The Orphan Problem in ZigBee Wireless Networks

Meng-Shiuan Pan, Chia-Hung Tsai, and Yu-Chee Tseng, *Senior Member, IEEE*

***Abstract*—ZigBee is a communication standard which is considered to be suitable for wireless sensor networks. In ZigBee, a device (with a permanent 64-bit MAC address) is said to join a network if it can successfully obtain a 16-bit network address from a parent device. Parent devices calculate addresses for their child devices by a distributed address assignment scheme. This assignment is easy to implement, but it restricts the number of children of a device and the depth of the network. We observe that the ZigBee address assignment policy is too conservative, thus usually making the utilization of the address pool poor. Those devices that can not receive network addresses will be isolated from the network and become *orphan* nodes. In this paper, we show that the *orphan problem* can be divided into two subproblems: the *bounded-degree-and-depth tree formation (BDDTF)* problem and the *end-device maximum matching (EDMM)* problem. We then propose algorithms to relieve the orphan problem. Our simulation results show that the proposed schemes can effectively reduce the number of orphan devices compared to the ZigBee strategy.**

***Index Terms*—graph theory, IEEE 802.15.4, network formation, orphan problem, wireless sensor network, ZigBee.**

I. Introduction

The recent progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks (WSNs)* feasible. A lot of research works have been dedicated to this area, including energy-efficient MAC protocols [11][27], routing and transport protocols [8][13], self-organizing schemes [16][24], sensor deployment and coverage issues [14][22], and localization schemes [6][23]. Applications of WSNs include habitat monitoring [2], wildfire monitoring [1], mobile object tracking [21][25], and navigation [20][26].

Recently, several WSN platforms have been developed, such as MICA, MICAz, Imote2, TelosB [4], TI CC2431 [5], and Jennic JN5121 [3]. For interoperability purpose, most platforms have adopted ZigBee [29] as their communication protocols. ZigBee adopts IEEE 802.15.4 standard [15] as its physical and MAC protocols and solves the interoperability issues from the physical layer to the application layer.

ZigBee supports three kinds of network topologies, namely star, tree, and mesh networks. A *ZigBee coordinator* is responsible for initializing, maintaining, and controlling the network. In a star network, all devices have to directly connect to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network is formed by one ZigBee coordinator and multiple *ZigBee routers*. A device can join a network as an *end device* by associating with the coordinator or a router. Fig. 1 shows a ZigBee tree network.

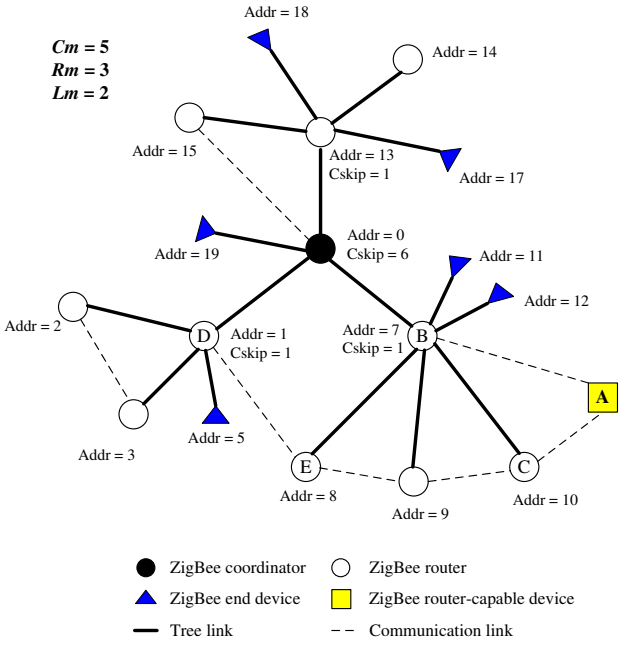


Fig. 1. An example ZigBee tree network.

In ZigBee, each node has a permanent 64-bit MAC address. A device is said to successfully join a network if it can obtain a 16-bit network address from the coordinator or a router. Using a short network address is for simplicity and for saving communication bandwidths. Before forming a network, the coordinator needs to decide three important system parameters: the maximum number of children of a router (Cm), the maximum number of child routers of a router (Rm), and the depth of the network (Lm). Note that a child of a router can be a router or an end device, so $Cm \geq Rm$. Given Cm, Rm and Lm, ZigBee has suggested a distributed address assignment scheme. While simple, the scheme may prohibit a node from accepting a child router/device as constrained by these parameters. We say that a node becomes an *orphan node* when it can not associate with any parent router but there are still unused address spaces remaining. We call this the *orphan problem*. For example, in Fig. 1, the router-capable device A has two potential parents B and C. Router B can not accept A as its child because it has reached its maximum capacity of $Cm = 5$ children. Router C can not accept A either because it has reached the maximum depth of $Lm = 2$. So A will become an orphan node. The orphan problem will worsen as

Manuscript received September 21, 2007; revised November 7, 2008.

M.-S. Pan is with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan. E-mail: mspan@cs.nctu.edu.tw

C.-H. Tsai is with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan. E-mail: chiahung@cs.nctu.edu.tw

Y.-C. Tseng is with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, and Department of Information and Computer Engineering, Chung-Yuan Christian University, Chung-Li, 32023, Taiwan. E-mail: yctseng@cs.nctu.edu.tw

the network scares up. We will further support this claim in Section II-B through simulations and real experiments. The orphan problem can be relieved if proper actions are taken. For example, in Fig. 1, if router E is connected to router D, router B will have capacity to accept A.

Given Cm, Rm, and Lm, we show that the orphan problem can be divided into two subproblems: 1) connecting as many routers as possible to form a tree and 2) connecting as many end devices as possible to the above tree. The first subproblem involves the router-capable devices only and can be modeled as a *bounded-degree-and-depth tree formation (BDDTF)* problem. We prove that this subproblem is in fact NP-complete. The second subproblem needs to connect as many end devices to the above tree as possible constrained by router's capacities and can be modeled as an *end-device maximum matching (EDMM)* problem. We prove that the EDMM problem is computationally feasible and then exist an optimal algorithm to solve it. To summarize, our approach involves two stages. The first stage will try to relieve the BDDTF problem by connecting more routers. Based on the result, the second stage will be able to connect the largest number of end devices to the tree.

Several works have investigated the *bounded-degree spanning tree* problem. Reference [10] proposes polynomial-time graph algorithms when additional connectivity and maximum degree of a graph are given. However, the depth constraint is not considered. Reference [18] introduces an approximation algorithm, which can find a spanning tree with a maximum degree of $O(K+log|V|)$, where K is the degree constraint and V is the set of nodes in the graph. The result is not applicable to our case because it does not consider the depth constraint and the number of children of a node is not bounded. In [17], a polynomial time algorithm is proposed to construct a spanning tree with a bounded degree and a bounded diameter. However, this algorithm is designed for complete graphs, which is not the case in a ZigBee network. Also, these works are not tailored to ZigBee specifications. Some works have focused on address configuration. Reference [7] proposes a network address assignment scheme based on the address assignment rule for an n-dimensional hypercube. Interestingly, when the ZigBee network structure is close to an n-cube, this scheme can indeed reduce the waste of address space. However, in practice, a WSN is typically randomly deployed on a 2D plane, which is unlikely to be similar to a high-dimensional n-cube. In fact, the scheme still suffers from the compatibility issue when the n-cube is incomplete and the orphan problem may still exist. Besides, additional overhead will be incurred to ensure that no duplicate addresses are assigned to nodes. Reference [19] organizes a network into concentric tiers around the sink and does not employ unique per-node addressing. When transmitting, a node will randomly choose an identifier for one-hop routing. This scheme is address-light, but it is only suitable for reporting scenarios and can not support point-to-point routing. In [28], an adaptive block addressing scheme is introduced for network auto-configuration purpose. It takes into account the actual network topology and thus is fully topology-adaptive. However, because the size of the address pool allocated by the coordinator is depended on the topology, addition of new nodes can cause the whole network to conduct address update. Moreover, this scheme needs two phases to initialize its adaptive tree, which is different from the ZigBee association procedure and is thus not compatible with ZigBee.

