PAPER

Tree Node Switching Algorithm for Minimum Energy Cost Aggregation Tree Problem in Wireless Sensor Networks

Hiroshi MATSUURA^{†a)}, *Member*

SUMMARY Data aggregation trees in wireless sensor networks (WSNs) are being used for gathering data for various purposes. Especially for the trees within buildings or civil structures, the total amount of energy consumption in a tree must be reduced to save energy. Therefore, the minimum energy-cost aggregation tree (MECAT) and MECAT with relay nodes (MECAT_RN) problems are being discussed to reduce energy consumption in data aggregation trees in WSNs. This paper proposes the tree node switching algorithm (TNSA) that improves on the previous algorithms for the MECAT and MECAT_RN problems in terms of energy efficiency. TNSA repeatedly switches nodes in a tree to reduce the number of packets sent in the tree. Packets are reduced by improving the accommodation efficiency of each packet, in which multiple sensor reports are accommodated. As a result of applying TNSA to MECATs and MECAT_RNs, energy consumption can be reduced significantly with a small burden.

key words: data gathering tree, data aggregation, MECAT, MECAT_RN

1. Introduction

There are many data aggregation tree applications in wireless sensor networks (WSNs) including forest fire detection, health monitoring in a hospital, and inventory management in a warehouse [1]. This is because data aggregation using a tree structure is considered to consume less energy than cluster or grid-based aggregation [2]. In a data aggregation tree, sensor reports from all the sensors are sent to the root node of the tree through the tree branches. Multiple sensor reports are capsuled in packets with an aggregation rate, q , which is defined as the maximum size of sensor reports aggregated in one packet. The maximum packet sizes are determined by individual wireless standards and are different from each other. In addition, even within the same wireless standard, the maximum packet size is dependent on the sensor application. Therefore, q is determined in accordance with the wireless standard and the sensor application.

The data aggregation tree lifetime, which is defined as the number of data gathering cycles before one of the sensors depletes its energy, is very important when the sensors are operated unattended for some applications such as weather forecasting [3], environment monitoring [4], and ecological studies [5]. However, these data aggregation trees are now being used in various buildings [6]–[8], civil structures [9], farms [10], and these sensors may have an AC power plug. In these cases, tree lifetimes do not have to be considered, but the total energy consumption on each tree

needs to be reduced.

When it comes to reducing power consumption in data aggregation trees, sleep scheduling is mainly applied [11], [12]. In sleep scheduling, each sensor is made dormant by using the duty-cycle scheme [13]. A dynamic power management in each sensor depending on various sleeping states has also been researched [14]. Sleep scheduling and power management, however, can be optimized even after a tree topology is determined, so they are independently considered from the minimum energy-cost aggregation tree (MECAT) and MECAT with relay nodes (MECAT_RN) problems.

The main aim of MECAT and MECAT_RN problems is also to reduce the power consumption in data aggregation trees, but their approach is different from those of sleep scheduling and power management mentioned above. That is, the MECAT/MECAT_RN problems have been defined as creating a data aggregation tree with the minimum number of data packets transmitted in the tree, and both have proved to be non-deterministic polynomial (NP) complete problems [15].

In a MECAT, the tree consists of sensors alone, and each sensor sends a report to the root node for each data gathering cycle. On the other hand, in a MECAT_RN, the tree includes relay nodes that do not send their own reports and that dedicate themselves to relaying the packets to their parent nodes in the tree. A sensor or a relay node in an aggregation tree is defined as a tree node.

Sparse random sampling (SRS) has recently been applied to WSN, and in each data gathering cycle, only randomly selected source nodes construct an aggregation tree and send their reports to the root node through the tree, so that the total energy consumption is reduced [16]. In MECAT/MECAT_RN, however, the tree structure is fixed and each sensor sends its report to the root. Another feature of MECAT/MECAT_RN is that the report size from each sensor may differ depending on the sensor, and this is different from the constant-size reports in trees in other papers [17]–[19]. This paper proposes the tree node switching algorithm (TNSA) for MECAT and MECAT_RN problems, and its contributions are summarized as follows.

First, this paper has applied tree node switching for the first time to approach the optimal solution for the MECAT/MECAT_RN problems, and this centralized algorithm, TNSA, has a light burden while significantly reducing energy consumption. The running time of the proposed TNSA switching process is $O(|V_T|^2 QHC_{max})$, where $|V_T|$ is

Manuscript received January 23, 2020.

Manuscript revised June 8, 2020.

Manuscript publicized August 3, 2020.

[†]The author is with NTT, Musashino-shi, 180–8585 Japan.

a) E-mail: hiroshi.matsuura.gt@hco.ntt.co.jp

DOI: 10.1587/transinf.2020EDP7010

the number of tree nodes in the tree, Q is the maximum number of neighboring tree nodes for a tree node, H is the maximum number of hop-count in the tree, and C_{max} is a parameter value for TNSA.

Secondly, this paper proposes a new tree creation algorithm for MECAT_RN problem, which can significantly reduce tree creation times of the previously proposed algorithms as well as their time complexities. If TNSA is applied to a MECAT_RN created by the proposed algorithm, energy consumption in the tree is smaller than those in other trees created with previously proposed algorithms while keeping much shorter tree creation times.

Thirdly, in TNSA, a tree node is switched from one subtree to another after finding the switch that has the largest packet reduction effect in the subtree. TNSA, however, allows a “zero-packet reduction switch” if there are no switches in the subtree that can reduce the number of packets. This zero-packet reduction switch gives a significant packet reduction effect at the end of the TNSA process.

The rest of the paper is organized as follows. Section 2 discusses related work, while Sect. 3 formulates the problem solved in this paper. Section 4 gives details on TNSA, and Sect. 5 gives a new tree creation algorithm applied to MECAT_RN problem. Section 6 discusses the simulation results, and Sect. 7 concludes the paper.

2. Related Works

Many data aggregation trees have been researched [18]–[23] because a data aggregation tree’s structure is advantageous in collecting data from every tree node to its root node. These algorithms, however, focus on the tree lifetime, rather than the total energy consumption in the tree.

Liu and Cao proved that for any aggregation rate q , if a shortest-path tree is used for gathering data, the total energy consumption in the tree will not be more than $\frac{5}{4}$ times the optimum solution [17]. However, they assumed that each tree node sends a report that has the same data size, which is not necessarily true in some applications.

Lin and Uster used a mixed-integer linear programming (MILP) model to create data-gathering clusters [24]. Even though they considered the total energy consumed by all the clusters, the topology for the multiple clusters is different from that of a data aggregation tree. In addition, the MILP problem cannot be solved in polynomial time, so the burden for the computation is large.

Kuo et al. defined the MECAT and MECAT_RN problems [15] and proved that every shortest path tree algorithm is a 2-approximation algorithm for the MECAT problem. For the MECAT_RN problem, they also applied the (3, 2)-light approximate shortest-path tree (LAST) algorithm [25] or an algorithm for the capacitated network design (CND) problem [26] for creating a MECAT_RN, which proved to be 7-approximation and 2λ -approximation algorithms, respectively. Here, the λ -approximation algorithm is assumed to be used for the CND problem. They, however, did not further optimize the algorithms for the MECAT and

MECAT_RN problems.

