases: the network initialization phase for establishing the nodes' schedules and data collection cycles. Figure 1 shows the order of two phases.

Note that nodes maintain their schedules during their lifetime, so that the network initialization phase occurs only once for all nodes. In the first phase, nodes exchange their slots information using CSMA/CA. All nodes are in wake-up status during this phase. Our approach uses a token-passing scheme to control slots assignment process. Node executes slots assignment only when it holds a token and meanwhile its buffer is not empty. After network initialization is finished, all nodes perform data collection tasks according to their schedules. The data is delivery to the sink periodically.

Figure 2 shows the node's schedule structure. Each node's schedule includes a set of slots that represent the active time of the node. Note that the node's slots are not always contiguous in time. In our approach, the first part (SYN) is used for time synchronization. Nodes perform time synchronization using their corresponding slots. It is worthy to point out that sink only has the slot for sending the synchronization packet and the node that has no children only has slot for receiving the synchronization packet. At the data collection (DC) part, nodes switch to the receiving mode in their received slots and to send mode in their transmission slots for transmitting data. Otherwise, they switch to the sleep mode for saving energy. When nodes finish all data activities in a data collection cycle, they will keep in sleep mode until they wake up again in the next data collection cycle.

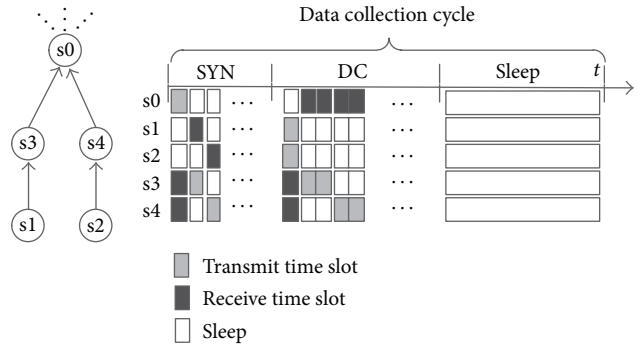


Figure 2: The structure of the node's schedule. The node's schedule consists of three parts: time synchronization (SYN), data collection (DC), and sleep period (SLEEP).

In this paper, we assume that all nodes in the network already form a data collection tree, in which nodes send their data to sink. To our best knowledge, many well-designed routing protocols [33–35] can be used to build the data collection tree.

3.2. Establishing Nodes' Schedules. The establishing of nodes' schedules aims at assigning slots to nodes for time synchronization and data collection. Our approach uses the depth-first-search (DFS) technique for slot assignment. Since DFS could be used to limit the number of slots assigned to each node, our approach avoids the slots assignment that exceeds the limitation of maximum consecutive slots. In this work, we define maximum consecutive slots as the maximum number of slots that could be assigned to each node when it executes slot allocation. For clarity, Notations shows some notations used in this paper. Note that CT is set when node is initialized and its value is customized by the user according to the application requirements.

After nodes initialization, sink generates a token for slot assignment. The token is passed to each node using DFS technique, and it is always passed to the child that has not received the token before and has the smallest node ID. When node receives the token, it first assigns slot ts_{SYN} for time synchronization. The slot ts_{data} is assigned when the node meets either of the following two conditions: the node has no children or RCT = 0 for at least one of its *ancestors*. We define the *ancestors* as the set of nodes on the path from a node to sink. After node completes the slot assignment, the slot assignment information must be propagated to its one- and two-hop neighbors using *schedule exchange* scheme and its ancestors then assign forward slots in sequence. When slot assignment finishes, nodes need to execute *time adjustment* to determine the start time of each part in a data collection cycle.

3.2.1. Slot Assignment. This section discusses the slot assignment for time synchronization and data collection. The flow chart is shown in Figure 3.

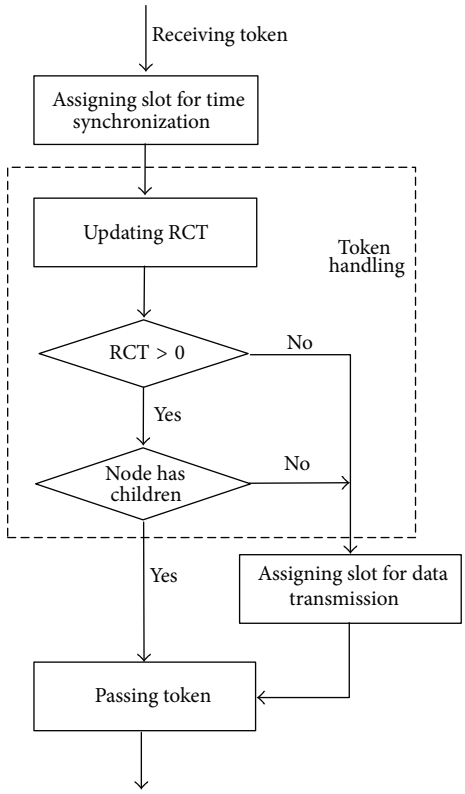


Figure 3: The flow chart of slot assignment.

(a) Slot for Time Synchronization. The slot ts_{SYN} is assigned to a node when it receives the token from its parent node. It first lists the ts_{SYN} included in the token in its receiving slot list (RL) in order to keep synchronization with its parent. If the node has children, it assigns a new slot (i.e., the maximum ts_{SYN} incremented by one) to its children. The node lists the new slot to its transmit slot list (TL). This new slot number will be included in the token, and then token is passed to its children.

(b) Token Handling. Before node assigns slots for data transmission, it needs to handle the token as follows. It updates RCT according to (1), where N_{data} denotes the number of ts_{data} used for data transmission. We have (2) for N_{data}, where w denotes the volume of data and L_{packet} denotes the length of packet transmitted in a slot. In this work, we consider that the volume of data produced by a node in one data collection cycle is small and it could be transmitted within a slot. Thus, the N_{data} is initialized to 1. Note that the RCT is carried by the token. Consider

$$\text{RCT} = \text{RCT} - N_{data}, \tag{1}$$

$$N_{data} = \frac{w}{L_{packet}}. \tag{2}$$

Baed on RCT, the token is handled by using following two rules.

(R1) Passing token to its child: if RCT > 0, the node gives up slot ts_{data} assignment and passes the token to the child that has the smallest node ID among its children. A node will be assigned slot ts_{data} by force if it has no children, to ensure that this node can be scheduled for data transmission at least one time.

(R2) Holding the token and assigning transmit slot: if RCT = 0, the node immediately assigns slot ts_{data} to itself.

Figure 4 illustrates the token passing and slots assignment.

(c) Slot for Data Transmission. After a node deals with the token, it should select the lowest slot number, which is not yet used by its interference nodes, as its slot ts_{data}. Our approach uses slot negotiation to implement slot reuse. It works as follows.

