Low-delay Node-disjoint Multi-path Routing using Complementary Trees for Industrial Wireless Sensor Networks

Luming Liu[1,2], Zhihao Ling[1,2] and Yun Zuo[1,2]
[1] School of Information Science and Engineering, East China University of Science and Technology
Shanghai, 200237 - China
[2] Key Laboratory of Advanced Control and Optimization for Chemical Process, Ministry of Education
Shanghai, 200237 - China
[e-mail: llm361163@sohu.com, zhhling@ecust.edu.cn, happyxiaoyunzi@126.com]
*Corresponding author: Luming Liu

Received May 13, 2011; revised August 26, 2011; accepted September 21, 2011; published November 29, 2011

Abstract

Complementary trees are two spanning trees rooted at the sink node satisfying that any source node's two paths to the sink node on the two trees are node-disjoint. Complementary trees routing strategy is a special node-disjoint multi-path routing approach. Several complementary trees routing algorithms have been proposed, in which path discovery methods based on depth first search (DFS) or Dijkstra's algorithm are used to find a path for augmentation in each round of path augmentation step. In this paper, a novel path discovery method based on multi-tree-growing (MTG) is presented for the first time to our knowledge. Based on this path discovery method, a complementary trees routing algorithm is developed with objectives of low average path length on both spanning trees and low complexity. Measures are employed in our complementary trees routing algorithm to add a path with nodes near to the sink node in each round of path augmentation step. The simulation results demonstrate that our complementary trees routing algorithm can achieve low average path length on both spanning trees with low running time, suitable for wireless sensor networks in industrial scenarios.

Keywords: Node-disjoint multi-path routing, complementary trees, path discovery method, average path length, time complexity

The work was supported by the National High Technology Research and Development Program of China (863 Program), under Grants 2007AA041201-4 and 2009AA04Z144 and the Shanghai Leading Academic Discipline Project under Grant B504.

DOI: 10.3837/tiis.2011.11.010

1. Introduction

A reliable, low-delay and low-complexity routing algorithm is necessary to wireless sensor networks in process automation industry. Firstly, the nodes in harsh industrial environment may fail for some reasons such as electromagnetic interference, so there should be at least two node-disjoint paths from a source node to the sink node. Secondly, the sensory data should be transmitted to the sink node in low delay for some time-critical tasks, so the path length (hops) from a source node to the sink node should be as low as possible. Thirdly, the complexity of a practical routing algorithm should not be large. In industrial wireless sensor networks (IWSN), centralized multipath routing and TDMA scheduling should be used together to provide reliable and deterministic end-to-end delivery of sensory data, as required by WirelessHART which is a well-known wireless communication standard specifically designed for process measurement and control applications, ratified by the HART Communication Foundation in September 2007 [1].

Multipath routing (MPR) is an effective strategy to achieve reliability in which data can be transmitted over multiple paths [2][3][4]. To improve the transmission reliability and avoid shared-node failures, the multiple paths from a source node to a destination node in MPR can be selected to be node-disjoint [5][6][7][8][9].

In conventional multipath routing algorithms, the complete path information is embedded in every packet, or every intermediate node must maintain a routing table of considerable scale. To reduce the packet overhead and node routing-table overhead (hence reduce lookup time), complementary trees routing can be employed. The complementary trees denote two spanning trees (called blue tree and red tree respectively) rooted at the sink node satisfying that any source node's two paths to the sink node on the two trees are node-disjoint. The routing table at any node has only two entries. A packet transmitted from a source node is marked with one of the two colors. An intermediate node that receives the packet forwards it based on the color field in the packet. **Fig. 1** illustrates two complementary trees rooted at the sink node, where the tree with blue edges (red edges, respectively) is the blue tree (red tree, respectively).

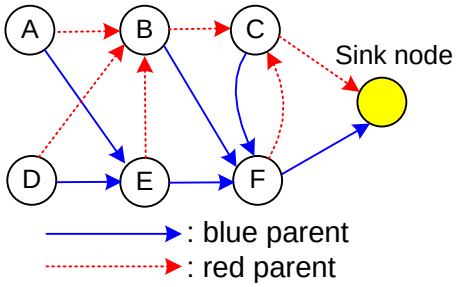


Fig. 1. An example of two complementary trees

1.1 Related Works

1.1.1 Path Augmentation Technique

Path augmentation technique [11] is a basic principle to construct two complementary trees for a two-node-connected network *G*, and the technique is used in all complementary trees routing algorithms. The path augmentation technique starts by choosing an arbitrary directed cycle in *G* with at least three nodes. If this cycle does not include all the nodes of *G*, then a path that

starts and ends on distinct nodes on the blue tree and that passes through k ($k \geq 1$) nodes not on the tree is chosen for augmentation. The path augmentation continues until all the nodes in the network are on the blue tree.

1.1.2 Path discovery method

As the key step of the path augmentation technique, the path discovery method determines the average path length performance and the complexity of the complementary trees algorithm. The existing complementary trees algorithms differ mainly in path discovery methods.

Medard et al. [11] presented the basic path augmentation technique, but they didn't explicitly discuss the path discovery method. And their algorithm selects a cycle and successive paths at random without considering the path length performance.

The linear time algorithms in [12][13][14] find paths successively using a DFS tree. In [15], depth first search from a node already on the blue tree is performed to find a path in each round of path augmentation step. The centralized version of the algorithm in [15] referred to as the RKK algorithm with time complexity $O(|N|(|N|+|L|))$ is used for comparison purpose in this paper. The algorithms in [12][13][14][15] can not find a path with nodes near to the sink node in each round of path augmentation step, so the average path length of node on the resulted complementary trees can be further reduced significantly.

Algorithm 5 in [16] is referred to as the XCT algorithm with time complexity $O(|N|^2(|N|\log|N|+|L|))$. In the XCT algorithm, a path with minimum total path length increase on blue tree is found for augmentation using Dijkstra's algorithm in each round of path augmentation step. The XCT algorithm only considers the path length minimization on the blue tree.

The BR algorithm in [17] improves the XCT algorithm in that it tries to minimize the path length on both blue tree and red tree. Dijkstra's algorithm is performed multiple times in each round of path augmentation step to find a path with minimum total path length increase on both trees. The BR algorithm is known to perform well in terms of average path length on both trees in existing complementary trees algorithms, but it has a high time complexity $O(|N|^2(|N|\log|N|+|L|))$, therefore the algorithm will cost large running time, especially when the network size is large. If breadth first search (BFS) is used as the path discovery method in the BR algorithm, its time complexity is still $O(|N|^2(|N|+|L|))$.