So far, tree nodes in a data aggregation tree have been switched for maximizing the tree lifetime [18], [19], [27], [28]. Generally, each switch is conducted to move nodes from a burdened subtree to an unburdened subtree, so that the tree node burdens are flattened throughout the tree. This type of tree node switching, however, has never been applied to MECAT/MECAT_RN problems.

3. Problem Formulation

This section formally defines the MECAT/MECAT_RN problem.

3.1 Network Model

In a WSN, sensor/relay nodes $v_1, \dots, v_N$ are assumed to be randomly deployed with the root node v_0 for monitoring an area. These nodes form a connective undirected graph, $G(V, E)$, where V is the set of nodes and E is the set of edges, which are neighbor relationships among nodes. If two nodes v_i and v_j are within a radio communication range, they have a neighbor relationship. Each node v_i , except v_0 and relay nodes, sends a report of size $s(v_i)$ to v_0 in each data-gathering cycle. A MECAT has no relay nodes, whereas a MECAT_RN has at least one.

A data aggregation tree constructed for $G(V, E)$ is defined as $T(V_T, E_T)$, where $V_T \subseteq V$ and $E_T \subseteq E$, and all the sensor nodes in $G(V, E)$ must be included in V_T . A tree node v_i can transmit a packet to a tree node v_j only if v_j is the parent of v_i in the tree. Each tree node consumes energy mainly in communication, and E_{tx} and E_{rx} are defined as the energy required to transmit and receive a packet in the tree, respectively. Both E_{tx} and E_{rx} include startup energy, which is used to activate the corresponding circuit in a sensor from the sleep status. Though E_{tx}/E_{rx} depends on the transmitting/receiving packet size, in this paper, they are assumed to have constant values in one $T(V_T, E_T)$ like in other papers [15], [18], [19], [27], [28]. This is because compared with the startup energy, the other energy required for transmitting or receiving a packet is much smaller when the packet size is small. For example, according to Wang et al. [29] when the transmitter startup time is 10^{-2} seconds, the startup energy is 10^{-4} J, whereas the energy required for transmitting a 100-bit packet is 10^{-6} J. Generally, in a WSN, the maximum packet size is small, such as 127 bytes for IEEE 802.15.4, thus the startup energy is the dominant factor to determine E_{tx} and E_{rx} values regardless of the packet size.

Some papers assume E_{tx} changes depending on the transmission distance, because the required power for the amplifier depends on the transmission distance [21], [30]. On the other hand, other papers assume E_{tx} has a constant value because sensor nodes are assumed to be allocated densely so that each sensor does not have to change amplifier power in accordance with the transmission distance. This paper adopts the latter assumption, assuming E_{tx} has a

Fig. 1 Data-aggregation tree example.

constant value regardless of its transmission distance.

Figure 1 shows an example of $T(V_T, E_T)$ having eight sensor nodes and one relay node v_2 , and q is set to 5. The left-hand number within the parentheses indicates $s(v_i)$, and the right-hand number indicates the total size of reports from the subtree rooted by v_i , which is defined as $s_T(v_i)$. An arrow between two tree nodes indicates an edge, $e \in E_T$, and a dotted line indicates an edge, $e \in E - E_T$. The number beside an arrow indicates the number of packets transmitted along the edge for each data-gathering cycle. For example, v_1 sends 3 packets because “ $s_T(v_1) = 11$ ” means the total size of reports from the subtree is 11, and each packet accommodates up to size-5 reports. A feature of a relay node v_i is that it does not send its own reports: $s(v_i) = 0$ as shown in the case of v_2 in Fig. 1.

3.2 Problem Statement

Under these assumptions, minimizing the energy consumption in $T(V_T, E_T)$ is represented by

$$\min_{T \in T(G)} \sum_{e \in E_T} (E_{tx} + E_{rx}) \left\lceil \frac{z(e)}{q} \right\rceil, \quad (1)$$

where $z(e)$ is the size of reports transmitted along e , and $\left\lceil \frac{z(e)}{q} \right\rceil$ is the number of packets along e . As is shown in Fig. 1, $z(e)$ is exactly the same as $s_T(v_i)$, where v_i is the source node of e . From this and because $(E_{tx} + E_{rx})$ has a constant value, (1) is equivalent to

$$\min_{T \in T(G)} \sum_{v_i \in V_T - \{v_0\}} \left\lceil \frac{s_T(v_i)}{q} \right\rceil. \quad (2)$$

From (2), the MECAT/MECAT_RN problem is equivalent to minimizing the total number of packets sent for one data-gathering cycle by the nodes in V_T .

In (2), it is assumed that all the nodes on $T(V_T, E_T)$ send their reports in the data gathering cycle without malfunctions. In addition, the energy consumption for creating $T(V_T, E_T)$ is not considered in (2), because the routing topology of $T(V_T, E_T)$ is assumed to be determined by its root node before data-gathering operation begins.

4. Proposed Algorithm: TNSA

This section details TNSA by explaining its function block, switching concept, pseudo code, a switching example, and

Fig. 2 TNSA function block.

analyses of its time complexity and approximation ratio.

4.1 TNSA Function Block

Figure 2 shows an implementation of TNSA in a root node, which determines the routes of a data aggregation tree in the WSN in a centralized manner. Within the root node, the neighbor info search function requests the nodes neighboring the root node to send their information such as their node ID and report size $s(v_i)$. After receiving the information, the neighboring nodes are registered to the sensor list by level, which is the minimum hop count from the root node to the node.

If all the level-1 nodes are registered to the sensor list, the pieces of information for level-2 nodes are taken by requesting the level-1 nodes to contact the neighboring level-2 nodes. As a result, the information for all the tree nodes in the WSN can be gained by the root node level by level, as shown in the sensor list.

The route-calculation function creates an initial tree using the information in the sensor list. Kuo et al. [15] proposed a shortest-path tree for a MECAT and an MST-based tree for a MECAT_RN, so it is supposed that either is created as an initial tree.

After the initial tree is created, the route-calculation function calculates $s_T(v_i)$ for each tree node by adding all the report sizes below v_i . After that, in the node-switching process, it switches nodes in the tree if the switches satisfy the conditions set with TNSA. Generally, a switch is allowed if it is considered to contribute to the packet reduction in the tree. After each switch, $s_T(v_i)$ values changed due to the switch are renewed by the $s_T(v_i)$ setting block in the route-calculation function.

This series of switches is continued until there is no switching effect gained or a switching limit set by the algorithm comes. After the series of switches, the final tree is determined, and the determined data aggregation tree is requested to be created in the real WSN through the routing-request function.

Fig. 3 Packet reduction effect by TNSA.

4.2 TNSA Switching Concept

Basically, TNSA repeatedly switches a tree node belonging to one subtree, $S(T, v_{top})$, to another. The subtree root node v_{top} is randomly selected for each switch. In each switch, the moved node is selected from $D(T, v_{top})$, which is the set of descendant nodes below v_{top} in $S(T, v_{top})$. TNSA tries to find the switching node that has the largest packet reduction effect in $D(T, v_{top})$.

Figure 3 shows a TNSA switching effect supposing $q = 5$. The changed values of $s_T(v_i)$ and the number of packets for an edge after a switch are circled. It is assumed that v_1 happens to be selected as v_{top} . In this case, the node $v_6 \in D(T, v_1)$ is moved from under v_3 to under v_4 . This is because, as a result of this switch, the number of packets in the tree is reduced by two as shown in Fig. 3. No other node in $D(T, v_1)$ can reduce the packets by two.

