Regular Paper

Scheduling Beacon Transmission to Improve Delay for Receiver-initiated-MAC Based Wireless Sensor Networks

Akihiro Fujimoto[1,a) Yukari Masui[2] Takuya Yoshihiro[2,b)

Received: May 10, 2017, Accepted: September 5, 2017

Abstract: Many low-energy MAC and routing protocols have been proposed for Wireless Sensor Networks (WSNs) since reducing energy consumption is a primary concern to meet the requirements of practical applications. Reducing delivery delay is, indeed, another promising requirement in WSNs because a large part of applications of WSNs intends to watch the sensed objects in real time although the amount of allowable delay differs according to cases. In this paper, we deal with a class of receiver-initiated MAC protocols, which is a promising energy-efficient MAC mechanism, and propose to schedule beacon timings so as to reduce the delivery delay of data packets. The key idea is to schedule beacon timings in the sequence of distance from leaf node to the sink node and reduce the time of packets staying at each node. Specifically, each sensor node selects a time slot to send beacon in a distributed manner based on their distances (i.e., hop counts) from the sink node, and autonomously adjusts its beacon timing to avoid frame collisions. Computer simulations show that the proposed protocol can collect data in a shorter time with less energy consumption than the conventional receiver-initiated MAC protocols.

Keywords: beacon scheduling, wireless sensor networks, receiver-initiated MAC protocols, low delivery delay

1. Introduction

In recent years, wireless sensor networks (WSNs) have attracted a lot of attention as an infrastructure for various monitoring applications, such as environmental monitoring [1], health care [2] and surveillance [3]. In general, WSNs consist of a large number of low-cost battery-powered devices (called "nodes") that have sensing and wireless communication functions. While WSNs can be easily deployed due to the battery-powered nodes, reducing energy consumption has been a primary concern to meet the requirements for network life-time of practical applications.

Many researchers propose MAC protocols utilizing "duty cycling" [4], [5] for energy saving in WSNs. In the duty cycling technique, each node turns its radio modules on and off according to its active/sleep schedules. Duty cycling can drastically reduce energy consumption, since, in general, radio modules consume most of the energy in WSNs. For example, TelosB TRP2420CA [6] consumes about 23 mA, 21 μA, and 1 μA in receive mode, idle mode, and sleep mode, respectively.

Receiver-initiated MAC protocols [7], [8], especially, can achieve good duty-cycling properties compared with sender-initiated MAC protocols [9], [10], [11]. In these protocols, receiver nodes are responsible for starting communication. When a node has data frames in its transmission queue, it transits to active mode and waits for the receiver to be also active. On the other hand, the receiver node informs the sender nodes that it is ready to receive data frames by small beacon frames. When the sender node can receive the beacon frame, it transmits the data frames to the receiver. Note that if no frame arrives after the beacon frame, the receiver can return to sleep state quickly. By reducing signaling duration, the receiver-initiated protocols can improve the energy efficiency and the spatial reusability.

Improving data collection delay is another promising requirement in WSNs, because a large part of monitoring applications intends to watch the sensed objects in real time although the amount of allowable delay differs depending on situations. In general, sensed data is delivered to a sink node through a tree-shaped multi-hop network. Therefore, to deal with increasing demand for real-time monitoring applications, it is required to relay data frames with short waiting time at each hop. However, although receiver-initiated MAC protocols are promising approaches, most of these protocols are not designed to consider the per-hop delay.

In this paper, we propose a novel receiver-initiated MAC protocol to improve the data delivery delay. The key idea of the proposed protocol is to schedule beacon timings in the sequence of distance from leaf nodes to the sink node so that data frames are forwarded with small duration staying at each node. The proposed protocol constructs a tree-shaped network for data collection, and schedules beacon timing of each node based on topology information. Specifically, in the proposed protocol, an operation cycle (that corresponds to beacon interval) is divided into time slots. Each sensor node selects a slot to send beacon in a distributed manner based on their distances (i.e., hop counts) from the sink node. In addition, in order to avoid frame collisions within a slot, each node adjusts its beacon transmission timing within the selected slot.

The contributions of this paper are as follows: 1) *guaranteed*

[1] Center for Information Science, Wakayama University, Wakayama, Wakayama 640–8510, Japan
[2] Faculty of System Engineering, Wakayama University, Wakayama, Wakayama 640–8510, Japan
a) fujimoto@center.wakayama-u.ac.jp
b) tac@sys.wakayama-u.ac.jp

delay bound: The proposed protocol assigns beacon transmission of each node into the time slot corresponding to its hop count. This means that the maximum per-hop delay is guaranteed to be less than the slot duration. As a result, the proposed protocol can improve the leaf-to-sink delivery delay. 2) *improved power efficiency:* The proposed protocol schedules beacon timings with strict time offsets. Therefore, each node can predict its parent's next beacon timing strictly so that it can sleep until just before the predicted time. As a result, the proposed protocol can improve the energy efficiency by reducing idle listening.

The reminder of this paper is as follows. Related work is introduced in Section 2. Section 3 introduces receiver-initiated paradigm, which is the basis of our proposed protocol. Section 4 describes the detail of the proposed protocol, and it is evaluated in Section 5. Section 6 concludes this paper.