1.2 Features of our work

In this paper, we propose a complementary trees algorithm with objectives of low average path length on both trees and low time complexity. A novel path discovery method called multi-tree-growing (MTG) is designed, which finds a path in a different way from existing path discovery methods. The algorithms in [16][17] are greedy methods in that they find a path with minimum total path length increase in each round of path augmentation step, while our algorithm attempts to add the path with nodes near to the sink node.

The contributions of this paper are summarized as follows:

1) A novel path discovery method is developed and used in our complementary trees algorithm, making the latter perform well in both average path length and time complexity.

2) To augment a path with nodes near to the sink node, the traditional path augmentation technique is improved in that a cycle containing the sink node can be added in any round of path augmentation step, and multiple trees grow in level order to find a path for augmentation in each round of path augmentation step.

3) The work in [11] stated that a network must be two-node-connected to obtain two

complementary trees, while we make the supplement in this paper that our complementary trees algorithm also works for a special kind of one-node-connected network.

The rest of the paper is organized as follows. Section 2 gives the network model and problem definition. Section 3 presents our complementary trees routing algorithm. Section 4 gives the theoretical analysis of our algorithm. Section 5 presents the simulation results and describes the performance comparison of our algorithm with existing complementary trees algorithms. The conclusions are presented in Section 6.

2. Problem Formulation

Consider a two-node-connected network $G(N, L, s)$ composed of a set of nodes N and a set of links L, where $s \in N$ is the sink node. A link exists between two nodes if they are in the receiving range of each other.

Problem definition. For the network $G(N, L, s)$, the goal is to construct two complementary trees B and R (called blue tree and red tree, respectively) rooted at s with minimum average path length from a source node to s on both trees such that the node-disjoint path constraint is satisfied.

Let P_u^B (P_u^R, respectively) denote the directed path from a source node u to s on B (R, respectively).

Node-disjoint path constraint: if the path from u to s on the blue tree traverses node v, then the path from u to s on the red tree does not traverse node v, i.e.,

$$\forall u \in N \setminus \{s\} \text{ and } \forall v \in N \setminus \{s, u\} \quad v \in P_u^R \Rightarrow v \notin P_u^B \tag{1}$$

For a source node u, u's path to the sink node on B (R, respectively) is called u's blue path (red path, respectively), and u's next hop node in its blue path (red path, respectively) is called its blue parent (red parent, respectively). Dual path of a source node denotes its blue path and red path together.

Integer linear program (ILP) can be used to obtain an optimal solution to the above question only for small-scale networks, its application to large-scale networks is impractical due to its prohibitive computational time [15]. Therefore, we designed a heuristic approach to solve the above question in this paper.

3. Low-delay and Low-complexity Complementary Trees Algorithm

Our complementary trees algorithm is called MTG algorithm for short. We first give some definitions used in the algorithm. We called the path or cycle discovered in a path augmentation step an ear, in other words, an ear is a path whose end nodes lie on the blue tree (or red tree) but whose internal nodes do not. If an ear has two different end nodes, the ear is called a normal ear. If both of an ear's two end nodes are the sink node s, the ear is called a sink ear.

Table 1 presents some notations used in the algorithm. The information of all nodes is maintained in a one-dimensional array D and the array index for each node is used as the node's identification number (ID). C_2 denotes the IDs of nodes on the blue tree (or red tree) and C_x denotes the complement of C_2 with respect to the universal set of N. In the following paragraphs, "a node in C_2 (C_x, respectively)" means that the node's ID is in C_2 (C_x, respectively).

Table 1. Notations used in the MTG algorithm

Notation	Type	Comment
D	node array	stores information for all nodes of N
C_2	node ID list	stores the IDs of nodes on the blue tree
C_x	node ID list	complement of C_2 with respect to the universal set of N
s, u, v, w	node ID	s is the sink node's ID
D[u].con	integer	state of node u
D[u].root	node ID	root node of the 1-connected tree containing node u
D[u].parent	node ID	temporary parent node of node u
D[u].nbrlist	node ID list	u's neighbor list

3.1 Working of the Algorithm

The MTG algorithm is presented in **Fig. 2**, it works as follows:

1) In the initial stage (line 1 - line 7), the blue tree and red tree only contain the sink node *s*.
2) Perform a round of multi-tree-growing procedure (line 9 - line 25) to find an ear (sink ear or normal ear) and then add the ear to the blue tree and red tree.
3) If the blue tree includes all nodes in *G*, the algorithm stops, else goes to step 2.

In the initial stage, all nodes excluding *s* are stored in C_x list in level order so that the nodes with low levels (near to the sink node) can be processed first in each round of multi-tree-growing procedure, and each node's neighbor nodes are recorded in its neighbor list in level order too. Breadth first search (BFS) is used to determine all nodes' levels where the level of the sink node *s* is 0.

The multi-tree-growing procedure is performed repeatedly in the MTG algorithm. During each round of multi-tree-growing procedure, nodes in C_x connect with the blue tree in level order and therefore multiple trees grow level by level, when two different trees meet at a node *u* for the first time, *u*'s upstream paths in the two trees form an ear. Either a sink ear or a normal ear can be found in each round of multi-tree-growing procedure if the blue tree does not include all nodes of the network, and a sink ear can be found in the first round of multi-tree-growing procedure, to be proved in section 4. After an ear is found, it is augmented to the blue tree and red tree in the way specified in the following ear augmentation rule, and the voltage values will be assigned to the nodes of the ear according to the following node voltage rule.

The ear discovery and augmentation procedure above will be performed repeatedly until all nodes of the network are on the blue tree. The flowchart of the algorithm is shown in **Fig. 3**. Some algorithmic details are specified as follow.

