PAPER Special Section on Sensing, Wireless Networking, Data Collection, Analysis and Processing Technologies for Ambient Intelligence with Internet of Things

Simple and Complete Resynchronization for Wireless Sensor Networks

Hiromi YAGIRI[†], *Nonmember and* **Takeshi OKADOME**[†a)], *Member*

SUMMARY The methods proposed in this paper enable resynchronization when a synchronization deviation occurs in a sensor node without a beacon or an ack in a wireless sensor network under ultra-limited but stable resources such as the energy generated from tiny solar cell batteries. The method for a single-hop network is straightforward; when a receiver does not receive data, it is simply placed in recovery mode, in which the receiver sets its cycle length T_B to $(b \pm \gamma)T$, where b is non-negative integer, $0 < \gamma < 1$, and T is its cycle length in normal mode, and in which the receiver sets its active interval W_B to a value that satisfies $W_B \geq W + \gamma T$, where W is its active interval in normal mode. In contrast, a sender stays in normal mode. Resynchronization methods for linear multi-hop and tree-based multi-hop sensor networks are constructed using the method for a single-hop network. All the methods proposed here are *complete* because they are always able to resynchronize networks. The results of simulations based on the resynchronization methods are given and those of an experiment using actual sensor nodes with wireless modules are also presented, which show that the methods are feasible.

key words: *asynchronous neighbor discovery, resynchronization, ultra-limited resources, wireless sensor network.*

1. Introduction

In the Internet of Things (IoT) era, sensor networks gather ambient information from the environment using sensors. We believe that, in the future, some sensor networks will be comprised of microscopic sensors. One such example is the sensor network that senses the state of blood. This sensor network is composed of microscopic sensing devices drifting down with the flow of blood in blood vessels. Such microscopic devices are required to be able to operate using very limited resources, that is, using less physical devices and software under low energy consumption.

This study focuses on *resynchronization* of wireless sensor networks with a synchronous communication protocol under ultra-limited resources*.

The constraint of ultra-limited resources requires us to adopt a simple synchronous communication protocol, in which sensor nodes with their own counter clocks iterate between a long-term battery charging mode and a short-term data communication mode. That is, the sensor nodes charge in sleep mode, and then send or receive data in a short-term active mode for normal communication (see Fig. 1). If rich resources are available, a sensor node may take a sufficiently long active interval to receive data; this enables us to resynchronize the node with the network when a synchronization deviation occurs in the sensor node, where a *synchronization deviation* denotes a state in which, even if the active interval of a sender overlaps with that of a receiver, the overlapping interval is not sufficient for the receiver to receive data from the sender. Under ultra-limited resources, however, a sensor node may not always have a long enough active interval to resynchronize.

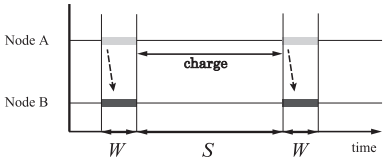


Fig. 1 Synchronized communication between two nodes with sleep mode in normal communication, where W denotes the active interval in which sensor nodes send or receive data, and S denotes the sleep interval. Node A is a data sender and node B a data receiver.

Previous studies have proposed practical algorithms for neighbor discovery that are based on slotted discovery schedules [3], [6]–[8]. For example, Kandhalu, Lakshmanan, and Rajkumar [6] presented a complete algorithm named U-Connect that always discovers a neighbor node for any relative phase offset. When a synchronization deviation occurs in a sensor node, the neighbor discovery algorithms can be used for recovery of synchronization.

Recovery of synchronization using the neighbor discovery algorithms, however, requires a beacon** between a sender and a receiver (or an acknowledgment (ack) from a receiver to a sender) in the normal communication mode if the sender and receiver use the minimum slot. The minimum slot is defined as the slot with the shortest length among those that allow sensor nodes to send data and/or receive data. This is because, when a synchronization deviation occurs, the sender may not detect the occurrence of the synchronization deviation without a beacon; consequently, it cannot switch to a slotted discovery schedule.

Even if the sensor nodes do not use a beacon or an ack,

Manuscript received May 8, 2018.
Manuscript revised July 20, 2018.
Manuscript publicized October 15, 2018.

[†]The authors are with Kwansei Gakuin University, Sanda-shi, 669-1337 Japan.
a) E-mail: tokadome@acm.org

DOI: 10.1587/transcom.2018SEP0002

*This paper is a full version of a conference paper [7] that focuses only on the single-hop sensor network. In addition to the single-hop sensor network, this paper deals with linear multi-hop and tree-based multi-hop one-way sensor networks. It also describes an experimental evaluation with the physical devices, which is not described in [7].

**We assume that a receiver transmits a beacon outside a slot before and after the slot.

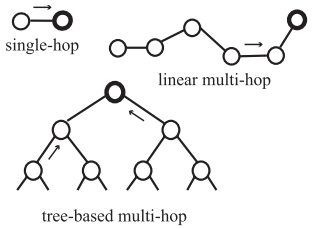


Fig. 2 Sensor network topologies.

the neighbor discovery algorithms may permit us to recover slot synchronization if the sensor nodes have a slot with at least twice the minimum slot length. This is because of the following reason. For any amount of synchronization deviation, there exists an overlap between the active slot of the sender and that of the receiver that is longer than the minimum slot length. If the receiver can estimate the amount of synchronizaition deviation during the active-slot overlap, the receiver can synchronize its slot to that of the sender. In either case, the neighbor discovery algorithms require additional resources.

The methods proposed in this paper are *not* based on slotted discovery schedules. In particular, the most basic method for a single-hop network requires *no* beacon or ack. The methods are complete; they are always able to resynchronize networks for any continuous phase offset. Furthermore, they are simple; they reduce the power required for resynchronization computations.

In this study, we focus on single-hop, linear multi-hop, and tree-based multi-hop one-way sensor networks. Focusing on the theoretical aspect of resynchronization, we first present a resynchronization method for one-to-one single-hop networks. We then extend this method to linear and tree-based sensor networks. Figure 2 shows schematic representations of the networks.

For simplicity, we begin with the following assumptions, which we relax later.

Assumption 1. The sleep and active intervals are constants with no fluctuation.

Assumption 2. After a synchronization deviation occurs in a sensor node, no node experiences further disturbance of synchronization until the node resynchronizes.

Besides, we assume, without loss of generality, that the sleep interval is significantly longer than the active interval. This assumption corresponds to the long-term battery charge and short-term data transceiver modes.

Section 3 discusses the relaxation of the assumptions mentioned above after Sect. 2 presents the resynchronization methods in detail. Section 4 describes simulations based on the resynchronization methods and an experiment using actual sensor nodes with wireless modules. Section 5 describes related work, and Sect. 6 concludes the paper. We present the proof of Property 1 in Sect. 2, but we describe them of Properties 2 to 12 in Appendix.

2. Resynchronization Methods

This section describes a few methods of synchronization recovery when a sensor node cannot communicate with the other nodes in a wireless sensor network that adopts a synchronous communication protocol.