The notations used in this paper are summarized in Table 1. As a TNSA parameter, a non-negative integer parameter δ is set to determine the smallest number of packets reduced by one switch. If there is no switch to reduce the packets by δ or more in $D(T, v_{top})$, TNSA goes to another subtree to find a switch by which δ or more packets are reduced. If δ is set to 0, there may possibly be a switch that has no packet reduction effect by itself. Thus, we call this switch a “zero-packet reduction switch” and explain why zero-packet reduction switches effectively reduce the total number of packets at the end of TNSA.

4.3 TNSA Pseudo Codes

Algorithm 1 shows the main function for TNSA. The input to TNSA is $G(V, E)$ and its output is the created tree by TNSA. V_a indicates the node group from which v_{top} is selected, and V_b indicates the node group from which a moved node is selected. The high level idea of Algorithm 1 is that v_{top} is randomly selected as long as it has not been selected C_{max} times. After that, tree nodes in $D(T, v_{top})$ are set into V_b as long as they have not moved M_{max} times. Each tree node in V_b is evaluated by the Cal_Packet_R function, and the most effective node in $D(T, v_{top})$ is moved from below v_{top} to another subtree. After each move, each $s_T(v_i)$ value changed by the move is updated by the Update_Subtree function.

Table 1 Summary of notations.

Symbol	Definition
$G(V, E)$	Graph of nodes V and edges E
V_T	Set of tree nodes in T
E_T	Set of edges in T
$s(v_i)$	Report size sent from v_i
$s_T(v_i)$	Total size of reports from subtree rooted by v_i
v_{top}	Subtree root below which node is moved to another subtree
$c(v_i)$	Number of times that v_i has been selected as v_{top}
C_{max}	Upper limit of v_{top} selections for each node
$D(T, v_{top})$	Set of descendants below v_{top} in T
$m(v_d)$	Number of times that v_d has been moved
M_{max}	Upper limit of moves for each node
$S(T_{new}, v_{top}, v_p)$	Smallest subtree that includes v_{top} and v_p in T_{new}
$R(S(T_{new}, v_{top}, v_p))$	Total reduced number of packets in $S(T_{new}, v_{top}, v_p)$ after one switch
$r(T_{new}, v_i, v_j)$	Root node of $S(T_{new}, v_i, v_j)$
$B(T, [v_i, v_j])$	Branch in T between v_i and node just before v_j
δ	Non-negative TNSA parameter: Number of packets that must be reduced by one switch
$S(T, v_i)$	Subtree rooted at v_i in T

 Algorithm 1: TNSA(G)

```

1  Give an initial tree,  $T$ , in  $G$ ;
2  Calculate  $s_T(v_i)$ , and set  $m(v_i) \leftarrow 0$ ,  $c(v_i) \leftarrow 0$ , for each  $v_i \in V_T$ ;
3  for  $\forall v_i \in V_T$ 
4  if ( $v_i \neq v_0$  and  $v_i \neq \text{leaf}$  and  $c(v_i) \neq C_{max}$ ) do  $V_a \leftarrow V_a \cup v_i$ ;
5  while  $V_a \neq \emptyset$  do
6  Select  $v_i \in V_a$  as  $v_{top}$ ;  $V_a \leftarrow V_a - v_i$ ;  $c(v_i) + 1$ ;
7  for  $\forall v_d \in D(T, v_{top})$ 
8  if  $m(v_d) < M_{max}$  do  $V_b \leftarrow V_b \cup v_d$ 
9   $v_{m,d} \leftarrow \text{null}$ ;  $v_{n,p} \leftarrow \text{null}$ ;  $R_{max} \leftarrow 0$ ;
10 while  $V_b \neq \emptyset$  do
11 Take  $v_d$  from  $V_b$ 
12  $\{v_p, R(S(T_{new}, v_{top}, v_p))\} = \text{Cal\_Packet\_R}(v_d, v_{top}, T, G)$ ;
13 if ( $R(S(T_{new}, v_{top}, v_p)) \geq \delta$ ) then
14 if ( $v_{n,p} = \text{null}$  or  $R(S(T_{new}, v_{top}, v_p)) > R_{max}$ ) then
15  $v_{m,d} = v_d$ ;  $v_{n,p} = v_p$ ;  $R_{max} = R(S(T_{new}, v_{top}, v_p))$ 
16 if  $v_{m,d} \neq \text{null}$  then  $v_{m,d}$  is moved under  $v_{n,p}$ ;  $m(v_{m,d}) + 1$ ;
17 Old parent of  $v_{m,d}$  in  $T$  is set as  $v_{o,p}$ ;
18 Update_SubTree( $T_{new}, v_{o,p}, v_{n,p}$ );  $T = T_{new}$ ;
19 else continue;
20 return  $T$ ;
    
```

Algorithm 2 describes the Cal_Packet_R function. The total number of reduced packets after v_d is moved to be a child of v_i is calculated for a possible new parent v_i , and the most effective parent candidate is returned as v_p with its number of reduced packets R_{max} .

Figure 4 shows an example for the procedure of the Cal_Packet_R function. In this example, $v_d \in D(T, v_{top})$ has

Algorithm 2: Cal_Packet_R(v_d, v_{top}, T, G)

```

1  $v_p = \text{null}; R_{\max} = 0;$ 
2 for  $\forall v_i$  neighboring  $v_d$  in  $V_T$  do
3   if  $v_i \in S(T, v_{top})$  then
4     continue;
5   else
6     if ( $v_p = \text{null}$  or  $R(S(T_{\text{new}}, v_{top}, v_i)) > R_{\max}$ ) then
7        $v_p = v_i; R_{\max} = R(S(T_{\text{new}}, v_{top}, v_i));$ 
8 Return ( $v_p, R_{\max}$ )
```

Fig. 4 Example of Cal_Packet_R function.**Algorithm 3:** Update_Subtree($T_{\text{new}}, v_{o_p}, v_{n_p}$)

```

1 Find smallest subtree:  $S(T_{\text{new}}, v_{o_p}, v_{n_p})$  in  $T_{\text{new}}$ ;
2 for  $\forall v_i \in B(T_{\text{new}}, [v_{o_p}, r(T_{\text{new}}, v_{o_p}, v_{n_p})])$ 
    $\cup B(T_{\text{new}}, [v_{n_p}, r(T_{\text{new}}, v_{o_p}, v_{n_p})])$ 
3   do Update  $s_{T_{\text{new}}}(v_i)$ ;
4 Delete relay nodes which do not have any sensor nodes in  $T_{\text{new}}$ ;
5 for  $\forall v_i \in V_{T_{\text{new}}}$  if ( $v_i \neq v_0$  and  $v_i \neq \text{leaf}$  and  $c(v_i) < C_{\max}$ )
6   do  $V_a \leftarrow V_a \cup v_i;$ 
```

two edges unused in T : one is the edge to v_4 , and the other is the edge to v_p . It is supposed that the Cal_Packet_R function found that the move of v_d to be a child of v_p reduces packets more than the move to be a child of v_4 . Thus, in this case, the pair between v_p and $R(S(T_{\text{new}}, v_{top}, v_p))$ is returned.