Step 1. The node (*requestor*) propagates the assigned slot information to its one-hop neighbors by sending *inform_slot* packet. The *inform_slot* packet contains the assigned slot information: the destination address, the parent ID, the first slot number, the number of slots, and the level counter. This information is represented using a quintuple: ID, PID, SN, SUM, and LC. The consecutive slots (SN, SN + SUM − 1) are called slot interval. The one-hop neighbors then check PID. If the node's ID is PID (i.e., the node is the parent of *requestor*), it sends a *reply* packet to the *requestor* immediately, or else it switches into listening mode. The *reply* packet contains the following information: a flag (Flag), and a new slot number (NSN). Therefore, the *reply* indicates whether or not the slot is reused by the parent's interference nodes. Based on the parent's TL, RL, and UL, the Flag and NSN are set as follows. Note that UL includes the node's two-hop neighbors' slot information that is collected in the schedule exchange phase.

(1) All slots in (SN, SN + SUM − 1) are not listed in the parent's TL, RL, and UL. The Flag and NSN are set as 1 and the *requestor*'s first slot number (i.e., SN), respectively. The details of slot negotiation are shown in Figure 5(a).

(2) At least, one slot in (SN, SN + SUM − 1) is listed in the parent's TL, RL, and UL. The parent set Flag to 0, and NSN is set as the lowest slot number that is not listed in the parent's TL, RL, and UL. The details of slot negotiation are shown in Figure 5(b).

Step 2. When *requestor* receives the *reply* packet, it reply *ack* packet. According to the *reply* packet, the field in *ack* packet is set as follows.

(1) The NSN is not listed in the *requestor*'s TL, RL, and UL; the Flag and FSN are set as 1 and NSN, respectively. FSN is defined as the first slot number of consecutive time slots.

(2) The NSN is listed in the *requestor*'s TL, RL, and UL. the Flag is set as 0, and FSN is set as the lowest slot number that is not listed in the *requestor*'s TL, RL, and UL. Note that FSN should be larger than NSN.

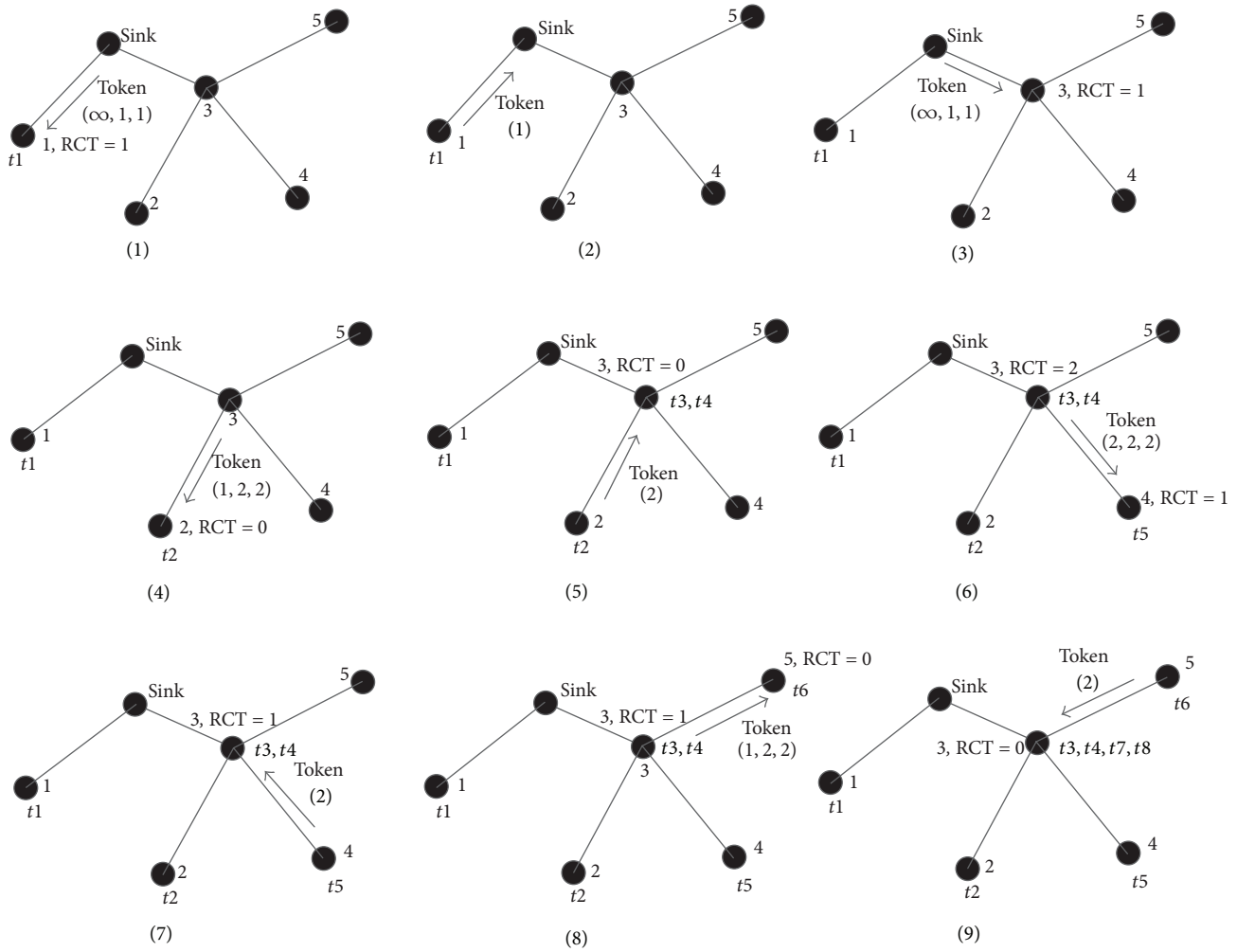


FIGURE 4: Token passing and slot assignment. We assume that node has $N_{data} = 1$ and CT = 2. The token contains following information: RCT, ts_{SYN}, and maximum ts_{SYN}. When the token is passed to a child node, all information except maximum ts_{SYN} in the token is valid. The maximum ts_{SYN} is valid when token returns to the node's parent. When a node receives the token, it updates RCT using (1). Based on the two rules, if a node has no children and RCT > 0 (R1), it assigns slot ts_{data} (see (1) and (6)); if a node has children and RCT > 0 (R1), it passes the token to its child that has the smallest node ID (see (3), (6), and (8)); if RCT = 0 (R2), the node is assigned slot ts_{data} immediately (see (4) and (8)).

Step 3. When the one-hop neighbors of *requestor* receive the *ack* packet, they check the Flag and then know whether to execute *schedule exchange*. If Flag = 1, the one-hop neighbors add (FSN, FSN + SUM − 1) to their UL and then execute *schedule exchange*. For the parent of *requestor*, it lists (FSN, FSN + SUM − 1) to its RL. If Flag = 0, slot negotiation has to be executed again. Figure 5(c) shows the details. During *schedule exchange*, SN is updated to FSN, and thus the new slot interval included in *inform-slot* packet is (FSN, FSN + SUM − 1).