3.1.1 Node State Rule

Every node can be in one of the following three states: 0-connected, 1-connected and 2-connected. At the beginning of the algorithm, the sink node *s* is set 2-connected and stored in C_2, while all other nodes are set 0-connected and stored in C_x. During a multi-tree-growing procedure, the 0-connected nodes in C_x are processed and become 1-connected (configured with a temporary parent) with the blue tree in level order, and the other nodes unprocessed in C_x are still 0-connected. We define a 1-connected tree as a tree satisfying the following two constraints: 1) the root of the tree is either a 2-connected node (excluding *s*) in C_2 or a 1-connected node in C_x which is a neighbor of *s*, 2) the other nodes of the tree are 1-connected nodes in C_x. As the above process continues, multiple 1-connected trees grow level by level. When two different 1-connected trees meet at a node *u* for the first time, *u*'s upstream paths in

the two 1-connected trees form an ear. Then the 1-connected nodes of the ear (called ear nodes for short) will be configured with blue parent and red parent and set 2-connected, and these ear nodes will be deleted from C_x and added into C_2. At the end of the multi-tree-growing procedure, all 1-connected nodes left in C_x will be set 0-connected (line 25) ready for the next round of multi-tree-growing procedure.

```
MTG algorithm
1  for each node u in N
2    D[u].con=0;  D[u].root= Null;  D[u].parent= Null;
3  D[s].con=2;  D[s].root= s;  D[s].parent= Null;
4  C2 = {s};
5  BFS is used to arrange all node IDs excluding s in Cx in level order;
6  for each node u in N
7    sort D[u].nbrlist in level order;
8  if (there are nodes left in Cx )
9    for each node u in Cx
10     for each neighbor node v of node u
11       if (v=s)
12         D[u].con=1;  D[u].root= u;  D[u].parent= s;
13       else if (D[v].con>0 and D[v].root != D[u].root)
14         D[u].con++;
15         if(D[u].con=1)
16           D[u].root=D[v].root;  D[u].parent=v;
17         else if (D[u].con=2)
18           construct an ear using u and v's upstream paths;
19           for each 1-connected node e in the ear
20             D[e].con=2;  D[e].root=e;
21             set blue parent and red parent for e by ear augmentation rule;
22           delete the ear nodes from Cx and insert them into C2 by node voltage rule;
23           reassign each node's new list index in C2 to the node's voltage value;
24           for each node w left in Cx
25             D[w].con=0;  D[w].root= Null;  D[w].parent= Null;
26           goto 8;
27 exit;
```

Fig. 2. The MTG algorithm to construct two complementary trees

Note that the two end nodes of an ear are 2-connected nodes while the internal nodes of an ear are 1-connected nodes. The nodes in C_2 are 2-connected, and we use C_0, C_1 to denote the 0-connected and 1-connected nodes, then they satisfy the following constraint:

$$\bigcup_{0\le i\le 2} C_i = N,\ C_i \bigcap C_j = \varnothing \text{ and } C_x = C_0 \bigcup C_1 \quad i \ne j \text{ and } 0 \le i, j \le 2 \tag{2}$$

3.1.2 Ear Augmentation Rule

When a sink ear $(s,v_1,\ldots,v_k,s)$ is found in a multi-tree-growing procedure where $k \ge 2$, for any node $v_i \in \{v_1,v_2,\ldots,v_k\}$, if $i<k$, v_i's blue parent is set to v_{i+1}, else v_i's blue parent is set to s. And if $i>1$, v_i's red parent is set to v_{i-1}, else v_i's red parent is set to s.

When a normal ear $(x,v_1,\ldots,v_k,y)$ is found in a multi-tree-growing procedure where $x, y \in C_2$, $k \ge 1$ and $x \prec y$ ("$x \prec y$" means that node x lies before node y in C_2.), for any node $v_i \in \{v_1,v_2,\ldots,v_k\}$, if $i<k$, v_i's blue parent is set to v_{i+1}, else v_i's blue parent is set to y. And if $i>1$,

v_i's red parent is set to v_{i-1}, else v_i's red parent is set to x.