2. Related Work

Only a few receiver-initiated MAC protocols considering the data collection delay have been proposed [12], [13]. In these protocols, each node schedules its active/sleep state leveraging the topology information (i.e., hop counts from the sink node).

Wada et. al. extends RI-MAC by applying the stair-like sleep mechanism [12], where RI-MAC [7] is the representative receiver-initiated MAC protocol. They found that heavy contention due to many competing senders wastes significant listening time, resulting in increasing delay as well as wasting energy. Note that several studies such as Ref. [15] reported that the idle listening consumes significant power and is a major factor to reduce network lifetime. Their idea in Ref. [12] for this problem is to determine sleep-time duration according to the distance (i.e., hop-count) from the sink node, i.e., nodes closer to the sink wake up more frequently. They showed that the wasteful listening time is reduced due to a large number of competing nodes, resulting in low delay and improved network capacity. However, their method in turn requires near-sink nodes to wake-up frequently, which reduces efficiency in power consumption.

REA-MAC [13] schedules beacon transmissions in a similar way to our proposed method, namely, each node selects a slot to broadcast a beacon based on its hop count from the sink node so that frames are forwarded in a pipeline manner. When some frames remain in the queue at the end of a time slot, the corresponding node pair can use the succeeding time slot to send the remaining frames.

The main difference between REA-MAC and the proposed method is the method to avoid beacon collision. While REA-MAC avoids beacon collision by randomizing beacon timings within the selected slot, the proposed method coordinates beacon timings with neighbor nodes. Since the senders can easily predict receivers' beacon timings in the proposed method, they can sleep until just before the predicted timings. As a result, the proposed method can save idle listening time, and energy.

3. Receiver-Initiated MAC Protocol (RI-MAC) [7]

This paper proposes to schedule beacon transmissions on the basis of the receiver-initiated paradigm, because it has potential to significantly improve the energy efficiency of WSNs. The receiver-initiated paradigm allows nodes to operate at low duty cycle. In this section, we briefly introduce RI-MAC, which is the representative receiver-initiated MAC protocol. **Figure 1** shows the frame transmission of RI-MAC. Note that this is the common and fundamental operation in receiver-initiated MAC protocols. In the figure, black and white boxes represent transmitted frame and received frame, respectively. Boxes written as "B" and "AB" represent beacons and ACK beacons, respectively, where ACK beacon has both roles of beacons and ACK frames. Active period of each node is shown in gray area, whereas sleep period in white area. In RI-MAC, each node broadcasts a short beacon frame when it wakes up and is ready to receive data frames. The beacon works as the trigger to start data transmissions. If no data packet arrives during a certain period after the beacon transmission, it means that no neighboring node has data frames to send. Therefore, the node quickly returns to sleep state and reduces the energy consumption.

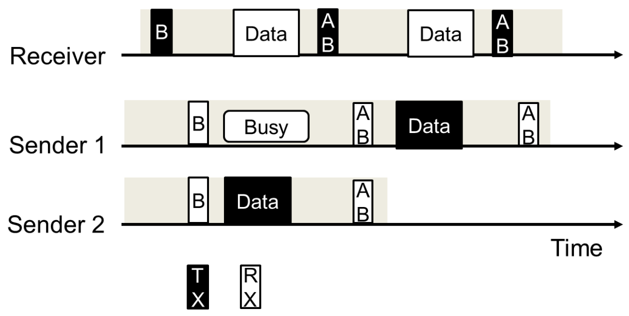

Fig. 1 Frame transmission in RI-MAC.

On the other side, a sender who has data frames in its transmission queue listens to the channel and waits for beacons. If it receives a beacon from the receiver of the next frame, it transmits the frame after the random back-off. Note that the idle listening time can be significantly reduced if the sender wakes up just before the beacon by predicting the timing of the beacon. On receiving the data frame, the receiver sends back an ACK beacon frame that returns acknowledgment to the sender while inviting another data frame transmission.

Since beacon frames have an important role in frame transmissions, beacon transmissions should be carefully scheduled to avoid collisions. In RI-MAC, each node schedules next beacon transmission after random time between $[\frac{1}{2}T, \frac{3}{2}T]$, where T is the average beacon interval, to avoid continuous beacon collisions between two nodes.

Another notable property of the receiver-initiated paradigm is that receivers can control senders' behaviors with beacons. RI-MAC [7] includes the back-off window size in beacons to control senders' back-off strategy for the improved communication efficiency. Specifically, RI-MAC sets initial back-off window size to zero, so that the receiver can quickly detect whether anyone attempts to send or not. When the receiver detects frame collisions, it re-calculates the back-off window size based on the traffic demand and notifies it by the next beacon that invites frame retransmission. This strategy can efficiently reduce energy consumption in light-traffic environment. Note that some receiver-

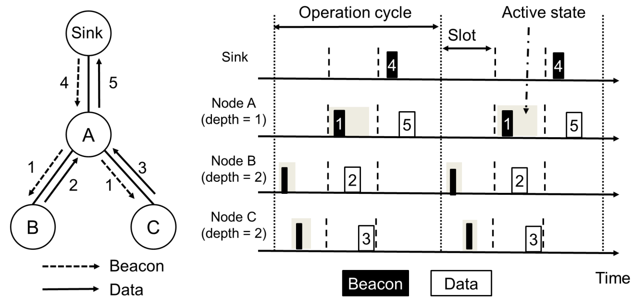

Fig. 2 Frame transmission order of the proposed protocol.

initiated MAC protocols adopt different strategies. For example, RC-MAC [8] achieves collision-free transmissions by designating the next sender within ACK beacon. These protocols efficiently utilize radio resources and avoid retransmissions to improve the energy efficiency of communications.