(d) Schedule Exchange. The schedule exchange is triggered when a node broadcasts its assigned slot(s) to its one-hop neighbors, and then each of these one-hop neighbors propagates the slot(s) information again to their one-hop neighbors (i.e., two-hop neighbors for the claimed node). Therefore, the assigned slot(s) information is propagated to all nodes within the node's interference range.

When a node receives the *inform-slot* packet in the slot assignment phase, the *ack* packet actually decides whether or not to perform *schedule exchange*. The *schedule exchange* is controlled by the level counter (LC) which indicates the status of packet propagation. If CT ≤ 2, the node will list the slot interval (SN, SN + SUM − 1) into UL. If CT ≤ 1, the node increments the level counter and then propagates the *inform-slot* packet to its one-hop neighbors; if not, the schedule exchange is completed.

After the schedule exchange finishes, the ancestors on the path to sink assign forward consecutive slots to transmit the data stored in their buffers. Note that the forward slot number must be larger than the received slot number. The slot assignment for data collection as described before is repeated until sink receives the *inform_slot* and executes schedule exchange. The slot assignment ends when the token is returned to sink and all nodes in the network finish their slot assignments.

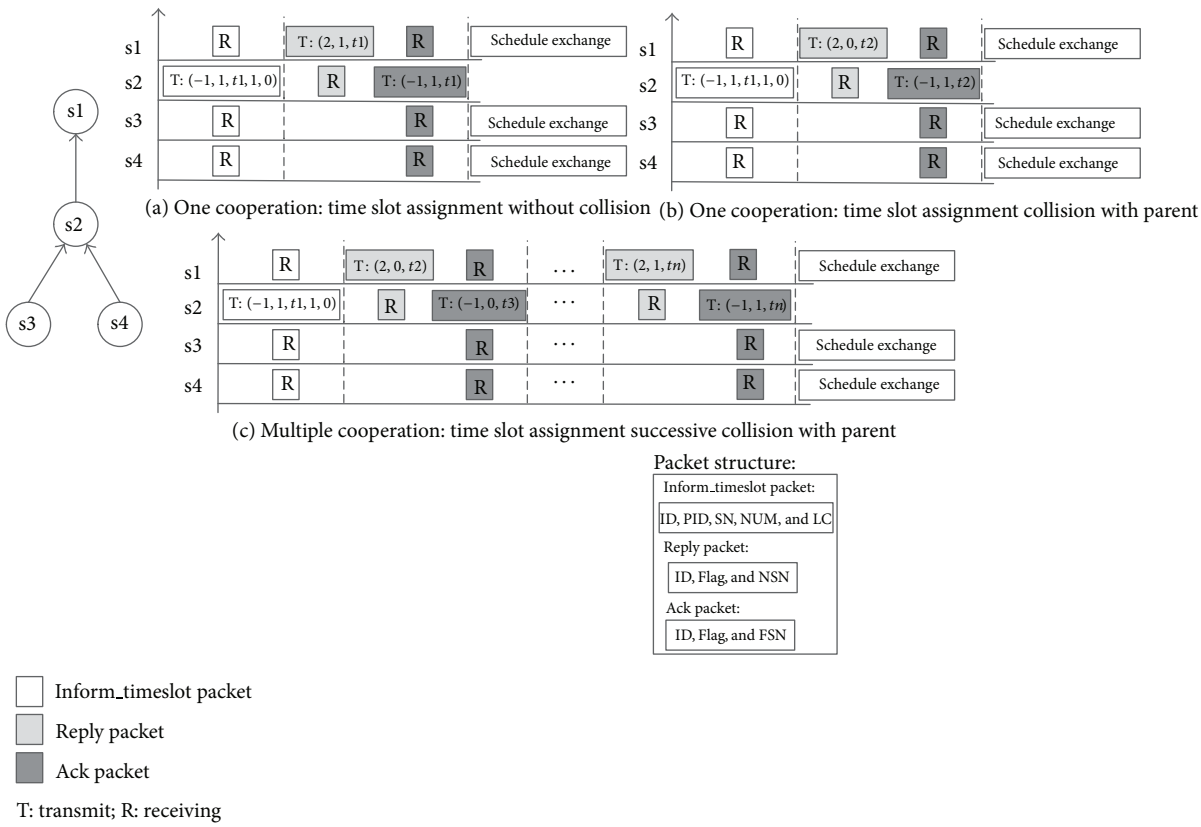


Figure 5: A timing diagram for slot negotiation between *requestor* s2 and parent s1 when the *requestor* tries to assign slot for data transmission.

3.2.2. Time Adjustment. When slot assignment is done, sink knows two important parameters on the nodes' schedules: the amount of slots for time synchronization (say N_{SYN}) passed by the token and the amount of slots for data collection (say N_{DC}). As our approach performs slot assignment for time synchronization and data collection at the same time when token is passed to nodes using the depth-first-search technique, it is necessary to adjust time related with the nodes' schedules in order to differentiate time synchronization phase from data collection phase with the time arrangement.

Hence, the start and end time for data collection can be calculated using (3). Consider

$$t_{\mathrm{DC}}^{\mathrm{start}} = N_{\mathrm{SYN}} * t_{\mathrm{SYN}},$$
$$t_{\mathrm{DC}}^{\mathrm{end}} = t_{\mathrm{DC}}^{\mathrm{start}} + N_{\mathrm{DC}} * t_{\mathrm{DC}}, \tag{3}$$

where t_{SYN} denotes the duration time for transmitting a time synchronization packet and t_{DC} denotes the duration time of one slot for data collection.

For nodes in the network, we use (4) to calculate the final start and end time of a slot. Consider

$$t_{\mathrm{DC}}^{\mathrm{is}} = t_{\mathrm{DC}}^{\mathrm{start}} + (i-1) * t_{\mathrm{DC}}, \tag{4}$$

$$t_{\mathrm{DC}}^{\mathrm{ie}} = t_{\mathrm{DC}}^{\mathrm{is}} + i * t_{\mathrm{DC}}, \tag{5}$$

where i denotes the ith slot.

For the node's schedule, we use (6) to calculate the duration time of node's schedule. Consider

$$T_{\mathrm{sch}} = N_{\mathrm{SYN}} * t_{\mathrm{SYN}} + N_{\mathrm{DC}} * t_{\mathrm{DC}}. \tag{6}$$

Our approach performs time adjustment after the slot assignment is finished. Initially, the sink generates a token for time adjustment. With the token passed to nodes using depth-first-search technique, each node adjusts the time arrangement of each phase, and meanwhile it obtains the duration of the whole schedule for data collection (i.e., T_{sch}). When the token is returned to sink, the time adjustment is completed.

3.3. Time Synchronization. Time synchronization is critical in TDMA-based scheduling because the nodes involved in the same slot must wake up at the right time to accomplish data exchange. In this work, our approach performs time synchronization locally, and parent only needs to be synchronized with its children in the assigned slots. The time synchronization works as follows.

In the assigned slot, parent broadcasts its time information using SYN packet at first, and its children adjust their clock when they receive the SYN packet. Since a child only needs to be synchronized with its parent, time synchronization is simple, as the clock drift of nodes can be minimized by periodic time synchronization during each data collection cycle.

