Wireless Netw (2018) 24:993–1005
https://doi.org/10.1007/s11276-017-1550-x

Constructing an optimally balanced tree to maximize data throughput with multiple channels

Trang Tien Nguyen[1] · **Hoon Oh**[1]

Published online: 7 July 2017

Abstract It is critical to improve throughput in real-time wireless sensor networks in order to reduce delay in data gathering and satisfy more real-time applications with tighter deadlines. One approach to achieving this is to construct a tree, such that the sizes of sub-trees of a root are well balanced, and to then assign a unique channel to each sub-tree. However, it is not easy to construct a balanced tree, because each node has to know the current connectivity status of other nodes in the network, which is not reasonable due to the increase in control overhead. We prove that building an optimal balanced tree is NP-Complete. Thus, we devise a heuristic algorithm to efficiently construct a balanced tree that is almost comparable to an optimally balanced tree. Furthermore, we suggest a way that each sub-tree is constructed to increase parallel transmission within itself. We apply the slotted sense multiple access (SSMA) protocol to each sub-tree, and evaluate the new approach with SSMA using a single channel and multi-channel lightweight medium access control. According to simulation results, SSMA using a balanced tree significantly outperforms other protocols in terms of packet delivery ratio and energy consumption.

Keywords Multiple channels · Throughput · Real-time · Reliable transmission · Wireless sensor networks

List of symbol

$T(i)$	$\{i\} + \{j \mid j$ is a descendant of node $i\}$
$level(i)$	The level of node i
R	The transmission range of a node
d(i,j)	The distance of node i and j
$P(i)$	The parent of node i
PCL(i)	Parent candidate list: the list of parent candidates of node i: $PCL(i) = \{k \mid d(k,i) \leq R, level(k) = level(i) - 1)\}$
CH(i)	Channel: the channel number of node i
PCCL(i)	Parent candidate channel list: the list of channels that the parent candidates of node i hold
ACL(i)	Allocated channel list: the list of channels allocated to the nodes at level i
NCL(i)	Neighbor channel list: the list of channels allocated to the neighbors at the same level
G(l)	The set of nodes located at level l
S(i)	The set of siblings of node i, including itself
C(i)	The list of children for node i
d(x)	Degree of node x: the number of edges outgoing from node x
TCR(S)	The tree construction request message issued by a sink: $TCR(S) = (level(S), (a, b, c, \ldots))$
JREQ	A join request message: $JREQ(v) = (v, level(v), CH(v))$
JRES	A join response message: $JRES(v) = (v, level(v), CH(v))$
ACK	An acknowledgment message: $ACK(v) = (v, level(v), CH(v))$

1 Introduction

Wireless sensor networks (WSNs) are expanding their application areas toward industrial fields in which applications monitor work environments, and provide workers with a service for safety, convenience, work efficiency, and so

✉ Hoon Oh
hoonoh@ulsan.ac.kr

[1] Ubicom Lab, School of Computer Engineering and Information Technology, University of Ulsan, P.O. Box 18, Ulsan 680-749, Korea

on. In this type of industrial application, the size of WSN is relatively small where each sink (or gateway) covers the number of nodes less than 40 nodes typically and the network can be extended by deploying more number of sinks. A server collects data periodically from sensor nodes via multiple sinks and provides a time-constrained service based on the analysis of the acquired data. This computing system is often called a *real-time context-aware system* in which *service quality relies on both the correctness of service decision and the timeliness of service provision.* A server should be able to acquire data in a reliable and timely manner in order to improve service quality. Considering a large spectrum of applications with different time constraints, it would be desirable to have a data gathering period as short as possible when designing a data transmission protocol in order to satisfy the time constraints of more context-aware applications. However, it is not easy to reduce data gathering time for wireless sensor networks with a limited bandwidth in which many nodes have to compete to transmit data through wireless multi-hop.

Many researchers have tried to improve data throughput to reduce the data-gathering period while ensuring real-time and reliable data transmission in WSNs [1–8]. Some of the research has focused on using a *spatial slot reuse* technique to improve throughput [2, 9, 10]. However, the spatial slot reuse technique suffers from the inefficiency of a precise slot schedule against the inherently unstable links of industrial wireless sensor networks and also the irregular interference[1] of different communication signals [9]. Recently, researchers have been showing more interest in the *channel-assisted slot reuse* technique, in which different transmitting nodes use different channels to provide parallel transmission [11–14]. However, the channel and slot schedule largely depend on network topology, which can be changed according to variations in interference factors. Meanwhile, another type of slot reuse, the *opportunistic slot reuse*[2] technique, based on a sharable slot within which multiple nodes compete for data transmission using carrier sense multiple access (CSMA), was proposed for the purpose of improving reliable data transmission and throughput, besides timely transmission [15]. This approach is promising, in that the slot schedule is made independent of any change in topology.

In this paper, we deal with the problem of improving throughput using multiple channels by using the opportunistic slot reuse approach, in which multiple nodes use a sharable slot. In this approach, a sharable slot is allocated to each tree level to constrain data transmission time, and the nodes at the same level try to perform reliable data transmission through competition using CSMA. However, since most of the nodes at the same level are blocked by the control messages [request to send (RTS) and clear to send (CTS)] of the handshake for channel contention between sending node and receiving node, the effect of opportunistic slot reuse is limited. To overcome this problem, we divide a tree into a number of sub-trees, and allocate a unique channel to each sub-tree, so that data transmissions among different sub-trees are secured independently. Nevertheless, the effect of using multiple channels will be limited unless a balance in sub-tree sizes is achieved. Therefore, this paper narrows the discussion to devising a tree construction and channel allocation algorithm that builds an optimally balanced tree, such that multiple sub-trees are balanced in size, and it allocates a unique channel to each sub-tree efficiently. We prove the optimality of tree balancing, and verify the effect of the algorithm through simulation.

According to our simulation results, it is shown that data delivery ratio, energy consumption, and data collection period are improved by 47.4, 34.7 and 20.0%, respectively when a balanced tree is used, compared to that of a randomly constructed tree, respectively.

The rest of the paper is organized as follows. In Sect. 2, we discuss the background for the proposed approach. The problem of the optimally balanced connection (OBC) between two adjacent tree levels is defined in Sect. 3. In Sect. 4, we give the details of a heuristic algorithm to balance the tree. The performance evaluation is given in Sect. 5. In Sect. 6, we make concluding remarks.

2 Background

2.1 Network and protocol model

We consider a wireless sensor network that consists of one sink and a number of sensor nodes. A sink has multiple transceivers so that it can simultaneously communicate with multiple sensor nodes using multiple channels. The sensor nodes form a tree originating from the sink, and may reach the other nodes via multiple wireless hops, since the transmission range of a node is limited for spectrum efficiency and to save battery power. A node senses data from the environment, periodically or on a need basis, and sends it to the sink along the tree path. Two nodes that can communicate with each other directly are said to have a *link*. A node that belongs to a tree is called a *tree-node*, a link between a parent and its child is a *tree-link*, and other links are an *ordinary link*.

[1] Irregular interference occurs since the interference range of a wireless signal is always farther than the communications range.

[2] *Opportunistic slot reuse*: Suppose that there are multiple (sender and receiver) pairs that need data transmission using CSMA. If senders and receivers are not blocked by RTS's of the other senders or CTS's of the other receivers, the senders can transmit data simultaneously, thus reusing a sharable slot opportunistically.