2.1 Resynchronization for Single-Hop Networks

We first attempt resynchronization for a simple single-hop network consisting of two sensor nodes– the data sender node and the receiver node. Let node A be a sender and node B a receiver. They have *no* beacon. Node B does *not* return an ack signal to A. When a synchronization deviation occurs in either A or B, node B does not receive data from node A; node B then enters recovery mode, in this mode, node B attempts to receive data from node A. Once node B succeeds in receiving data, it can return to normal communication. We say this process of recovering synchronization *resynchronization*. Note that node A cannot detect the loss of a synchronization owing to the lack of a beacon or ack signal. Thus node A continues to send data in normal communication (normal mode), even after synchronization has been lost.

Let W be the active interval of nodes A and B in normal mode, W_B be the receive interval of node B in recovery mode, respectively, S be the sleep interval in normal mode and S_B be the sleep interval of node B in recovery mode. Further, let T be the cycle interval of nodes A and B in normal mode and T_B be the cycle interval of node B in recovery mode. By definition, $T = S + W$ and $T_B = S_B + W_B$ hold. Furthermore, without loss of generality, we assume that $T > S \gg W$, and thus $T_B > S_B \gg W_B$ holds.

For a synchronization deviation, the following method enables us to resynchronize the nodes.

Method SN (Single-hop Network). Let $\alpha = b + \gamma$ be a positive constant, where $b \geq 0$ is the integer part of α and $\gamma > 0$ is its fractional part. If node B does not receive data, Method SN puts node B in recovery mode, where we set

$$T_B = \alpha T = (b + \gamma)T \tag{1}$$

and we set W_B to a positive number that satisfies either of the following

$$W_B \geq W + \gamma T, \tag{2}$$

$$W_B \geq W + (1 - \gamma)T; \tag{3}$$

we repeat the longer sleep and active cycles. On the other hand, node A stays in normal mode. If node B receives data from node A while in recovery mode, the node judges itself to be resynchronized and changes its mode to normal mode.

Note that

$$\gamma = \frac{T_B}{T} - \left\lfloor \frac{T_B}{T} \right\rfloor, \tag{4}$$

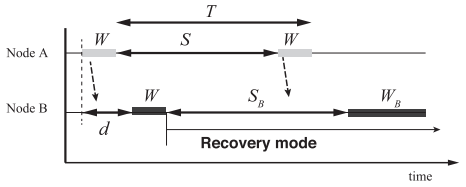


Fig. 3 Operations of nodes A and B in Method SN. W: active interval of nodes A and B in normal mode, W_B: active interval of node B in recovery mode, S: sleep interval in normal mode, S_B: sleep interval of node B in recovery mode, $T = W + S$: cycle interval in normal mode, d: magnitude of synchronization deviation.

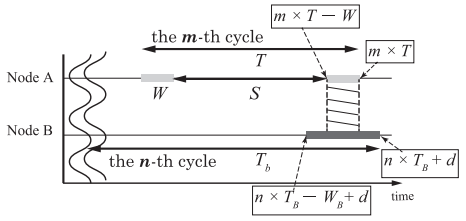


Fig. 4 A sufficient condition for resynchronization. A state occurs in which the active interval of node B covers that of node A. W: active interval of nodes A and B in normal mode, W_B: active interval of node B in recovery mode, S: sleep interval in normal mode, $T = W + S$: cycle interval in normal mode, d: magnitude of synchronization deviation.

where $\lfloor\ \rfloor$ denotes the floor function. Figure 3 illustrates the operations of nodes A and B in Method SN. Method SN can recover synchronization between nodes A and B in a finite time. We show this with the following property.

Property 1. *Let the amount of synchronization deviation be d[†]. (see Fig. 3). Then, for any d, $0 < d < T$, Method SN can resynchronize nodes A and B in a finite number of active–sleep cycles in recovery mode.*

Proof. Assume that, just after node B enters recovery mode, nodes A and B repeat m and n cycles whose intervals are T and T_B, respectively. Method SN succeeds in resynchronizing if a state occurs in which the active interval of node B covers that of node A (Fig. 4). The state is represented by

$$m \times T - W \geq n \times T_B - W_B + d \quad \text{and}$$
$$m \times T \leq n \times T_B + d,$$

which are equivalent to

$$nT_B - W_B + W + d \leq mT \leq nT_B + d. \tag{5}$$

A sufficient condition for resynchronization is therefore that, for any deviation d in a cycle, $0 < d < T$, there exist positive integers m and n that satisfy inequality (5). We first assume that W_B satisfies condition (2). Let us here consider the following inequality

$$-\gamma T + d \leq mT - nT_B \leq d. \tag{6}$$

[†]For $d < 0$, we can use $d' = T + d > 0$ as the magnitude of synchronization deviation instead of d. Thus, without loss of generality, we can assume $0 < d < T$.

Because $-(W_B - W) \leq -\gamma T$ holds from (2), we can see that if, for any deviation d, there exist positive integers m and n that satisfy (6), then the integers m and n also satisfy (5). Inequality (6) is equivalent to

$$(n-1)\gamma + \frac{d}{T} \leq m - nb \leq n\gamma + \frac{d}{T}. \tag{7}$$

Thus, it is sufficient to show that, for any deviation d in a cycle, there exist positive integers m and n that satisfy (7). We can easily observe the following facts from inequality (7). (a) The difference between the extreme left-hand and right-hand sides is the positive constant $\gamma < 1$ for each $n = 1, 2, \ldots$. (b) Both the extreme left-hand and right-hand sides monotonically increase as n increases, and the increment is the positive constant $\gamma < 1$. (c) The extreme left-hand side for $(n + 1)$ is equal to the extreme right-hand side for n for all $n = 1, 2, \ldots$. Hence, for some positive integer n, there exists a positive integer $\tilde{m}$ that satisfies

$$(n-1)\gamma + \frac{d}{T} \leq \tilde{m} \leq n\gamma + \frac{d}{T}. \tag{8}$$

Set $m = \tilde{m} - nb$, where $\tilde{m}$ and n are positive integers that satisfy (8). The positive integers m, n then obviously satisfy (7).

Next, we assume that W_B satisfies condition (3). We can then see that, from inequality (5) and (3), it is sufficient to show that there exist positive integers m, n that satisfy the following:

$$(n+1)\gamma - 1 + \frac{d}{T} \leq m - nb \leq n\gamma + \frac{d}{T}. \tag{9}$$

Just as described above, we can prove this assertion. □

Note that Property 1 is *not* a necessary condition for resynchronization. In Sect. 3, we shall give a necessary condition for resynchronization if sensor nodes work under the minimum active interval, where the minimum active interval is defined as the shortest active interval in length among those that allow the sensor nodes to send data and/or to receive data stably.

Let us consider the resynchronization method presented here, which uses a positive, non-integer constant α and W_B satisfying either (2) or (3). We again call the method *Method SN*.