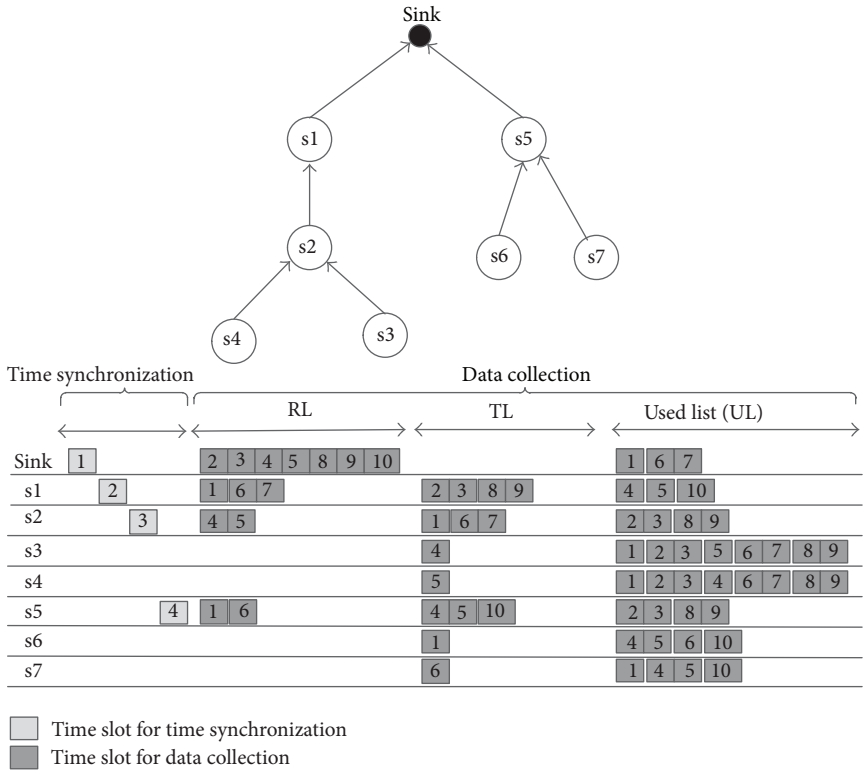


Figure 6: As an example of our approach, each node produces one packet at an operation periodically and the maximum consecutive slots assignment requirement (i.e., the node's buffer limitation) is two packets at one time.

3.4. Example. We now illustrate our approach using the data collection tree shown in Figure 6. We assume that node produces one packet at an operation periodically. We also assume CT is 2.

Initially, the slot assignment token is generated by sink and it first passes token to the node s1. When s1 obtains, it first records the slot t1 for receiving time synchronization packet; s1 assigns slot t2 for sending its time synchronization packet because it has a child (i.e., s2). s1 then checks its parent's RCT that is piggybacking by token. Since s1 needs to reserve one slot for its own data produced in an operation, hence, the RCT is updated to 1. As RCT $>$ 0 and s1 has child s2, s1 passes the token to s2 with RCT $=$ 1. In addition, slot 2 used for time synchronization is also included in the token.

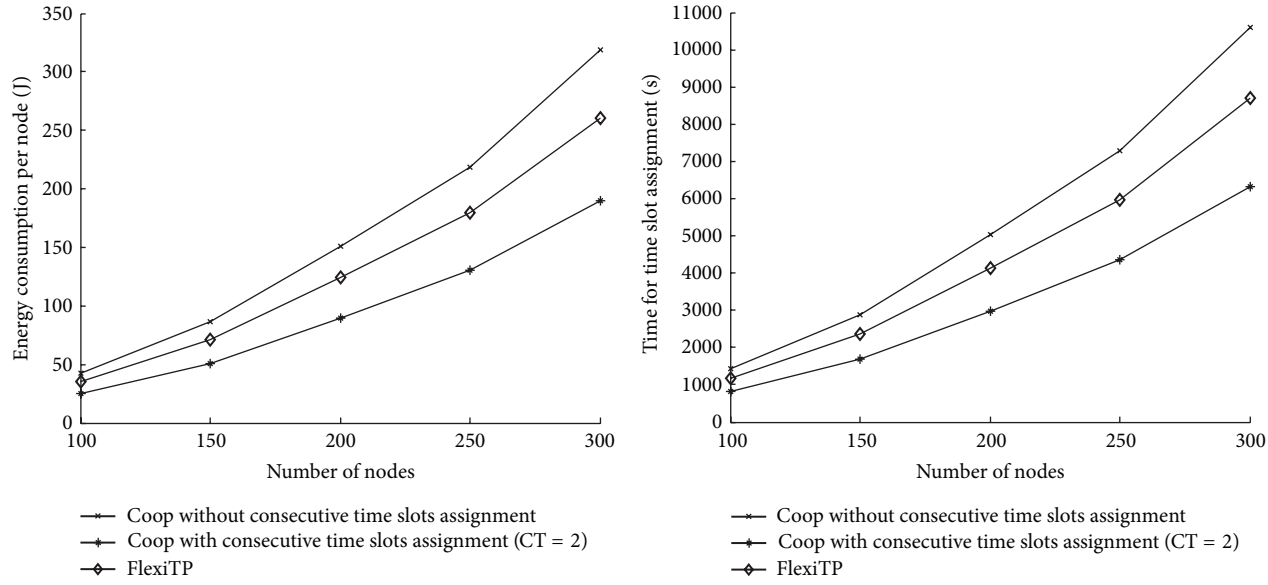
When s2 receives the token, it records slot t2 for receiving time synchronization packet and then assigns slot t3 for sending its time synchronization packet. According to RCT included in the token, s2 knows that parent's (i.e., s1) CT will reach 2 if it transmits one packet to s1. Therefore, s2 immediately claims slot (i.e., slot 1) for transmitting data and lists this slot in its TL after finishing the cooperation between s1 and s2. s2 and then propagates the slot to its two-hop neighbors using schedule exchange (Section 3.2.1).

As s1 is the parent of s2, it records slot 1 in its RL, whereas s3 and s4 list slot 1 in their ULs as shown in Figure 4. Again, s1, s3, and s4 broadcast slot information to their one-hop neighbors, and hence sink knows that slot 1 is used and then lists it in its UL. After the schedule exchange is completed, s2's parent, s1, will assign forward slots (i.e., slot 2 and 3) to forward two packets received from s2 and produced by itself.

When the token is passed to node s3, the RCT is set as 2 because both s1 and s2 do not have packets. However, since s3 has no children, it executes slot assignment immediately and then the token is returned to s2. When s2 receives the token from s3, the token is passed to s4 because its RCT is 1. The token-passing is replicated until all nodes receive the token. Note that node s6 can reuse slot 1 because slot 1 does is not used by its two-hop neighbors. Thus, in slot 1, s2 and s6 can send their data without collision, and s1 and s5 can receive data from their child in that slot. Similarly, slots 4 and 5 are reused by s3, s4, and s5.