The main contributions of this paper are threefold. First, this is the first work that points out the orphan problem in ZigBee wireless networks. Second, we show that the existence orphan is an inherent concern no matter how one sets the Cm, Rm, and Lm constraints. We verify this claim through different configurations and parameter settings. A larger Cm or Rm will impose more memory requirement on routers and packets, while a larger Lm will also induce longer network delays. Third, we connect the orphan problem to NP-complete and classical algorithms and then propose network formation heuristics that can effectively reduce the number of orphans with given Cm, Rm, and Lm.

The rest of this paper is organized as follows. Preliminaries are given in Section II. Section III and Section IV present our algorithms. Simulation results are given in Section V. Finally, Section VI concludes this paper.

II. Preliminaries

A. Overview of IEEE 802.15.4 and ZigBee Standards

IEEE 802.15.4 [15] specifies the physical and data link protocols for *low-rate wireless personal area networks (LR-WPAN)*. In the physical layer, there are three frequency bands with 27 radio channels. Channel 0 ranges from 868.0 MHz to 868.6 MHz, which provides a data rate of 20 kbps. Channels 1 to 10 work from 902.0 MHz to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located from 2.4 GHz to 2.4835 GHz, each with a data rate of 250 kbps.

IEEE 802.15.4 devices are expected to have limited power, but need to operate for a longer period of time. Therefore, energy conservation is a critical issue. Devices are classified as *full function devices* (FFDs) and *reduced function devices* (RFDs). IEEE 802.15.4 supports star and peer-to-peer topologies. In each PAN, one device is designated as the *coordinator*, which is responsible for maintaining the network. A FFD has the capability of serving as a coordinator or associating with an existing coordinator/router and becoming a router. A RFD can only associate with a coordinator/router and can not have children.

According to ZigBee standard [29], a ZigBee network is formed by the following procedures. Devices that are coordinator-capable and do not currently join a network can be a candidate of a ZigBee coordinator. A device that desires to be a coordinator will scan all channels to find a suitable one. After selecting a channel, this device broadcasts a beacon containing a PAN identifier to initialize a PAN. A device that hears a beacon of an existing network can join this network by performing the association procedures and specifying its role, as a ZigBee router or an end device. If the device hears multiple beacons, it chooses the beacon sender with the smallest hop count to the coordinator. The beacon sender will determine whether to accept this device or not by considering its current capacity and its permitted association duration. If

the device is successfully associated, the association response will contain a short 16-bit address for the request sender. This short address will be the network address for that device.

In ZigBee, network addresses are assigned to devices by a distributed address assignment scheme. The coordinator determines Cm, Rm, and Lm. The coordinator and each router can have at most Rm child routers and at least $Cm - Rm$ child end devices. Devices' addresses are assigned in a top-down manner. For the coordinator, the whole address space is logically partitioned into $Rm+1$ blocks. The first Rm blocks are to be assigned to the coordinator's child routers and the last block is reserved for the coordinator's own child end devices. From Cm, Rm, and Lm, each node computes a parameter called $Cskip$ to derive the starting addresses of its children's address pools. The $Cskip$ for the coordinator or a router in depth d is defined as:

$$Cskip(d) = \begin{cases} 1 + Cm \times (Lm - d - 1) & \text{if } Rm = 1 \\ \frac{1 + Cm - Rm - CmRm^{Lm-d-1}}{1 - Rm} & \text{otherwise.} \end{cases} \quad (1)$$

The coordinator is said to be at depth 0; a node which is a child of another node at depth d is said to be at depth $d+1$. Address assignment begins from the ZigBee coordinator by assigning address 0 to itself. If a parent node at depth d has an address A_{parent}, the n-th child router is assigned to address $A_{parent} + (n-1) \times Cskip(d) + 1$ and n-th child end device is assigned to address $A_{parent} + Rm \times Cskip(d) + n$. An example of the address assignment is shown in Fig. 1. The $Cskip$ of the coordinator is obtained from Eq. (1) by setting $d = 0$, $Cm = 5$, $Rm = 3$, and $Lm = 2$. Then the child routers of the coordinator will be assigned to addresses $0+(1-1)\times 6+1 = 1$, $0+(2-1)\times 6+1 = 7$, $0+(3-1)\times 6+1 = 13$, etc. The address of the only child end device of coordinator is $0+3\times 6+1 = 19$. Note that, in ZigBee, the maximum network address capacity is $2^{16} = 65536$. This restricts that the coordinator can not decide the Cm, Rm, and Lm arbitrarily.

B. The Orphan Problem

By the above rules, the coordinator and routers can accept more routers and devices if they still have capacities. However, when a node can not join the network because all its neighbors have run out of their address capacities, we say the node has become an orphan. This situation may be relieved if there are remaining address spaces in other places of the network. Fig. 1 is a small-scale orphan problem. Here, we present some real implementation results of the ZigBee network formation procedure based on Jennic JN5121 [3]. Fig. 2 shows a deployment of 49 routers on a 360 $cm \times 360$ cm grid area. The grid size is 60 $cm \times 60$ cm. Nodes' transmission power is set to 150 mW, which can reach a transmission range around 100 to 200 cm. For each combination of (Rm, Lm), we conduct five experiments and observe the average number of orphans and the average end-to-end delay from the deepest node to the coordinator. Table I shows our experimental results. We can see that regardless of different (Rm, Lm) combinations, there always exist $30\% \sim 70\%$ orphans. Although a smaller Rm can lead to fewer orphans, it also results in longer end-to-end delay.

TABLE I
THE PERCENTAGES OF ORPHANS AND END-TO-END DELAYS UNDER DIFFERENT COMBINATIONS OF (Rm, Lm) IN THE TEST SCENARIO OF FIG. 2.

(Rm, Lm)	Orphans	Orphan Ratio	Delay
(6, 2)	35	71.4%	0.360s
(5, 3)	31.4	64.1%	0.447s
(4, 4)	30	61.2%	0.597s
(3, 5)	21.2	43.3%	0.681s
(2, 6)	27.8	56.7%	0.8125s
(3, 7)	17.2	35.1%	0.991s
(2, 8)	20.8	42.4%	1.197s

Since it is infeasible to conduct large-scale real tests, we also use simulations to make more observations. In Fig. 3, 800 nodes are randomly deployed on a circular field with a radius of 230 m. Nodes' transmission range is 25 m. To reduce orphans, given an Rm, we will set Lm to the maximum possible value. So we set (Rm, Lm) = (4, 7), (3, 9), and (2, 15) (these Lm values are the maximum possible ones for the given Rm) and $Cm = Rm$ (which means no end devices). In Fig. 3(a), since $Lm = 7$, the network cannot grow too deep, so a lot of nodes are left as orphans. In Fig. 3(b), since a larger $Lm = 9$ is used, there are much fewer orphans. However, there are still a lot of nodes at the edge unable to connect to the network. In Fig. 3(c), with a larger $Lm = 15$, orphans are significantly reduced. However, with the same setting, Fig. 3(d) shows a more extreme case where all neighboring nodes of one of the coordinator's children have been associated with other routers, making it a leaf node. This actually wastes a lot of address spaces. A smaller Rm may result in a non-shortest path from a router to the coordinator, thus causing a longer transmission delay and even more orphans if their routing path lengths exceed the constraint of Lm. In fact, assuming $Cm = Rm$, a router at depth d serving as a leaf implies a loss of $\frac{1-Rm^{Lm-d+1}}{1-Rm}$ address spaces. This is why a larger part of the network at the lower right side is unable to join the network. Note that this could happen because the ZigBee tree formation is asynchronous and nodes will compete to connect to nearby routers. These observations motivate us to design our schemes by trying to maintain sufficient children for nodes nearby the coordinator.