Property 2. *Let α, γ, and T_B be the constants defined in Method SN. (a) If we set W_B to $W + \gamma T$, which is the minimum value that satisfies (2), then, after recovery mode starts, Method SN resynchronizes in at most $N_1 = \lceil \frac{1}{\gamma} \rceil$ cycles in recovery mode, where $\lceil\ \rceil$ is the ceiling function. Thus it takes Method SN at most $T_B N_1$ to resynchronize. Furthermore, the average latency of Method SN is expressed as*

$$L_1^{ave} = \sum_{k=1}^{\lfloor \frac{1}{\gamma} \rfloor} k\gamma T_B + \left\lceil \frac{1}{\gamma} \right\rceil \left(\frac{1}{\gamma} - \left\lfloor \frac{1}{\gamma} \right\rfloor \right) \gamma T_B \tag{10}$$

if the amount of synchronization deviation obeys the uniform

distribution.

(b) If we set W_B to $W + (1-\gamma)T$, which is the minimum value that satisfies (3), it resynchronizes in at most $N_2 = \lceil \frac{1}{1-\gamma} \rceil$ cycles in recovery mode. Thus it takes Method SN at most $T_B N_2$ to resynchronize. The average latency of Method SN is represented by

$$L_2^{ave} = \sum_{k=1}^{\lfloor \frac{1}{1-\gamma} \rfloor} k(1-\gamma)T_B + \left\lceil \frac{1}{1-\gamma} \right\rceil \left(\frac{1}{1-\gamma} - \left\lfloor \frac{1}{1-\gamma} \right\rfloor \right)(1-\gamma)T_B \quad (11)$$

if the amount of synchronization deviation obeys the uniform distribution.

Let us give some numerical examples of the recovery time. First, set $W = 10.0$ ms, $T = 1{,}000$ ms, and $b = 1$. If we have $\gamma = 0.002$, then $W_B = W + \gamma T = 12.0$ ms, which is the the minimum value that satisfies (2) in this setting, and it takes Method SN 4.18 min on average to resynchronize and at most 8.35 min. If $\gamma = 0.001$, then 8.35 min on average and at most 16.7 min, and if $\gamma = 0.0005$, then 16.68 min on average and at most 33.35 min. Next, set $W = 5.0$ ms, $T = 600$ ms, $b = 1$, and $\gamma = 0.0025$. Then, it takes Method SN 2.01 min on average to resynchronize and at most 4.01 min.

Note that, in addition to clock fluctuation, a packet loss leads to the beginning of the recovery mode. Even if the recovery mode is caused by the packet loss, Method SN succeeds in resynchonizing; however, theoretically, it is the worst recovery time to resynchronize the network. In practice, we can avoid this worst recevery time by slightly modifying the procedure in Method SN. That is, when a receiver does not receive any data, it maintains the normal communication mode instead of entering the recovery mode immediately, and if the receiver fails to receive data several times, it starts the recovery mode.

In closing this section, we describe two properties that characterize Method SN.

Property 3. *Assume the constant setting of Method SN. If W_B satisfies condition (2), then the increment of S, $\Delta S = S_B - S$, satisfies $\Delta S \leq (b-1)T$. The equality $\Delta S = (b-1)T$ holds if and only if the equality $W_B = W + \gamma T$ holds in (2).*

Note that $\Delta S \geq 0$ for $b \geq 1$, but $\Delta S < 0$ for $b = 0$. In contrast, the increment of W is always positive because $W_B > W$.

Property 4. *The condition where, with the assumption $S_B > W_B$, (1) holds and either (2) or (3) holds, where b is a non-negative integer and $0 < \gamma < 1$ is equivalent to that where either*

$$T_B = (b^+ + \bar{\gamma})T \quad (12)$$

or

$$T_B = (b^+ - \bar{\gamma})T \quad (13)$$

holds and

$$W_B \geq W + \bar{\gamma}T \quad (14)$$

holds, where b^+ is a positive integer and $0 < \bar{\gamma} < 1$.

2.2 Resynchronization for Linear Multi-Hop Networks

This section describes a method for resynchronization in one-way and linear multi-hop networks. The method uses Method SN for single-hop networks that was presented in the previous section. In a one-way and linear multi-hop network, data are sent from the terminal node to the sink node and data send alternates with data receive in each node. That is, a cycle in the operation of nodes except the terminal and sink nodes consists of two phases: data send and data receive. Thus, as shown in Fig. 5, the cycle interval T_2 is equal to $2(W + S) = 2T$, where W and S are the active and sleep intervals. The terminal node simply sends data to its succeeding node once every cycle. The root node also simply receives data from its preceding node once every cycle. In the following discussion, a receiver does not need to return an ack to a sender.

The following is a method for resynchronization in a one-way and linear multi-hop network.

Method LN (Linear Network). Use the cycle interval T_2 instead of T and define α, b, and γ as constants defined in Method SN. In recovery mode, Method LN changes the active intervals for data send to sleep intervals as shown in Fig. 6. If a node does not receive data from its preceding node, Method LN puts the node in recovery mode just as in Method SN using T_2 instead of T.

Note that in Method LN, once a node, say node B,

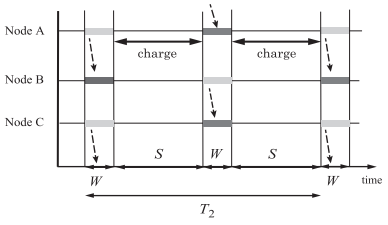


Fig. 5 The operations of sensor nodes in normal mode in simple synchronous communication in linear multi-hop networks. W: the active interval of nodes A and B, S: the sleep interval, $T_2 = 2(W + S)$: the cycle interval.

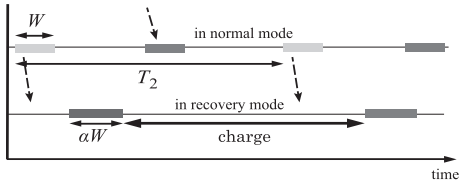


Fig. 6 The operations of sensor nodes in recovery mode in linear multi-hop networks. W: the active interval in normal mode, W_B: the active interval in recovery mode, $T_2 = 2(W + S)$: the cycle interval in normal mode.

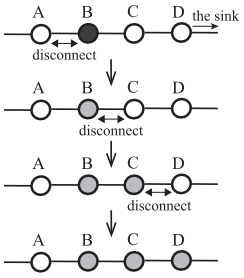


Fig. 7 The propagation of recovery mode in linear multi-hop networks. A synchronization deviation occurs in node B. Gray nodes are in recovery mode.

enters recovery mode, its succeeding node, say C, also enters recovery mode because after entering recovery mode, node B does not send data, and then node C does not receive data from node B. The same relation holds between the adjacent two nodes from B to the sink and thus they all enter recovery mode. Figure 7 illustrates the propagation of recovery mode.