4. Receiver-initiated MAC Protocol with Depth-aware Beacon Scheduling

We propose a receiver-initiated MAC protocol that schedules beacon transmissions based on *node depth*, i.e., the node's distance from the sink node in hop-counts, so that data frames can be smoothly forwarded with small delay. In this paper, we extend RI-MAC [7] to achieve the above property. The major extensions are twofold: 1) integration of topology management, and 2) slot-based static beacon scheduling considering node depth.

In addition, we extend back-off mechanism of RI-MAC to support control messaging. Different from original RI-MAC, our protocol uses a non-zero value as the initial back-off window size. When a node attempts to send a control message, it sends the message immediately after the beacon reception. In this way, control messages have higher priority than data frames.

4.1 Frame Transmissions

Figure 2 shows the behavior of the proposed protocol on frame transmissions, where ACK beacon is omitted for visibility. Nodes share a common network-wide *operation cycle* and divide it into N time slots. We assume that the length of the operation cycle corresponds to the beacon interval.

Each node transmits beacons in a suitable time slot corresponding to its depth, in which nodes farther from the sink send beacons earlier so that the staying time of data at each node becomes small. For example, the operation cycle is divided into three frames (i.e., $N = 3$) in Fig. 2. Nodes B and C first send beacons in the 1st time slot. (This time no data frame transmission occurs since B and C are the deepest nodes in this network.) Then, node A, whose depth is 1, wakes up to send a beacon in the 2nd slot. After receiving beacons from A, B and C send data frames to A according to the RI-MAC manner (see Section 3), and fall into the sleep state. In the same way, node A sends those data frames as well as its own one to the sink during the 3rd time slot. As we have shown in this example, data frames are smoothly forwarded from leaf nodes to the sink node within a single *operation cycle*, i.e., within a certain delay bound.

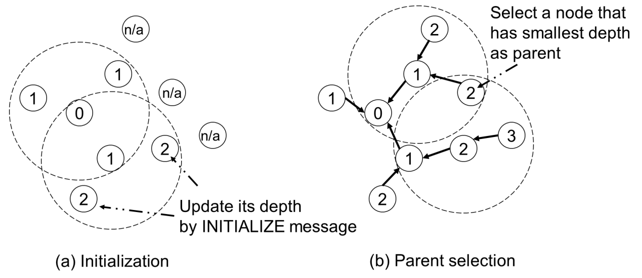

Fig. 3 Tree construction.

4.2 Topology Management

4.2.1 Tree construction

When starting a data collection application, the sink node initializes the network by flooding INITIALIZE messages to construct a data collection tree. An INITIALIZE message contains parameters to schedule beacon timings: operation cycle T, start time of operation cycle t_{start}, number of slots N, and sub-slot length δ. Note that sub-slot is used to adjust beacon timings between nodes with the same depth, which is described in Section 4.3. Node depth (i.e., hop count from the sink node) of the forwarder is also included in it. INITIALIZE messages are sent repeatedly to avoid frame losses.

Figure 3 shows an example of the tree construction. Circles and numbers inside of them mean nodes and their depth, respectively. In particular, the node with depth 0 is the sink node. Depth n/a means that the depth is not determined. Each dashed circle is the communication range of the sender. In other words, nodes within the range can receive the INITIALIZE message. When a node receives the INITIALIZE message, the node transits to *tree construction* state. In tree construction state, each node initializes the beacon scheduling parameters with the informed values in the received INITIALIZE message. The node also checks the sender and depth field of it to record them as the neighbor information. If the informed depth, $d - 1$, is smaller than any known depths of the other neighbors, the node updates its depth to d (Fig. 3 (a)). The neighbor nodes whose depth is $d - 1$ are marked as parent candidates. Each node broadcasts an INITIALIZE message with its depth whenever its depth is updated.

When no INITIALIZE message arrives during certain period, T_{init}, the node selects one parent node from its parent candidates (Fig. 3 (b)). Then, the node sends a JOIN message to the selected parent. When the joining node can receive an ACCEPT message from the parent node, the node transits to *beacon scheduling* state. Otherwise, the node contacts another parent candidate. Note that the best node that should be selected as parent differs depending on applications and/or routing policies. Therefore, the parent selection algorithm is out of scope of this paper.

Even when a node is in the tree construction state, it should send (or forward) sensed data to the sink node. In this paper, we assume that the node randomly selects a receiver from its parent candidates in each frame transmission, if parent node is not selected yet.