The protocol model used in this paper is based on a *slotted sense multiple access* (SSMA) mechanism, in which a sharable slot is allocated to each tree level, and the nodes at the tree level compete to send data to their respective parents using CSMA within the sharable slot. Data transmission is performed progressively, starting with the nodes at the highest level and proceeding to those at level 2 (the lowest sending level). The advantages of this approach are multi-fold. The slot schedule is independent of network topology, making the protocol robust against a change in topology; only the nodes at the same level compete for data transmission using CSMA, decreasing the competition significantly; they can send data in a reliable manner by using CSMA, as well as in a time-constrained manner by using a sharable slot; the approach can maximize data aggregation, and can effectively manage energy consumption. Refer to Oh and Azad [15] for detailed information regarding the scheduling of sharable slots.

2.2 Problem identification

In this paper, in a monitoring and control network that consists of one sink and a number of sensor nodes (typically less than 40 nodes), we discuss a way of improving the throughput of data transmission using multiple channels when the SSMA protocol is used. In order to improve the throughput, we consider a method to allocate multiple channels to a tree structure. Broadly, there are two channel allocation methods. One is to allocate a unique channel to each tree level in the tree's vertical direction, referred to as a *vertical channel allocation* (VCA) approach [16], and another is a *horizontal channel allocation* (HCA) approach [17], in which a tree is logically divided into multiple sub-trees of a sink, and then a unique channel is allocated to each sub-tree.

In the VCA approach, every node sends data to its parent in its one upper level using its parent channel. In this case, since a sensor node cannot send and receive data at the same time, only the nodes at every other level can send data simultaneously. For example, in Fig. 1, nodes 6 (at level 2) and 8 (at level 4) can send data using different channels at the same time. In addition, Vinh and Oh optimized the number of channels [16]. For example, when $k \geq 0$, the nodes at level $4\ k + 1$ and at level $4\ (k + 1) + 1$ can use the same channel without interfering with each other. In this case, two channels are enough to achieve maximum parallel transmission. However, if the tree depth is not high, the effect of parallel transmission by this way of channel allocation will not be high. Furthermore, the nodes at the same level mostly fail to make parallel transmission due to blocking as discussed above.

Meanwhile, in the HCA approach, a tree is divided into multiple sub-trees of a sink, and a unique channel is assigned to each sub-tree [17]. The HCA approach is simple, but has several advantages. First, the nodes in one sub-tree can transmit data independently of the other sub-trees. Second, it can limit the growth of an aggregated packet when data aggregation is used. Third, it can significantly reduce the number of competing nodes at each level. Fourth, it can improve the lifetime of the network by balancing the energy consumption of the nodes. However, in this approach, balancing the sizes of sub-trees in tree construction is of great importance. Consider a tree topology that is constructed in an arbitrary manner, as shown in Fig. 1. Since the second sub-tree, ST_2, is overwhelmingly big, the performance of data transmission in the whole tree will be dominated by that in ST_2.

We consider the way of constructing a balanced tree in this paper. So far, some researchers have tried to find a balanced tree in wireless sensor networks [18, 19]. In a dynamic balanced spanning tree [18], the approach considers distance to the sink, residual energy, and a node's weight (the number of children) to construct an energy-aware balanced tree. Every node has to continuously check the residual energy of its neighbors using a hello message. This also requires periodic reconstruction of the tree, resulting in changes in the role of nodes. Thus, it may not be appropriate for use in real-time applications. Chung-Kuo et al. [19] based their algorithm on three rules: *few-children-first*, *few-neighbor-first*, and *short-network-distance-first*, with priority in that order. One problem is that it can increase the depth of a tree, being a data-routing path, because an orphan node will always try to join the node with the fewest children. Thus, it would not be desirable to apply the algorithm for tree construction under SSMA. In this paper, we consider a novel algorithm that can form a balanced tree, in terms of sub-tree size, and can allocate channels efficiently.

Additionally, we consider a way of enhancing the opportunistic slot reuse within a sub-tree in order to improve throughput. When SSMA is used, a tree like ST_2 is not suitable for the opportunistic slot reuse. Channel acquisition by one node tends to block most of the other nodes at two adjacent levels, thereby limiting the possibility of the opportunistic slot reuse. For example, suppose that the nodes at level 4 (3, 8, 13, 14) and the nodes at level 3 (4, 9, 10) are in transmission mode and receiving mode, respectively. When node 3 sends RTS to its parent 4, node 8 is blocked. If node 4 responds with CTS, nodes 4, 9, and 10 at level 3 are blocked. Accordingly, the transmission by node 3 does not allow any opportunistic slot reuse or parallel transmission. Thus, to improve the opportunistic slot reuse, we present a method to build a sub-tree such that the nodes at the same level in the same sub-tree are distanced as far as possible and thus they can have the higher opportunity to transmit data in parallel.

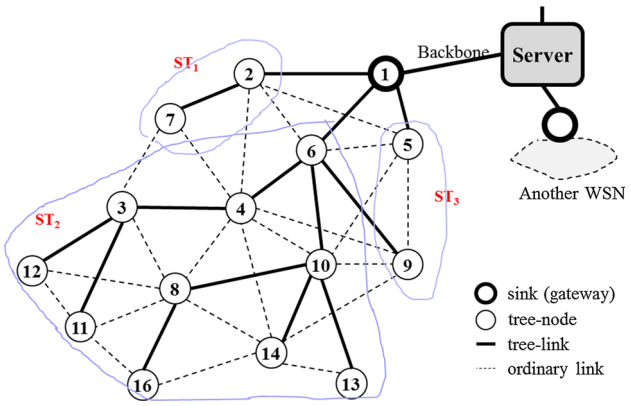


Fig. 1 Problem of data transmission with an unbalanced tree

2.3 Definitions and notations

As explained in the section on problem identification, it is desirable to construct a tree such that the sub-trees of a sink are balanced in size. For ease of explanation, we give some definitions with examples.

Definition 1 It is said that a covers b if there is an edge (a, b).

Definition 2 A tree level is *completely connected* if every node at level i is connected to all the nodes at level $i - 1$; it is *partially connected* if any node at level i is not connected to at least one of the nodes at level $i - 1$.

Definition 3 If tree level l is completely connected, the level is *optimally balanced* if every node, i, at level l has at least $\left\lfloor \frac{N(l)}{N(l-1)} \right\rfloor$ and at most $\left\lceil \frac{N(l)}{N(l-1)} \right\rceil$ children.

Definition 4 If tree level l is partially connected, the level is *optimally balanced* if, for every node i, $|C(p(i))| \leq \min\{|C(j)||j \in PCL(i)|\} + 1$. A tree is a *locally optimally balanced tree* (L-OBT) if every level in the tree is optimally balanced.

For example, Fig. 2(a) is not optimally balanced because node 1 has no child while nodes 2 and 3 each have two children. However, Fig. 2(a) satisfies Definition 4 since nodes 1, 2, and 3 have one, one, and two children, respectively.

Definition 5 A tree is an *optimally balanced tree* (OBT) if, when all nodes at a level l finish joining a tree, $\forall i \in \mathrm{G}(l)$ and $l(i) \geq 3$, $|T(i.sr)| \leq \min\{|T(j.sr)||j \in PCL(\mathrm{i}), j \neq P(i)\} + 1$, where $x.sr$ indicates the root of a sub-tree of a sink to which node x belongs.

The use of $PCL(i)$ excludes the special case that the formation of a sub-tree is independent of the formation of other sub-trees.

3 Optimally balanced connection problem

According to Definition 4, if all the levels of a tree are optimized in connectivity, the resulting tree becomes L-OBT. However, it is not easy to balance every tree level. The problem to balance the connections between two adjacent levels is referred to as an *optimally balanced connection* (hereafter, abbreviated as *OBC*) problem.