When node s7 claims slot, it first tries to assign slot 2 and then broadcasts to its one-hop neighbors. However, its parent, s5, knows that slot 2 is used by its two-hop neighbors and it cannot receive data from s7 in slot 2 correctly; hence, s5 replies s7 and assigns a new slot 6 that is not listed in its RL, TL, and UL. s7 then confirms the s5's reply using *ack* packet. If slot 6 is listed in its RL, TL, and UL, it will try to assign another slot and start a new cooperative process. The cooperation between s5 and s7 is similar to Figure 5. When the token is returned to sink permanently, the slot assignment is finished.

Afterwards, the sink initiates the time adjustment phase by generating a new token. All nodes in the network adjust their time arrangement such as the start and end time of each phase according to the time information included in



(a) Energy cost per node versus network scale

(b) Time cost of establishing nodes' schedules versus network scale

Figure 7: The time and energy cost of establishing nodes' schedules. We refer to the maximum consecutive slots requirement as CT.

the new token. After the nodes' schedules are establish finally, nodes perform regular data collection tasks based on their schedules.

4. Performance Evaluation

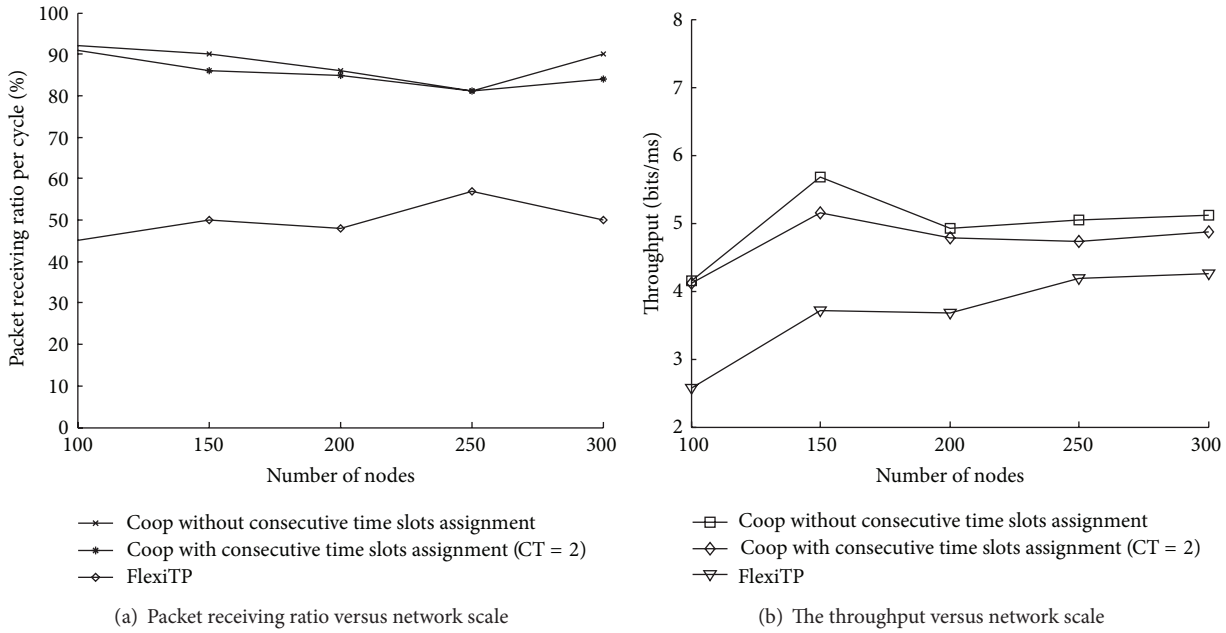
We evaluate our approach named Coop and compare its performance against existing TDMA scheduling FlexiTP [6]. In FlexiTP, sink first generates a token for slot assignment in network initialization phase. The token passing is finished using the depth-first-search technique. Each time, the token is also passed to the child that has the smallest ID. When node assigns a slot, it first announces a slot that is not listed in its TL, RL, and UL and then propagates it to one- and two-hop neighbors. FlexiTP assumes that nodes only buffer one packet at any time, and thus each node has to assign forward slot immediately. Abundant of energy will be consumed due to frequent slot assignment.

We implemented the prototypes of the two approaches in network simulator (NS-2). Our simulation results are based on the mean value of various random network topologies with up to 300 nodes distributed uniformly over a 500 m × 500 m area in all experiments. The location of sink was fixed at the top-center of the network topology. In our experiments, each node produces one packet at an operation in a data collection cycle. The simulation parameters for all experiments were based on Mica2 Mote hardware [6, 36]. Table 1 presents these simulation parameters.

4.1. Impact of Network Scale. We first measured the time and energy cost of establishing nodes' schedules. These costs are one-off costs because nodes' schedules will be maintained throughout their lifetime. Figure 7(a) shows the average energy consumption for assigning the nodes' schedule. The energy cost per node is less than 0.6% of the node's initial energy for all network scale. For Coop, when nodes are assigned with no consecutive slots, they spent more energy on slot assignment than those under FlexiTP protocol, in which many control packets are transmitted due to the cooperation between children and its parent. However, the results on packet receiving ratio (PRR), which is calculated by (7) and explained in detail later, in Figure 8(a) show that the extra energy cost plays a tradeoff for PRR. When the the feature of consecutive slots assignment is on, the cost of Coop on energy and time is much lower than FlexiTP because the number of control packets and schedule exchanges is reduced in Coop. Figure 7(b) shows the time overhead, which shares the similar trend on time cost with the energy consumption shown in Figure 7(a).

Table 1: Simulation parameters in NS-2.

Simulation parameters	Default value
Channel bandwidth	19.2 Kbps
Packet size	56 bytes (36 bytes for payload and 20 bytes for header)
Transmission range	100 m
Transmit power	63 mW
Receiving power	30 mW
Idle power	30 mW
Sleep power	0.003 mW
Transition power	30 mw
Transition time	2.45 ms
Slot size	27 ms
Node initial energy	54,000 J



(a) Packet receiving ratio versus network scale

(b) The throughput versus network scale

Figure 8: The packet receiving ratio and throughput versus network scale.