4.2.2 Joining Constructed Data Collection Tree

When a node attempts to join a constructed data collection tree, it should find and contact parent node without disturbing its neighbors' communication. **Figure 4** shows an example of the

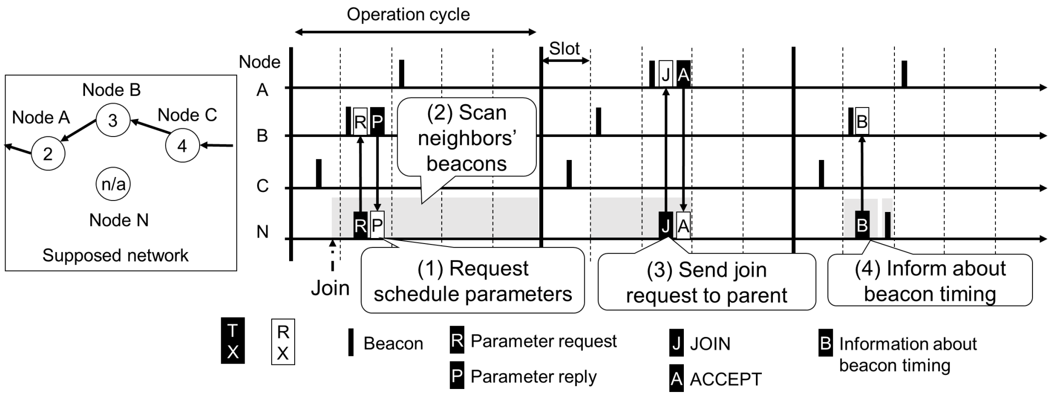
Operation cycle
Slot
Node
A
B
C
N
Node B
Node A
Node C
3
2
4
n/a
Node N
Supposed network
(2) Scan neighbors' beacons
R P
J A
B
Join
(1) Request schedule parameters
(3) Send join request to parent
(4) Inform about beacon timing
TX
RX
Beacon
R Parameter request
P Parameter reply
J JOIN
A ACCEPT
B Information about beacon timing

Fig. 4 Join process of new node N.

joining process, where newly joining node N is focused on. Node N first passively scans neighbors' beacons. When it receives the first beacon from any other node, it requests the scheduling parameters from the beacon's sender (see process (1) in Fig. 4). In addition, if the joining node has some data to send, it also sends the data to the beacon's sender. Contacting the first beacon's sender ensures communication within an operation cycle. Otherwise it may lose the opportunity for communication, since it does not know how many neighbors there are nor when they can communicate with it.

After receiving the first beacon, the joining node continues the passive scan to know its position (i.e., depth) on the data collection tree. During the passive scan (see process (2) in Fig. 4), the joining node records the neighbors' depths and their beacon timings. Obviously, each node can know all its neighbor nodes when it scans neighbors' beacons during operation cycle T. However, the characteristics of the depth-based beacon scheduling can shorten the scanning period. Since neighbor nodes will send beacons at similar timings, the joining node can recognize there will not be any other neighbor nodes when no beacon is detected during a certain period. In such a case, the joining node turns its radio module off. In this paper, we set this time period to the duration of two consecutive slots.

After the passive scan, the node determines its depth as d, when $d - 1$ is the minimum one among neighbors' depths. When the joining node determines its depth, it schedules to wake up at slots corresponding to depths d and $d - 1$. In the slot for depth d, the joining node collects neighbors' beacons to determine its beacon timing (that is described in Section 4.3). Then, it sends a JOIN message to a selected parent candidate triggered by the beacon of the parent candidate at the slot for depth $d - 1$ (see process (3) in Fig. 4).

To avoid frequent tree reconstructions, we assume a new node joins as a leaf node. This means no node sends data frames to the joining node. Therefore, the joining node need not inform deeper nodes about its existence. On the other hand, when the neighbors of the joining node are at depth d, they have to know the beacon timing of the joining node. This is because the beacon timing of the joining node has to be taken into consideration in beacon timing adjustment (see Section 4.3) to avoid frame collisions. Unfortunately, however, it is hard for the joining node to broadcast a message to the neighbor nodes. Although all of them wake up to send beacons at slot for depth d, their wake-up (and beacon transmission) timings are different from each other to avoid collisions. Therefore, the joining node waits for its neighbors' beacons during the slot for depth d, and informs each node about its own beacon timing by unicast (see process (4) in Fig. 4).

4.3 Scheduling Beacons Based on Node Depth

The proposed beacon scheduling method consists of two phases: initial phase and adjustment phase. In the initial phase, nodes roughly schedule their slots to send beacons by selecting time slots corresponding to their depth on the data collection tree. After that, in the adjustment phase, each node adjusts its time to send beacons to avoid frame collisions. The time adjustment behavior is triggered when a node finds beacons of the neighbors colliding with its own ones. When the collision is detected, the node changes the beacon timing randomly within its own time slot to avoid collision with its neighbors in the next operation cycle. Note that, during these two phases, a node has to record the time of beacons received from its parent (in the tree) and other neighbors. To do this, it has to stay in active state during the slot corresponding to the node.

In the following part, we describe the specific operations in the initial and the adjustment phases.

Initial Phase:

After joining the data collection tree, each node starts scheduling its beacon timing. In initial phase of beacon scheduling, each node determines its own slots to send beacons based on its depth in the collection tree, which it knows about during the joining process.