In Fig. 2, level l is partially connected. Suppose that the nodes at level l select the nodes at level $l - 1$. Since the selected node has to respond with a response message to the selector, the nodes at level l that are connected to the selected node can overhear the response message with the number of children that the selected node has. Accordingly, as in Fig. 2(a), if node 4 selects node 2, node 5 can record that node 2 has one child. However, it has no choice but to have node 2 or node 3 as its parent. In consequence, node 1 will not have any children, failing to satisfy Definition 3. However, as in Fig. 2(b), if node 5 selects its parent earlier than node 4, the optimal balance is achieved.

The OBC problem cannot be solved deterministically if a node does not have its *two-level topology information* that includes the link information for all the nodes at its own level and its one upper level. To build two-level topology information, every node has to send its link information to all nodes at those two levels periodically. This will cause a lot of overhead. Thus, assuming that a node does not have two-level topology information, we prove that the OBC problem is NP-Complete in the rest of this section.

Suppose there are two node groups, A and B. If all nodes in group A have a connection to at least one node in B, the optimization problem is defined as follows.

OBC Problem: *Given two node sets A and B, the OBC problem is to find* $E'(A, B) \subseteq E(A, B)$ *such that* $\forall a \in A$, *where a covers only one node in B and maxd*(B) - *mind*(B) *is minimized.*

Theorem 1 *The OBC problem is NP-Complete.*

To prove this, we take a simple instance of the OBC problem with $maxd(B)$ - $mind(B) = 0$ and show that the OBC problem can be reduced to a 3-partition problem [20] that is known to be NP-Complete. The 3-partition problem is defined as follows: given integers $S = \{v_1, v_2, ..., v_{3k}\}$, the problem to partition S into $S_1 \dot{U} S_2 \dot{U} \ldots \dot{U} S_k$ such that $\forall i$, $|S_i| = 3$ and $\forall i, \forall j, \sum_{x \in S_i} x = \sum_{x \in S_j} x$ is NP-Complete. Now, let us consider the instance of an OBC problem such that $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_k\}$. We introduce $A' = \{A'_1, A'_2, \ldots, A'_{3k}\}$ such that $\forall i$, $A'_i \subseteq A$. Then, we construct a set $S = \{v_1, v_2, \ldots, v_{3k}\}$ as follows: $v_i = |A'_i|$; and

$S = S_1 \dot{U} S_2 \dot{U} \ldots \dot{U} S_k$ where, $\forall i$, $S_i = \{x | x \in S\}$ such that $|S_i| = 3$.

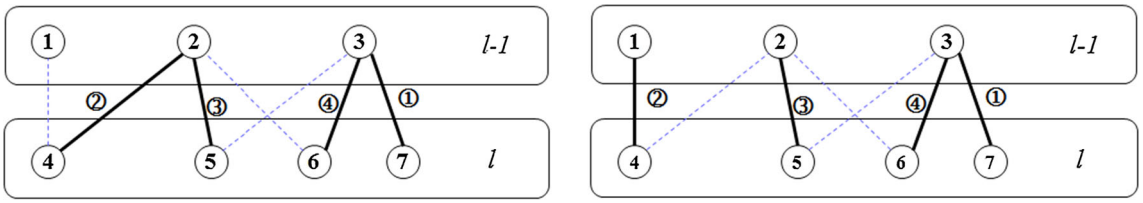


Fig. 2 An optimally balanced connection for a partially connected tree. **a** Optimally balanced connection. **b** Unbalanced connection

We prove that S admits a 3-partition if and only if the sets A and B have an OBC.

(only if) Suppose that S can make a 3-partition with subsets $\sum_{x\in S_i} x = \sum_{x\in S_j} x \quad \forall i, \forall j$. Then, we can assign the distinct $|A'_i|$ nodes in A to A'_i in A'. Consequently, $\sum_{x\in S_1} x$ nodes are connected to b_1, resulting in $(b_i) = d(b_j)$, $\forall i$, j. This implies that maxd(B) − mind(B) = 0.

(if) Suppose that the sets A and B have an OBC. Then, given $d(b_i)$, we can always generate three disjoint subsets $\left(A'_x, A'_y, A'_z\right)$ such that $A'_x \subseteq A, A'_y \subseteq A, A'_z \subseteq A$, and $|A'_x| + |A'_y| + |A'_z| = d(b_i)$ in polynomial time. Then, A' becomes 3-partition. Therefore, we prove that the OBC is NP-Complete. □

Figure 3 gives an example of a simple OBC problem that can be reduced to a 3-partition problem in polynomial time. Figure 3(a) shows the OBC such that each node in B has four connections. This can be transformed to the 3-partition problem, $S = \{2, 1, 1, 3, 1, 0, 4, 0, 0, 0, 1, 2, 1, \ldots\}$ where $S = A'$.

By Theorem 1, the problem to find an OBC for the given two sets is NP-Complete. Therefore, we focus on devising a heuristic algorithm to construct a balanced connection among the nodes at two adjacent levels.

3.1 Formulation of the OBC problem for integer linear programming

In this section, we introduce an integer linear programming formulation for finding an OBC. The objective is to minimize the deviation values of the degrees of the nodes in B, given two node sets, $A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_n\}$, and the set of edges $E = \{E_1, E_2, \ldots E_k\}$, where $E_i = \{(a, b) \mid a \in A, b \in B, d(a, b) \le R\}$. When a node in B has to choose one in A, let us introduce a decision variable x_{ij} as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } a_i \text{ chooses } b_j \text{ as its parent} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

For a node $i \in A$, let us define B_{ij} as $B_{ij} = \{j | d(i, j) \le R, j \in B\}$. Since $i \in A$ must have only one parent, the following constraint holds:

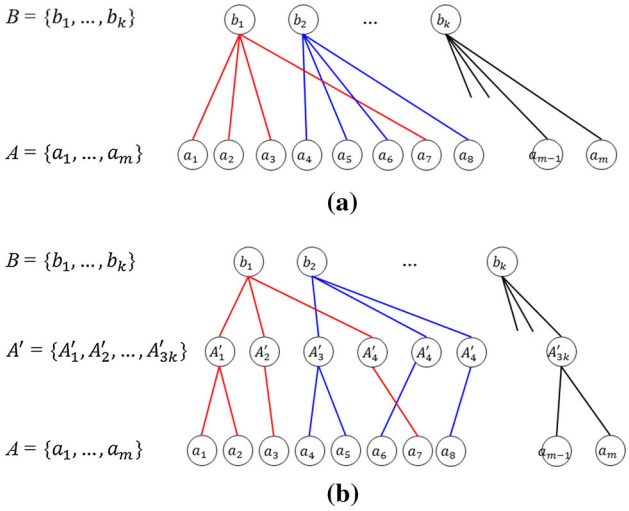


Fig. 3 An example of the OBC problem reduced to the 3-partition problem. **a** OBC problem with $d(b_i) = 4$. **b** The OBC problem with $d(b_i) = 4$ is transformed to the equivalent 3-partition problem with $\sum_{x\in S_i} x = 4$, where $S_i = \{x | x \in A'\}$ such that $|S_i| = 3$, $1 \le i \le 3\ k$

$$\sum_{j\in B_{ij}} x_{ij} = 1, \quad \forall i \in A \tag{2}$$

Since the total number of children is equal to the number of nodes in A, the following equality holds:

$$\sum_{i\in A, j\in B} x_{ij} = |A| \tag{3}$$

Then, the integer linear programming minimizes Γ:

$$\Gamma = \max\left\{ \left| \sum_{i\in A} x_{ij} - \sum_{i\in A} x_{ik} \right| j \in B,\ k \in B,\ j \ne k \right\} \tag{4}$$

subject to Eq. 1 and Eqs. 2 and 3.