The changes of $s_T(v_i)$ are restricted in $S(T_{\text{new}}, v_{top}, v_p)$, because the other nodes are not affected by the move of v_d . Even in $S(T_{\text{new}}, v_{top}, v_p)$, the changes of $s_T(v_i)$ are restricted on the branch, $B(T_{\text{new}}, [v_{top}, v_2])$, and the branch, $B(T_{\text{new}}, [v_p, v_2])$, where v_2 is the root node for $S(T_{\text{new}}, v_{top}, v_p)$. The node belonging to $B(T_{\text{new}}, [v_{top}, v_2])$ is only v_{top} , and the nodes belonging to $B(T_{\text{new}}, [v_p, v_2])$ are v_p and v_6 . Therefore,

$$\begin{aligned}
R(S(T_{\text{new}}, v_{top}, v_p)) = & \{ \lceil s_T(v_{top})/q \rceil + \lceil s_T(v_p)/q \rceil + \lceil s_T(v_6)/q \rceil \} - \\
& \{ \lceil s_{T_{\text{new}}}(v_{top})/q \rceil + \lceil s_{T_{\text{new}}}(v_p)/q \rceil + \lceil s_{T_{\text{new}}}(v_6)/q \rceil \}.
\end{aligned}$$

Algorithm 3 describes the Update_Subtree function that updates the values of $s_{T_{\text{new}}}(v_i)$ in $S(T_{\text{new}}, v_{o_p}, v_{n_p})$ after each switch. In addition, in the case of MECAT_RN, unnecessary relay nodes are deleted as a result of a switch. After that, nodes in V_a are renewed.

Figure 5 shows an example where this relay node deletion is applied in Algorithm 3. In this example, after v_d

Fig. 5 Example of relay node deletion after switch in MECAT_RN. **Fig. 6** Example of TNSA applied to MECAT_RN.

is moved to be a child of v_p , no sensor node is relayed by v_{top} , so v_{top} is deleted. After being deleted from the tree, the deleted relay node will not be used in the TNSA process again.

4.4 TNSA Application Example

This subsection gives an application example of TNSA. Figure 6(a)–(d) show a transition of the initial MECAT_RN shown in Fig. 1, when TNSA is applied. The transition of the tree from (a) to (b) shows the same switch conducted in Fig. 3. The parameter δ is set to 0 in this example, which means that a zero-packet reduction switch is allowed if there is no switch to reduce the number of packets.

In (b), v_2 is assumed to be selected as v_{top} . In this case, the node $v_4 \in D(T, v_2)$ is moved from under v_2 to under v_1 even though this move does not reduce packets as shown in (c). This is because any other move from $D(T, v_2)$ increases the number of packets in the tree.

After the move of v_4 , no child nodes are below v_2 , which is a relay node, as shown in (c). In this case, v_2 is deleted as shown in Algorithm 3. After the deletion of v_2 , the tree is as shown in (d). Here, all the non-leaf nodes except v_0 (namely, v_1, v_3 , and v_4) are set into V_a in Algorithm 3.

Fig. 7 F_k^3 network topology.

Thus, these three nodes are selected as v_{top} one by one in Algorithm 1.

However, below these tree nodes, no nodes can be moved because each move will increase the number of packets in the tree. Therefore, TNSA ends at line 20 in Algorithm 1, because there is no node left in V_d after v_1 , v_3 , and v_4 are removed from V_d . Thus, the final output tree is the one in (d).

4.5 Time Complexity for TNSA

In this subsection, the time complexity of TNSA is analyzed by Theorem 1.

Theorem 1 The running time of the TNSA switching process is $O(|V_T|^2 QHC_{max})$.

Proof: In each switch, for every $v_i \in D(T, v_{top})$, Cal_Packet_R function is called, and each Cal_Packet_R function takes $O(QH)$ time. This is because there are at most Q neighbors for v_d at line 2 in Algorithm 2, and for each neighbor v_i , $R(S(T_{new}, v_{top}, v_i))$ is calculated. Each calculation for $R(S(T_{new}, v_{top}, v_i))$ takes $O(H)$ time, because $s_T(v_i)$ values are changed on just two branches in $S(T_{new}, v_{top}, v_i)$.

Here, $|D(T, v_{top})| = O(|V_T|)$, so each switch takes $O(|V_T|QH)$ time. The number of times each node is selected as v_{top} is at most C_{max} , so in total v_{top} is selected $O(|V_T|C_{max})$ times. Therefore, the time complexity of TNSA is $O(|V_T|^2 QHC_{max})$. ■

Here, C_{max} , which is a positive integer parameter in TNSA, is usually set to 10 or less, so it is not a big factor for the time complexity, and H is the hop-count in T and thus generally much smaller than $|V_T|$.

4.6 Lower Bound of TNSA Integrity Ratio

In this subsection, a lower bound of the integrity ratio: $\frac{T_{TNSA}}{T_{OPT}} = 1 + \Omega\left(\frac{1}{|V_T|}\right)$ is proved. Here, T_{OPT} is the number of packets required for a data-gathering cycle in the optimal tree and T_{TNSA} is the number of packets in the tree created with TNSA. For this purpose, a WSN, F_k^3 , shown in Fig. 7 is used. In Fig. 7, v_{i-j} indicates a node that is i hops away from v_0 and the j th node from the leftmost node. In F_k^3 ,

Fig. 8 Number of packets in TNSA tree compared with optimal tree.

there are $k + 1$ hops from v_0 to $v_{k+1,1}/v_{k+1,2}$, and v_{i-j} has neighbor relationships with v_{i-1-j} and v_{i+1-j} if $1 < i < k$. In addition, $v_{k+1,1}$ has neighbor relationships with $v_{k,1}$ and $v_{k,2}$, and $v_{k+1,2}$ has neighbor relationships with $v_{k,2}$ and $v_{k,3}$.

In addition, the effect of zero-packet reduction switches is also discussed by explaining how zero-packet reduction switches can prevent TNSA from outputting the worst case tree in F_k^3 .

Theorem 2 A lower bound, $\frac{T_{TNSA}}{T_{OPT}} = 1 + \Omega\left(\frac{1}{|V_T|}\right)$ exists as the integrity ratio of TNSA.

Proof: Consider the trees in F_k^3 in Fig. 8. Here, q is set to 5, and δ is set to 1, which does not allow any zero-packet reduction switches. In addition, the value, $s(v_{i-j}) = 5$, is assumed for each v_{i-j} ($1 \leq i \leq k - 1$).

It is supposed that the tree shown in (a) is an initial tree given at line 1 in Algorithm 1. In this case, TNSA cannot move $v_{k+1,1}$ from under $v_{k,2}$ to under $v_{k,1}$ or move $v_{k+1,2}$ from under $v_{k,2}$ to $v_{k,3}$. This is because neither move reduces the packets in the tree. The optimal tree has k fewer packets than tree (a). This is because, each edge on the branch between v_0 and $v_{k,2}$ in the optimal tree has one packet fewer than that in tree (a) as shown in Fig. 8.