For data gathering applications, a large number of data need to be sent to sink periodically. The scheduling approach can reduce the duration time of data collection by reusing slot among nodes. On the other hand, data would be received by sink as soon as possible. In this paper, we use (7) to calculate the packet receiving ratio (PRR). Consider

$$p = \frac{N_{\text{recv}}}{N_{\text{send}}}, \tag{7}$$

where N_{recv} denotes the number of packets received correctly by sink and N_{send} denotes the number of packets sent from the source nodes in the network. As we assume each node produces one packet during a data collection cycle, we have $N_{\text{send}} = N$ for a network composed of N nodes. We show the performance of PRR in Figure 8(a). Compared with FlexiTP, our cooperation mechanism can avoid most communication collisions under the protocol interference model, and thus Coop performs better than FlexiTP on the metric of PRR. It is worthy to note that the assumption on interference in this paper cannot fully guarantee collision-free slots to be reused due to the irregularity of wireless interference, which is the reason why Coop cannot reach 100% PRR. To measure network throughput, we use (8). Consider

$$C = \frac{N_{\text{recv}} * L_{\text{packet}}}{T_{\text{DC}}} = \frac{N_{\text{send}} * p * L_{\text{packet}}}{T_{\text{DC}}}, \tag{8}$$

where N_{recv} and N_{send} were used in (7). We have $N_{\text{send}} = N$ for a network with N nodes, p is PRR calculated by (7), and T_{DC} denotes the duration of data collection. In this paper, $T_{\text{DC}} = N_{\text{DC}} * t_{\text{DC}}$, where N_{DC} and t_{DC} are defined in (6) and L_{packet} denotes the length of a packet. In this paper, L_{packet} is 56 bytes according to Table 1.

Figure 8(b) shows the throughput performance. We can draw a conclusion that Coop achieves better throughput than FlexiTP. The reason for this is that Coop has better packet receiving ratio (i.e., higher p) than FlexiTP.

In the most cases, sensor nodes have constrained energy supply. We measured the maximum energy consumption that involves network lifetime when the first node in the network dies in a data collection cycle. Furthermore, in order to evaluate the energy efficiency of data collection for each scheduling approach, we use (9) to calculate the energy consumption for data transmission. Consider

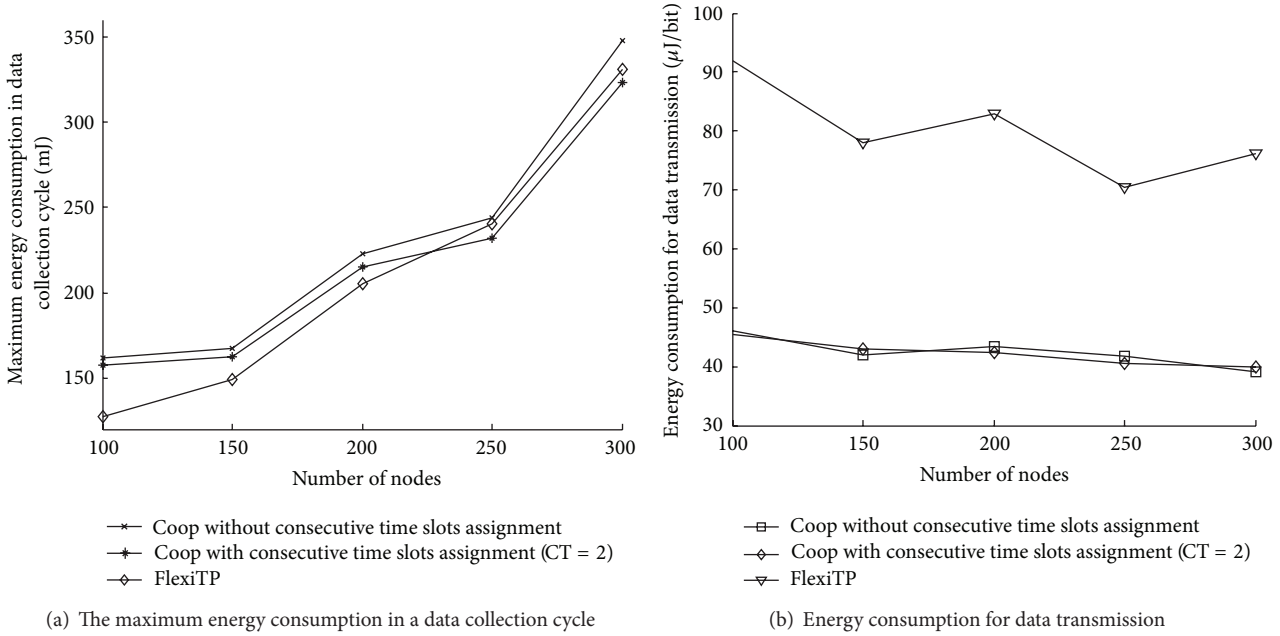
$$E_{\text{bit}} = \frac{\sum_{i=1}^{N} e_i}{N_{\text{send}} * p * \text{Payload}}, \tag{9}$$

where e_i denotes the total energy consumption during the data collection cycle and N_{send} and p were used in (8). Payload is the useful data in a packet. In this paper, Payload is 36 bytes according to Table 1.

Figure 9(a) shows the maximum energy consumption. As shown in the figure, when the feature of consecutive slots assignment is on, nodes can gather at least one packet before executing slot assignment; hence, the consecutive slots are assigned to nodes and reduce the number of node state switching. The energy spent on node state switching is thus saved. As FlexiTP protocol has lower PRR than Coop, nodes consumed lower energy on packet transmission than Coop.

In Figure 9(b), we show the energy consumption for data transmission. The result shows that the Coop's energy consumption for a bit is lower than FlexiTP about 30 μJ and we attribute this to the Coop's high packet delivery ratio.

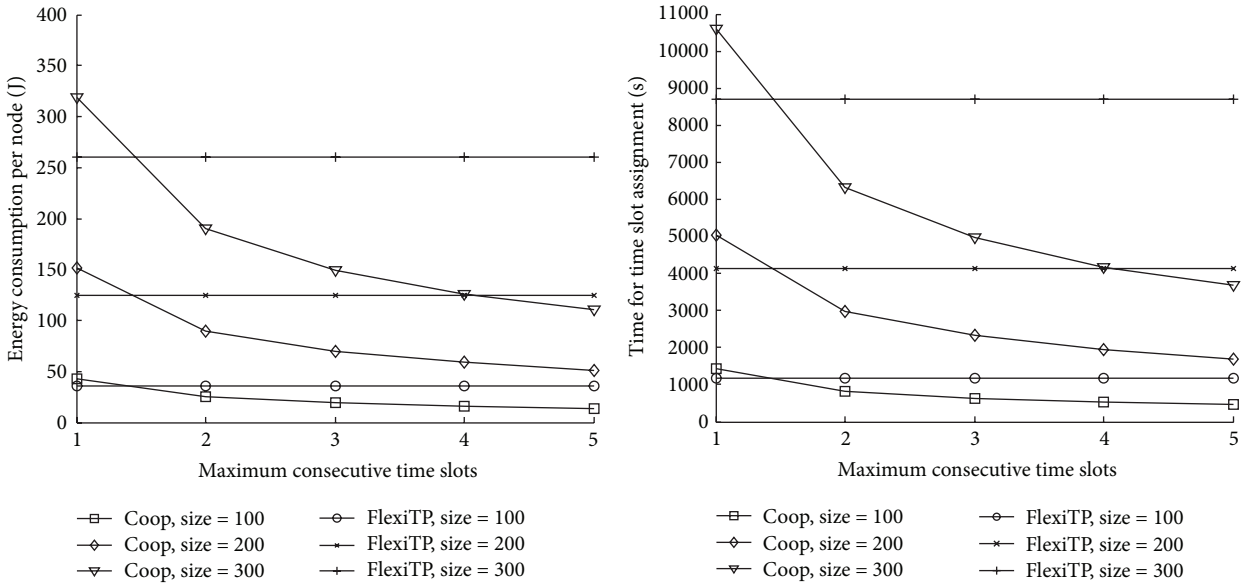
4.2. Impact of Consecutive Slots Assignment. In this simulation, we studied the impact of maximum consecutive slots



(a) The maximum energy consumption in a data collection cycle

(b) Energy consumption for data transmission

Figure 9: The maximum and average energy consumption for data collection versus network scale.