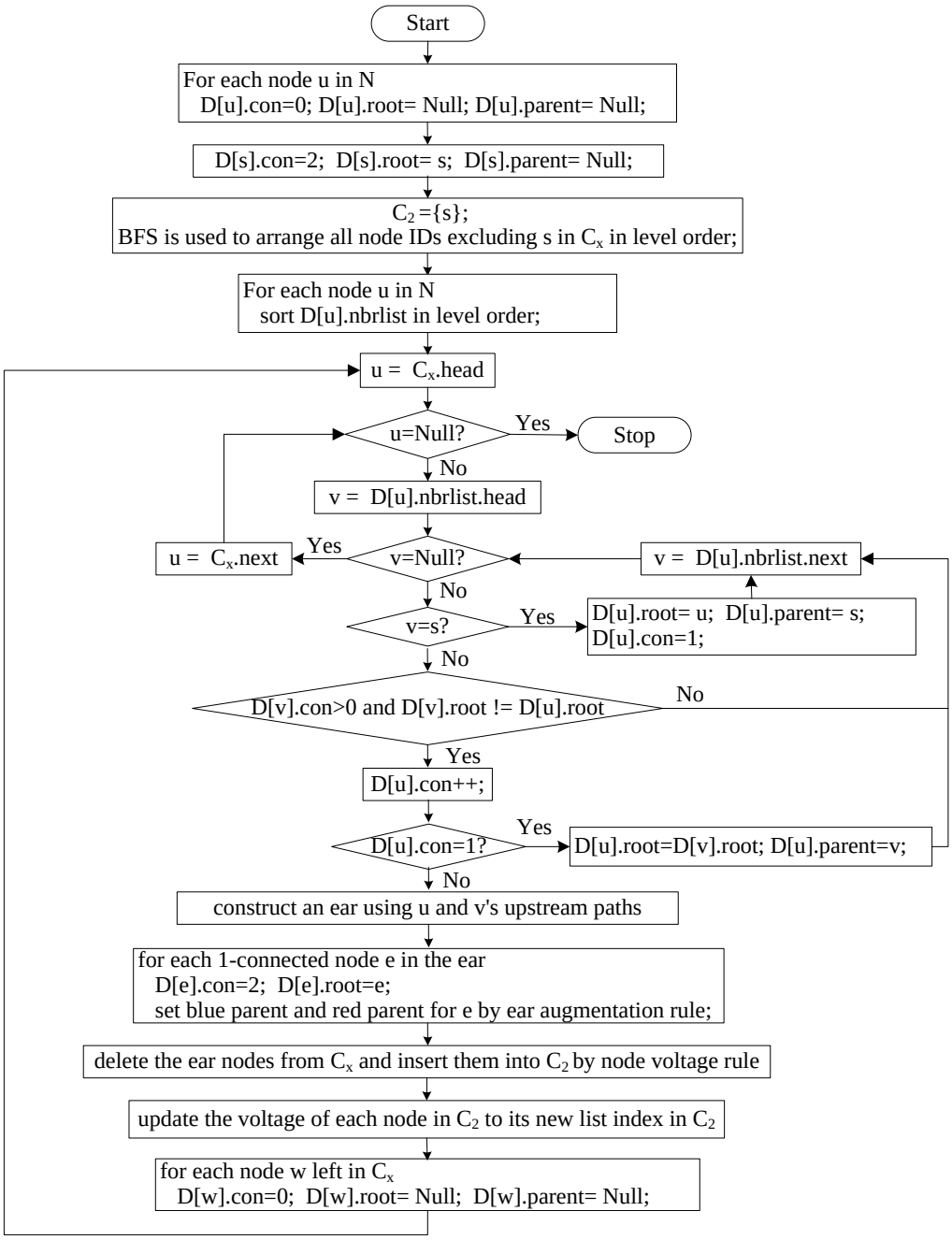


Fig. 3. The flowchart of the MTG algorithm

3.1.3 Node Voltage Rule

The list index of a node ID in C_2 is used as the voltage value for the node. If a sink ear $(s,v_1,\ldots,v_k,s)$ is found where $s \in C_2$, the ear nodes $(v_1,v_2,\ldots,v_k)$ will be deleted from C_x and added at the list tail of C_2 in sequence, and the algorithm will then update the voltage value of each node in C_2 to its new list index.

If a normal ear $(x,v_1,\ldots,v_k,y)$ is found where $x,y \in C_2$ and $x \prec y$, the ear nodes $(v_1,v_2,\ldots,v_k)$ will be deleted from C_x and inserted before y in C_2 sequentially, and the algorithm will then update the voltage value of each node in C_2 to its new list index.

The working of the MTG algorithm is illustrated on an example network shown in **Fig. 4**. The numbers in parenthesis represent the levels of nodes while those outside the parenthesis represent the node IDs. In **Fig. 4** (a), a sink ear $\{s,1,2\}$ is found and augmented. From **Fig. 4** (b) to **Fig. 4** (c), two 1-connected trees $\{1,3,4,7,8\}$ and $\{2,5,6\}$ meet at node 9 and a normal ear $\{1,3,8,9,5,2\}$ is found and augmented. Node 4, node 6 and node 7 will then be augmented in turn. The two complementary trees constructed at the end of the algorithm are shown in **Fig. 4** (d).

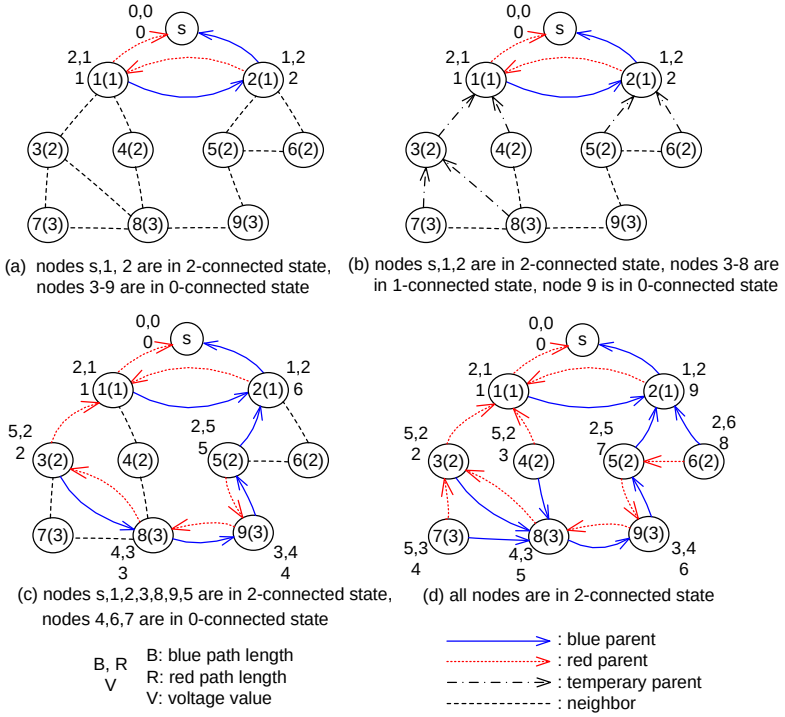


Fig. 4. An example showing the working of the MTG algorithm

4. Theoretical Analysis

A Type-A 1-connected tree is defined as a 1-connected tree whose root is a 1-connected node in C_x which is a neighbor of s. A Type-B 1-connected tree is defined as a 1-connected tree whose root is a 2-connected node excluding s in C_2. We analyze the algorithm as follows.

Theorem 1. A sink ear can be found in the first round of multi-tree-growing procedure. Either a sink ear or a normal ear can be found in each round of multi-tree-growing procedure if there are nodes in C_x.

Proof. In the initial stage, $C_2=\{s\}$, $C_x = N\backslash\{s\}$, and all nodes in C_x are set 0-connected. In the first round of multi-tree-growing procedure, the 0-connected nodes in C_x will become 1-connected with the blue tree (or red tree) level by level, therefore multiple Type-A 1-connected trees grow in the nodes of C_x. Because the network is two-node-connected, at least two Type-A 1-connected trees will meet. When two Type-A 1-connected trees meet at a

node *u* for the fist time, a sink ear appears which is composed of *u*'s upstream paths in the two trees and *s*.

Suppose that an ear has been augmented in the *nth* ($n>1$) round of multi-tree-growing procedure and there are still nodes left in C_x, let's consider the *(n*+1*)th* round of multi-tree-growing procedure. At the beginning of the *(n*+1*)th* round multi-tree-growing procedure, all nodes left in C_x are 0-connected (set at the end of the last round of multi-tree-growing procedure). The 0-connected nodes in C_x will become 1-connected with the blue tree (or red tree) level by level, therefore, multiple 1-connected trees grow in the nodes of C_x. Because the network is two-node-connected, at least two 1-connected trees will meet, and there are possibly three cases as follows.

Case 1: If two Type-A 1-connected trees meet at a 0-connected node *u* for the fist time, a sink ear appears which is composed of *u*'s upstream paths in the two trees and *s*.