While both routers and end devices may become orphans, there capabilities are different. A router may accept more routers/devices, while an end device cannot. Further, their address calculation rules are also different as reviewed in Section II-A. For these reasons, we divide the orphan problem into two subproblems: BDDTF and EDMM problems. In the first BDDTF problem, we consider only router-capable devices and model the network by a graph $G_r = (V_r, E_r)$, where V_r consists of all router-capable devices and the coordinator t and E_r contains all symmetric communication links between nodes in V_r. Given parameters Cm, Rm, and Lm such that $Cm \geq Rm$, the goal is to assign parent-child relationships to nodes such that as many vertices in V_r can join the network as

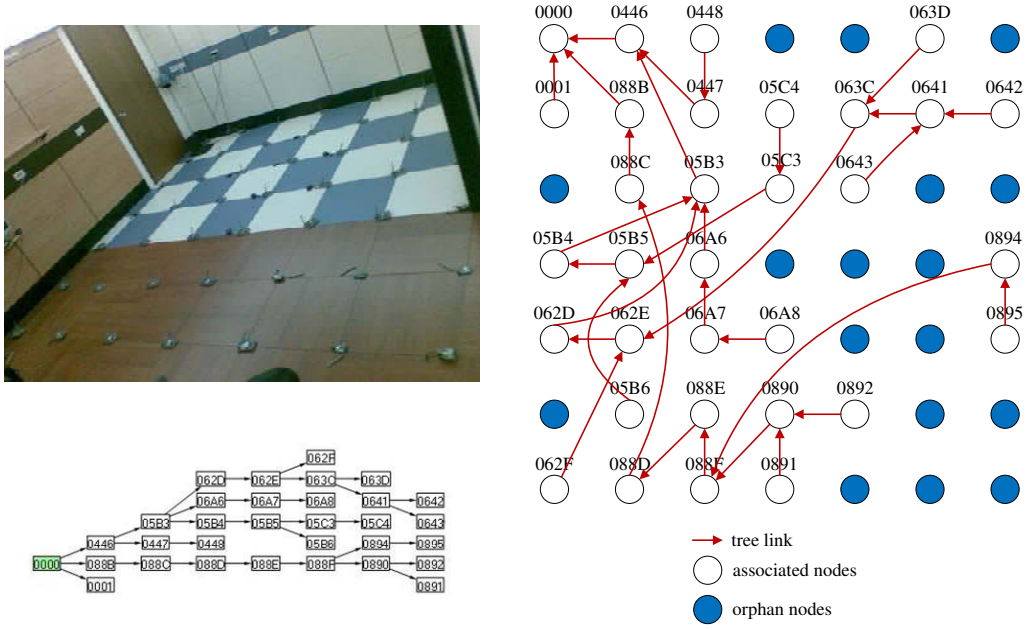


Fig. 2. A real-world ZigBee network formation example based on JN5121 in a 7x7 grid structure.

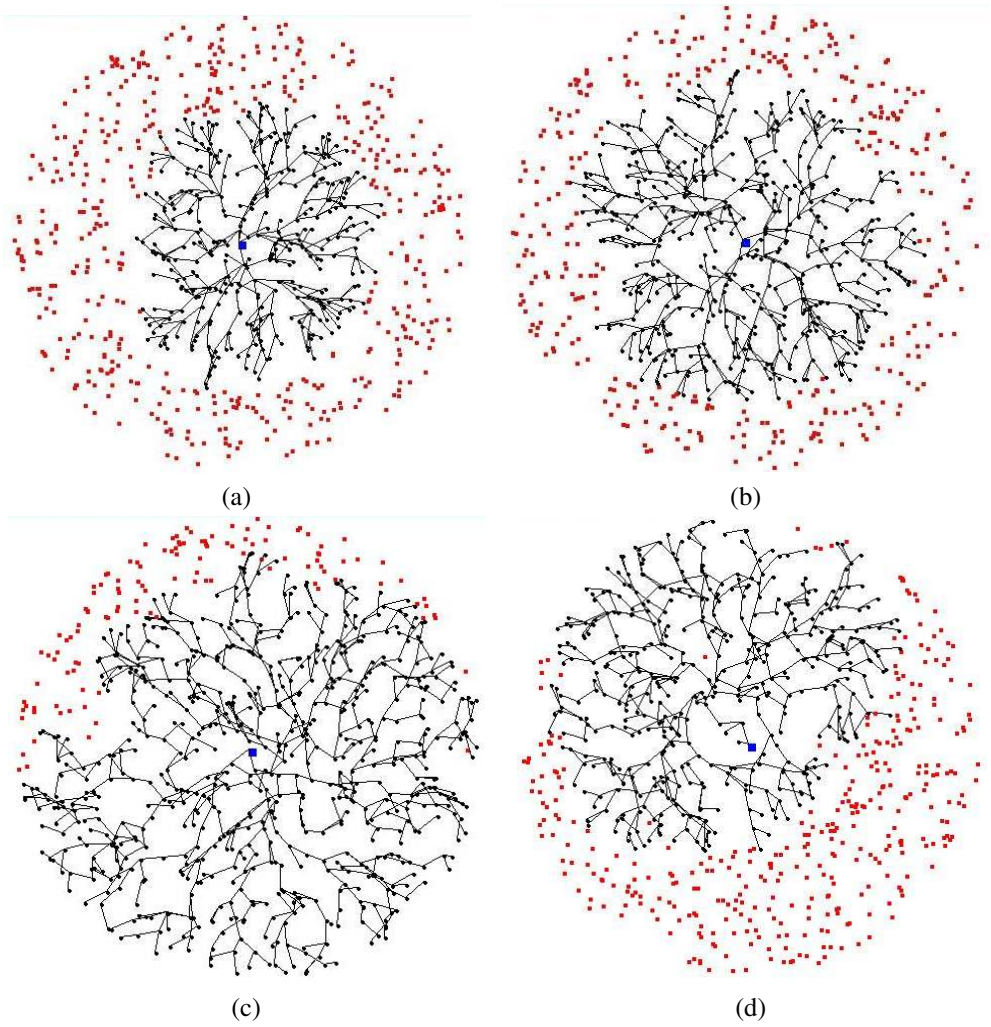


Fig. 3. ZigBee network formation examples with (Rm, Lm) equal to (a) (4, 7), (b) (3, 9), and (c-d) (2, 15). There are 461, 341, 120, and 351 orphan nodes, respectively.

possible. Below, we formulate this problem to a tree formation problem.

Definition 1: Given $G_r = (V_r, E_r)$, Rm, Lm, and an integer $N \leq |V_r|$, the *Bounded-Degree-and-Depth Tree Formation (BDDTF)* problem is to construct a tree T rooted at t from G_r such that T satisfies the ZigBee tree definition and T contains at least N nodes.

In [12], it is shown that the *Degree-Constrained Spanning Tree (DCST)* as defined below is NP-complete.

Definition 2: Given $G_c = (V_c, E_c)$ and a positive integer $K_c \leq |V_c|$, the *Degree-Constrained Spanning Tree (DCST)* problem is to find a spanning tree T_c from G_c such that no vertex in T_c has a degree larger than K_c.

Theorem 1: The BDDTF problem is NP-complete.

Proof: To prove that the BDDTF problem is NP-complete, we first show that the problem belongs to NP. Given a tree T in G_r, it is easy to check whether T satisfies the constraints of Rm and Lm and contains more than N nodes in polynomial time. Next, to prove that the BDDTF problem is NP-complete, we reduce the DCST problem to it. Let $G_c = (V_c, E_c)$ and integer K_c represent an arbitrary instance of the DCST problem. We can transform G_c to an instance of the BDDTF problem G_r by setting $V_r = V_c$, $E_r = E_c$, $N = |V_c|$, $Rm = K_c$, and $Lm \rightarrow \infty$ in polynomial time. We now claim that we can find a T_c for the DCST problem if and only if we can find a ZigBee-conformed tree T containing N nodes. To prove the *if* part, if there is a ZigBee-conformed tree T in G_r to connect $N = |V_c| = |V_r|$ nodes with parameters $Rm = K_c$ and $Lm \rightarrow \infty$, we can find a tree T_c in G_c to connect $N = |V_c|$ nodes as a spanning tree in G_c such that no vertex in T_c has a degree larger than K_c. Conversely, to prove the *only if* part, suppose that there is a spanning tree T_c to connect the nodes in G_c. Since $Rm = K_c$ and $Lm \rightarrow \infty$, there must exist a ZigBee-conformed tree $T = T_c$ in G_r containing $N = |V_c| \leq |V_r|$ nodes. So the theorem is proved. □

By Theorem 1, we can see that the first subproblem is intractable. Definition 1 and Theorem 1 imply the orphan problem is inevitable with any Rm and Lm. This also implies that there is no optimal decision for choosing Cm, Rm, and Lm to avoid the orphan problem.