First, each node divides the operation cycle, whose length is T, into N slots. Note that the parameters T and N are informed by the sink node during the tree construction period. Then, the node selects its time slot to send beacons based on its depth. Specifically, when the operation cycle is divided into slots $\{S_0, S_1, \cdots, S_{N-1}\}$, a node with depth d selects slot S_i according to Eq. (1):

$$i = N - 1 - (d \bmod N). \qquad (1)$$

The node determines its initial offset in its slot to send beacons by randomly selecting a sub-slot within the selected slot S_i. The sub-slot selection behavior is illustrated in **Fig. 5**. In order

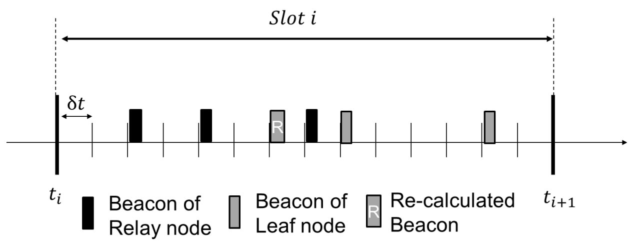

Fig. 5 Beacon timing selection within time slot.

Algorithm 1 Operation in Initial Phase
```
Input: operation cycle length $T$, number of slots $N$, sub-slot length $\delta t$,
  node depth $d$
  //Select a slot $S_i$ based on depth $d$
  $i \Leftarrow N - 1 - (d \bmod N)$
  $t_i \Leftarrow$ start time of $S_i$
  //Select a sub-slot of slot $S_i$
  Randomly select integer $r$, s.t. $r \times \delta t < \frac{T}{2N}$
  if relay node then
    $b_i \Leftarrow t_i + r \times \delta t$
  else
    $b_i \Leftarrow t_i + \frac{T}{2N} + r \times \delta t$
  end if
  Sleep until $b_i$, then broadcast a beacon
  Record neighbors' beacon timings $\boldsymbol{B} = \{b_i^{(n)}\}, n = 0, 1, \cdots$ during the time
  period from $b_i$ to $t_i + \frac{T}{N}$
  Record parent's beacon timing $b^p_{(i+1) \bmod N}$ during the time period from $t_i + \frac{T}{N}$
  to $t_i + \frac{2T}{N}$
  $t_i \Leftarrow t_i + T$
  Sleep until $t_i$ and transit to Adjustment Phase in next operation cycle
```

Fig. 6 Slot selection in the initial phase.

to smoothly forward data frames, it is preferable that relay nodes in the data collection tree have more opportunities to receive data frames from their children. Therefore, in the proposed scheduling method, nodes are to select the different parts of sub-slots as an initial beacon transmission offset based on whether they are relay node or not, i.e., relay nodes select the first half of sub-slots while leaf nodes do the second half. Specifically, when slot S_i starts at time t_i, the time for transmitting beacons b_i is calculated by Eq. (2):

$$b_i = \begin{cases} t_i + \delta t \times r & \text{(if relay node)}, \\ t_i + \dfrac{1}{2}\dfrac{T}{N} + \delta t \times r & \text{(otherwise)}, \end{cases} \tag{2}$$

where δt is the predefined time length of sub-slot that is sufficient to send several data frames, and r is a random integer value in range $\left[0, (\frac{1}{2}\frac{T}{N} - \delta t)/\delta t\right]$.

Since each node transits its state autonomously, it is possible that some nodes are in the tree construction state although the other ones are in the beacon scheduling state. If the depth is updated by INITIALIZE message, the node executes the above procedure, which formal description is shown in **Fig. 6**. When the depth is not updated for a certain period of time, the node transits to the *adjustment phase*.

Adjustment Phase:

After the initial phase finishes, the node transits to the adjustment phase, in which each node adjusts the offset (i.e., sub-slot) to transmit beacons such that no collision occurs with the neighbor's beacons. To judge whether a collision occurs or not, nodes utilize the time of neighbor beacons recorded in the *initial state*. Specifically, each node records the offset between its neighbors' and own beacon timings. For example, when node A sends the

Algorithm 2 Operation in Adjustment Phase
```
Input: operation cycle length $T$, number of slots $N$, sub-slot length $\delta t$,
  slot start time $t_i$, own beacon time $b_i$, neighbors' beacon times $\boldsymbol{B}$
  //Check whether there are beacons could collide with its beacon
  $f \Leftarrow 0$
  for all $b_i^{(n)} \in \boldsymbol{B}$ do
    if $b_i^{(n)} - b_i < \delta t$ then
      $f \Leftarrow 1$, then break loop
    end if
  end for
  if f == 1 then
    //Make a candidate list $\boldsymbol{C}$ for alternative beacon timing
    $\boldsymbol{C} \Leftarrow \emptyset$
    $u \Leftarrow t_i$
    while $u < t_i + \frac{T}{N}$ do
      $\boldsymbol{C} \Leftarrow \boldsymbol{C} \cup u$, if $b_i^{(n)} - u > \delta t, \forall b_i^{(n)} \in \boldsymbol{B}$
      $u \Leftarrow u + \delta t$
    end while
    $b_i \Leftarrow u$, which is randomly selected from $\boldsymbol{C}$
  end if
  Sleep until $b_i$, then broadcast a beacon
  Record neighbors' beacon timings $\boldsymbol{B} = \{b_i^{(n)}\}, n = 0, 1, \cdots$ during the time
  period from $b_i$ to $t_i + \frac{T}{N}$
  Record parent's beacon timing $b^p_{(i+1) \bmod N}$ during the time period from $t_i + \frac{T}{N}$
  to $t_i + \frac{2T}{N}$
  $t_i \Leftarrow t_i + T$
  if $f == 0$ and $\boldsymbol{B}$ is identical with previous $\boldsymbol{B}$ then
    Finish Adjustment Phase
    $b_i \Leftarrow b_i + T$
    Sleep until $b_i$
  else
    Sleep until $t_i$ and repeat Adjustment Phase
  end if
```