Case 2: If two Type-B 1-connected trees meet at a 0-connected node *u* for the fist time, a normal ear appears which is composed of *u*'s upstream paths in the two trees.

Case 3: If a Type-A 1-connected tree and a Type-B 1-connected tree meet at a 0-connected node *u* for the fist time, a normal ear appears which is composed of *u*'s upstream paths in the two trees and *s*.

Therefore, either a normal ear or a sink ear can be found in the *(n*+1*)th* round of multi-tree-growing procedure. By induction, an ear can be found in each round of multi-tree-growing procedure until there are no nodes left in C_x. □

Theorem 2. The time complexity of the MTG algorithm is $O(|N|(|N|+|L|))$.

Proof. In the initial stage, the operations in line 2 cost time $O(|N|)$, the operations in line 5 cost time $O(|N|+|L|)$ and the operations in line 7 cost time $O(|N||L|)$.

The operations from line 11 to line 16 are used to visit one of *u*'s neighbors and these operations can finish in constant time. In a multi-tree-growing procedure, the neighbor list of each node in C_x is traversed at most once. Since the sum of the lengths of all nodes' neighbor lists is $O(L)$, the operations from line 11 to line 16 are performed at most $O(L)$ times in a multi-tree-growing procedure. According to Theorem 1, the multi-tree-growing procedure will be executed $O(|N|)$ times since each round of the procedure deletes at least one node from C_x. As a result, the time complexity for operations from line 11 to line 16 is $O(|N||L|)$.

The operations from line 18 to line 25 are used to process an ear after it is found in a multi-tree-growing procedure. The running time of the operation in line 18 is $O(|N|)$ since the length of an ear is $|N|$ at the most. Similarly, the running time of the operations in each line between 19 and 25 in one multi-tree-growing procedure is $O(|N|)$ respectively. Since the multi-tree-growing procedure is executed $O(|N|)$ times, the time complexity of the operations from line 18 to line 25 is $O(|N|^2)$.

Therefore, the time complexity of the MTG algorithm is $O(|N|(|N|+|L|))$. □

Suppose that the MTG algorithm terminates after *K* rounds of multi-tree-growing procedures, and we use $C_0(n)$, $C_1(n)$, $C_2(n)$ and $C_x(n)$ to denote C_0, C_1, C_2 and C_x after *n* rounds of multi-tree-growing procedures where $n \le K$, then they satisfy the following Theorem.

Theorem 3. For any integer *n* ($0 \le n \le K$), $C_x(n)$ is the complementary set of $C_2(n)$ with respect to the universal set of *N*, and $C_0(n)$, $C_1(n)$, $C_2(n)$ and $C_x(n)$ satisfy the following constraints:

$$C_1(n)=\varnothing \text{ and } C_x(n)=C_0(n) \tag{3}$$

$$C_2(K)=N \text{ and } C_x(K)=\varnothing \tag{4}$$

Proof. The theorem can be deduced easily from the node state rule and node voltage rule, so details are omitted here. □

Theorem 4. After n ($0 < n \le K$) rounds of multi-tree-growing procedures, $\forall u \in C_2(n) \setminus \{s\}$, node u has a directed path P_{us}^B (P_{us}^R, respectively) to s through the blue parent (red parent, respectively) attribute of node, and the path consists of nodes in $C_2(n)$ with monotonically increasing (decreasing, respectively) voltage values (excluding s).

Proof. The theorem can be deduced easily from Theorem 1, the ear augmentation rule and node voltage rule, so details are omitted here. □

The correctness of the MTG algorithm is proved in Theorem 5.

Theorem 5. The MTG algorithm terminates with a pair of complementary trees.

Proof. According to Theorem 1 and Theorem 2, the algorithm can terminate in finite time. When the algorithm terminates, a pair of complementary trees can be constructed, which can be proved in a similar way as in [11]. Specifically, when the algorithm terminates, let B and R denote the two spanning graphs using the blue parent and red parent attributes of nodes respectively. According to Theorem 4, B and R satisfy two constraints: 1) There is no cycle in B (R, respectively) for a cycle would imply that the voltage values of the nodes traversed would decrease and then increase, 2) B (R, respectively) is a connected graph of G. Therefore, B and R are both spanning trees rooted at s.

In the same time, for any node $u \in N$, u's path P_{us}^B in B is node-disjoint with u's path P_{us}^R in R, since P_{us}^B consists of monotonically increasing voltage values and P_{us}^R consists of monotonically decreasing voltage values excluding s. Therefore B and R are a pair of complementary trees. □

We analyze the path length performance of the algorithm as follows.

Theorem 6. In each round of multi-tree-growing procedure before the MTG algorithm terminates, for any 0-connected node u in level h ($u \in C_x$ and $h \ge 1$), when it is u's turn to be processed, u will become 1-connected with a node in level (h-1).

Proof. Without loss of generality, let's consider the *nth* ($n > 0$) round of multi-tree-growing procedure.

At the beginning of this multi-tree-growing procedure, all nodes in C_x are 0-connected and these nodes are arranged in node level order. Suppose that the minimum level number for the nodes in C_x is m ($m \ge 1$), so the nodes in level (m-1) are not in C_x and they must have been set 2-connected and stored in C_2. And because any node's neighbor list is arranged in level order, all nodes of level m in C_x will be processed and become 1-connected with the nodes of level (m-1).

Suppose that the nodes of level (m+i) in C_x have become 1-connected with nodes of level (m+i-1) in this round of multi-tree-growing procedure, then the nodes of level (m+i) not in C_x must have become 2-connected and stored in C_2, so all nodes of level (m+i) in N are connected. Since any node's neighbor list is arranged in level order, the nodes of level (m+i+1) in C_x will be processed and become 1-connected with the nodes of level (m+i). By induction, Theorem 6 holds true. □

Theorem 7. In each round of multi-tree-growing procedure before the MTG algorithm terminates, for any node u in a 1-connected tree T rooted at node r, let l_u and l_r denote the levels of u and r respectively, and d_u denotes the relative distance (hops) from node u to r in T, then l_u, l_r and d_u satisfy the following constraint.

$$l_u=l_r+d_u \quad u \neq r \tag{5}$$

Proof. Theorem 7 can be deduced easily from Theorem 6. □

Theorem 8. When two 1-connected trees T_1 and T_2 meet for the first time at node e in a multi-tree-growing procedure, let p_1 (p_2, respectively) denote e's upstream path in T_1 (T_2, respectively) while r_1 (r_2, respectively) denote the root of T_1 (T_2, respectively), then p_1 (p_2, respectively) is the shortest path from e to r_1 (r_2, respectively) in T_1 (T_2, respectively).