Property 5. *Assume that a synchronization deviation occurs in a node and it does not receive data in a one-way and linear multi-hop network. Method LN enables us to resynchronize the network in a finite time.*

Property 6. *Assume that a synchronization deviation occurs in the k-th node from the sink node in a one-way and linear multi-hop network. Then just after the k-th node enters recovery mode, it takes Method LN at most* kT_BN *to resynchronize the network, where N is* N_1 *or* N_2 *defined in Property 2.*

Although Method LN is complete, the recovery time using this method increases in proportion to the number of nodes in a linear network, as shown in Property 6. We, however, have some ways to improve the speed of synchronization recovery drastically, if we allow the minimum use of an ack. For example, let nodes $n_1, n_2, \ldots, n_N$ consist of a linear multi-hop network, where node n_1 is the leaf node and node n_N is the sink node. We do not use an ack in the normal mode. Assume that the synchronization deviation occurs in node n_3 and that node n_4 cannot receive any data. If n_4 does not receive data from n_3, then n_4 sends a special message to n_5; if n_5 sends an ack back to n_4 in response to the special message, then n_4 can judge that a synchronization deviation did not occur in n_4 but it did in n_3. Thus, entering only n_3 into the recovery mode enables us to resynchronize the whole network without the propagation of the recovery mode. Hence, if we use this method for resynchronization in a large linear multi-hop network, the latency of synchronization recovery is of the same order as that in a single-hop two-node network.

2.3 Resynchronization for Tree-Based Sensor Networks

For simplicity, we assume a full binary tree of height H, where the top node is the root (sink) node and the nodes at the lowest level are leaves. In a tree-based sensor network,

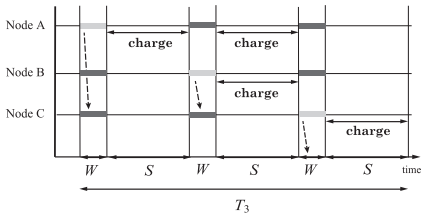
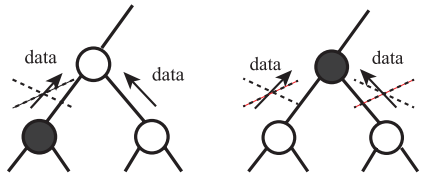


Fig. 8 Data transceiver in a tree-based sensor network in normal mode. Node C is the parent of nodes A and B. W: the active interval, S: the sleep interval, $T_3 = 3(W + S)$: the cycle interval.



(a) The deviation occurs in the left child.

(b) The deviation occurs in the node itself.

Fig. 9 A non-leaf node can determine whether a synchronization deviation occurs in the node or in one of the children. (a) If the deviation occurs in the left child, the node does not receive data from the left child. (b) If the deviation occurs in the node itself, the node does not receive data from the left or the right child.

every sensor node except the root node sends data to its parent in the active mode. Every non-leaf node except the root node receives data twice and sends data once in a transceiver phase because it has two children and a parent. That is, as shown in Fig. 8, node C receives data from its child nodes A and B and sends them to the parent in the active intervals. Thus the cycle interval T_3 in normal mode is equal to $3(W + S) = 3T$, where W and S are the active and sleep intervals. Leaf nodes simply send data to their parents in the active interval once every cycle. Further, the root node simply receives data from its children.

First, we describe a resynchronization method for the situation, in which a synchronization deviation occurs in a sensor node that is neither a leaf nor the parent of a leaf. In this case, a higher node need *not* return an ack to the lower ones. Note that a non-leaf node has two children that send data to the node. Thus Assumption 2 enables a non-leaf node to determine whether the synchronization deviation occurs or not in the node. As Fig. 9(a) shows, for example, the node determines that a synchronization deviation occurs in the left child because it does not receive data from it, but it does from the right child. Further, as Fig. 9(b) shows, the node determines that a synchronization deviation has occurred in the node because it does not receive data from the left or the right child.

Method TNU (Tree-based Network, Upper). Use the cycle interval T_3 instead of T and define α, b, and γ as constants defined in Method SN. In recovery mode, Method TNU changes the active interval for data send into the sleep interval and it also changes the active interval for one of the two receive phases into the sleep

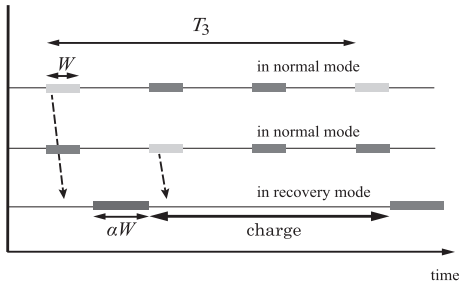


Fig. 10 The operations of a node in recovery mode and those of its children in normal mode in a tree-based sensor network. W: the active interval in normal mode, W_B: the active interval in recovery mode, $T_3 = 3(W + S)$: the cycle interval in normal mode.

interval as shown in Fig. 10. Let G be a node that is not a leaf or the parent of a leaf. If node G does not receive data from the children and if it determines that a synchronization deviation has occurred in node G, Method TNU puts node G in the same recovery mode as in Method SN using T_3 instead of T. The other nodes stay in normal mode.

Property 7. *Assume that a synchronization deviation occurs in a node that is* not *a leaf or the parent of a leaf in a tree-based sensor network. Method TNU permits us to resynchronize the network in a finite time.*

Now we turn to leaf nodes and their parents. If a leaf node does not receive the ack returned from its parent, it cannot detect the occurrence of synchronization deviation in it or in its parent because it simply sends data to the parent. Hence, resynchronization between a leaf node and its parent requires the ack to be returned from the parent[†]. Even when using an ack, a leaf node needs a particular operation for resynchronization because it has only one receiver (its parent), whereas a non-leaf node has two senders and a receiver.

Method TNL (Tree-based Network, Lower). Again, define α, b, and γ and use the cycle interval T_3 instead of T. Assume that a synchronization deviation occurs in a leaf node or in the parent of a leaf. Method TNL uses the same recovery mode as in Method TNU. Furthermore, it requires a *sync* message sent from a parent node to its child. Let node A be a leaf node and node C its parent.

i) Assume that the synchronization deviation occurs in leaf A. Then node A does not receive the ack returned from the parent C and thus node A enters recovery mode. Node C also detects the synchronization deviation and it changes its mode to *seminormal* mode, in which node C receives data from the other child, B, and sends data to its parent just as in normal mode, but node C sends a sync to node A at the time of receiving data from A in normal mode. The other nodes stay in normal mode. Figure 11 illustrates the operation of the nodes in the method.
After the synchronization between nodes A and C completes, they reenter normal mode.

ii) Assume that a synchronization deviation occurs in parent C. Then it enters recovery mode. The two child nodes, A and B, also enter recovery mode because they do not receive the ack returned from node C. Figure 12 shows the situation. In this method, parent D of node C changes its mode to seminormal mode, in which node D receives data from the other child and sends data to its parent just as in normal mode, but node D sends a sync to node C at the time of receiving data from node C in normal mode. The other nodes stay in normal mode. Figure 13 illustrates the operation of nodes A, B, C, and D after a synchronization deviation occurs in node C.
After the resynchronization between nodes C and D is completed, node C enters seminormal mode and sends a sync to its children A and B that are in recovery mode. Again, after node C resynchronizes its children, node C return to normal mode. Further, nodes A and B enter normal mode after they resynchronize node C.

[†]We can use the beacon instead of the ack. In particular, the beacon is more compatible with the resource-limited requirement if the size of the ack is larger than that of the beacon, although the beacon communication needs additional devices.