Fig. 7 Beacon timing adjustment in the adjustment phase.

beacon at $b^{(A)}$ and receives the beacon from neighboring node B at $b^{(B)}$, node A records the offset $b^{(B)} - b^{(A)}$. When some offsets are less than δt, the node judges that collisions could occur and reschedules its own beacon timing.

Nodes reschedule their beacon timings in a similar way as scheduling in the initial phase. The differences are the two restrictions that avoid new collision brought from moving the sub-slots of beacons. The first one is excluding sub-slots that have been occupied by another neighbor's beacons, which directly avoids collisions. The other is using all sub-slots instead of selecting a half part of them according to the type (i.e., relay or leaf) of nodes, which reduces the collision probability. Formally, we show the pseudo code of those operations in **Fig. 7**.

Note that one problem arises in avoiding collision: when multiple nodes transmit beacons simultaneously and collide, nodes around there cannot even tell which beacons collide. Namely, if a node sends a beacon and it collides, the node cannot detect the collision as well as the collided neighbor. To solve the problem, we in the proposed scheduling method introduce NACK frames. Specifically, we suppose three nodes A, B, and C, where node A is a hidden node of node C (and vice versa), and node B is in both of the communication ranges of nodes A and C. If A and C transmit beacons simultaneously, B receives an undecodable signal due to collisions. In this case, B broadcasts a NACK frame as the response of the signal to notify that the collision occurs. When nodes A and C receive the NACK frame as the response of their beacon, they start to move their sub-slots in the same way as described above.

During *adjustment phase*, every node keeps in active state to check the offset for neighbors' beacons until the end of its own slot. When a node observes that no neighboring node changes its beacon timing during two consecutive operation cycles, the node ends the adjustment phase and starts duty-cycling.

4.4 Duty-Cycling

In duty-cycling, nodes basically have only to periodically wake up and transmit beacon frames. Only when it has some data frames, it additionally wakes up to transmit the data frames slightly before the predicted parent's beacon timing. In this paper, we assume that nodes wake up 50 ms before the parents' beacon timings to absorb the influence of so-called clock drift. When the node receives ACK beacon after the data transmission, it checks its transmission queue. If some data frames remain, the node continuously attempts to send the remained frames. Otherwise, it returns to sleep state.

As long as senders have data frames in their queues, the receiver repeatedly transmits beacon frames (typically, as ACK beacons). However, if it is expected that data transmission would not be completed at the end of the time slot, the receiver sets the end flag in the beacon and transmits it. When senders receive a beacon with the end flag, they stop sending data frames and return to sleep state. Then the receiver waits for its parent's beacon.

A node may receive no signal (including error frames) from its parent on the scheduled beacon timing for several reasons, such as degraded channel condition or move of the parent node. In such case, it keeps wakeup state and waits for beacons of other parent candidates. When it receives a beacon from another parent candidate, it regards the beacon's sender as a temporal parent and forwards data frames. Note that if it cannot receive any beacon until the end of the parents' slot, it stores the data in its buffer.

If a node cannot receive any beacon from its parent during three consecutive operation cycles, it judges that the parent node becomes not available and attempts to rejoin the data collection tree. Since the rejoining node already knows the parent candidates, it just sends a JOIN message to another parent candidate. If any alternative parent is not available (i.e., the lost parent was the last parent candidate), the rejoining node reconstructs the subtree whose root is itself. In this reconstruction, the rejoining node increases its depth by one (i.e., its depth is updated to $d + 1$) and informs its children that it is no longer their parent by sending a beacon frame with the incremented depth. Receiving this beacon frame, the children of the rejoining node select another parent node. Then, the rejoining node sends a JOIN message to a selected neighbor node whose depth is d.

5. Performance Evaluation

We evaluated the performance of the proposed protocol through computer simulation. We used a handmade simulator written in Java, which simulates MAC protocols over a predefined collection tree. To evaluate the performance of the proposed protocol, we compared the proposed protocol with two methods: "Random" and "REA-MAC [13]," in terms of data collection delay and energy consumption. In Random method, each node randomly selects the time of its beacon transmission using whole operation cycle, instead of applying time slot. The other behaviors of Random method are the same as the proposed protocol.

Table 1 Parameters about wireless communication.

Parameter	Value
Bandwidth	250 [kbps]
SIFS	192 [μs]
CCA	128 [μs]
Data frame length	25 [Byte]
ACK frame length	variable
Beacon frame length	6 [Byte]
Beacon listening time	50 [ms]
Data listening time after beacon	0.3 [s]
Sensing interval	300 [s]

Table 2 Parameters about the proposed protocol.