Proof. Let u ($u \in T_1$) denote e's neighbor node in p_1 and suppose that e has another neighbor node v in T_1 with shorter upstream path to r_1 than u's. Let p_u and p_v denote u and v's upstream paths in T_1 respectively, and d_u and d_v denote the lengths of p_u and p_v respectively, and l_u and l_v denote the levels of u and v respectively, then $d_u > d_v$. According to the ear construction method of the algorithm, u is the first node in e's neighbor list which lies on T_1, and because the nodes in e's neighbor list are arranged in level order, $l_u \leq l_v$. Therefore, $d_u \leq d_v$ by Theorem 7, contradicting with the previous assumption. As a result, p_1 is the shortest path from e to r_1 in T_1. Similarly, p_2 is the shortest path from e to r_2 in T_2. □

The work in [11] stated that a network must be two-node-connected to obtain two complementary trees, we make the supplement that the MTG algorithm also works for a special kind of topology called approximate two-node-connected network.

Definition: If a network is composed of n ($n \geq 2$) two-node-connected blocks and the sink node is the only common node for all blocks, we call this kind of network an approximate two-node-connected network.

Fig. 5 shows an example of an approximate two-node-connected network which is composed of three two-node-connected blocks {s,1,2,3,4}, {s,5,6,7} and {s,8,9,10}. An approximate two-node-connected network is in fact one-node-connected because the sink node is a cutting node for the network.

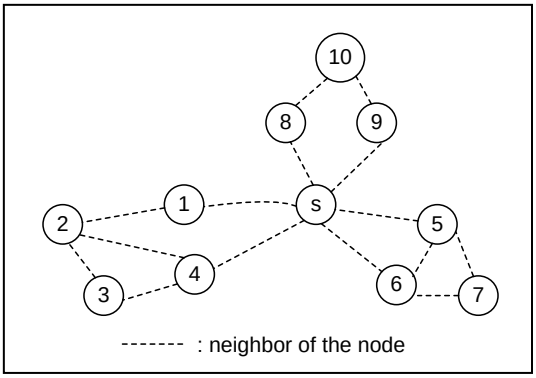


Fig. 5. An approximate two-node-connected network composed of 3 blocks

Theorem 9. The MTG algorithm can construct two complementary trees for an approximate two-node-connected network.

Proof. We consider an approximate two-node-connected network $P(N, L, s)$ composed of n ($n \geq 2$) two-node-connected blocks {B_1, B_2…B_n}.

1) *Two local complementary trees can form in each block of P.* In a round of multi-tree-growing procedure, multiple 1-connected trees grow in C_x continuously. Because all blocks of P are two-node-connected and they have no common node except s, at least two 1-connected trees can meet and therefore form an ear in a block before the multi-tree-growing procedure terminates. An ear can be found in a block during a multi-tree-growing procedure so

long as there are nodes in C_x. The ear discovery and augmentation procedure will happen alternatively and repeatedly in different blocks until two complementary trees form in each block of P.

2) *The union of all local complementary trees in each block forms two global complementary trees for P*. Let T_i^B (T_i^R, respectively) denote the blue subtree (red subtree, respectively) constructed in B_i ($1 \le i \le n$) which is rooted at s and spans all nodes of B_i, and let B (R, respectively) denote the union of T_i^B (T_i^R, respectively) for $1 \le i \le n$. According to the definition of B_i, T_i^B and T_i^R ($1 \le i \le n$), B (R, respectively) is a spanning tree for P. On the other side, for any node $u \in P$, u has two node-disjoint paths to s in the block containing u, therefore, B and R are complementary trees for P. □

5. Performance Evaluation

We demonstrate the effectiveness of the MTG algorithm by comparing the results to those obtained from the RKK, XCT and BR algorithms mentioned in section 1.

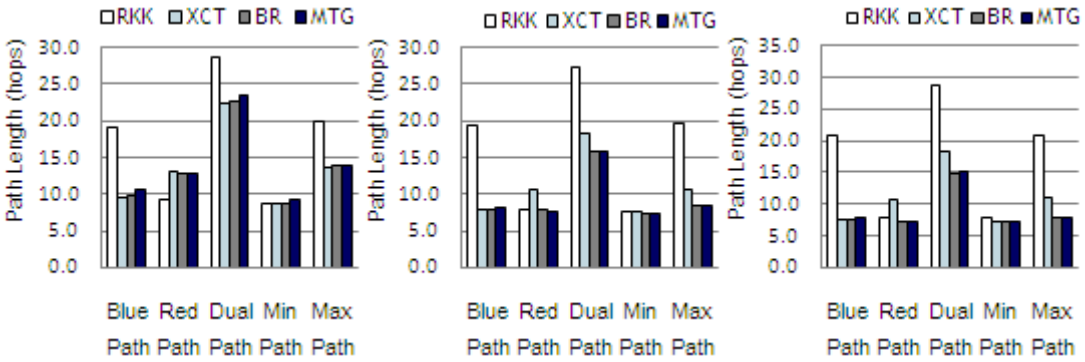
The performance metrics considered are: (1) algorithm solution time, (2) average blue/red/dual path length, (3) average minimum/maximum path length, (4) blue/red tree depth. The solution time of an algorithm is simply the running time of its program to obtain a solution. The average minimum (maximum, respectively) path length refers to the lowest (highest, respectively) path length among the blue path and red path, averaged over all nodes in the network.

Two kinds of network topologies are used to test these algorithms, random and grid. A link between two nodes exists if the distance between them is no more than $\sqrt{2}$ units. For random networks, we employ 100, 200 and 300 as the network size respectively, where N nodes are distributed in a square region of side 10 units. For grid topologies, 5×5, 10×10 and 15×15 networks are used where nodes are placed on a two-dimensional grid at integer coordinates. All networks generated are two-node-connected and the sink node is placed at the origin. We simulated these algorithms using a C program on a Pentium 4 desktop with 1 G memory. For each network size in random networks, 20 different topologies were simulated and the average results are reported.