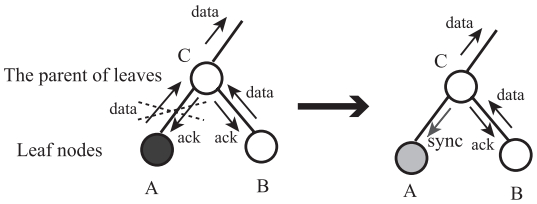


Fig. 11 A synchronization deviation occurs in a leaf node. Leaf-node A enters recovery mode. Node C slightly changes its mode into seminormal mode, in which it receives data from the other child B and sends data to its parent just as in normal mode, but node C sends a sync to node A. Gray nodes are in recovery mode.

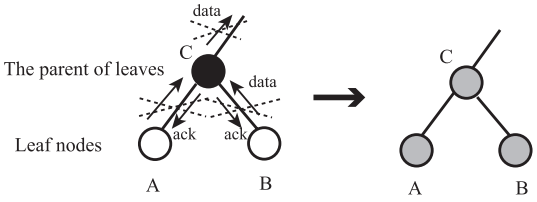


Fig. 12 When a synchronization deviation occurs in the parent of a leaf node, the parent enters recovery mode and its children also enter recovery mode. Gray nodes are in recovery mode.

Note that when a synchronization deviation occurs in the parent C, it enters recovery mode and its children A and B also enter recovery mode. The recovery of synchronization for the three nodes requires a sync from node D, the parent of C, as shown in Fig. 13.

Property 8. *Assume that a synchronization deviation occurs in a leaf node or in the parent of a leaf node in a tree-based sensor network. Method TNL permits us to resynchronize*

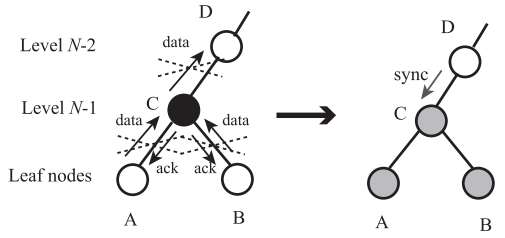


Fig. 13 When a synchronization deviation occurs in the parent, C, of a leaf node, the parent D of node C slightly changes its mode into seminormal mode, in which node D receives data from the other child and sends data to its parent just as in normal mode, but node D sends sync to node C at the time of receiving data from C in normal mode. Gray nodes are in recovery mode.

the network in a finite time.

From Properties 7 and 8, we obtain the following.

Property 9. *Assume that a synchronization deviation occurs in a node in a tree-based sensor network. Methods TNL and TNU permit us to resynchronize the network in a finite time.*

Property 10. *Assume that a synchronization deviation occurs in a node at level k in a tree-based sensor network. Methods TNL and TNU enable us to resynchronize the network in at most*

$$\tau = \begin{cases} \frac{2}{3}T_B N, & k < M-1, \\ 3T_B N, & k = M-1, \\ T_B N, & k = M, \end{cases}$$

where N is N_1 or N_2 defined in Property 2.

3. Discussion

We first discuss the constant setting in the resynchronization methods presented in this paper, relating it to the feasibility of the resynchronization methods. From Property 3, we find that the increment of the active time, ΔW, is always positive and that the increment, ΔS, of the sleep time satisfies $\Delta S \geq 0$ for $b \geq 1$, but $\Delta S < 0$ for $b = 0$. This means that if we set b to 0, we must use an active interval with a margin in normal mode Thus this setting might be physically infeasible. Even when we set b to 1, we must use an active interval with a small margin, but this setting is feasible as shown in the numerical example of the recovery time, where we set W = 10.0 ms, T = 1,000 ms, b = 1, γ = 0.001, and $W_B = W + \gamma T$ = 11.0 ms. In contrast, if we set b to 2 or higher, we do not need an active interval with a margin, but the recovery time of the methods becomes longer.

Next, we discuss the assumptions introduced in Sect. 1 and relax them. The simplified synchronous system requires constant sleep and active intervals without fluctuation, which is Assumption 1. Under such ultra-limited resources, even if we use state of the art technologies, the counter clock fluctuates (see, for example, [1]), which leads to unstable operation of the synchronous system. Thus, adopting the synchronous system is impractical, especially, for a large-scale sensor network. One of the solutions to the clock fluctuation problem is to use an active interval with a large margin. The use of a largely-margind active interval, however, contradicts the premise that the sensor network that we are focusing on works under ultra-limited resources.

The following property shows that a slightly modified version of Method SN enables us to apply resynchronization in a synchronous system with a fluctuating active/sleep interval if we modify the conditions of W_B and γ.

Property 11. *Let ε be an upper bound of the errors of the active/sleep interval. That is, we assume that, in both normal and recovery modes, the active/sleep interval fluctuates between the constant $\pm\varepsilon$. Set γ and W_B to satisfy either of the following:*

$$\gamma T > 2\varepsilon(b+1), \tag{15}$$

$$W_B \geq 2\varepsilon b + W + \gamma T \tag{16}$$

or

$$\frac{(b+4)\varepsilon}{T} + \frac{1}{2} < \gamma < 1 - \frac{(b+4)\varepsilon}{T}, \tag{17}$$

$$W_B \geq (2b+6)\varepsilon + W + (1-\gamma)T. \tag{18}$$

Furthermore, set $T_B = (b + \gamma)$, where b is a non-negative integer. Then, for any amount of synchronization deviation d, $0 < d < T$, Method SN with the above constant setting can resynchronize nodes A and B in a finite number of active–sleep cycles in recovery mode.

We can also relax Assumption 2: "After a synchronization deviation occurs in a sensor node, no node experiences further disturbance of synchronization until the node resynchronizes." Assume that synchronization deviation occurs as a Poisson arrival process, which is a commonly used model for random, mutually independent error occurrences. Then, the following property holds.

Property 12. *We assume that the number of synchronization deviations obeys a Poisson distribution; that is, the probability of exactly k synchronization deviations in the time interval between 0 and t is expressed as*

$$P(k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \tag{19}$$

where λ is the number of synchronization deviations per unit time. We also assume that the amount of synchronization deviation obeys the uniform distribution. Then, if we set $W_B = W + \gamma T$, the probability that Method SN resynchronizes without further synchronization deviation after a synchronization deviation occurs is given by

$$\sum_{n=1}^{\lfloor \frac{1}{\gamma} \rfloor} e^{-n\lambda T_B} \frac{\gamma T}{T} + e^{-\lambda \lceil \frac{1}{\gamma} \rceil T_B} \frac{(\frac{1}{\gamma} - \lfloor \frac{1}{\gamma} \rfloor)\gamma T}{T}. \tag{20}$$

If we set $W_B = W + (1-\gamma)T$, it is given by

$$\sum_{n=1}^{\lfloor \frac{1}{1-\gamma} \rfloor} e^{-n\lambda T_B} \frac{(1-\gamma)T}{T} + e^{-\lambda \lceil \frac{1}{1-\gamma} \rceil T_B} \frac{(\frac{1}{1-\gamma} - \lfloor \frac{1}{1-\gamma} \rfloor)(1-\gamma)T}{T}. \quad (21)$$

Let us give a numerical example of the probability of resynchronization. Set $W = 10.0$ ms, $T = 1{,}000$ ms, $b = 1$, $\gamma = 0.001$, and $W_B = W + \gamma T = 11.0$ ms. Then, the probability of resynchronization is greater than or equal to 87.29% if a synchronization deviation occurs on average every hour, 95.50% every 3 hours, 97.72% every 6 hours, 98.85% every 12 hours, 99.42% every 24 hours, and 99.81% every 72 hours.