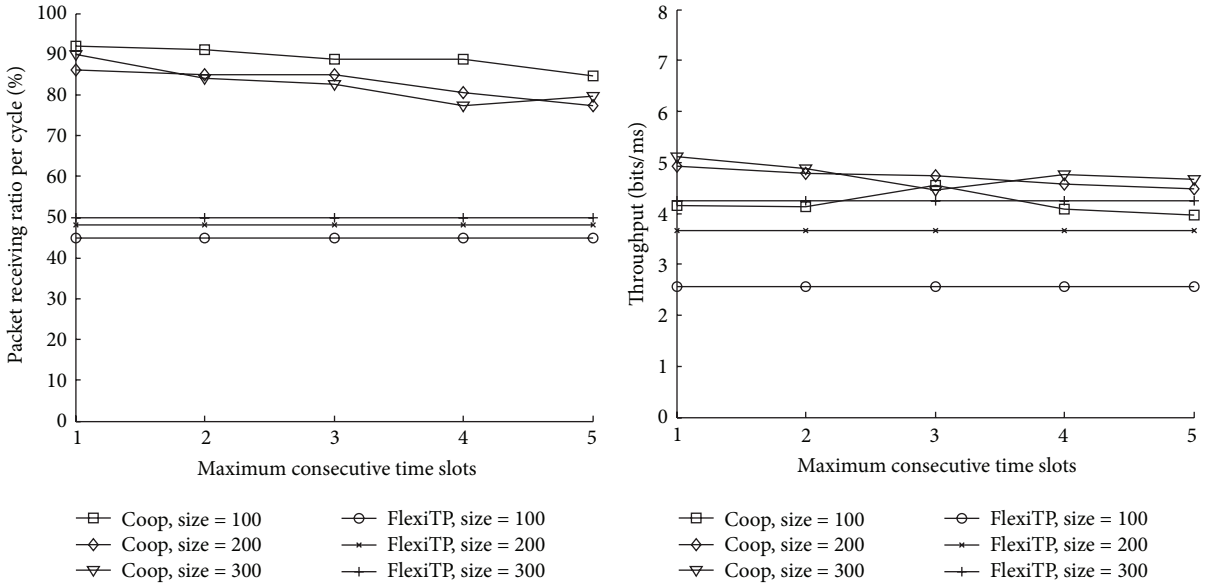
(a) Energy consumption per node

(b) The time overhead of assigning nodes' schedules

Figure 10: The average energy consumption and time overhead versus maximum consecutive slots requirement.

(CT) for the performance of Coop. We varied the CT from 1 to 5 and then measured PRR, energy consumption, and cost for establishing the nodes' schedules under three different networks scales, that is, networks with 100, 200, and 300 nodes, respectively. The results are based on the mean value of different random network topologies.

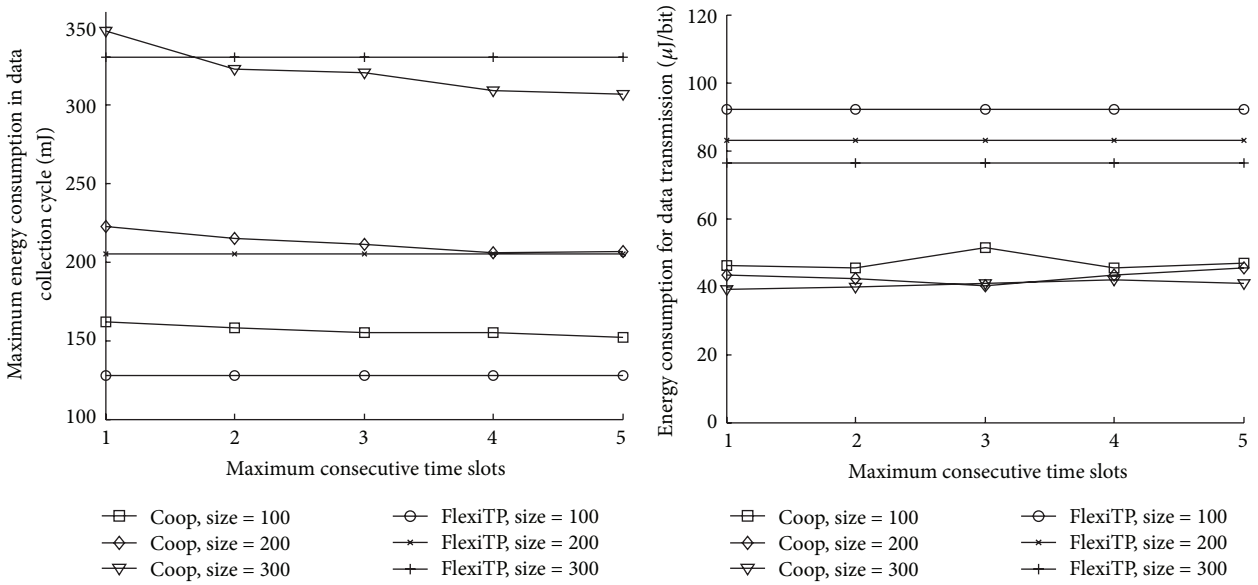
Figure 10 shows the experiment results of the time and energy cost when nodes have different CT. Both average energy consumption and time overhead are reduced when we increase the CT. However, two features of network lead to these costs falling finally limited with the increase of CT: the first feature is network topology; the other feature is that nodes in the network self-organize into a data collection tree in the network initialization phase, and hence each node in the tree has a fixed subtree rooted at the node. As the CT increases, each node can assign enough consecutive slots at a time to gather all packets that are sent by its descendant, and hence the gain of time and energy cost falling is reduced, and these costs finally trend to a constant when the gain is very small. We measured the reliability of data



(a) The PRR versus maximum consecutive slots

(b) Throughput versus maximum consecutive slots

Figure 11: The PRR and throughput versus maximum consecutive slots.



(a) Maximum energy consumption versus maximum consecutive slots

(b) Energy consumption for data transmission versus maximum consecutive slots

Figure 12: The energy consumption versus maximum consecutive slots.

transmission. As shown in Figure 11(a), the packet receiving ratio is reduced with the increasing of CT. The reason is that packets are lost due to continuous communication collision in the consecutive slots. However, Coop still outperforms TlexiTP on packet delivery. We also show the throughput in Figure 11(b). The throughput was calculated using (8). Even though PRR is decreased with the increasing of CT, Coop still has larger throughput than FlexiTP. The possible reason for this is that the throughput derived by (8) is mainly determined by the packet receiving ratio when there is little difference on the duration of data collection in the same network.