4 Heuristic rules to find balanced connection

In this section, we devise heuristic rules (HRs) to find the balanced connection. When a node selects its parent from the nodes one level lower, we use two heuristic rules to optimize the balance of the degrees of the nodes at the parent level.

- (HR 1) A node always chooses as its parent a node from among its parent candidates that has the least number of children.

- (HR 2) A node that has the least number of parent candidates has the higher priority when choosing its parent.

HR 1 was drawn from the expectation that Γ will be minimized if a node always selects, as its parent, a node that has been selected the least as a parent for other nodes. However, following this rule does not result in the OBC, since the nodes at two adjacent levels are not completely connected. Thus, we use *HR* 2 together with *HR* 1. *HR* 2 was drawn to complement *HR* 1, because when a node with fewer candidates for selecting a parent has the higher priority when choosing a parent, this will reinforce *HR* 1. However, this cannot resolve the situation where two or more nodes have the same number of parent candidates, as explained in Fig. 2.

Now, we present an algorithm to generate a balanced connection by employing the two heuristic rules. Given two sets of nodes at level $i-1$ and i, $S(i) = \{a_1, a_2, \ldots, a_{n_i}\}$ and $S(i-1) = \{b_1, b_2, \ldots, b_{n_{i-1}}\}$, the centralized algorithm to construct a balanced connection is given as follows Fig. 4:

4.1 Distributed balanced tree construction algorithm

Now, we describe the *distributed balanced tree construction* (DBTC) algorithm to construct a balanced tree by utilizing the heuristic rules.

```
//D: small constant
// TCR = (sink Id, level = 1): Tree construction request message
1:  At every node x at initialization time:
2:        PCL(x) = ∅;
3:        Level(x) = 0;
4:  At a sink node:
5:        broadcast TCR(level= 1);
6:  At every node x that receives TCR (level = 1) from sink s:
7:        level(x) = level(s) + 1;
8:        PCL(x) = {s};
9:        pcl = PCL(x); //just copy
10:       ∀y∈ pcl, C(y) = ∅;
11:       set delay_timer = random(0, 1);
12: At every node x that overhears JREQ(level) from node y for the first time:
13:       level(x) = level + 1;
14:       PCL(x) = PCL(x) ∪ {y};
15:       set level_wait_timer = TCstime + (l − 2) × TCslot;
16: At every node x that overhears JREQ(level) from node y:
17:       if (level(x) == level+1) then
18:             PCL(x) = PCL(x) ∪ {y};
19: At every node x that level_wait_timer expires:
20:       pcl = PCL(x); //just copy
21:       ∀y∈ pcl, C(y) = ∅;
22:       set delay_timer = |PCL(i)| × D + random(0, 1);
23: At node x that delay_timer expires:
24:       choose y in pcl such that |C(y)| = min {|C(v)|| v ∈ PCL(x)};
25:       pcl = pcl − {y};
26:       set wait_timer to RTT;
27:       send JREQ to y;
28: At node x that receives ACK from node y:
29:       P(x) = y;
30:       remove wait_timer;
31: At every node, say x, that overhears ACK from node y:
32:       C(y) = C(y) ∪ {x};
```

Algorithm 1. A distributed balanced tree construction algorithm

Fig. 4 A centralized algorithm to construct a balanced connection

// Assume that every node x knows $PCL(x)$.
Step 1: every node x starts with $count = 1$ and $\forall y \in \mathrm{PCL(x)}$, $\mathrm{C(y)} = \emptyset$;
Step 2: select node x in L(i) such that $|\mathrm{PCL(x)}| == \min\{|\mathrm{PCL(y)}|\ \mathrm{y} \in \mathrm{L(i)}\}$;
Step 3: node x chooses y in PCL(x) such that $|\mathrm{C(y)}| = \min\{|\mathrm{C(v)}|\ \mathrm{v} \in \mathrm{PCL(x)}\}$;
Step 4: $\mathrm{C(y)} = \mathrm{C(y)} \cup \{\mathrm{x}\}$ and $count = count + 1$;
Step 5: if $count \leq n_i$, go to step 2;

Tree construction has to be performed progressively from the lowest level to the highest level. In other words, the nodes at level $i + 1$ have to wait until those at level i finish joining a tree. We assume that time is synchronized globally (see the flooding time synchronization protocol [21]; in fact, this time synchronization is needed only once, since SSMA synchronizes the time locally). First, sink s at level 1 becomes a tree member by itself and initiates tree construction by sending $TCR(s) = (s)$ at tree construction start time, *TCs time*.

Upon receiving TCR, every orphan node sets its level to the level in *TCR*, plus one. Then, every node at level l sets *level_wait_timer* as follows:

$$level\_wait\_timer(l) = TCstime + (l - 2) \times TCslot \quad (5)$$

where *TCslot* is the time slot taken for all nodes at the same level to finish joining a tree. Setting the proper *TCslot* value may be controversial; however, since tree construction is conducted only at the beginning of network operation with SSMA, it may not cause any problem, even though *TCslot* is set to a sufficiently long time. In fact, *TCslot* will vary depending on the maximum number of neighbors that a node can have. A node at level l remains active for *TCslot*, starting from *level_wait_timer*(l) and updates its *PCL* whenever it receives *JREQ* from any node one level lower.

Now, we need to determine which node takes the higher priority in order to issue *JREQ* at the same level to reflect *HR* 2. Every node i sets its delay timer, *delay_timer*(i), as soon as its *level_wait_timer* expires.

$$delay\_timer(i) = |PCL(i)| \times D + r \quad (6)$$

where D is a constant time, and r is a random value in the range [0, 1]. $D \geq \max(r)$ (=1) must hold to have a node with a smaller |*PCL*| join a tree earlier.

Next, when node i's *delay_timer* expires, it selects a parent from its *PCL* according to *HR* 1. Note that at the start of its own *TC slot*, every parent candidate x in *PCL* has its children $C(x) = empty$. Thus, node i selects one randomly from its *PCL*(i). It then sends *JREQ* to its selected parent, $P(i)$, which has to reply with *JRES*. At this time, all nodes that overhear *JRES* will know that selected node $P(i)$ now holds one child. In this way, the heuristic rules *HR* 1 and *HR* 2 are applied to construct a balanced connection between two adjacent levels.

Algorithm 1 details the distributed algorithm to construct a balanced tree, referred to as the *DBTC* algorithm. A balanced tree is constructed by using *TCR*, *JREQ*, and *ACK* messages. By overhearing *JREQ* and *ACK*, a node can determine its *PCL* and the number of children of each node in its *PCL*. Therefore, each node at level $l + 1$ must overhear JREQ from all nodes on its PCL at level l before it starts transmitting *JREQ* by using *level_wait_timer*(l) in Eq. 5. *A node that has the smaller PCL should join a tree earlier at the same level.* Thus, the *delay_timer* in Eq. 6 is used to determine the node that has the higher priority to issue *JREQ* at the same level.

4.2 DBTC algorithm with channel assignment

Constructing a balanced tree is critical in achieving high throughput. In addition, in order to increase the possibility of parallel transmission, the construction shape of a sub-tree has to be taken into account.

Figure 5 shows two construction shapes for two balanced sub-trees, ST_1 and ST_2. The first sub-tree in Fig. 5(a) has a *near-preferred* formation, such that the siblings in the same sub-tree are located near each other, whereas the shape in Fig. 5(b) has a *far-preferred* formation such that the siblings in the same sub-tree are located far from each other. The latter allows more parallelism in transmission. For example, nodes 8 and 10 in Fig. 5(b) can transmit data simultaneously. Accordingly, we can see that the shape of a sub-tree is another factor for increasing throughput. In this section, we discuss an algorithm to assign a unique channel to a sub-tree and to exploit the channel number to have the preferred formation of sub-trees.