Finally, we give a stronger sufficient condition for resynchronization without proof.

Property 13. *Set T_B, γ, and W_B to satisfy either of the following.*

$$T_B = (b + \gamma)T, \quad (22)$$

$$\frac{W - W_B}{T} \leq \lfloor k\gamma \rfloor - k\gamma < 0, \quad (23)$$

where k is a positive integer, b is a non-negative integer, and $0 < \gamma < 1$, or

$$T_B = (b^+ - \gamma)T, \quad (24)$$

$$\frac{W - W_B}{T} \leq k\gamma - \lceil k\gamma \rceil < 0, \quad (25)$$

where k is a positive integer, b^+ is also a positive integer, and $0 < \gamma < 1$. Then, for any amount of synchronization deviation d, $0 < d < T$, Method SN with the above constant setting can resynchronize nodes A and B in a finite number of active–sleep cycles in recovery mode.

If we put $k = 1$ in (22) to (25), then Property 13 yields Property 1. Property 13 gives us the strongest sufficient condition for resynchronization in the sense that if we take T_B, γ, and W_B values that satisfy either (22) and (23) or (24) and (25), then there exist integers m and n that satisfy (5) and, conversely, if there exist integers m and n that satisfy (5), then T_B, γ, and W_B in (5) satisfy either (22) and (23) or (24) and (25). It also presents us a necessary condition for resynchronization if sensor nodes work under the minimum active interval, where the minimum active interval is defined as the shortest active interval in length among those that allow the sensor nodes to send data and/or to receive data stably.

4. Experiment and Simulation

Yagiri and Okadome [7] confirm that Method SN recovers from the synchronization in the single-hop networks through simulation. The simulation results show that (a) Method SN always succeeds in resynchronizing, and (b) the maximums of the resynchronization latencies nearly coincide with the worst-case estimations in any case.

We also perform a simple experiment for Method SN implemented by XBee modules with Arduino boards. Furthermore, we confirm that the resynchronization methods for the multi-hop networks, Methods LN, TNU, and TNL recover from synchronization deviation through simple simulation. We assume that no packet loss occurs through the experiment and simulation.

4.1 Experiment

This section describes the experiment for Method SN implemented by XBee modules with Arduino boards. We have implemented Method SN using XBee 802.15.4 wireless networking RF modules [9] with Arduino [10]. That is, the sender is an Arduino board with an XBee module and the receiver is also an Arduino board with a XBee module. The implementation details are as follows. (a) Using the Hibernate function on the XBee modules, we put the XBee modules to sleep. (b) For the active interval, a sender sends one byte at a time to a receiver. (c) We consider a data transfer to be succeessful when the serial port of the receiver's XBee module has the data.

Using Method SN implemented by XBee modules with Arduino, we performed the experiment. We set the active interval in normal mode to 25.0 ms, that in recovery mode to 30.0 ms, the cycle interval to 1,025 ms in normal mode, γ to 0.2, and b to 1. Because an XBee module transmits a one-byte packet in less than or equal to 10 ms [11], the active interval of 25.0 ms seems to be long enough for the transmission time. In fact, Method SN implemented by using XBee modules with Arduino works without any trouble.

Starting from normal mode in which the receiver can receive data transmitted by the sender, the receiver delays the start of the active interval by a uniformly generated random deviation and enters recovery mode. When the receiver succeeds in receiving data, it sends the random deviation and the corresponding recovery latency to a PC connected to the receiver by a USB cable. After that, the receiver delays the start of the active interval by another uniformly generated random deviation and enters recovery mode again. The receiver repeats the operations mentioned above.

A 73.5-day experiment shows that the method always succeeds in synchronization recovery. In the experiment, we obtain 58,784 data in total. The average and maximum of latencies are 108.0 s and 221.5 s, respectively. On the other hand, the theoretical average and maximum of latencies under the same setting are 106.8 s and 211.2 s, which are compatible with the experimental results.

4.2 Simulation

In the simulations, we simulate the methods for each setting of the parameters 10,000 times for uniformly generated random synchronization deviations in a cycle.

In the simulations of Method LN, we randomly choose a sensor node in which a synchronization deviation occurs. We set b to two, three, and four. Furthermore, we set the ac-

tive intervals to 15.0 ms and 30.0 ms and the cycle intervals to 4.5 s and 9.0 s in normal mode. We assume a linear multi-hop network consisting of 10 or 20 sensor nodes. The simulation results show that (a) the method always succeeds in resynchronizing and (b) the maximums of the resynchronization latencies nearly coincide with the worst case estimations in any case. For a linear multi-hop network consisting of 10 sensor nodes, the average, standard deviation, minimum, and maximum of latencies become 20207.9 s, 16913.4 s, 45.0 s, and 72090.0 s, respectively, for the setting W = 15 ms, T = 4.5 s, and b = 4.

In the simulations of Methods TNU and TNL, we use a full binary tree of height 4 levels. Again, we randomly choose a sensor node in which a synchronization deviation occurs. We set b to two, three, and four. Furthermore, we set the active intervals to 15.0 ms and 30.0 ms and the cycle intervals to 4.5 s and 9.0 s in normal mode. In the simulations, the simulation results show that (a) Methods TNU and TNL never fail to recover from synchronization deviation and (b) the maximums of the resynchronization latencies nearly coincide with the worst case estimations in any case. For the leaf nodes of a full binary tree of height 4 levels, the average, standard deviation, minimum, and maximum of latencies become 8226.5 s, 4698.4 s, 67.5 s, and 16213.5 s, respectively, for the setting W = 15 ms, T = 4.5 s, and b = 4.

For the parents of the leaf nodes and for the same setting of W, T, and b, the average, standard deviation, minimum, and maximum of latencies become 24175.0 s, 14017.8 s, 202.5 s, and 48640.5 s, respectively. For the root node or its children, when we set W to 15 ms, T to 4.5 s, and b to 4, the average, standard deviation, minimum, and maximum of latencies are 5449.8 s, 3087.2 s, 45.0 s, and 10809.0 s, respectively.

5. Related Work

Dutta and Cutter [3] have proposed a protocol named Disco for asynchronous neighbor discovery. Disco ensures the discovery of a neighbor node. The nodes select a pair of prime numbers such that the sum of their reciprocals is equal to the desired duty cycles. The nodes then wake up at multiples of the individual prime numbers.