After solving the BDDTF problem, we already have a tree T containing the coordinator and some routers. In the second EDMM subproblem, we will connect non-router-capable devices to the tree T constructed earlier following the ZigBee definition such that as many end devices are connected to T as possible. Toward this goal, we model the network by a bipartite graph $G_d = (\{\hat{V}_r \cup V_e\}, E_d)$, where $\hat{V}_r$ consists of the coordinator and all routers in T, excluding those at depth Lm (note that those at depth Lm are unable to accept more children), V_e consists of all end devices, and E_d contains all symmetric communication links between $\hat{V}_r$ and V_e. Each vertex $v \in \hat{V}_r$ can accept at most $C_v \geq (Cm - Rm)$ end devices. From G_d, we construct another bipartite graph $\tilde{G}_d = (\{\tilde{V}_r \cup \tilde{V}_e\}, \tilde{E}_d)$ as follows.

1) From each vertex $v \in \hat{V}_r$, generate C_v vertices v_1, v_2, ..., v_{C_v} in $\tilde{V}_r$.
2) From each vertex $u \in V_e$, generate a vertex u in $\tilde{V}_e$.
3) From each edge (v, u) in E_d, where $v \in \hat{V}_r$ and $u \in V_e$, connect each of the C_v vertices v_1, v_2, ..., v_{C_v} generated in rule 1 with the vertex u generated in rule 2. These edges form the set $\tilde{E}_d$.

Intuitively, we duplicate each $v \in \hat{V}_r$ into C_v vertices, and each edge $(v, u) \in E_d$ into C_v edges. These C_v vertices and C_v edges reflect the capability of router v to accept end devices. It is clear that $\tilde{G}_d$ is a bipartite graph with edges connecting vertices in $\tilde{V}_r$ and vertices in $\tilde{V}_e$ only. Since each vertex in $\tilde{V}_r$ is connected to at most one vertex in $\tilde{V}_e$, this translates the problem to a maximum matching problem as follows.

Definition 3: Given a graph $\tilde{G}_d = (\{\tilde{V}_r \cup \tilde{V}_e\}, \tilde{E}_d)$, the *End-Device Maximum Matching (EDMM)* problem is to find a maximum matching of $\tilde{G}_d$.

Given router tree T, the maximum matching problem in Definition 3 can be solvable in polynomial time [9]. Note that even with maximum matching, it does not guarantee that all end devices will be connected, so orphan end devices may still exist after solving the second subproblem. Below, we will propose several schemes for these two subproblems.

III. Algorithms for the BDDTF Problem

We propose two algorithms for the BDDTF problem. In our algorithms, we will repeatedly generate several BFS trees from G_r. For each tree being generated, we may decide to truncate some nodes if the tree is not conformed to the ZigBee definition. The truncation is done based on nodes' *association priorities* in the tree. Below, we show how such priorities are defined, given a BFS tree T in G_r:

- A node x has a higher priority than another node y if the subtree rooted at x in T has more nodes than the subtree rooted at y.
- If the subtrees rooted at nodes x and y have the same number of nodes, the one with less *potential parents* has a higher priority. A node regards a neighbor as a potential parent if this neighbor has a smaller hop count distance to the root in T than itself.

The above definitions are based on the considerations of address space utilization. The first rule is so defined because node x may have a better utilization. The second rule is so defined because a node with less potential parents is more likely to encounter difficulty when trying to attach to the network. For example, in Fig. 4, if $Rm = 3$, the coordinator will choose nodes A, B, and C as its child routers since they have larger subtrees. Similarly, B will choose D, E, and F as its child routers. However, if $Rm = 2$, the coordinator will choose A and B as its child routers. Further, B will choose D and E as its child routers. Node F is not selected because it has more (two) potential parents and thus has a higher probability to be connected in later stages of the formation.

A. Centralized Span-and-Prune Algorithm

Given a graph $G_r = (V_r, E_r)$, our goal is to find a tree $T = (V_T, E_T)$ from G_r conforming to the ZigBee tree definition.

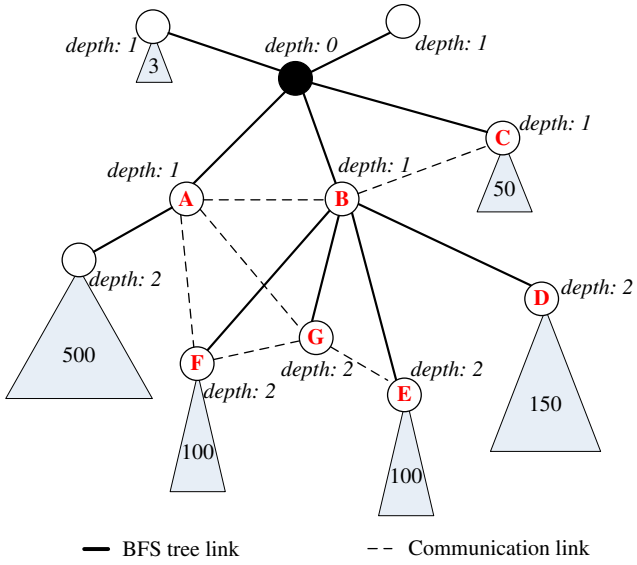


Fig. 4. Examples of priority assignment in our algorithm. (The numbers in triangles indicate sizes of the corresponding subtrees.)

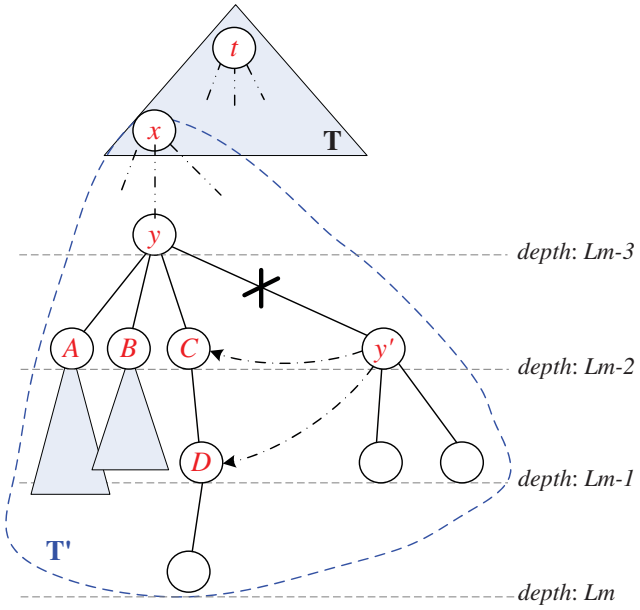


Fig. 5. An example of the Span-and-Prune algorithm.

The algorithm consists of a sequence of iterations. Initially, T contains only the coordinator t. Then in each iteration, there are two phases: *Span* and *Prune*. In the Span phase, we will pick a node in T, say x, and span from x a subtree T' to include as many nodes not yet in T as possible. Then we attach T' to T to form a larger tree. However, the new tree may not satisfy the ZigBee definition. So in the Prune phase, some of the newly added nodes in T' may be trimmed. The resulting tree is then passed to the next iteration for another Span and Prune phases. This is repeated until no more nodes can be added. Each node in the network will be spanned at most once. To keep track of the nodes yet to be spanned, a queue Q will be maintained. The algorithm is presented below.