We also measured the maximum energy consumption as shown in Figure 12(a). The result shows that the energy consumption spent on the node state switching is reduced because more consecutive slots are assigned to nodes when a node has larger CT. In addition, the energy for data transmission as shown in Figure 12(b) has no obvious change

with the increasing of CT. According to (9), the reason could be that the total energy consumption for data collection and the volume of data transmission are increased with the network scale, and meanwhile the PRR only has small changes.

5. Conclusion

Wireless sensor networks have limited hardware resource, such as energy, bandwidth, and memory. This paper proposes an efficient TDMA scheduling approach to improve packet receiving ratio yet with acceptable energy and time cost. Our approach uses slot negotiation mechanism to implement slot reuse under the protocol interference model. The slot negotiation produces low communication cost because it only occurs on each link. Furthermore, our approach assigns consecutive slots to nodes so that the time and energy cost are reduced. Our approach also integrates a local time synchronization mechanism to minimize clock drift among nodes. The simulation results show that our approach achieves high packet receiving ratio and energy efficiency and meanwhile decreases the time and energy cost by increasing the number of consecutive slots. In our future work, we would like to develop an efficient schedule adjustment strategy to deal with data transmission fault and network dynamic.

Notations

CT: The maximum number of consecutive slots
RCT: The remaining number of consecutive slots on the path from a node to the sink
RL: Receiving slot list
TL: Transmit slot list
UL: Used slot list
ts_{SYN}: Slot for time synchronization
ts_{data}: Slot for data transmission.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

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

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21 Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1567–1576, IEEE, June 2002.

[3] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 171–180, New York, NY, USA, November 2003.

[4] Q. Shi, C. Comaniciu, D. Wang, and U. Tureli, "Cross-layer MAC design for location-aware wireless sensor networks," *International Journal of Communication Systems*, vol. 24, no. 7, pp. 872–888, 2011.

[5] K.-W. Jang, "Meta-heuristic algorithms for channel scheduling problem in wireless sensor networks," *International Journal of Communication Systems*, vol. 25, no. 4, pp. 427–446, 2012.

[6] W. L. Lee, A. Datta, and R. Cardell-Oliver, "FlexiTP: a flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 851–864, 2008.

[7] W. Zhao and X. Tang, "Scheduling data collection with dynamic traffic patterns in wireless sensor networks," in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM '11)*, pp. 286–290, Shanghai, China, April 2011.

[8] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[9] C. Renner, V. Turau, and C. Weyer, "Performance of energy-efficient tdma schemes in data-gathering scenarios with periodic sources," in *Proceedings of the 7th International Conference on Networked Sensing Systems (INSS '10)*, pp. 187–194, 2010.

[10] Y. Wu, X.-Y. Li, Y. 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.

[11] K. Sato and S. Sakata, "A power-efficient distributed tdma scheduling algorithm with distance-measurement for wireless sensor networks," *Wireless Personal Communications*, vol. 75, no. 2, pp. 1511–1528, 2014.

[12] K. L. Bryan, T. Ren, L. DiPippo, T. Henry, and V. Fay-Wolfe, "Towards optimal tdma frame size in wireless sensor networks," Tech. Rep., University of Rhode Island, 2007.

[13] 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, 2012.

[14] S. Gandham, Y. Zhang, and Q. Huang, "Distributed minimal time convergecast scheduling in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 50, Lisboa, Portugal, July 2006.

[15] G. Brar, D. M. Blough, and P. Santi, "The SCREAM approach for efficient distributed scheduling with physical interference in wireless mesh networks," in *Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS '08)*, pp. 214–224, Beijing, China, July 2008.

[16] W. Ge, J. Zhang, J. E. Wieselthier, and X. Shen, "PHY-Aware distributed scheduling for ad hoc communications with physical interference model," *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2682–2693, 2009.

[17] T. Moscibroda, R. Wattenhofer, and A. Zollinger, "Topology control meets sinr: the scheduling complexity of arbitrary topologies," in *Proceedings of the 7th ACM International Symposium on Mobile ad hoc Networking and Computing*, pp. 310–321, ACM, 2006.

[18] J. Huang, S. Liu, G. Xing, H. Zhang, J. Wang, and L. Huang, "Accuracy-aware interference modeling and measurement in wireless sensor networks," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 172–181, Minneapolis, Minn, USA, July 2011.

[19] S. H. Cheng and C. Y. Huang, "Coloring-based inter-WBAN scheduling for mobile wireless body area networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 2, pp. 250–259, 2013.

[20] L. Tao and F. Yu, "Delay-jitter aware slot assignment for real-time applications in wireless multimedia ad hoc sensor networks," *Computer Communications*, vol. 35, no. 16, pp. 1967–1982, 2012.

[21] I. Amdouni, P. Minet, and C. Adjih, "OSERENA: a coloring algorithm optimized for dense wireless networks," *International Journal of Networked and Distributed Computing*, vol. 2012, 2012.

[22] F. Zhang, P. Cheng, J. Chen, Y. Sun, and X. Shen, "Distributed saturation degree based tdma scheduling algorithm for target tracking," in *Proceedings of the IEEE International Conference on Communications (ICC '11)*, pp. 1–5, June 2011.

[23] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '06)*, pp. 190–201, May 2006.

[24] K. Sato and S. Sakata, "A distance-measurement-oriented distributed TDMA scheduling algorithm for sensor networks," in *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '11)*, pp. 1–3, June 2011.

[25] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 181–192, November 2003.

[26] C. Yu and E. Fleury, "A distributed policy scheduling for wireless sensor networks," in *Proceeding of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1559–1567, Anchorage, Alaska, USA, May 2007.

[27] Y. Zhao, J. Wu, F. Li, and S. Lu, "On maximizing the lifetime of wireless sensor networks using virtual backbone scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1528–1535, 2012.

[28] L. Shi and A. O. Fapojuwo, "TDMA scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 7, pp. 927–940, 2010.

[29] H. Byun and J. Yu, "Adaptive duty cycle control with queue management in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1214–1224, 2013.

[30] X. Lin and S. B. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Proceedings of the 45th IEEE Conference on Decision and Control (CDC '06)*, pp. 1258–1263, December 2006.

[31] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1846–1859, 2009.

[32] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," in *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '07)*, vol. 35, pp. 313–324, June 2007.

[33] H. Zhang and H. Shen, "Balancing energy consumption to maximize network lifetime in data-gathering sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 10, pp. 1526–1539, 2009.

[34] W. Qiu, E. Skafidas, and P. Hao, "Enhanced tree routing for wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 3, pp. 638–650, 2009.

[35] 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 (SenSys '09)*, pp. 1–14, New York, NY, USA, November 2009.

[36] I. Raicu, L. Schwiebert, S. Fowler, and S. K. Gupta, "Local load balancing for globally efficient routing in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 1, no. 2, pp. 163–185, 2005.