Parameter	Value
Operation cycle length	20 [s]
Number of slots N	10
Minimum beacon gap δt	50 [ms]

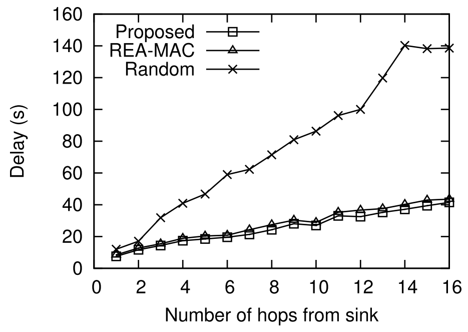

Fig. 8 Data collection delay with node depth.

5.1 Simulation Environment

In the simulations, each sensor node equips single omni-directional antenna. For simplicity, the simulation uses protocol interference model [14]. Both the transmission range and the interference range are set to 100 [m]. Nodes communicate with each other according to modified RI-MAC [7] in the proposed protocol and Random method. In modified RI-MAC, nodes periodically transmit their beacon, and Beacon-on-request function is omitted. The parameters over wireless communications are listed in **Table 1**. The sensor nodes are randomly placed in a 1,000 [m] $\times$ 1,000 [m] field. Note that the nodes do not move through simulations in this paper. As a data collection tree, we construct the tree that minimizes the number of relay nodes as a typical tree to collect data. When a node is isolated, those nodes are removed from the simulations. After the tree is constructed, nodes start to schedule their beacon transmissions via *initial* and *adjustment* states. The parameters of the proposed protocol are listed in **Table 2**. Under the above conditions, we repeated the simulation 30 times.

5.2 Simulation Results

First, we evaluate data collection delay of the proposed protocol, where the data collection delay is defined as the delay from the data generation time to the arrival time at the sink node. **Figure 8** compares the data collection delay of the three methods (i.e., the proposed protocol, Random method and REA-MAC,) for each depth of nodes when the number of nodes in the field is 200. In this figure, we see that the proposed protocol and REA-MAC have a similar performance that is far better than the ran-

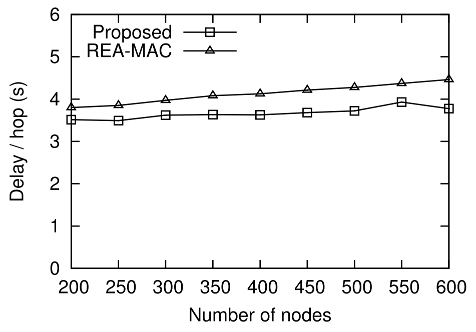

Fig. 9 Average delivery delay per hop with various number of nodes in the field.

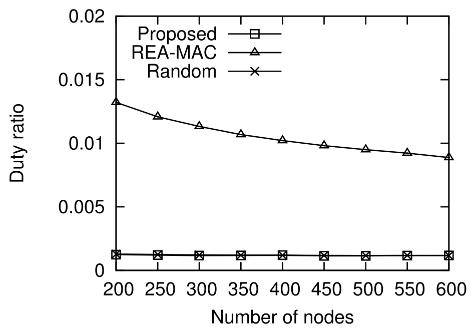

Fig. 10 Average duty ratio with various number of nodes.

dom one. Since both the proposed and REA-MAC schedule beacons according to the depth of nodes, data frames are smoothly forwarded without staying at a node for a long time, which results in low collection delay.

Figure 9 shows the average delivery delay per hop under the various numbers of nodes in the field. Note that the results of Random case is omitted due to visibility; the values in Random case is about 10 seconds in every case. From the results, we see that the proposed protocol slightly outperforms REA-MAC. This is because the proposed protocol assigns the sub-slots of relay nodes in the first half of a slot. As a result, larger parts of frames are transmitted earlier in the *operation cycle*, which reduces the delay performance per hop. We also see that the per-hop delay slightly increases as the number of nodes increases in both methods. This is because some data frames could not be transmitted within one operation cycle when nodes were densely placed. Here, we would note that the proposed protocol improves the performance larger than REA-MAC in terms of the increase rate, which indicates that the proposed protocol is better in treating collisions.

Finally, we evaluated the energy efficiency of the three methods. In this paper, we use the duty ratio, which is calculated by active time/whole simulation time, as the index of the energy efficiency. **Figure 10** shows the average duty ratio under various numbers of nodes. Although the proposed protocol induces additional overhead compared with Random method, they show almost the same performance. We also see that both of Random and the proposed protocol significantly outperform REA-MAC in terms of the duty ratio. In REA-MAC, nodes have to wake up before the start times of time slots which have the parents' beacon timings. On the contrary, nodes can easily wake up just before parents' beacon timings in both of Random and the proposed protocol due to their periodic beacon transmission nature. Since they can reduce idle listening time, it is expected that they are effective to reduce energy consumption.

Note that idle listening has a significant impact on energy consumption as well as the lifetime of sensor nodes [15], which would strengthen the efficacy of the proposed protocol against the conventional REA-MAC. Moreover, their duty ratio are highly depend on the listening time for parents' beacons, which is set to 50 ms in the simulations. Therefore, when a shorter listening time is applied, duty ratio could be improved. Note that, however, more precise synchronization will be required to use such a short listening time.