Each node i manages $CHINFO(i)$
$$= (PCCL(i), NCL(i), ACL(i), CH(i), P(i)). \quad (7)$$

Each node sends its channel list and level number when it sends *TCR*, *JREQ*, or *JRES* in the process of tree construction. First, a sink, say i, broadcasts $TCR(i, level(i), CL(i))$, and upon receiving this message, a node, say j, selects its own channel such that two neighbor nodes at the same tree level do not select the same channel and updates $CHINFO(j)$, and then sends $JREQ = (j, level(j), CH(j))$ to join.

Finally, node j finishes joining the tree upon receiving $ACK = (i, level(i), CH(i))$ from node i. This algorithm performs distributed balanced tree construction with

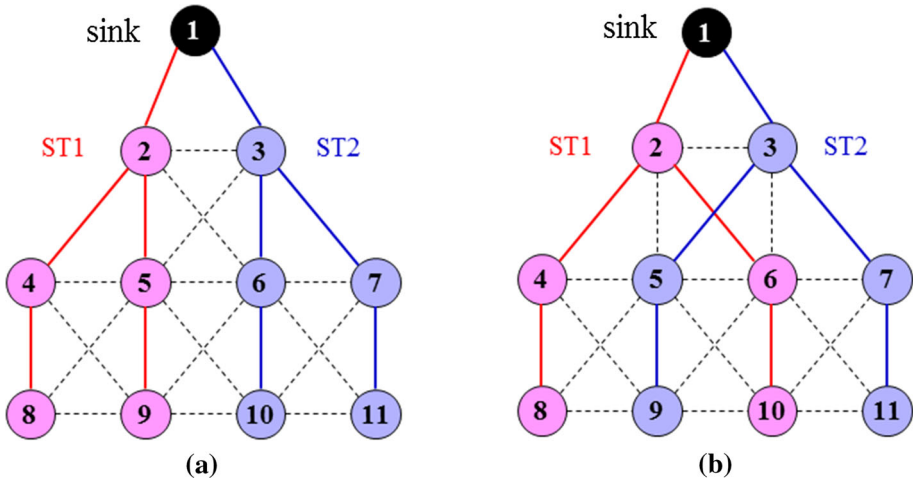
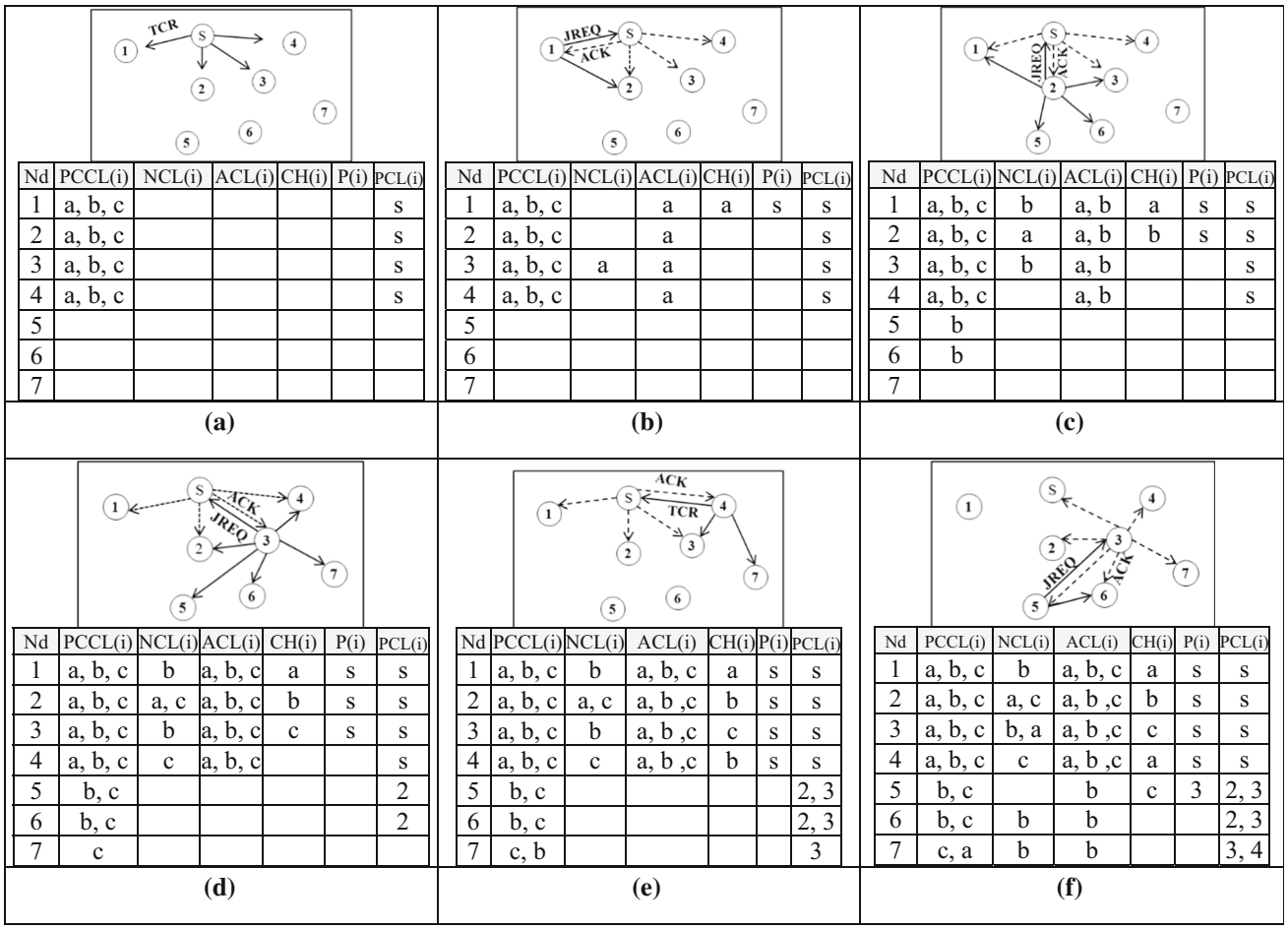


Fig. 5 The shapes of a sub-tree. **a** A optimally balanced tree. **b** A optimally balanced tree that maximizes a parallel transmission



Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c					s
2	a, b, c					s
3	a, b, c					s
4	a, b, c					s
5						
6						
7						

(a)

Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c		a	a	s	s
2	a, b, c		a			s
3	a, b, c	a	a			s
4	a, b, c		a			s
5						
6						
7						

(b)

Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c	b	a, b	a	s	s
2	a, b, c	a	a, b	b	s	s
3	a, b, c	b	a, b			s
4	a, b, c		a, b			s
5	b					
6	b					
7						

(c)

Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c	b	a, b, c	a	s	s
2	a, b, c	a, c	a, b, c	b	s	s
3	a, b, c	b	a, b, c	c	s	s
4	a, b, c	c	a, b, c			s
5	b, c					2
6	b, c					2
7	c					

(d)

Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c	b	a, b, c	a	s	s
2	a, b, c	a, c	a, b ,c	b	s	s
3	a, b, c	b	a, b ,c	c	s	s
4	a, b, c	c	a, b ,c	b	s	s
5	b, c					2, 3
6	b, c					2, 3
7	c, b					3

(e)

Nd	PCCL(i)	NCL(i)	ACL(i)	CH(i)	P(i)	PCL(i)
1	a, b, c	b	a, b, c	a	s	s
2	a, b, c	a, c	a, b ,c	b	s	s
3	a, b, c	b, a	a, b ,c	c	s	s
4	a, b, c	c	a, b ,c	a	s	s
5	b, c		b	c	3	2, 3
6	b, c	b	b			2, 3
7	c, a	b	b			3, 4

(f)

Fig. 6 Execution of the *DBTC-CA* algorithm

channel assignment. Thus, it is referred to as *DBTC with channel assignment* (*DBTC-CA*) algorithm.