The results of path length and tree depth for the four algorithms are shown in **Fig. 6-Fig. 9.** We can observe from **Fig. 6** and **Fig. 7** that the MTG and BR algorithms have similar path length performance, which is much better than that of the other two algorithms, especially in large-scale networks. The performance results from **Fig. 8** and **Fig. 9** follow similar trend, in other words, the MTG and BR algorithms perform well on both complementary trees in terms of tree depth, compared with the other two algorithms.

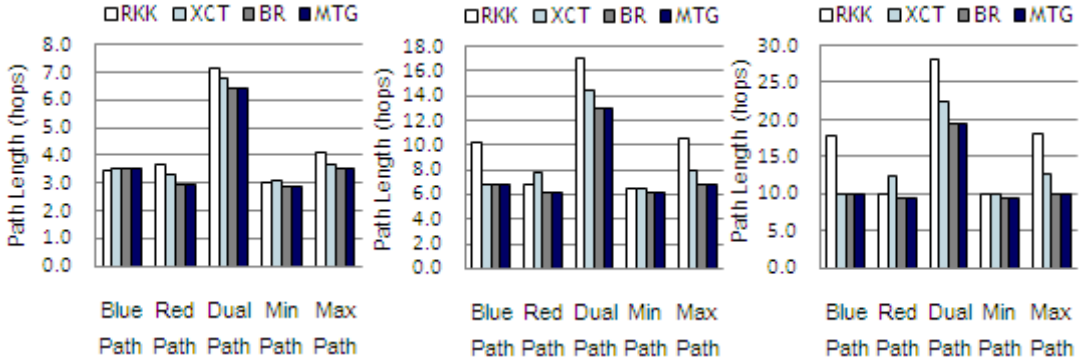
The results of running time for the four algorithms are shown in **Table 2**, where a random network with 100 (200 and 300, respectively) nodes is denoted as R-100 (R-200 and R-300, respectively). We can observe from **Table 2** that, the BR and XCT algorithms need to cost large time to construct two complementary trees especially in large-scale networks, while the MTG and RKK algorithms can finish well under 50 ms for all scenarios, so the MTG and RKK algorithms perform much better than the other two algorithms in terms of running time.

In summary, the MTG and BR algorithms have similar path length performance which is much better than that of the other two algorithms, but the MTG algorithm costs far less running time (under 50 ms for all scenarios) than the BR algorithm. Therefore, in the four algorithms, the MTG algorithm can perform well in both running time and average path length, suitable for industrial wireless sensor networks scenarios.



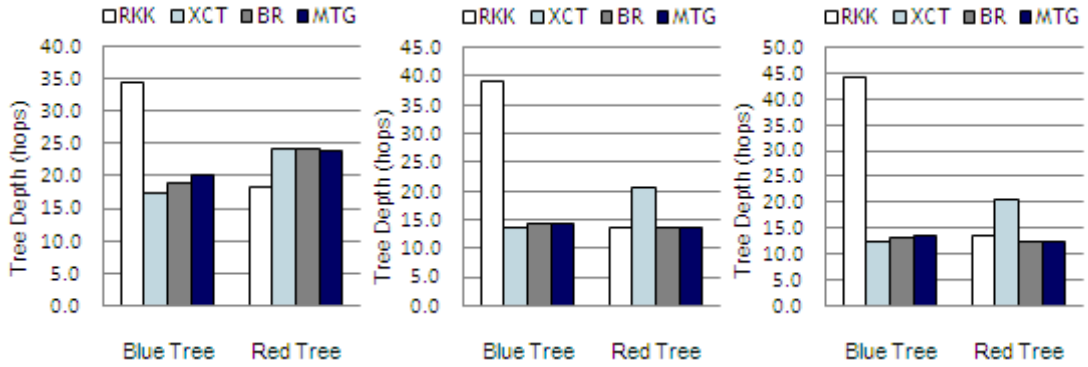
(a) 100-node network (b) 200-node network (c) 300-node network

Fig. 6. Average path length on random networks



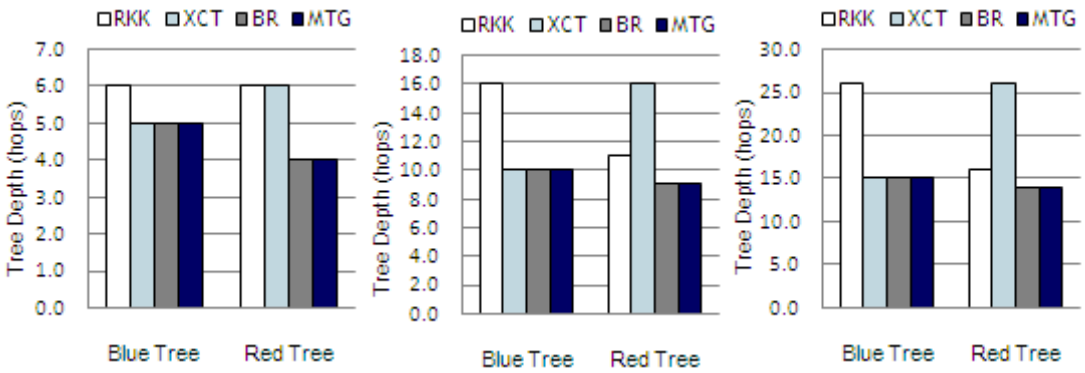
(a) 5×5 network (b) 10×10 network (c) 15×15 network

Fig. 7. Average path length on grid networks



(a) 100-node network (b) 200-node network (c) 300-node network

Fig. 8. Blue/red tree depth on random networks



(a) 5×5 network (b)10×10 network (c) 15×15 network

Fig.9. Blue/red tree depth on grid networks

Table 2. Running time (ms) of different complementary trees algorithms on random/grid networks

Network	Average solution time				Network	Average solution time			
	RKK	XCT	BR	MTG		RKK	XCT	BR	MTG
R-100	<1	594.50	427.70	7.05	**5×5**	<1	31.90	31.50	1.00
R-200	2.15	8560.90	7867.00	18.60	**10×10**	<1	1152.25	1155.30	6.25
R-300	6.55	37604.20	35495.85	38.35	**15×15**	1.00	10920.55	10938.65	20.60

6. Conclusion

In this paper, we proposed a complementary trees routing algorithm (MTG algorithm) with time complexity $O(|N|(|N|+|L|))$. Effective measures have been taken to reduce the average path length, hence reduce the data delivery delay for nodes. Compared with existing known complementary trees algorithms, the MTG algorithm can perform well in both running time and average path length especially in large-scale networks, as evidenced by the simulation. In addition, the MTG algorithm can also work for some special one-node-connected networks, as demonstrated in the theoretical analysis. These advantages make the algorithm practical for industrial wireless sensor networks.