Therefore, (3) is gained.

$$T_{TNSA} - T_{OPT} = k. \quad (3)$$

In addition, from (a), T_{TNSA} is represented by (4).

$$\begin{aligned} T_{TNSA} &= 3(2 + 3 + \cdots + k + (k + 1)) + 2 \\ &= 3 * \frac{k}{2} * ((k + 1) + 2) + 2. \end{aligned} \quad (4)$$

The k value is $(|V_T| - 3)/3$ from the tree structure shown in Fig. 8 (a), thus from Eqs. (3) and (4), (5) is gained.

$$\begin{aligned} \frac{T_{TNSA}}{T_{OPT}} &= \frac{\frac{3k(k+3)}{2} + 2}{\frac{3k(k+3)}{2} + 2 - k} \\ &= 1 + \frac{k}{\frac{3k(k+3)}{2} + 2 - k} \end{aligned}$$

$$= 1 + \Omega\left(\frac{1}{|V_T|}\right). \quad (5)$$

Though Fig. 8 shows a MECAT, each sensor node v_{i-j_-} ($1 \leq i < k-1$) in the tree can be replaced with a relay node. Thus, a MECAT_RN has the same lower bound in F_k^3 .

So far it is proven that $\frac{T_{TNSA}}{T_{OPT}} \geq 1 + c\left(\frac{1}{|V_T|}\right)$ in F_k^3 for an infinite number of $|V_T|$, where c is a constant. This is equivalent that TNSA has the integrity ratio: $\frac{T_{TNSA}}{T_{OPT}} = 1 + \Omega\left(\frac{1}{|V_T|}\right)$. ■

In this way, TNSA may not be able to create the optimal tree in F_k^3 . If zero-packet reduction switches are allowed by setting δ to 0, however, the probability of falling into the worst tree shown in Fig. 8 (a) is much smaller. Theorem 3 demonstrates this probability.

Theorem 3 The probability of TNSA for not creating the optimal tree is less than $\left(\frac{1}{2}\right)^{M_{max}/2}$ in F_k^3 , if zero-packet reduction switches are allowed to be used.

Proof: Figure 9 shows the effect of zero-packet reduction switches. For simplicity, k is set to 2 in this example, but this proof can be applied to any k .

In Fig. 9 (a), the initial tree for a MECAT is shown and if $\delta = 0$, a zero-packet reduction switch is conducted under $v_{2,2}$. Two zero-packet reduction switch candidates are under $v_{2,2}$: one is to move $v_{3,1}$ to be a child of $v_{2,1}$, and the other is to move $v_{3,2}$ to be a child of $v_{2,3}$. Either switch is conducted with 50% probability. In this example, it is assumed that $v_{3,2}$ is moved to be a child of $v_{2,3}$ as shown in (b).

In the tree shown in (b), even if $v_{1,1}$ is selected as v_{top} , there is no node to be moved below v_{top} . Thus, there is a 50% probability to select $v_{1,3}$ or $v_{2,3}$ as v_{top} , and there is another 50% probability to select $v_{1,2}$ or $v_{2,2}$ as v_{top} . If $v_{1,2}$ or $v_{2,2}$ is chosen as v_{top} , $v_{3,1}$ is moved to be a child of $v_{2,1}$, because by this switch, the number of packets is reduced by two as shown in (d). Tree (d) is the final output of TNSA, because either $v_{3,1}$ or $v_{3,2}$ does not go back under $v_{2,2}$ due to the packet increase by the switch.

Fig. 9 Zero-packet reduction switch effect for avoiding worst tree.

If $v_{1,3}$ or $v_{2,3}$ is selected as v_{top} in (b), $v_{3,2}$ is moved back under $v_{2,2}$, and as a result, tree (c), which is exactly the same as tree (a), is recreated. In tree (c), $v_{3,1}$ is moved to be a child of $v_{2,1}$ or $v_{3,2}$ is moved to be a child of $v_{2,2}$ with 50% probability each as explained above. Therefore, it is possible that $v_{3,2}$ is repeatedly moved between under $v_{2,2}$ and under $v_{2,3}$. If this switching repetition between tree (c) and tree (b) continues until the maximum moving time, M_{max} for $v_{3,2}$ is reached, the final output tree for TNSA can be tree (c) or (b) but not the optimal tree (d)

However, the probability TNSA outputs a tree other than the optimal tree (d) is less than $\left(\frac{1}{2}\right)^{M_{max}/2}$. This is because $v_{2,3}$ has to move up to M_{max} times, and for each even-time move, $v_{1,3}$ or $v_{2,3}$ must be selected as v_{top} instead of $v_{1,2}$ or $v_{2,2}$, and this probability is 50%. In addition, in the middle of the TNSA process, $v_{3,1}$ may be chosen and moved to be a child of $v_{2,1}$ in tree (c). In this case, $v_{2,1}$ has to be moved back to under $v_{2,2}$, which has 50% probability. Therefore, the probability TNSA outputs a tree other than the optimal tree is less than $\left(\frac{1}{2}\right)^{M_{max}/2}$. ■

From Theorem 3, it is clear that even though TNSA with $\delta = 0$ has the same lower bound, $\frac{T_{TNSA}}{T_{OPT}} = 1 + \Omega\left(\frac{1}{|V_T|}\right)$, it has a much higher probability to have the optimal tree in F_k^3 . Even in other WSNs rather than F_k^3 , the same type of energy reduction effect is gained by zero-packet reduction switches, which is demonstrated in the evaluation section.

5. New Tree Creation Algorithm for MECAT_RN

In this section, a tree creation algorithm for the MECAT_RN problem is proposed to replace previous algorithms that have long tree creation times.

5.1 Previous MECAT_RN Algorithms and Their Problem

For the MECAT_RN problem, the (3, 2)-LAST algorithm or an algorithm for CND problem (CND algorithm) was applied for creating a MECAT_RN [15]. Either one of the previous two algorithms is based on a minimum spanning tree (MST), so at first it creates a Steiner tree with an MST-based algorithm [31].

Figure 10 shows how an MST-based Steiner tree is created in a WSN shown in (a). First, a logical network with sensor nodes alone is created in (b), in which each edge has the cost of the shortest hop-count including relay nodes between the two sensor nodes. In (c), Prim's algorithm [32]

Fig. 10 Process of creating an MST in logical WSN.

Algorithm 4: Proposed_MECAT_RN(G)

```

1 Each neighboring node  $v_i$  of  $v_0$  is entered into  $V_c$  with path-cost
  from  $v_0$  to  $v_i$  and its parent tree node ( $v_0$ ).
2 while (there is a sensor node unlisted to  $T$ ) do
3   Select  $v_j$  that has smallest path-cost in  $V_c$ , and  $v_j$  is listed to  $T$ 
   as new tree node.
4   Each neighboring node  $v_i$  of  $v_j$  is entered into  $V_c$  with path-cost
   from  $v_0$  to  $v_i$  and its parent tree node ( $v_j$ ). If  $v_i$  is already in  $V_c$ ,
   it is replaced only when new path-cost( $v_i$ ) is smaller.

```

is run to create an MST in the logical network. After the MST is created, the (3, 2)-LAST algorithm replaces each MST path with the shortest path to v_0 , if the MST path-cost is more than three times larger than the shortest path-cost. In the CND algorithm, each MST path from a tree node or subtree to v_0 is replaced with the shortest path if the report size from the node or subtree is equal to or larger than $q/2$.