Figure 6 illustrates the execution of the *DBTC-CA* algorithm for a network that has one sink, seven sensor nodes, and three available good channels: *a*, *b*, and *c*.

Each node can record the number of children of its parent. For example, in Fig. 6(f), when node 5 selects node 3 as a parent, upon overhearing *ACK* from node 3, nodes 6 and 7 can record that node 3 has one child. Accordingly, node 6 will select node 2 as a parent rather than node 3, and

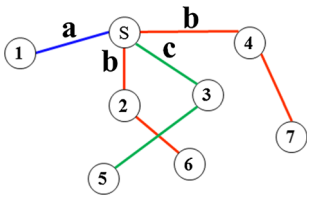


Fig. 7 A balanced tree finally obtained with the DBTC-CA algorithm

node 7 will select node 4. As a result, we obtain the final tree seen in Fig. 7.

4.3 Comparison of DBTC-CA and random construction

The DBTC-CA algorithm tries to exclude situations where a node chooses, as a parent, a candidate that already has many children by having parent candidates send to prospective children one level higher the number of children they have. In addition, it also tries to prevent the neighboring nodes at the same level from being allocated an identical channel by sharing channel allocation information among themselves. The former contributes to balancing the tree, while the latter helps to improve parallel data transmission. For example, nodes 3 and 14 in the same sub-tree can transmit data in parallel. In Fig. 8(a), if node 7 selects node 6, node 4 selects node 2 as a parent. If node 9 selects node 5, node 10 selects node 6 arbitrarily, since nodes 2, 5, and 6 all have one child. This algorithm is based on the principles that nodes at one level select their parents before the nodes at the higher levels, and a node with the least number of parent candidates always selects its parent before the other nodes at the same level. In this way, a node that has more parent candidates can select its parent by considering the balance of connectivity between two adjacent levels.

For example, in Fig. 8(b), if node 13 selects node 10, node 14 has no choice but to select node 10. As a result, node 10 has two children, whereas node 9 has no children, resulting in unbalanced connectivity. With the tree in Fig. 8(b), we can collect data from all nodes during 11 data transmissions, whereas with the tree in Fig. 8(a), we can do it during only five data transmissions, considering parallel data transmission.

5 Performance evaluation

5.1 Simulation model

For convenience, the SSMA protocol using a balanced tree constructed via the DBTC-CA algorithm is referred to as SSMA-DBTC-CA. We evaluated SSMA-DBTC-CA against its ancestor, SSMA and the Multi-Channel Lightweight Medium Access Control (MC-LMAC) [13] which is a well-known tree-based multi-channel protocol. We used the Qualnet simulator version 5.0.2 [22], which has been used commercially. Table 1 shows the key simulation parameters and values.

5.2 Performance metrics

In this simulation, we want to evaluate the soundness of balance in tree construction when a new tree construction algorithm is used. When the set ST of sub-trees in tree T is expressed as $ST = \{ST_1, ST_2, \ldots, ST_k\}$ where ST_i indicates the i^{th} sub-tree of a sink, a balance index δ is given as a standard deviation for the sizes of sub-trees in a constructed tree T as follows:

$$\delta = \sqrt{\frac{1}{|ST|}\sum_{i=1}^{|ST|}(|ST_i| - \mu)^2}, \quad \mu = \frac{N-1}{|ST|} \tag{8}$$

where |ST| is the number of sub-trees, $|ST_i|$ is the number of nodes of the ith sub-tree, and N is the total number of nodes in the network.

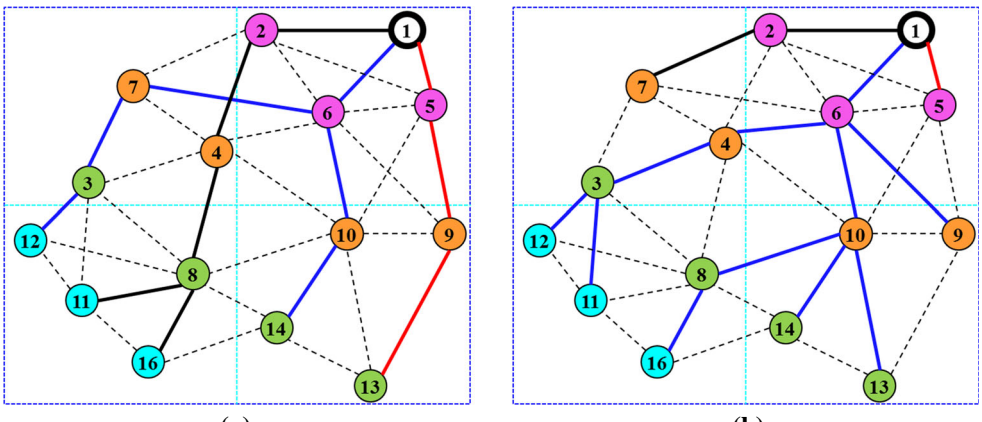


Fig. 8 Balanced and unbalanced trees. **a** A tree constructed with the DBTC-CA algorithm and three available good channels. **b** A tree constructed with the random algorithm

Table 1 Key simulation parameters and values

Parameter	Value
Number of nodes, *nNodes*	15–100
Payload length	32 bytes
Transmission range, *R*	25 m
Number of frequencies	4
Path loss model	2-ray
Shadowing model	Constant (*Mean* = 4 dB)
Noise factor	10 dB
Sensor energy model	MicaZ
Battery model	Linear
Simulation time, *T*	600 s

We also examine the size of maximum sibling group for a constructed tree T with a sink *s* as follows.

$$nSiblingsMax = \max\{|S(i)|i \in T(s)\} \quad (9)$$

The bigger nSiblingsMax indicates that a node in the same sub-tree has more children. This is not good since more nodes will compete to transmit data to their parent, increasing delay of packet.

Next, we compare some protocols in terms of data transmission throughput and energy consumption. In general, throughput indicates the number of bits processed per a unit time. Since we use time slot in protocols for the timely delivery of data, it is not meaningful to evaluate throughput. Thus, we evaluate throughput indirectly by checking how quickly each protocol can deliver data packet upon decreasing a superframe size (data gathering period). Finally, we evaluate energy consumption for the tree protocols, expecting that a balanced tree incurs less competition.

5.3 Evaluation results

5.3.1 Soundness of tree balance

In this simulation, 15–100 nodes are randomly distributed in an area of 100 × 100 m^2 to form trees using each of three different tree construction algorithms: *OBT*, *DBTC*, and random selection(*RS*). We generated optimal trees by using the formulation in Sect. 3.1 and the GNU Linear Programming Kit for Windows [23].

Figure 9 shows the balance index δ for the three tree construction methods. It is shown that the balance index of DBTC is almost comparable to that of OBT. This implies that the proposed heuristic algorithm works well. On the other hand, the RS algorithm shows high imbalance and increasing gap with increase in the number of nodes. Figure 10 shows the maximum number of siblings, nSiblingsMax, for nodes under OBT, DBTC and RS. Note that DBTC and OBT show the identical curve. It is natural three graphs show an increasing pattern as the number of nodes increases. The graphs also show that every node with OBT and DBTC has two or fewer siblings with less than 30 nodes and at most five nodes with 100 nodes. In conclusion, we can say that OBT and DBTC improve the maximum number of siblings by almost 50% overall compared with RS.