1) Initially, let queue Q contains only one node t. Let the depth of t to zero. Also, let the initial tree $T = (\{t\}, \emptyset)$.
2) (*Span Phase*) Check if Q is empty. If so, the algorithm is terminated and T is the final ZigBee tree. Otherwise, let $x = \mathit{dequeue}(Q)$ and construct a spanning tree T' from x as follows. Assuming the depth of x in T to be $\mathit{depth}(x)$, we try to span a subtree from x with height not exceeding $Lm - \mathit{depth}(x)$ in G_r in a breadth-first manner by including as many nodes in $V_r - V_T \cup \{x\}$ as possible. Let the resulting tree be T'.
3) (*Prune Phase*) Attach T' to T by joining node x. Still, name the new tree T. Since some of the nodes in T' may violate the Rm parameter, we traverse nodes in T' from x in a breadth-first manner to trim T.
   a) When visiting a node, say y, set y as "traversed" and check the number of children of y. If y has more than Rm children, we will compute their priorities based on T' (refer to the definitions of nodes' priorities in a tree given in the beginning of this section). Only the Rm highest prioritized children will remain in T, and the other children will be pruned from T.
   b) When each node, say y', that is pruned in step 3(a) or 3(b), let $\mathit{tree}(y')$ be the pruned subtree rooted at y'. Since $\mathit{tree}(y')$ is pruned, we will try to attach y' to another node n in T' if n satisfies the following conditions: 1) n is neighboring to y' but not a descendant of y', 2) n is not traversed yet, and 3) $\mathit{depth}(n) + 1 + \mathit{height}(\mathit{tree}(y')) \leq Lm$. If so, we will connect the subtree $\mathit{tree}(y')$ to node n. If there are multiple such candidates, the one with a lower depth is connected first. If no such node n can be found, y prunes all its children. Then for each pruned child, we recursively perform this step 3(b) to try to reconnect it to T'. This is repeated until no further reconnection is possible.
4) After the above pruning, call the resulting tree T. For nodes that are newly added into T in step 3, insert them into queue Q in such a way that nodes with lower depth values are inserted first (these nodes will go through Span and Prune phases again). Then, go back to step 2.

To summarize, step 3(a) is to prune those nodes violating the Rm constraint. In order to allow more vertices to join the network, step 3(b) tries to recursively reconnect those pruned subtrees to T'. Step 4 prepares newly joining nodes in Q for possible spanning in step 2.

Fig. 5 illustrates an example. When being traversed, y decides to prune y' and keep A, B, and C as children. Step 3(b) will try to reconnect y' to C or D, which are the neighbors of y' in T' and are not traversed. In this example, only C can be considered because connecting to D violates the depth constraint Lm.

The computational complexity of this algorithm is analyzed as follows. The iteration from step 2 to step 4 will be executed at most $|V_r|$ times. In each iteration, the complexity of constructing the tree T' in step 2 is $O(N^2)$, where $N = |V_r| - |V_T|$ is the number of nodes still not connected to T. Step 3 checks all nodes in T' and will be executed at most $O(N)$ times. For

a run in Step 3 (assume visiting node y), the cost contains: 1) In step 3(a), y can use a linear search method to find Rm highest prioritized children and the computational cost is $O(D)$, where D is the degree of G_r. 2) Since the subtree size of y is at most $O(N)$ and a pruned node checks at most $O(D)$ neighbors to find its new parent, the cost of step 3(b) in a run is $O(ND)$. So, in one iteration, the time complexity of step 3 will be $O(N(D + ND)) = O(N^2D)$. Step 4 sorts new nodes of T according to their depth values, so the time complexity is $O(N^2)$. The complexity in each iteration is $O(N^2 + N^2D + N^2) = O(N^2D) = O(|V_r|^2D)$. Since there are at most $|V_r|$ iterations, the overall time complexity of this algorithm is $|V_r| \times O(|V_r|^2D) = O(|V_r|^3D)$. Although this complexity looks somewhat too high, we believe that using $|V_r|$ to bound N is too strong. Our experimental experience reveals that the value of N will degrade quickly because most nodes will be connected to the tree T after several iterations. So, the time complexity of an iteration is quite small in practice[1].

B. Distributed Depth-then-Breadth-Search Algorithm

The above Span-and-Prune algorithm is a centralized one. In this section, we present a distributed algorithm, which does a depth-first search followed by a breadth-first-like search. The depth-first search tries to form some long, thin backbones, which are likely to pass through high-node-density areas. Then from these backbones, we span the tree in a breadth-first-like manner. The algorithm is presented below.

1) (Depth Probing) Given a graph $G_r = (V_r, E_r)$, the coordinator t needs to probe the depth of the tree first. A Probe(*sender_addr, current_depth,* Lm) packet is used for this purpose. The Probe packets are flooded in a BFS-like manner, until a depth Lm is reached. Note that following the definition of ZigBee, before the final tree is determined, nodes will use their 64-bit MAC addresses to communicate with each other in this stage.
   This algorithm begins by the coordinator t flooding a Probe(*Addr*(t), 0, Lm) packet in the network, where *Addr*(t) is t's address. When a node v receives a Probe(*sender_addr*, *current_depth*, Lm) packet, it does the following:
   a) If this is the first time v receiving a Probe() packet, v sets its parent *par*(v) = *sender_addr* and its depth *depth*(v) = *current_depth* + 1. If *depth*(v) < Lm, v rebroadcasts a Probe(*Addr*(v), *depth*(v), Lm) packet.
   b) If this is not the first time v receiving a Probe() packet, it checks if *depth*(v) > *current_depth*+1 is true. If so, a shorter path leading to the coordinator is found. So v sets its parent *par*(v) = *sender_addr* and its depth *depth*(v) = *current_depth* + 1. If *depth*(v) < Lm, v rebroadcasts a Probe(*Addr*(v), *depth*(v), Lm) packet.
   Note that to ensure reliability, a node may periodically rebroadcast its Probe() packet. And each node can know the number of its potential parents by the Probe() packet.
2) (Probe Response) After the above probing, a BFS-like tree is formed. Each node then reports to its parent a Report() packet containing (i) the size of the subtree rooted by itself and (ii) the height of the subtree rooted by itself. In addition, each node v will compute a *tallest_child*(v), which records the child of v whose subtree is the tallest among all child subtrees.
3) (Backbone Formation) After the coordinator t receives all its children's reports, it will choose at most Rm children with the larger subtree sizes as backbone nodes. This is done by sending a Backbone() message to each of the selected children. When a node v receiving a Backbone() message, it further invites its child with the tallest subtree, i.e., node *tallest_child*(v), into the backbone by sending a Backbone() packet to *tallest_child*(v). After this phase, t has constructed a backbone with up to Rm subtrees, each as a long, thin linear path.
4) (BFS-like Spanning) After the above backbone formation, the coordinator can broadcast beacons to start the network. A node can broadcast beacons only if it has successfully joined the network as a router (according to ZigBee, this is achieved by exchanging Association_Request and Association_Response with its parent). In our rule, a backbone node must associate to its parent on the backbone, and its parent must accept the request. For each non-backbone node, it will compete with each other in a distributed manner by its association priority, where the association priority is defined by the size of the subtree rooted by this node in the BFS-like tree formed in step 1. A non-backbone node sends its association requests by specifying its priority. A beacon sender should wait for association requests for a period of time and sorts the received requests by their priorities. Then the beacon sender can accept the higher-priority ones until its capacity (Rm) is full.

Compared to the ZigBee protocol, this algorithm requires nodes to broadcast three extra packets (Probe(), Report(), and Backbone()). A Probe() packet needs to flood to the whole network and thus needs an efficient broadcast scheme (this is beyond the scope of this paper). Let n be the total number of nodes in the network. Below, we will show that the additional time and message complexity against to ZigBee are $O(Lm)$ and $O(n)$, respectively .

To see the additional time complexity, observe that the coordinator t will issue Probe() to check the depth of the tree and a node v will rebroadcast it only when *depth*(v) < Lm or it can find a shorter path to t. So, the additional time complexity will be bounded by $O(Lm)$. In the process of finding the *tallest_child*, each node will report to its parent. Because the reporting is started from leaf nodes, the additional time complexity is also bounded by $O(Lm)$. Finally, the backbone formation will be triggered by t to construct a long, thin linear path. Again, the additional cost is $O(Lm)$. Overall, the additional time complexity of our algorithm against ZigBee

[1]By our simulation, in average, almost 75% of nodes can be connected to the tree T in first iteration. After second iteration, almost 88% of nodes can be connected to the tree T.

is bounded by $O(Lm)$.