In either algorithm, the dominant process to determine the time complexity is the process of creating a logical network shown in Fig. 10(b), because from each sensor node in (a), the shortest hop-counts to the other sensor nodes should be calculated. If Dijkstra's algorithm [33] is used for the calculation, the time complexity of each algorithm is $O(SNQ \log N)$, in which S is the number of sensor nodes and N is the total number of nodes in G including the unused relay nodes for the tree. This is because the time complexity for Dijkstra's algorithm is $O(NQ \log N)$ and it has to be run at least S times in each algorithm.

5.2 Proposed Algorithm for Creating MECAT_RN

The concept of the proposed MECAT_RN-creation algorithm is to seek the tree that has the smallest path-cost from v_0 to each sensor tree node. A path-cost is the sum of node-costs on the path. Each node has its node-cost, but a relay node has a larger node-cost than a sensor node. This is because relay nodes do not have to be used for a tree, while all sensor nodes should be listed in a tree. Although the proposed algorithm is similar to Dijkstra's algorithm, it does not use link-costs on the links and uses node-costs to determine a path-cost.

Algorithm 4 describes the flow of the proposed algorithm. At lines 1 and 4, a path-cost is determined by

$$\text{pathcost}(v_i) = \text{pathcost}(v_j) + \text{nodecost}(v_i), \quad (6)$$

where $\text{pathcost}(v_0) = 0$. A node-cost is determined by

$$\text{nodecost}(v_i) = 1 + \text{weight}, \quad (7)$$

where $\text{weight} = 0$ if v_i is a sensor node. If v_i is a relay node, the value of weight is determined by

$$\text{weight} = a * q, \quad (8)$$

where a is a non-negative value parameter.

The weight should be set proportional to q . This is because if q is large, effective data aggregation with a high accommodation rate in each packet is more important than the

minimum hop-count to v_0 from each tree node. If the weight is set to a large value, the number of used relay nodes on a created tree will decrease, so that a larger data-aggregation effect is expected at the cost of more hop-count from each tree node to v_0 . In contrast, if the weight is set to 0, a minimum hop-count tree is created from each tree node to v_0 and is ideal for data-non-aggregation trees ($q = 1$). The a in Formula (8) was set to 0.04 in the next evaluation section because it had a relatively larger energy reduction effect than other values.

The time complexity of the proposed algorithm is $O(NQ \log N)$, which is the same as that of the Dijkstra's algorithm. Thus, the proposed algorithm has much smaller time complexity than the previous algorithms for MECAT_RN problems. After creating a MECAT_RN by the proposed algorithm, TNSA can be applied to further reduce the energy consumption in the tree.

6. TNSA Evaluation

In this section, TNSA is applied to MECATs created with a shortest path tree creation algorithm, and the burden of TNSA and the energy reduction effects are evaluated. TNSA is also applied to MECAT_RNs created with the proposed algorithm, and the energy consumptions on the created MECAT_RNs are compared with those created with the previous algorithms. The burden of the proposed MECAT_RN creation algorithm is also evaluated.

6.1 Simulation Environment

In a HP ProLiant ML110 server [34], a computer simulation environment was created using JBoss [35], which is a Java-based development platform, and all the algorithms were implemented and run on the JBoss platform. In the simulation, sensor/relay nodes were randomly placed with the minimum distance between two nodes set to 10 m, on a square area: 200×200 m, 300×300 m, 400×400 m, or 600×600 m. The number of nodes was varied from 250 to 2000 on these square areas. For the MECAT environment, sensor nodes alone were placed to create a WSN on a square area, but for the MECAT_RN environment, relay nodes were placed in addition to sensor nodes on a square area.

After the creation of a WSN, the tree root node v_0 was determined from the central position of the square area. The maximum transmission range of all the nodes was set to 25 m. If two nodes v_i and v_j were within the radio communication range, they had a neighbor relationship. A random integer between 1 and 5 was allocated as $s(v_i)$ for each sensor tree node v_i in the tree. After the creation of an initial tree rooted by v_0 using a previous algorithm or the proposed algorithm in the MECAT/MECAT_RN environment, TNSA was then applied to the tree.

6.2 MECAT Evaluation

In this subsection, the effects and burden of TNSA in

Table 2 WSNs used for evaluation.

	Square	Number of sensors	Average neighbors
WSN_1	200 × 200 m	250	9.82
WSN_2	300 × 300 m	500	11.92
WSN_3	400 × 400 m	1000	11.33

Fig. 11 TNSA energy reduction effect on MECATs ($5 \leq q \leq 50$).

MECATs are evaluated. First, a shortest-path tree was created by taking a breadth first traversal [28], because it is proven that any shortest path tree creation algorithm is a 2-approximation algorithm for the MECAT problem [15], and there is no alternative algorithm proposed. Table 2 shows the WSNs used for this evaluation. In each WSN, an initial shortest-path tree was created, and TNSA with zero-packet reduction switches ($\delta = 0$) or without them ($\delta = 1$) was applied to the initial tree. The TNSA parameters C_{max} , and M_{max} were both set to 10.

Figure 11 shows the TNSA energy reduction effects in the three WSNs. In the figure, in each WSN, the total energy consumptions for one data gathering cycle are compared between the initial shortest-path tree and the trees to which a TNSA switching process is applied. The aggregation rate, q , is changed in the range of $5 \leq q \leq 50$ with intervals of 5.

As a result, the trees to which TNSA without zero-packet reduction switches is applied have 4.29% lower average energy than their initial shortest path trees. On the other hand, the trees to which TNSA with zero-packet reduction switches is applied have 6.76% lower average energy than their initial shortest path trees.

In terms of the effects of the values of q , in WSN_1, which is the smallest WSN, TNSA effect peaks within the range of $q < 30$. WSN_3, however, has 1000 nodes, and the TNSA effect peaks with a larger q such as $40 \leq q$. It is deduced that in WSN_3, the number of switching patterns to approach the optimal energy solution increases as q nears 40. Therefore, the initial tree is far from the optimal tree and TNSA especially with zero-packet reduction switches is beneficial for approaching the optimal tree.

Figure 12 shows the average processing times for the creation of the shortest-path trees, and the TNSA switching processes with and without zero-packet reduction switches in each WSN. From Fig. 12, except for the process of the zero-packet reduction switches in WSN_3, every switching process has a processing time shorter than 2.3 seconds, and it is much shorter than its initial shortest path tree creation

Fig. 12 TNSA processing times for each WSN.**Table 3** Number of switches in TNSA.

	WSN_1	WSN_2	WSN_3
Without zero-packet reduction switches	13.8	29.5	63.5
With zero-packet reduction switches	1196.1	2158.3	4548.5

times if zero-packet reduction switches are not allowed. Therefore, the burden for a TNSA switching process is generally very light.

If zero-packet reduction switches are allowed, many switches are conducted and TNSA may continue until some nodes reach the M_{max} -time move limit or some nodes are selected as v_{top} by C_{max} times. Therefore, if there are many nodes in a WSN, as is the case for WSN_3, the burden for the switching process becomes larger. On the other hand, if zero-packet reduction switches are not allowed, the TNSA switching process ends when there is no switch that can reduce the number of packets in the tree, thus the total number of switches is reduced significantly. Table 3 indicates the average number of switches conducted by a TNSA switching process in each WSN. As is shown in the table, there is a distinctive difference between TNSA with and without zero-packet reduction switches in each WSN, especially in WSN_3.