5.3.2 Throughput evaluation

In this simulation, we compared the three approaches, SSMA, SSMA-DBTC-CA, and MC-LMAC, in terms of packet delivery ratio (PDR) according to the variation of *nNodes* and superframe size. Both SSMA-DBTC-CA and MC-LMAC used four channels equally.

As *nNodes* varied from 30 to 70 nodes with superframe fixed at 1000 or 1600 ms, we evaluated PDR. Referring to Fig. 11, upon comparing SSMA-DBTC-CA and SSMA

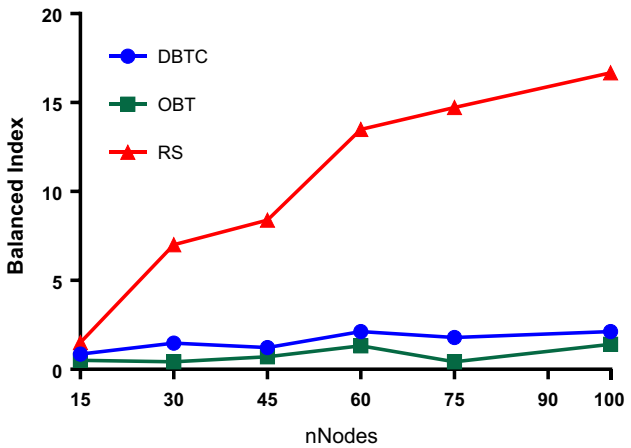


Fig. 9 Comparison of tree balance

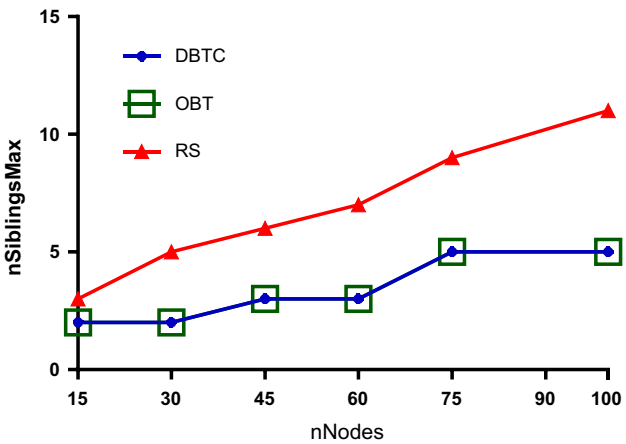


Fig. 10 Comparison of the maximum number of siblings

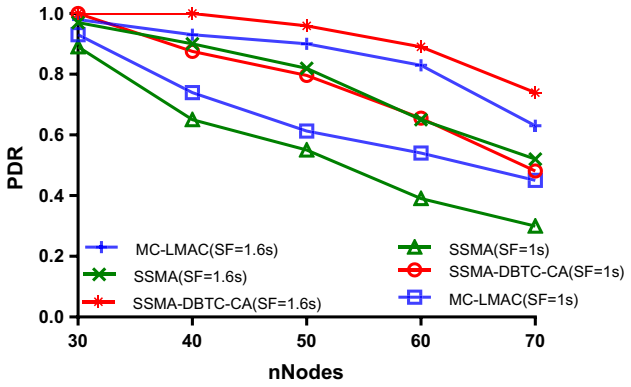


Fig. 11 Packet delivery ratio according to variation of nodes

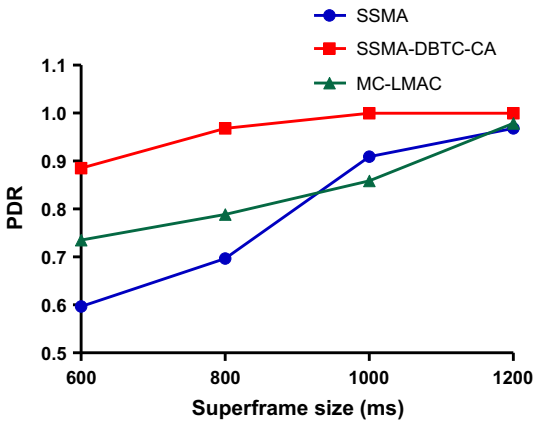


Fig. 12 Packet delivery ratio according to variation of superframe size

which use the same data transmission technique for reliability and timely transmission, the graphs indicate that SSMA-DBTC-CA can process packets faster than SSMA for the same superframe size. In fact, considering that all these protocols target a small monitoring and control network with less than 40 nodes, the proposed protocol shows highly dependable performance even with SF = 1.0 s. Note that the PDR of SSMA are sharply decreasing as the number of nodes decreases, especially when the superframe size is small (SF = 1.0 s). In Fig. 12, as the superframe size increases from 600 to 1200 ms, the graphs show PDRs for the three approaches. It shows that the proposed protocol can survive well with the small superframe size of 600 ms compared with the other two's. This tells that the proposed protocol can process packets much faster than the other ones, indirectly indicating good throughput.

5.3.3 Energy consumption

Energy consumption depends on the number of competing nodes, duty cycle, control overhead, etc. SSMA manages

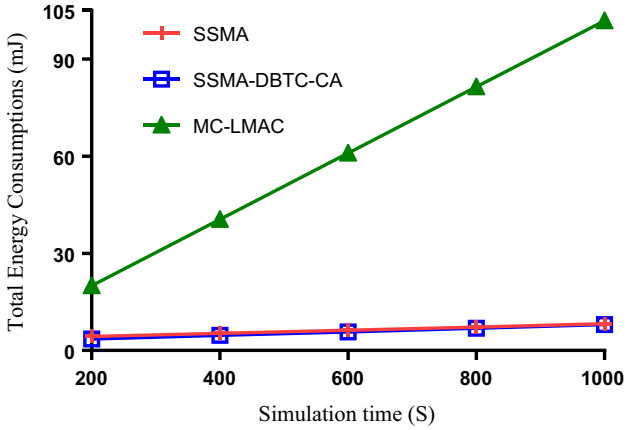


Fig. 13 Cumulative energy consumption with time (SF = 1 s)

energy of node very effectively by using time slots for data transmission and by allowing nodes to enter a sleep state as soon as they finish data transmission within the allocated sharable slot. For the first 200 s, SSMA consumes more energy since it involves tree construction and time synchronization. Then, it consumes a small amount of energy for the other 200 s intervals, resulting in the very slow increase in the curve. SSMA-DBTC-CA shows a graph of a similar pattern; however, it consumes energy slightly less than SSMA since it handles the smaller number of nodes with less competition in data transmission (in the figure, two graphs are shown equally because of the scale of graph). On the contrary, a node in MC-LMAC has to remain active during the common frequency period in every slot because it does not know when potential packets will arrive. Thus, it consumes a lot of energy consumption as Fig. 13.

6 Conclusions

Building an optimally balanced tree was proven to be NP-Complete. Thus, a heuristic distributed balanced tree construction algorithm (DBTC) was proposed to construct a balanced tree efficiently. We proved by simulation that the balance of a tree generated by DBTC is almost the same as that by the optimal balanced tree algorithm that constructs an optimally balanced tree. We applied DBTC to the multi-channel SSMA protocol to maximize the efficiency of using multiple channels. According to simulation, compared with SSMA and MC-LMAC the new multi-channel protocol could reduce superframe size greatly with the sustained PDR, implying indirectly that the new approach increases throughput significantly. It was also shown that it could improve PDR greatly by reducing the competition of data transmission.