From the above results, we found that the proposed protocol can collect data frames in a short time with low energy consumption.

6. Conclusion

In this paper, we proposed a receiver-initiated MAC protocol that schedules beacon transmissions to improve data collection delay. The proposed protocol was designed to leverage the information of the data collection trees. By selecting a time slot for beacon transmission based on the nodes' depth, the proposed protocol smoothly forwards data frames from its children to its parent without holding the frames at each node for a long time. In addition, in order to avoid collisions, each node adjusts its beacon timing within the selected slot. Through computer simulations, we confirmed that the proposed protocol collects data frames with slightly better delay performance than the conventional method, but with significant improvement on energy consumption.

As future work, we develop a method to divide operation cycle into slots adaptively based on the traffic demand.

Acknowledgments This work was supported by Japan Society for the Performance of Science (JSPS) under Grant KAKENHI 15H02691.

References

[1] Oliveira, L.M. and Rodrigues, J.J.: Wireless sensor networks: A survey on environmental monitoring, *Journal of Communications*, Academy Publisher, Vol.6, No.2, pp.143–151 (Apr. 2011).

[2] Zhang, Y., Sun, L., Song, H. and Cao, X.: Ubiquitous wsn for healthcare: Recent advances and future prospects, *Internet of Things Journal*, Vol.4, No.1, pp.311–318, IEEE (2014).

[3] Chen, M., Gonzalez, S., Cao, H., Zhang, Y. and Vuong, S.T.: Enabling low bit-rate and reliable video surveillance over practical wireless sensor network, *Journal of Supercomputing*, Vol.65, No.1, pp.287–300, Springer (2013).

[4] Doudou, M., Djenouri, D. and Badache, N.: Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks, *Communication Surveys and Tutorials*, Vol.15, No.2, pp.528–550, IEEE (2013).

[5] Carrano, R.C., Passos, D., Magalhaes, L. and Albuquerque, C.: Survey and taxonomy of duty cycling mechanisms in wireless sensor networks, *Communication Surveys and Tutorials*, Vol.16, No.1, pp.181–194, IEEE (2014).

[6] CC2420 Single-chip 2.4 GHz rf transceiver, available from ⟨http://focus.ti.com/lit/ds/symlink/cc2420.pdf⟩.

[7] Sum, Y., Gurewits, O. and Johnson, D.B.: RI-MAC: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks, *Proc. 6th ACM Conference on Embedded Networked Sensor Systems* (*SenSys'08*), pp.1–14 (2008).

[8] Huang, P., Wang, C. and Xiao, L.: RC-MAC: A receiver-centric mac protocol for event-driven wireless sensor networks, *Trans. Computers*, Vol.64, No.4, pp.1149–1161, IEEE (Mar. 2015).

[9] Polastre, J., Hill, J. and Culler, D.: Versatile low power media access

for wireless sensor networks, *Proc. 2nd ACM Conference on Embedded Networked Sensor Systems* (*SenSys 2004*), pp.95–107 (2004).

[10] El-Hoiydi, A. and Decotignie, J.: WiseMAC: An ultra low power mac protocol for multi-hop wireless sensor networks, *Proc. 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks* (*ALGOSENSORS 2004*), pp.18–31 (2004).

[11] Buettner, M., Yee, G., Anderson, E. and Han, R.: X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks, *Proc. 4th ACM Conference on Embedded Networked Sensor Systems* (*SenSys'06*), pp.307–320 (2006).

[12] Wada, T., Lin, I.T. and Sasase, I.: Asynchronous receiver-initiated mac protocol with the stair-like sleep in wireless sensor networks, *Proc. 22nd International Symposium on Personal Indoor and Mobile Radio Communications* (*PIMRC*), pp.854–858 (2011).

[13] Tang, H., Cao, J., Sun, C. and Lu, K.: REA-MAC: A low latency routing-enhanced asynchronous duty-cycle mac protocol for wireless sensor networks, *Journal of Central South University*, Vol.20, No.3, pp.678–687 (2013).

[14] Gupta, P. and Kumar, P.R.: The capacity of wireless networks, *Trans. Inf. Theory*, Vol.46, No.2, pp.388–404, IEEE (2000).

[15] Jelicic, V., Magno, M., Brunelli, D., Bilas, V. and Benini, L.: Analytic comparison of wake-up receivers for WSNs and benefits over the wake-on radio scheme, *Proc. 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks* (*PM2HW2N '12*), pp.99–106 (2012).

Akihiro Fujimoto received his B.E., M.I, and Ph.D. degrees from Osaka University in 2008, 2010, 2013, respectively. He was adopted as an assistant professor at Wakayama University in 2013. His research interest includes multimedia streaming system, peer-to-peer communication, and wireless networks. He is a member of IEEE, IEICE, and IPSJ.

Yukari Masui received her B.E. degree from Wakayama University in 2015. She is currently working with SCSK Corporation.

Takuya Yoshihiro received his B.E., M.I, and Ph.D. degrees from Kyoto University in 1998, 2000, 2003, respectively. He was an assistant professor at Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in graph theory, distributed algorithms, computer networks, wireless networks, medical applications, bioinformatics, etc. He is a member of IEEE, IEICE, and IPSJ.