The parameter setting for C_{max} and M_{max} sometimes effectively reduces the burden of a TNSA switching process without reducing the energy reduction effects. Figure 13 shows the energy reduction effects and processing times for the TNSA switching process with zero-packet reduction switches. The C_{max} and M_{max} values are changed between 2 and 20, but when one parameter value is changed, the other value is fixed to 10. The q value is fixed to 30, and WSN_3 is used.

As shown in Fig. 13, a small value such as 6 for C_{max} and M_{max} is clearly sufficient to obtain a similar energy reduction effect with other values of C_{max} and M_{max} . In addition, if one parameter is set to 6, it has a much shorter processing time, such as 2–3 seconds, than if both parameters are set to 10. This is deduced that because a large WSN such as WSN_3 contains many different switching nodes to

Fig. 13 Effects of C_{max} and M_{max} values on energy reduction and burden.

Table 4 WSNs used for MECAT_RN evaluation.

	Square	Num. of sensors	Num. of relay nodes	Average neighbors
WSN_4	400 × 400 m	200	800	11.49
WSN_5	600 × 600 m	400	1600	10.63

reduce the total energy, so small values for C_{max} and M_{max} are sufficient to gain a near-optimal tree. Therefore, setting C_{max} or M_{max} to a value smaller than 10 effectively shortens the TNSA processing time while maintaining the energy efficiency.

6.3 MECAT_RN Evaluation

In this subsection, the effects and burden of the proposed tree creation algorithm and TNSA in MECAT_RNs are evaluated. Table 4 shows the WSNs used for this evaluation. In each WSN, an MECAT_RN creation algorithm, namely the (3,2)-LAST algorithm, the CND algorithm, or the proposed algorithm, created MECAT_RN, which has to include all the sensor nodes in the WSN. Each relay node, however, does not have to be included in an MECAT_RN and is used only when necessary in each algorithm. In reality, many relay nodes were not included in each created MECAT_RN. An MECAT_RN created by the proposed algorithm had an option to apply TNSA, so the TNSA effect on the MECAT_RN problem is evaluated.

Figure 14 shows the energy reduction effects and processing times of each algorithm in WSN_4 and WSN_5. In this evaluation, TNSA's zero-packet reduction was allowed and both C_{max} and M_{max} were set to 10. The total energy consumption in a tree created by the CND algorithm or the proposed algorithm with or without TNSA was compared with that in a tree created with the (3,2)-LAST algorithm, and their energy reduction rates are shown.

As shown in (a), in the relatively small network, WSN_4, the proposed algorithm alone cannot reduce the energy if q is set to a large value, but with TNSA

Fig. 14 Energy reduction rates and processing times in MECAT_RNs.

(proposed + TNSA), it has much higher energy reduction rates. On average, the proposed algorithm reduces the energy by 2.16%. With TNSA, however, its average reduction rate is 14.23%, whereas the CND algorithm's average reduction rate is 12.64%.

The proposed algorithm clearly has a better energy reduction rate when q is set to a small value. It is deduced that the proposed algorithm emphasizes the smaller hop-counts from v_0 rather than the total tree cost as its Dijkstra-based nature, and if the aggregation rate is small, it is more important to reach v_0 in a short hop-count rather than a packet accommodation rate. In (a), the proposed algorithm with TNSA can create trees faster than the (3,2)-LAST and CND algorithms, thus it not only can reduce the energy in WSN_4 but also has a faster algorithm speed.

From (b), a clearer advantage to use the proposed algorithm with TNSA is shown. The proposed algorithm's average energy reduction rate is 16.99%. In addition, with TNSA, it further increases the energy reduction rate to 24.95%, whereas the CND algorithm has a 22.23% average reduction rate. In (b), the proposed algorithm's most noticeable advantage is that its average processing time is more than four and six times shorter than those of the CND and (3,2)-LAST algorithms, respectively. Even combined with TNSA, its average processing time is more than three and four times shorter than those of the CND and (3,2)-LAST algorithms.

This result follows the time complexity gap between the proposed algorithm and (3,2)-LAST/CND algorithm discussed in Sect. 5. The proposed algorithm is theoretically more than S times faster than these previous algorithms, because each previous algorithm runs Dijkstra's algorithm at least S times, which is the number of sensor nodes. From this fact, the more sensor nodes, the wider the processing time gap between the proposed algorithm and the others. Because a tree-creation algorithm is run while the data-gathering operation is stopped, a long algorithm processing

Fig. 15 TNSA energy reduction switching effect in WSN_4 ($5 \leq q \leq 50$).

time will lengthen the operation downtime.

The energy reduction effects and processing times of TNSA with zero-packet reduction switches are shown in Fig. 15. TNSA is applied to a tree created with the proposed MECAT_RN creation algorithm. As a result, the trees to which TNSA without zero-packet reduction switches is applied have a 10.41% lower average energy than their initial trees. On the other hand, the trees to which TNSA with zero-packet reduction switches is applied have a 12.09% lower average energy than their initial trees. These energy reduction rates with TNSA are much higher than those when TNSA is applied to MECATs.

This larger effect of TNSA on MECAT_RNs than on MECATs is thought to come from a MECAT_RN generally containing fewer neighboring tree nodes. For example, the average number of tree nodes in WSN_4 is 324.8, which is much fewer than the case in of WSN_3, which has 1000 tree nodes, even though they are both the same size, 400×400 m. In this case, a small number of switches of the tree nodes may largely improve the energy reduction effect.

On the other hand, the effect of the zero-packet reduction switches is smaller than the case in a MECAT. It is deduced that if a tree has many neighboring tree nodes such as in the case of MECAT, the energy reduction effect by the zero-packet reduction switches is more likely to increase, because the switching patterns to reach the optimal tree are increased by the large number of neighboring tree nodes.

Figure 15 (b) shows the processing times for tree creations with TNSA. Their average processing time is less than 3 seconds. In addition, the average processing time for the zero-packet reduction switches is about 0.3 seconds, which is almost negligible, because a MECAT_RN has far fewer neighboring tree nodes than a MECAT, and this largely reduces the total number of switches.

7. Conclusions

The proposed tree node switching algorithm (TNSA) applies a series of tree node switches to a data aggregation tree to approach the optimal trees for the MECAT/MECAT_RN problems. In this paper, by analyzing its time complexity, integrity ratio, and the effect of zero-packet reduction switches, it was proven that TNSA can come closer to the

optimal energy consumption than existing algorithms while having a light burden.

For the MECAT_RN problem, a new tree creation algorithm was proposed and demonstrated to have a much smaller time complexity and algorithm processing times compared with previously proposed algorithms.

In the evaluation section, it was demonstrated that TNSA can significantly reduce the energy consumptions in MECATs and MECAT_RNs within a short processing time. The parameter values C_{max} and M_{max} were also analyzed, and it was shown that small values of them do not degrade the energy reduction effect much in a large WSN and contribute to a lighter burden in the TNSA switching process.

A quantitative analysis for the zero-packet reduction switches was also conducted in the evaluation section. From the analysis, this paper deduces that the zero-packet reduction switches have a larger energy reduction effect in a MECAT than in a MECAT_RN, because in a MECAT each tree node generally has more neighbor tree nodes.