To see the additional message complexity of our algorithm, observe that the probing step is similar to a BFS tree construction, so the message complexity is $O(n)$. In the step of finding *tallest_child*, because each node will only report to its parent once, the message cost is also $O(n)$. Finally, t will send Backbone() packets to its Rm selected children, who will further invite their children with the tallest subtrees. The cost is at most $O(Rm + Rm \times (Lm - 1))$. Overall, the additional message complexity against ZigBee is $O(n)$.

IV. Algorithms for the EDMM Problem

In Section II-B, we have formulated the EDMM problem as a maximum matching problem in a bipartite graph. It is already known that there exists optimal polynomial-time algorithms to solve this problem. Below, we show how to use the maximum matching algorithm in [9] to solve our problem. Recall that after connecting routers in the BDDTF problem, we will obtain a bipartite graph $\tilde{G_d} = (\{\tilde{V_r} \cup \tilde{V_e}, \tilde{E_d}\})$. From $\tilde{G_d}$, we can find a maximum matching as following.

1) Try a greedy approach by first matching those vertices with small degrees. We denote this matching edge set as M. Then, we transform the undirected graph $\tilde{G_d}$ to a directed graph $\tilde{G_d}'$ by directing the edges in M to point from $\tilde{V_e}$ to $\tilde{V_r}$ and directing the edges not in M to point from $\tilde{V_r}$ to $\tilde{V_e}$.
2) Apply a DFS search on $\tilde{G_d}'$ starting from any node of $\tilde{V_r}$. If $\tilde{G_d}'$ has any *alternating path* [9] P staring from $\tilde{V_e}$ and ending at $\tilde{V_r}$, we mark all edges of P belonging to M as not belonging to M, and vice versa. (It is easy to see that P must be of an odd length.) Then we reconstruct $\tilde{G_d}'$ based on the new M.
3) Repeat step 2 until each node in $\tilde{V_r}$ has been searched once. Then the final M is a maximum matching of $\tilde{G_d}$.

As shown in [9], the complexity of the above procedure is $O((|\tilde{V_r}|)(|\tilde{V_r}| + |\tilde{V_e}| + |\tilde{E_d}|))$.

The above algorithm is a centralized one. In practice, we need a distributed algorithm to allow routers to connect end devices in a decentralized way. Below, we present a distributed algorithm, which has a *greedy phase* followed by a *probing phase*. In the greedy phase, the routers will accept end devices which have less potential associable routers. Then, each orphan router will try to probe a 3-hop alternating path P as discussed above to relieve its orphan situation. The probing process can be executed before a timer T_{probe} expires. After T_{probe} expires, an end device can not change its parent.

1) (Greedy phase) Each router will periodically broadcast beacon packets with a reserve bit to indicate whether it still has capacity to accept more end devices. Each end device e will overhear beacons from routers and, based on these beacons, compute the number N_e of its neighbor routers with their reserve bits on. In the case of $N_e = 0$, e is a potential orphan. If $N_e > 0$, e will try to perform the association procedure by providing its N_e value to routers. Routers simply accept as many end devices as possible with smaller N_e first (intuitively, a smaller N_e means less potential parents).
2) (Probing phase) After the greedy phase, each associated end device will broadcast its new N_e value (note that this value counts its parent as well as those neighboring routers which still have remaining capacities). For an orphan end device e (with $N_e = 0$), it can try to resolve its situation as follows:
   a) A Probe() packet[2] can be sent by e to any neighboring router r.
   b) When r receives the Probe() from e, r can check if it has a child end device e' such that $N_{e'} \geq 2$. If so, r will send a Probe() packet to e' to ask e' to switch to another router.
   c) On reception of r's Probe(), e' will try to associate with another router other than its current one. If it succeeds, a Probe_Ack() will be returned to r; otherwise, a Probe_Nack() will be returned.
   d) When r receives the result from e', a Probe_Ack() or a Probe_Nack() will be returned to e accordingly. In the former case, e will associate with r. In the latter case, e will try another router by going back to step (a), until timer T_{probe} expires.

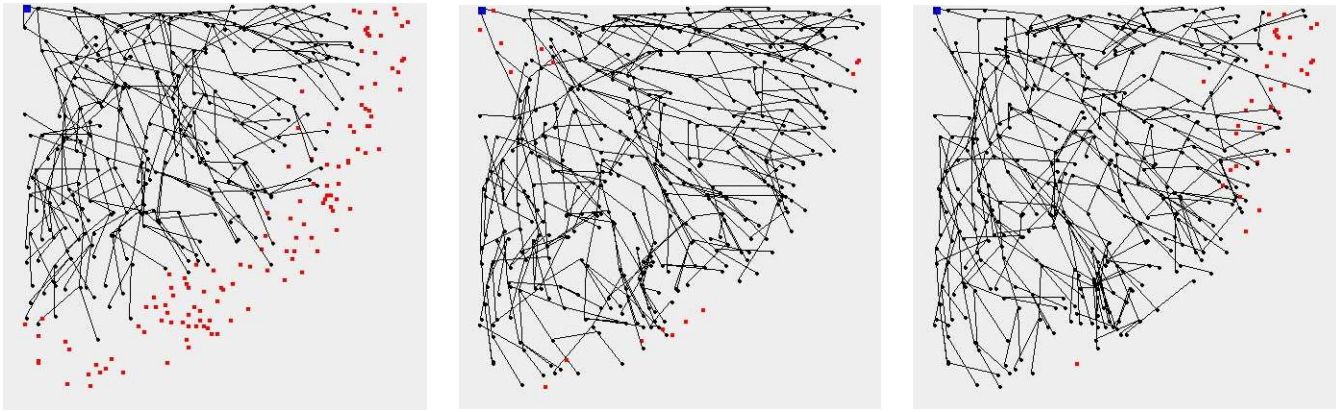
The above protocol allows an orphan to probe 3-hop paths. It is not hard to extend this protocol to allow probing longer paths at higher costs (we leave it to the audience). Next, we analyze the additional time and message complexity required for this protocol against the original ZigBee. The additional time complexity will be bounded by $O(T_{probe})$. The additional message complexity is incurred by the *probing phase*. Our protocol has a progressive property because each probe may reduce one orphan end device. So the extra cost will be bounded by a polynomial function of the number of end devices. If longer alternating paths are explored, the cost will be higher. However, one may use the timer T_{probe} to bound the cost.

V. Simulation Results

A simulator has been implemented based on Java. First, we compare our Span-and-Prune algorithm (SP) and Depth-then-Breadth-Search algorithm (DBS) against the ZigBee algorithm (ZB) in their capabilities to relieve the BDDTF problem under random and regular node deployment. Next, through varying the combinations of Cm, Rm, and Lm, we further show the superiority of SP and DBS even under different node density environment. We also investigate in more details the advantage of the backbone probing in our DBS scheme. Finally, we will show the performance of our distributed EDMM scheme to connect end devices.