Kandhalu, Lakshmanan, and Rajkumar [5] have presented an algorithm named U-Connect. In U-Connect, the period between consecutive listen-slots is determined by the desired duty cycle. The protocol in U-Connect has a constraint that the period value needs to be a prime in the number of slots. Such wake-ups of the node at multiples of prime numbers ensure deterministic discovery latency which is an outcome of the Chinese Remainder Theorem.

Purohit, Priyantha, and Lie [6] have presented WiFlock, which uses a unified and collaborative beaconing mechanism to achieve both neighbor discovery and group maintenance goals. WiFlock's combined protocol can achieve high energy efficiency and low latency.

Zhang et al. [8] have proposed Acc, which serves as an on-demand generic discovery accelerating middleware for many deterministic neighbor discovery schemes. Acc leverages the discovery capabilities of neighbor devecies, supporting both direct and indirect neighbor discoveries. Further, they have presented a proactive online rendezvous maintenance mechanism, which is used to reduce delays for the detection of leaving of neighbors.

Many of the previous studies on synchronizaion in wireless sensor networks have focused on clock synchronization, that is, providing a common notion of time across the nodes of wireless sensor networks (for example, [2], [4]). This is because clock synchronization is viewed as a critical factor in maintaining the healthy functioning of wireless sensor networks.

6. Conclusion

Focusing on the theoretical aspects of resynchronization for wireless sensor networks with a synchronous communication protocol, this paper presents the methods that permit us to resynchronize single-hop, linear multi-hop, and tree-based multi-hop one-way sensor networks with ultra-limited resources. The methods proposed here are simple and thus they reduce the power consumed by resynchronization. The methods are complete because they always recover synchronization. This paper also describes the results of an experiment using actual sensor nodes with wireless modules. These results of the experiment show that the methods proposed in this paper are feasibile.

References

[1] I.F. Akyildiz and M.C. Vuran, Wireless Sensor Networks, John Wiley & Sons, 2010.
[2] L. Barenboim, S. Dolev, and R. Ostrovsky, "Deterministic and energy-optimal wireless synchronization," ACM Trans. Sen. Netw., vol.11, no.1, Article 13, 2014.
[3] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," Proc. 6th ACM Conference on Embedded Network Sensor Systems (SenSys'08), pp.71–84, 2008.
[4] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," Proc. 10th ACM Conference on Embedded Network Sensor Systems (SenSys'12), pp.1–14, 2012.
[5] A. Kandhalu, K. Lakshmanan, and R.R. Rajkumar, "U-connect: A low-latency energy-efficient asynchronous neighbor discovery protocol," Proc. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'10), pp.350–361, 2010.
[6] A. Purohit, B. Priyantha, and J. Liu, "WiFlock: Collaborative group discovery and maintenance in mobile sensor networks," Proc. 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'11), pp.37–48, 2011.
[7] H. Yagiri and T. Okadome, "Recovery of synchronization for wireless sensor networks," Accepted for a presentation in The 11th International Conference on Mobile Computing and Ubiquitous Networking (ICMU2018), 2018.
[8] D. Zhang, T. He, Y. Liu, Y. Gu, Y. Ye, R.K. Ganti, and H. Lei, "Generic neighbor discovery accelerations in mobile applications," ACM Trans. Sen. Netw., vol.11, no.4, Article 63, 2015.
[9] http://www.digi.com/products/xbee-rf-solutions
[10] https://www.arduino.cc/
[11] http://knowledge.digi.com/articles/Knowledge_Base_Article/Timin

g-Comparison-For-Data-Transfer-By-Product

Appendix: Proofs of Properties 2 to 12

Proof of Property 2. (a) Assume $W_B = W + \gamma T$. Inequality (5) is then equivalent to (7). Now just after the first cycle of recovery mode in node B, if there does not exist an integer m that satisfies (7) for $n = 1$, we have $\gamma + \frac{d}{T} < 1$, because the extreme left-hand side of (7) is $\frac{d}{T}$, where $0 < \frac{d}{T} < 1$. Recall that the extreme right-hand side of (7) monotonically increases as n increases, and its increment is γ. Thus, it reaches a value greater than or equal to 1 after at most $\lceil \frac{1}{\gamma} \rceil$ cycles, which implies Method SN requires at most $\lceil \frac{1}{\gamma} \rceil$ cycles to recover the synchronization. Next, we show that the average latency is given by (10). Note that if $d \approx T$, then $n = 1$ satisfies (7), which is the first cycle of recovery mode. Assume that, for a d, $\hat{n}$ is the minimum value that satisfies (7). Then when d decreases by γT, the minimum value of n that satisfies (7) is $\hat{n} - 1$ except when $0 < d < (\frac{1}{\gamma} - \lfloor \frac{1}{\gamma} \rfloor)\gamma T$, where $\frac{1}{\gamma}$ is not an integer. Hence, the probability that a synchronization deviation occurs for which it takes Method SN n cycles to resynchronize is $\frac{\gamma T}{T}$ if the amount of synchronization deviation obeys the uniform distribution. For $0 < d < (\frac{1}{\gamma} - \lfloor \frac{1}{\gamma} \rfloor)\gamma T$, where $\frac{1}{\gamma}$ is not an integer, the minimum value of n that satisfies (7) is equal to $\lceil \frac{1}{\gamma} \rceil$ and consequently its probability is $\frac{(\frac{1}{\gamma} - \lfloor \frac{1}{\gamma} \rfloor)\gamma T}{T}$. Thus, we obtain (10).

(b) The same reasoning for the case $W_B = W + (1-\gamma)T$ leads us to the conclusion that it recovers synchronization after at most $\lceil \frac{1}{1-\gamma} \rceil T_B$ and after L_2^{ave} represented by (11) on average. □

Proof of Property 3. Because $S_B = T_B - W_B$ by definition, we obtain

$$S_B \leq (b-1)T + T - W \tag{A·1}$$

from (2) with (1). This relation leads us to

$$\Delta S \leq (b-1)T \tag{A·2}$$

because, by definition, $S = T - W$ and $\Delta S = S_B - S$. Clearly, if the equality in (2) holds, then the equality in (A·2) also holds and vice versa. □

Proof of Property 4. First, assume that (1) holds and either (2) or (3) holds with the assumption $S_B > W_B$. (a) Let us assume $b \geq 1$. If (2) holds, put $b^+ = b$ and $\bar{\gamma} = \gamma$. Then we obtain (12) and (14) from (1) and (2). If (3) holds, put $b^+ = b + 1$ and $\bar{\gamma} = 1 - \gamma$. Then $(b+\gamma)T = (b + 1 - 1 + \gamma)T = b^+ - \bar{\gamma}$ and $W_B \geq W + \bar{\gamma}$. Thus we obtain (13) and (14) from (1) and (3). (b) Let us assume $b = 0$. In this case, (2) does not hold. This is because $T_B = \gamma T$ and thus, if (2) holds, $W_B \geq W + \gamma T > T_B > S_B$, which contradicts the assumption $S_B > W_B$. If (3) holds, put $b^+ = 1$ and $\bar{\gamma} = 1 - \gamma$. Then we obtain (13) and (14) from (1) and (3). Next assume that either (12) or (13) holds and (14) holds. If (12) holds, then we obtain (1) and (2). If (13) holds, put $b = b^+ - 1$ and $\gamma = 1 - \bar{\gamma}$. Then we also obtain (1) and (3) from (13) and (14). □