Acknowledgement This research was supported by the 2015 Research Fund of University of Ulsan.

References

1. Oh, H., & Van Vinh, P. (2013). Design and implementation of a MAC protocol for timely and reliable delivery of command and data in dynamic wireless sensor networks. *Sensors, 13*(10), 13228.
2. Wen-Zhan, S., Renjie, H., Shirazi, B., & Lahusen, R. (2009). TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks. In *IEEE international conference on pervasive computing and communications,* 2009. *PerCom* 2009, 9–13 *March* 2009 (pp. 1–10). doi:10.1109/PERCOM.2009.4912757.
3. Suriyachai, P., Brown, J., & Roedig, U. (2010). Time-critical data delivery in wireless sensor networks. In R. Rajaraman, T. Moscibroda, A. Dunkels, & A. Scaglione (Eds.), *Distributed computing in sensor systems: Proceedings of the 6th IEEE international conference, DCOSS 2010, Santa Barbara, June 21–23, 2010* (pp. 216–229). Berlin: Springer.
4. Petersen, S., & Carlsen, S. (2011). WirelessHART versus ISA100.11a: The format war hits the factory floor. *Industrial Electronics Magazine, IEEE, 5*(4), 23–34. doi:10.1109/MIE.2011.943023.
5. Singh, B. K., & Tepe, K. E. (2009). Feedback based real-time MAC (RT-MAC) protocol for wireless sensor networks. In *Global telecommunications conference,* 2009. *GLOBECOM* 2009. *IEEE, Nov.* 30 2009–*Dec.* 4 2009 (pp. 1–6). doi:10.1109/GLOCOM.2009.5425620.
6. Hoesel, L. F., & Havinga, P. J. M. (2004). A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In *Paper presented at the 1st international workshop on networked sensing systems, INSS 2004*, Tokio.
7. Jungsook, K., Jaehan, L., Pelczar, C., & Byungtae, J. (2008). RRMAC: A sensor network MAC for real time and reliable packet transmission. In *IEEE international symposium on consumer electronics,* 2008. *ISCE* 2008, 14–16 *April* 2008 (pp. 1–4). doi:10.1109/ISCE.2008.4559491.
8. Vinh, P., & Oh, H. (2012). RSBP: A reliable slotted broadcast protocol in wireless sensor networks. *Sensors, 12*(11), 14630.
9. Malhotra, B., Nikolaidis, I., & Nascimento, M. A. (2010). Aggregation convergecast scheduling in wireless sensor networks. *Wireless Networks, 17*(2), 319–335. doi:10.1007/s11276-010-0282-y.
10. Chaudhary, M. H., & Scheers, B. (2012). High spatial-reuse distributed slot assignment protocol for wireless ad hoc networks. In *Communications and information systems conference* (*MCC*), 2012 *Military,* 8–9 *Oct.* 2012 (pp. 1–8).
11. Jovanovic, M. D., & Djordjevic, G. L. (2007). TFMAC: Multi-channel MAC protocol for wireless sensor networks. In *Proceedings of the 8th international conference on telecommunications in modern satellite, cable and broadcasting services,* 2007. *TELSIKS 2007, 26–28 Sept. 2007* (pp. 23–26). doi:10.1109/TELSKS.2007.4375929.
12. Seungku, K., & Doo-Seop, E. (2014). Link-state-estimation-based transmission power control in wireless body area networks. *IEEE Journal of Biomedical and Health Informatics, 18*(4), 1294–1302. doi:10.1109/JBHI.2013.2282864.
13. Incel, O. D., van Hoesel, L., Jansen, P., & Havinga, P. (2011). MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. *Ad Hoc Networks, 9*(1), 73–94. doi:10.1016/j.adhoc.2010.05.003.
14. Borms, J., Steenhaut, K., & Lemmens, B. (2010). Low-overhead dynamic multi-channel MAC for wireless sensor networks. In J. S. Silva, B. Krishnamachari, & F. Boavida (Eds.), *Proceedings of the wireless sensor networks: 7th European conference, EWSN 2010, Coimbra, February 17–19, 2010* (pp. 81–96). Berlin: Springer.
15. Oh, H., & Azad, K. M. A. (2016). A big slot scheduling algorithm for the reliable delivery of real-time data packets in wireless sensor networks. In Q.-A. Zeng (Ed.), *Wireless communications, networking and applications: Proceedings of WCNA 2014* (pp. 13–25). New Delhi: Springer.
16. Van Vinh, P., & Oh, H. (2015). O-MAC: An optimized MAC protocol for concurrent data transmission in real-time wireless sensor networks. *Wireless Networks, 21*(6), 1847–1861. doi:10.1007/s11276-015-0887-2.
17. Yafeng, W., Stankovic, J. A., Tian, H., & Shan, L. (2008). Realistic and efficient multi-channel communications in wireless sensor networks. In *INFOCOM 2008. The 27th conference on computer communications. IEEE, 13–18 April 2008*. doi:10.1109/INFOCOM.2008.175.
18. Avokh, A., & Mirjalily, G. (2010). Dynamic balanced spanning tree (DBST) for data aggregation in wireless sensor networks. In *Proceedings of the 2010 5th international symposium on telecommunications* (*IST*), *4–6 Dec. 2010* (pp. 391–396). doi:10.1109/ISTEL.2010.5734058.
19. Chung-Kuo, H., Guey Yun, C., & Jang-Ping, S. (2012). Load-balanced trees for data collection in wireless sensor networks. In *Proceedings of the 41st international conference on parallel processing workshops* (*ICPPW*), *2012, 10–13 Sept 2012* (pp. 474–479). doi:10.1109/ICPPW.2012.65.
20. Garey, M. R., & Johnson, D. S. (1990). *Computers and intractability: A guide to the theory of NP-completeness*. London: W. H. Freeman & Co.
21. Maroti, M., Kusy, B., Simon, G., & Ledeczi, A. (2004). The flooding time synchronization protocol. In *Paper presented at the proceedings of the 2nd international conference on embedded networked sensor systems*. Baltimore.
22. QualNet 5.0.2 network simulator. http://web.scalable-networks.com/qualnet.
23. GPLK for windows. http://winglpk.sourceforge.net/.

Trang Tien Nguyen received the B.S. degree in control and automation engineering from Department of Electrical Engineering at the Hanoi University of Science and Technology, Vietnam, in 2013. He is currently a graduate student with the Department of Electrical and Computer Engineering, University of Ulsan, South Korea, where he is working toward the M.S./Ph.D. degree. His research interests lie in embedded systems, and mobile ad hoc networks.

Hoon Oh received the B.S.E.E. degree from the Sung Kyun Kwan University, Seoul, and the M.SC. degree and the Ph.D. degree in computer science from Texas A&M University at College Station, Texas, in 1993 and 1995, respectively. From 1983 to 1989 and 1996 to 2000, he worked as a software engineer and software architect in the Corporate Research Center of Samsung Electronics. He was involved in developing communication protocols for the data services of the CDMA and IMT2000 handset products. Currently, he is a professor of the Department of Computer Engineering and Information Technology and a director of the Vehicle IT Convergence Technology Research Center in University of Ulsan, Korea. He has been serving as a department head of IT convergence department from 2016, University of Ulsan. He received a Best Paper Award from the National Academy of Science, USA in 1995. He published over fifty refereed journals for the last decade and made many technology transfers to the local IT companies through University-industry cooperation program. His research interests lie in embedded systems, mobile ad hoc networks, real-time computing, context-aware computing. He is a member of IEEE, IEICE, and ICASE, and a lifetime member of KICS and KISA. Wireless Netw (2015) 21:1847–1861.