With regard to the known wireless communication standards for industrial automation such as WirelessHart, we further envisage that the complementary trees approach can also be used to facilitate the following works:

1) TDMA scheduling scheme. For example, the transmitting activities of nodes can be scheduled level by level, or tree by tree.

2) Parallel data delivery. For example, two nodes in different levels can transmit simultaneously to enhance the spatial reuse of the network, hence reduce the data delivery delay.

We will conduct these works in our future work.

References

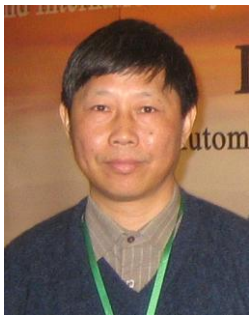
[1] HART Communication Foundation, “WirelessHART,” http://WirelessHART.hartcomm.org/, Feb. 2010.

[2] R.C. Shah, J.M. Rabaey, “Energy Aware Routing for Low Energy Ad Hoc Sensor Networks,” in *Proc. of IEEE Wireless Communications and Networking Conference*, pp. 350-355, Mar. 17-21, 2002. Article (CrossRef Link)

[3] X.X. Huang, Y.G. Fang, “Multiconstrained QoS Multipath Routing in Wireless Sensor Networks,” *Wireless Networks*, vol. 14, no. 4, pp. 465-478, Aug. 2008. Article (CrossRef Link)

[4] Y.S. Chen, T.Y. Juang, Y.W. Lin, I.C. Tsai, "A Low Propagation Delay Multi-path Routing Protocol for Underwater Sensor Networks," *Journal of Internet Technology*, vol. 11, no. 2, pp. 153-165, Mar. 2010.
[5] Y.M. Lu, V.W.S. Wong, "An Energy-Efficient Multipath Routing Protocol for Wireless Sensor Networks," *International Journal of Communication Systems*, vol. 20, no. 7, pp. 747-766, Jul. 2007. Article (CrossRef Link)
[6] S.U. Rehman, W.C. Song, G.L. Park, "Associativity-Based On-Demand Multi-Path Routing in Mobile Ad Hoc Networks," *KSII Transactions on Internet and Information Systems*, vol. 3, no. 5, pp. 475-491, Oct. 2009. Article (CrossRef Link)
[7] S. Oh, D. Kim, H. Kang, H.J. Jeong, "SMSR: A Scalable Multipath Source Routing Protocol for Wireless Sensor Networks," in *Proc. of 6th International Conference on Ubiquitous Intelligence and Computing*, pp. 121-135, Jul. 07-09, 2009. Article (CrossRef Link)
[8] K. Xiong, Z.D. Qiu, Y.C. Guo, H.K. Zhang, "Multi-Constrained Shortest Disjoint Paths for Reliable QoS Routing," *ETRI Journal*, vol. 31, no. 5, pp. 534-544, Oct. 2009. Article (CrossRef Link)
[9] L. Shu, Y. Zhang, Z.W. Yu, L.T. Yang, M. Hauswirth, N. Xiong, "Context-Aware Cross-Layer Optimized Video Streaming in Wireless Multimedia Sensor Networks," *Journal of Supercomputing*, vol. 54, no. 1, pp. 94-121, Oct. 2010. Article (CrossRef Link)
[10] A.M. Abbas, B.N. Jain, "Path Diminution in Node-Disjoint Multipath Routing for Mobile Ad Hoc Networks is Unavoidable with Single Route Discovery," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 5, no. 1, pp. 7-21, 2010. Article (CrossRef Link)
[11] M. Medard, S.G. Finn, R.A. Barry, R.G. Gallager, "Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs," *IEEE-ACM Transactions on Networking*, vol. 7, no. 5, pp. 641-652, Oct. 1999. Article (CrossRef Link)
[12] W.Y. Zhang, G.L. Xue, J. Tang, K. Thulasiraman, "Linear Time Construction of Redundant Trees for Recovery Schemes Enhancing QoP and QoS," in *Proc. of 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 2702-2710, Mar. 13-17, 2005. Article (CrossRef Link)
[13] W.Y. Zhang, G.L. Xue, J. Tang, K. Thulasiraman, "Faster Algorithms for Construction of Recovery Trees Enhancing QoP and QoS," *IEEE-ACM Transactions on Networking*, vol. 16, no. 3, pp. 642-655, June 2008. Article (CrossRef Link)
[14] S. Ramasubramanian, M. Harkara, M. Krunz, "Linear Time Distributed Construction of Colored Trees for Disjoint Multipath Routing," *Computer Networks*, vol. 51, no. 10, pp. 2854-2866, Jul. 2007. Article (CrossRef Link)
[15] S. Ramasubramanian, H. Krishnamoorthy, M. Krunz, "Disjoint Multipath Routing using Colored Trees," *Computer Networks*, vol. 51, no. 8, pp. 2163-2180, June 2007. Article (CrossRef Link)
[16] G.L. Xue, L. Chen, K. Thulasiraman, "Quality-of-Service and Quality-of-Protection Issues in Preplanned Recovery Schemes using Redundant Trees," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1332-1345, Oct. 2003. Article (CrossRef Link).
[17] R. Balasubramanian, S. Ramasubramanian, "Minimizing Average Path Cost in Colored Trees for Disjoint Multipath Routing," in *Proc. of 15th International Conference on Computer Communications and Networks*, pp. 185-190, Oct. 09-11, 2006. Article (CrossRef Link).

Luming Liu is currently working towards his Ph.D. degree at School of Information Science and Engineering, East China University of Science and Technology, China. His research interests are in the areas of wireless networks in industrial automation.



Zhihao Ling is a professor at East China University of Science and Technology (ECUST), China. He received his Ph.D. degree in Control Science and Engineering from ECUST in 2005. He is an executive director of Process Control and Instrumentation technical committee of China Instrument and Control Society and a director of Shanghai Association of Automation. His research interest is networked measurement and control system in industrial automation.

Yun Zuo is currently working towards her Ph.D. degree at School of Information Science and Engineering, East China University of Science and Technology, China. Her research interests include routing and scheduling methods in industrial wireless sensor networks.