Proof of Property 5. Note first that we can apply Method SN to two adjacent nodes in a linear network because it does not depend on the cycle interval. Assume that synchronization fails between adjacent nodes A and B shown in Fig. 7. Then Property 1 assures synchronization recovery between them with Method SN. After the resynchronization completes, node B returns to normal mode, which, in turn enables us to resynchronize nodes B and C, the other node adjacent to B. The same recovery of synchronization propagates to the sink node, and thus the network is resynchronized. □

Proof of Property 6. From Property 2, it takes Method LN at most $T_B N$ to resynchronize two neighboring nodes, because it uses Method SN to recover synchronization between them. If a synchronization deviation occurs in the k-th node from the sink, Method LN uses Method SN k times to recover synchronization of the network. Hence it takes Method LN at most $kT_B N$ to resynchronize the network. □

Proof of Property 7. Note that we can apply Method SN to a parent–child pair in a tree-based network because it does not depend on the cycle interval. Assume that a synchronization deviation occurs in node G that is not a leaf or the parent of a leaf and node G enters recovery mode. Then Method SN resynchronizes node G with its children, which results in synchronization recovery among node G and the other nodes in the network because the other nodes, especially the children, stay in normal mode in Method TNU. □

Proof of Property 8. When a synchronization deviation occurs in a leaf node, Method TNL puts the node in recovery mode and its parent in semi-normal mode, in which the parent sends a sync to the leaf once every cycle that is synchronized with the cycles of the nodes in normal mode. Hence, Method SN ensures resynchronization of the leaf and its parent.

Next consider when a synchronization deviation occurs in the parent, node C, of a leaf node. In Method TNL, node C enters recovery mode and the parent of node C changes its mode into semi-normal mode, in which the parent sends sync to its child C once every cycle that is synchronized with the cycles of the nodes in normal mode. Again, Method SN ensures resynchronization of node C and its parent. Likewise, Method TNL permits resynchronization of node C and its children using Method SN. □

Proof of Property 9. This property is obtained from Properties 7 and 8 immediately. □

Proof of Property 10. Methods TNU and TNL basically use Method SN for recovering synchronization between a node and its parent. Assume first that a synchronization deviation occurs in a node at level $k < H - 1$. After the node enters recovery mode, it tries to receive data from its two children in the active interval of every cycle. Because the two children are in normal mode, the amount of the synchronization

deviation that occurred in the node is at most $\frac{2}{3}T_B$. Thus it takes at most $\frac{2}{3}T_B N$ to resynchronize the node and one of its two children, which results in the resynchronization of the network.

Next assume that a synchronization deviation occurs in a node at level $H-1$. The parent in semi-normal mode sends a sync to the node once every cycle (the interval: T_B) and it takes at most $T_B N$ to resynchronize the nodes. After resynchronization between the node and its parent, the node sends sync to its children for resynchronization among them and it takes at most $2T_B N$ to recover synchronization among them because the node sends a sync to a particular child once every cycle. Thus, it takes at most $3T_B N$ to resynchronize the network.

Finally, assume that a synchronization deviation occurs in a leaf node at level H. The parent of the leaf in semi-normal mode sends sync to the node once every cycle and it takes at most $T_B N$ to resynchronize the nodes, which coincides with the time of resynchronization of the network. □

Proof of Property 11. We outline a proof of this property. Let $\sigma_X(k)$ be the total error of the active interval of a node named X in k cycles and $\tau_X(k)$ that of the sleep interval. Then, a sufficient condition for resynchronization is that, for any deviation d in a cycle, there exist positive integers m and n that satisfy

$$nT_B - W_B + W + d + \epsilon^-_{m,n} \le mT \le nT_B + d + \epsilon_{m,n}, \quad (\mathrm{A}\cdot 3)$$

where

$$\epsilon^-_{m,n} = \sigma_B(n) + \tau_B(n-1) - \sigma_A(m) - \tau_A(m-1) \text{ and}$$
$$\epsilon_{m,n} = \sigma_B(n) + \tau_B(n) - \sigma_A(m) - \tau_A(m).$$

We first assume (15) and (16). It is then sufficient to show that, for any deviation d, there exist positive integers m and n that satisfy

$$\gamma(n-1) + \frac{-2\varepsilon b + d + \epsilon^-_{n,m}}{T} \le m - nb \le \gamma n + \frac{d + \epsilon_{n,m}}{T}. \quad (\mathrm{A}\cdot 4)$$

We can prove the following facts regarding inequality (A·4). For any synchronization deviation d, (a) the extreme left-hand side for n is less than or equal to the extreme right-hand side for n, (b) the extreme left-hand side for $n+1$ is less than or equal to the extreme right-hand side for n, (c) the extreme left-hand side for $n+1$ is greater than that for n, and (d) the extreme right-hand side for $n+1$ is greater than that for n. Set $m = b$ and $n = 1$. If (A·4) is not satisfied, then increase m to the minimum value of $m - nb$ greater than the extreme right-hand side for n. Next, increase n with $m - nb$ kept constant by increasing m by b when increasing n by one. Then, from the facts (a) to (e) mentioned above, both the extreme left-hand and right-hand sides increase as n increases and thus there exist m and n that satisfy (A·4) because $m - nb$ is kept constant. This implies that the integers m and n satisfy (A·3).

Likewise, if we assume (17) and (18), we can show that there exist positive integers m and n that satisfy the following:

$$\gamma n - (1-\gamma) + \frac{-(2b+6)\varepsilon + d + \epsilon^-_{m,n}}{T} \le m - nb \le \gamma n + \frac{d + \epsilon_{m,n}}{T},$$

which implies that the integers m and n satisfy (A·3). □

Proof of Property 12. Assume that we set $W_B = W + \gamma T$. Note that, if no further synchronization deviation occurs after a synchronization deviation d occurs, Method SN resynchronizes in n cycles if d satisfies $T - n\gamma T \le d < T - (n-1)\gamma T$ and it resynchronizes in $\lceil \frac{1}{\gamma} \rceil$ cycles if d satisfies $0 < d < (\frac{1}{\gamma} - \lfloor \frac{1}{\gamma} \rfloor)\gamma T$, where $\frac{1}{\gamma}$ is not an integer. Hence, from (19), we obtain (20) as the probability that Method SN resynchronizes without further synchronization deviation after a synchronization deviation occurs if d obeys the uniform distribution. Likewise, if we set $W_B = W + (1-\gamma)T$, we can show that the probability of resynchronization is given by (21). □

Hiromi Yagiri received the B.S. degree from Kwansei Gakuin University in 2017. He is now a graduate student in Kwansei Gakuin University.

Takeshi Okadome received Doctor of Science From the University of Tokyo in 1988. During 1998–2009, he was a researcher in NTT Lablatories. He is now a professor of Kwansei Gakuin University.