References

- [1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A Survey," *Computer Networks* (Elsevier) Journal, vol.38, no.4, pp.393–422, March 2002.
- [2] J. NituElza and A. Jyotsna, "A survey on energy efficient tree-based data aggregation techniques in wireless sensor networks," *IEEE ICIRCA*, 2018.
- [3] C.-J. Liu, H.-C. Lee, J. Yang, J.-T. Huang, Y.-M. Fang, B.-J. Lee, and C.-T. King, "Development of a long-lived, real-time automatic weather station based on WSN," *ACM SenSys*, pp.401–402, Nov. 2008.
- [4] J. Yang and X. Li, "Design and implementation of low-power wireless sensor networks for environmental monitoring," *IEEE WCNIS*, June 2010.
- [5] E. Stattner, N. Vidot, P. Hunel, and M. Collard, "Wireless sensor network for habitat monitoring: A counting heuristic," *IEEE LCN Workshops*, Oct. 2012.
- [6] A. Orestis, A. Dimitrios, D. Dimitrios, and C. Ioannis, "Smart energy monitoring and management in large multi-office building Environments," *ACM PCI*, pp.219–226, Sept. 2013.
- [7] F. Jia, Y. Sun, and J. Yu, "A home monitoring system for elderly people based on MEMS sensors and wireless networks," *IEEE Sensors*, 2013.
- [8] Y. Zatout, E. Campo, and J.-F. Llibre, "Toward hybrid WSN architecture for monitoring people at home," *ACM MEDES*, pp.308–314, Oct. 2009.
- [9] S.A. Rajba, T. Rajba, and P. Raif, "The Implementation of wireless sensor networks for environmental monitoring of water facilities," *IEEE IDAACS*, Sept. 2015.
- [10] M. Murad, K.M. Yahya, and G.M. Hassan, "Web based poultry farm monitoring system using wireless sensor network," *ACM FIT*, pp.1–5, Dec. 2009.
- [11] R. Chauha and V. Gupta, "Energy efficient sleep scheduled clustering & spanning tree based data aggregation in wireless sensor network," *IEEE RAIT*, 2012.
- [12] Q. Chen, H. Gao, S. Cheng, J. Li, and Z. Cai, "Distributed non-structure based data aggregation for duty-cycle wireless sensor networks," *IEEE INFOCOM*, 2017.
- [13] W. Feng and L. Jiangchuan, "On reliable broadcast in low duty-cycle wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol.11, no.5, pp.767–779, May 2012.
- [14] A. Sinha and A. Chandrakasan, "Dynamic power management in

- wireless sensor networks,” *IEEE Des. Test. Comput.*, vol.18, no.2, pp.62–74, March/April 2001.
- [15] T.-W. Kuo, K.C.-J. Lin, and M.-J. Tsai, “On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms,” *IEEE Trans. Comput.*, vol.65, no.10, pp.903–916, Oct. 2016.
- [16] X. Yu and S.J. Baek, “Energy-efficient collection of sparse data in wireless sensor networks using sparse random matrices,” *ACM Trans. Sen. Netw. (TOSN)*, vol.13, no.3, pp.1–36, Aug. 2017.
- [17] C. Liu and G. Cao, “Distributed Monitoring and Aggregation in Wireless Sensor Networks,” *IEEE INFOCOM*, 2010.
- [18] S.K.A. Imon, A. Khan, M.D. Francesco, and S.K. Das, “Energy-efficient randomized switching for maximizing lifetime in tree-based wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol.23, no.5, pp.1401–1415, Oct. 2015.
- [19] H. Matsuura, “Maximizing Lifetime of Data-Gathering Sensor Trees in Wireless Sensor Networks,” *IEICE Trans. Commun.*, vol.E102-B, no.12, pp.2205–2217, Dec. 2019.
- [20] Y. Wu, Z. Mao, S. Fahmy, and N.B. Shroff, “Constructing maximum-lifetime data-gathering forests in sensor networks,” *IEEE/ACM Trans. Netw.*, vol.18, no.5, pp.1571–1584, Oct. 2010.
- [21] H. Inanlou, K.S. Shourmasti, H. Marjani, and N.A. Rezaei, “FFDA: A tree based energy aware data aggregation protocol in wireless sensor networks,” *IEEE WINSYS*, 2010.
- [22] J. Zhang, Q. Wu, F. Ren, T. He, and C. Lin, “Effective data aggregation supported by dynamic routing in wireless sensor networks,” *IEEE ICC*, 2010.
- [23] H. Wang and H. Chan, “On the construction of data aggregation tree with maximized lifetime in wireless sensor networks,” *IEEE WCSP*, Oct. 2019.
- [24] H. Lin and H. Üster, “Exact and Heuristic Algorithms for Data-Gathering Cluster-Based Wireless Sensor Network Design Problem,” *IEEE/ACM Trans. Netw.*, vol.22, no.3, pp.903–916, June 2014.
- [25] S. Khuller, B. Raghavachari, and N. Young, “Balancing minimum spanning trees and shortest-path trees,” *Algorithmica*, vol.14, pp.305–321, 1995.
- [26] R. Hassin, R. Ravi, and F.S. Salman, “Approximation algorithms for a capacitated network design problem,” *Algorithmica*, vol.38, pp.417–431, 2004.
- [27] C. Buragohain, D. Agrawal, and S. Suri, “Power aware routing for sensor databases,” *IEEE INFOCOM*, 2005.
- [28] J. Liang, J. Wang, J. Cao, J. Chen, and M. Lu, “An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks,” *IEEE INFOCOM*, 2010.
- [29] A. Wang, S. Cho, C. Sodini, and A. Chandrakasan, “Energy efficient modulation, and MAC for asymmetric RF microsensor systems,” *Proc. Int. Symp. Low Power Electron. Des.*, pp.106–111, 2001.
- [30] H. Matsuura, “New routing framework for RPL: Constructing power-efficient wireless sensor network,” *IEEE NOMS*, 2014.
- [31] L. Kou, G. Markowsky, and L. Berman, “A Fast Algorithm for Steiner Trees,” *Acta Informatica*, vol.15, pp.141–145, Springer-Verlag, 1981.
- [32] R.C. Prim, “Shortest Connection Networks and Some Generalization,” *Bell Sys. Tech. J.*, vol.1, pp.1389–1401, 1957.
- [33] E.W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, vol.1, pp.269–271, 1959.
- [34] HP ProLiant ML110 G4 Server - Overview, 9 July 2019. https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c01105516
- [35] Red Hat JBoss Enterprise Application Platform, 14 May 2020. <https://www.redhat.com/en/technologies/jboss-middleware/application-platform>

Hiroshi Matsuura received his B.E. degree from Kyushu University, Fukuoka, Japan, in 1989. He joined NTT Laboratories in 1989, after which he worked on network design and management. He joined NTT West Research and Development Center in 1999, where he was engaged in managing storage area networks and IP networks. He returned to NTT Laboratories in 2003, after which he tackled with GMPLS routing architecture, the Steiner tree problem, and k shortest path algorithms. Currently, his main interests are energy saving and maximizing the lifetime of data-gathering sensor trees in WSNs.