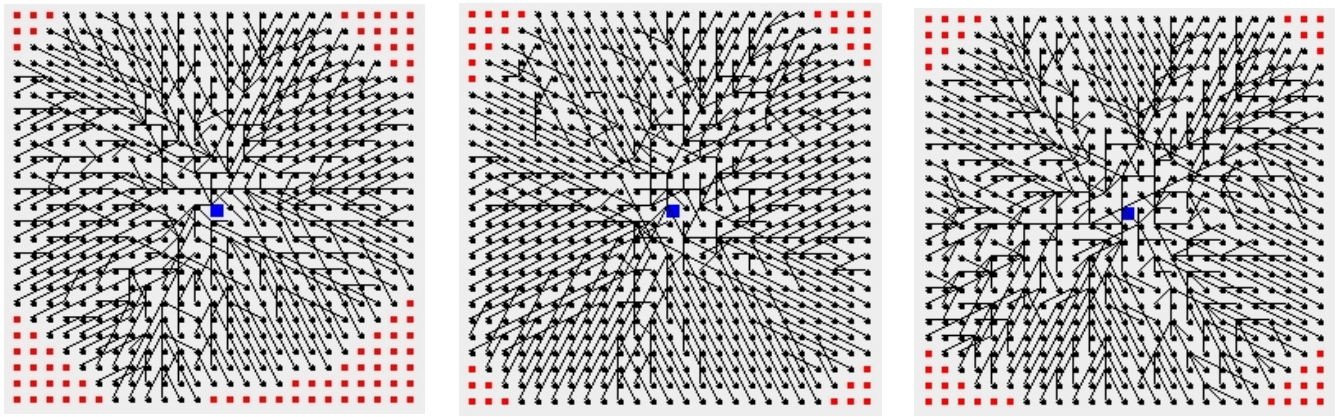
A) Random vs. Regular Networks: In Fig. 6, we test a $90°$-sector area with a radius of 200 m and with 400 randomly deployed router-capable nodes each with a transmission range of 32 m. We set $Cm = Rm = 2$ and $Lm = 8$. ZB, SP, and DBS algorithms incur 110.2, 13.7, and 37.9 orphan routers, respectively, in average. DBS only incurs slightly more orphans than the centralized SP does. In particular, we see that both SP and DBS may leave some nodes nearby the

[2]This Probe() should be distinguished from the Probe() in Section III-B.



(a) (b) (c)

Fig. 6. Network formation results in a 90°-sector area by (a) ZB, (b) SP, and (c) DBS.



(a) (b) (c)

Fig. 7. Network formation results in a 25×25 grid area by (a) ZB, (b) SP, and (c) DBS.

coordinator unconnected due to the Rm constraint but can reach farther nodes. Fig. 7 considers a 25×25 regular grid with a grid distance of 10 m. Nodes' transmission distances are 23 m. We set $Cm = Rm = 4$ and $Lm = 7$. In this case, ZB, SP, and DBS incur 70.2, 37.2, and 40.4 orphan routers, respectively. ZB performs the worst. DBS performs closely to the centralized SP.

B) Impact of Link Density on the BDDTF Problem: We simulate 800 randomly distributed router-capable devices in a circular region with a radius of 200 m with the coordinator at the center. We restrict $Cm = Rm$ and vary Rm and Lm to observe the number of orphan routers. Table II shows the address spaces of different combinations of Cm, Rm, and Lm, which can clearly accommodate much more than 800 nodes ideally. We set nodes' transmission ranges to 35 m and 60 m. Since the network area and the number of nodes are fixed, a larger transmission range actually means denser links among neighboring nodes. As Fig. 8 shows, denser links do lead to much less orphans. However, transmission range depends on hardware features as well as deployment needs, which are sometime uncontrollable. In addition, we see that in all cases, SP performs the best, followed by DBS and then ZB.

C) Impact of Rm and Lm on the BDDTF Problem: The above Fig. 8 indicates that increasing Lm can more effectively reduce orphan routers as opposed to increasing Rm. In Fig. 9, we further fix Lm and vary Rm to conduct our tests. We see that the orphan situation can benefit less by enlarging Rm under low link density. However, as the link density is higher, enlarging Rm is still quite effective. This is because a higher link density will allow a node to have more potential children. Our scheme can save space for Rm and thus allow a larger space for Lm. For example, in Fig. 8(a), SP incurs nearly the same number of orphan routers in the (3, 7) case (resp., the (3, 8) case) as ZB does in the (3, 8) case (resp., the (3, 9) case). In Fig. 9(b), SP incurs nearly the same number of orphan routers when $Rm = 6$ as ZB does when $Rm = 11$. Saving the space for Rm can allow a larger Lm, which can in turn relieve the orphan problem. This shows the benefit of our SP scheme.

D) Impact of the Backbone Formation in DBS: In DBS, there is a backbone formation to choose subtrees of larger sizes. We modify DBS to a DBS-NB (NB = non-backbone) scheme, which works similar to DBS but does not form backbones as in DBS (i.e., all nodes in step 4 are considered as non-backbone ones). The results are in Fig. 10, which clearly shows the importance of the formation process.

TABLE II
IDEAL ADDRESS SPACES OF VARIOUS COMBINATIONS OF Rm AND Lm ($Cm = Rm$).

($Cm = Rm, Lm$)	(3, 7)	(3, 8)	(3, 9)	(4, 6)	(4, 5)	(5, 5)	(6, 5)
Total address spaces	3280	9841	29524	5461	1365	3906	9331
($Cm = Rm, Lm$)	(6, 4)	(7, 4)	(8, 4)	(9, 4)	(10, 4)	(11, 4)	(12, 4)
Total address spaces	1555	2801	4681	7381	11111	16105	22621

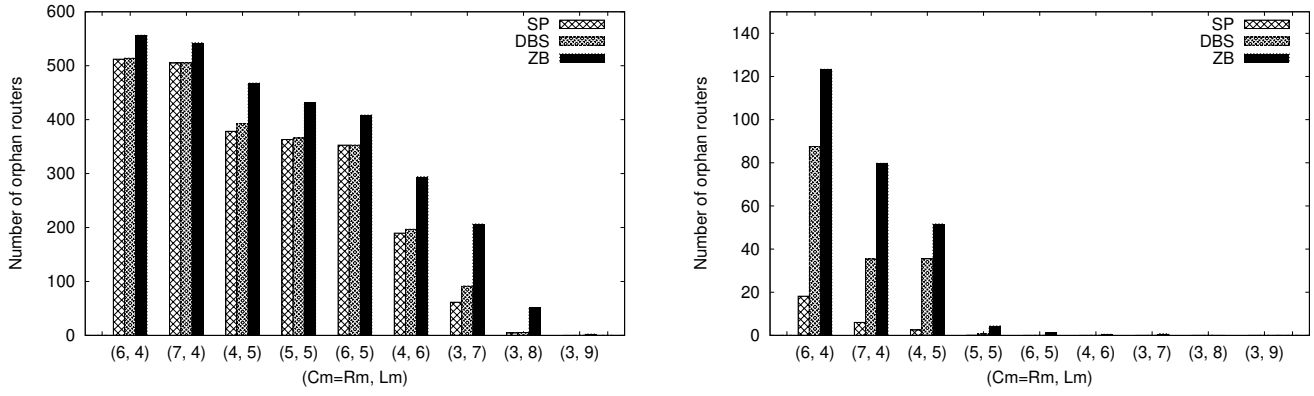


Fig. 8. Comparison on the number of orphan routers when the transmission range is (a) 35 m and (b) 60 m.

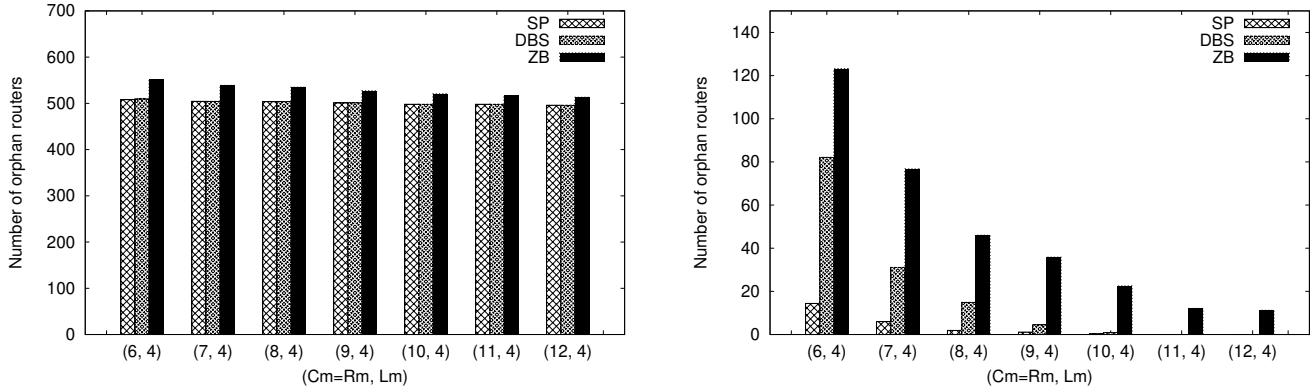


Fig. 9. Comparison on the number of orphan routers by fixing Lm and varying Rm when the transmission range is (a) 35 m and (b) 60 m.

E) The EDMM Problem: In these experiments, we simulate the networks with both routers and end devices. We randomly place 800 routers and 8000 end devices in a circular area of radius 200 m with the coordinator at the center. Routers' transmission ranges are 35 m, and end devices' are 15 to 30 m. An end device can only associate to a router located within its transmission range. We set $Cm = 15$, $Rm = 3$, and $Lm = 8$. We use SP to connect routers and then apply the centralized maximum matching scheme (Max-Match), our distributed matching scheme (Dis-Match), or ZigBee (ZB) to connect end devices. In all cases of end devices' transmission ranges, Fig. 11 shows that Dis-Match can significantly reduce orphan end devices as opposed to ZB, and perform quite close to Max-Match.

VI. CONCLUSIONS

In this paper, we have identified a new orphan problem in ZigBee-based wireless sensor networks. We show that the problem is non-trivial because a device is not guaranteed to join a network even if there are remaining address spaces in other places of the network. We model this orphan problem as two subproblems, namely the BDDTF problem and the EDMM problem. We prove that the BDDTF problem is NP-complete and propose a two-stage network formation policy, which can effectively relieve the orphan problem. Compared to the network formation scheme defined in ZigBee, our algorithms can significantly reduce the number of orphan routers. Contrarily and interestingly, we show that the EDMM

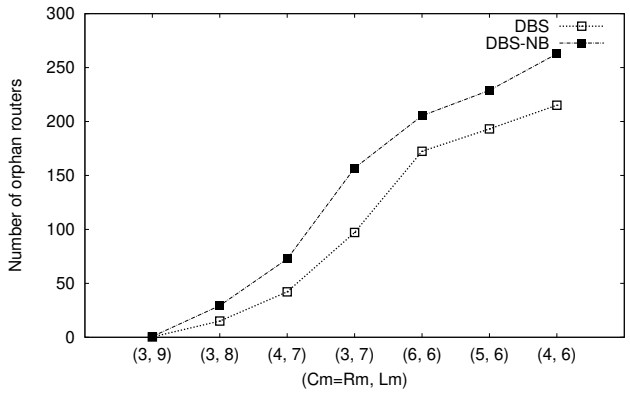


Fig. 10. Comparison of DBS and DBS-NB schemes (800 routers; a circular sensing field of a radius of 200 m; nodes' transmission range 35 m).

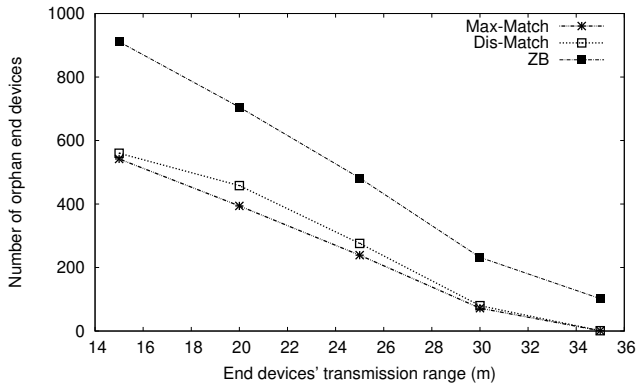


Fig. 11. Comparison on the number of orphan end devices at various transmission ranges.

problem is solvable in an optimal way in polynomial time by a centralized algorithm and propose a distributed matching algorithm. Our simulations also show that our distributed algorithm performs quiet closely to the maximum matching algorithm. These results are expected to significantly enhance the connectivity of ZigBee networks.

ACKNOWLEDGMENT

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 95-2221-E-009-058-MY3, 96-2218-E-009-004, 97-3114-E-009-001, 97-2221-E-009-142-MY3, and 97-2218-E-009-026, by MOEA under grant 94-EC-17-A-04-S1-044, by ITRI, Taiwan, and by III, Taiwan.

REFERENCES

[1] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.
[2] Habitat monitoring on great duck island. http://www.greatduckisland.net/technology.php.
[3] Jennic JN5121. http://www.jennic.com/.
[4] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/.
[5] Texas Instruments CC2431. http://www.ti.com/.
[6] J. Bachrach, R. Nagpal, M. Salib, and H. Shrobe. Experimental results and theoretical analysis of a self-organizing global coordinate system for ad hoc sensor networks. *Telecommunications Systems Journal*, 26(2-4):213–234, 2004.
[7] G. Bhatti and G. Yue. Structured addressing scheme for wireless networks. United States Patent 20070060134, 2007 (pending).
[8] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.
[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.
[10] A. Czumaj and W.-B. Strothmann. Bounded degree spanning trees. In *Proc. of European Symposium on Algorithms (ESA)*, 1997.
[11] T. Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.
[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
[13] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. of Hawaii Int'l Conference on Systems Science (HICSS)*, 2000.
[14] C.-F. Huang, Y.-C. Tseng, and L.-C. Lo. The coverage problem in three-dimensional wireless sensor networks. *Journal of Interconnection Networks*, 8(3):209–227, 2007.
[15] IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), 2003.
[16] M. Kochhal, L. Schwiebert, and S. Gupta. Role-based hierarchical self organization for wireless ad hoc sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.
[17] J. Konemann, A. Levin, and A. Sinha. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2):117–129, 2004.
[18] J. Konemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proc. of ACM Symposium on Theory of Computing (STOC)*, 2000.
[19] S. Kulkarni, A. Iyer, and C. Rosenberg. An Address-Light, Integrated MAC and Routing protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 14:793–806, 2006.
[20] Q. Li, M. DeRosa, and D. Rus. Distributed algorithm for guiding navigation across a sensor network. In *Proc. of ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.
[21] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(8):1044–1056, 2006.
[22] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of IEEE INFOCOM*, 2001.
[23] D. Niculescu and B. Nath. DV based positioning in ad hoc networks. *Telecommunications Systems Journal*, 22(1-4):267–280, 2003.
[24] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, October 2000.
[25] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. *The Computer Journal*, 47(4):448–460, 2004.
[26] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. *IEEE Computer*, 39(7):55–62, 2006.
[27] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2002.
[28] J. Zheng and M. J. Lee. A resource-efficient and scalable wireless mesh routing protocol. *Ad Hoc Networks*, 5:704–718, 2007.
[29] ZigBee specification version 2006, ZigBee document 064112, 2006.

Meng-Shiuan Pan received the B.S. and M.S. degrees from the National Chung Cheng University and National Tsing Hua University, Taiwan, in 2001 and 2003, respectively. He obtained his Ph.D. in the Department of Computer Science, National Chiao Tung University, Taiwan, in 2008. His research interests include mobile computing and wireless communication.

Chia-Hung Tsai received his B.S. and M.S. degrees in Computer Science and Information Engineering from the National Chiao-Tung University, Taiwan, in 2004 and 2006, respectively. He is currently pursuing Ph.D. in the Department of Computer Science, National Chiao-Tung University, Taiwan. His research interests include wireless communication and sensor networks.

Yu-Chee Tseng obtained his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He is Professor (2000–preset), Chairman (2005–present), and Associate Dean (2007–present) at the Department of Computer Science, National Chiao-Tung University, Taiwan. He is also Adjunct Chair Professor at the Chung Yuan Christian University (2006–present). Dr. Tseng received the Outstanding Research Award, by National Science Council, R.O.C., in both 2001–2002 and 2003–2005, the Best Paper Award, by Int'l Conf. on Parallel Processing, in 2003, the Elite I. T. Award in 2004, and the Distinguished Alumnus Award, by the Ohio State University, in 2005. His research interests include mobile computing, wireless communication, and parallel and distributed computing.

Dr. Tseng serves on the editorial boards for Telecommunication Systems (2005–present), IEEE Trans. on Vehicular Technology (2005–2009), IEEE Trans. on Mobile Computing (2006–present), and IEEE Trans. on Parallel and Distributed Systems (2008–present).